

CYBER TOOLBOX

Release 2.1

Users Guide

目次

1.はじめに.....	1
CyberToolbox とは?	1
2.インストール.....	1
必要システム構成	1
CTB のインストール	1
CTB の起動.....	1
3.基本操作.....	2
ユーザーインターフェイス概要	2
ワールドウインドウ.....	3
視点透視ウインドウ	3
正射影ウインドウ / プリティブウインドウ.....	3
ダイアグラムウインドウ / モジュールウインドウ.....	3
仮想空間へのオブジェクト追加/保存	4
オブジェクト追加	4
オブジェクト保存	6
ジオメトリオブジェクト編集.....	7
ジオメトリオブジェクト選択	7
ジオメトリオブジェクト移動	7
マテリアル/テキストチャ設定	8
仮想空間での移動.....	9
歩行移動	9
視点位置の初期化	10
シーングラフの編集.....	10
ノードの追加.....	10
ノードの移動.....	11
ノードの削除.....	12
ノードの編集.....	12
イベントダイアグラム編集.....	13
ダイアグラムの追加	14
モジュールの追加.....	14
モジュール同士の接続	15

シミュレーションの実行/停止.....	16
4. チュートリアル.....	1
掛け算.....	1
掛け算動作の定義	1
ダイアグラム動作の説明.....	3
シミュレーションの実行.....	4
回転する立方体.....	4
回転する立方体.....	5
立方体オブジェクトの追加.....	5
回転動作の定義.....	6
ダイアグラム動作の説明.....	9
シミュレーションの実行.....	9
光る球体.....	10
球体オブジェクトの追加	10
点滅動作の定義.....	11
ダイアグラム動作の説明.....	14
シミュレーションの実行.....	14
インターネットでのコンテンツ公開.....	15
コンテンツ関連ファイルのアップロード.....	15
5. 操作説明.....	1
ワールドウインドウ.....	1
ワールドツリー	1
ツールバー.....	9
透視投影ウインドウ.....	14
仮想空間の移動.....	14
オブジェクトの選択	16
ツールバー.....	17
正射影ウインドウ.....	20
オブジェクト編集.....	20
表示平面/領域	25
ツールバー.....	28
ダイアグラムウインドウ.....	31
ダイアグラム名.....	31

モジュール操作	32
データフローライン操作	34
ツールバー	36
6.動作定義.....	1
動作のしくみ	1
イベント.....	2
システムイベント.....	3
ユーザー定義イベント.....	4
ダイアグラム.....	9
モジュール概要	12
モジュール詳細	15
String クラス.....	15
Numeric クラス.....	23
Math クラス.....	29
Filter クラス.....	36
Geometry クラス.....	41
Boolean クラス.....	44
Object クラス.....	47
Material クラス.....	50
Light クラス	54
Viewpoint クラス.....	59
Interpolator クラス.....	62
Misc クラス.....	64
7.互換性.....	1
VRML ブラウザ/プラグイン	1
Java サポート.....	1
ROUTE 処理順序	2

1.はじめに

CyberToolbox とは?

CyberToolbox(CTB)は VR(Virtual Reality)初心者から中級者向けの VRML(Virtual Reality Modeling Language) 2.0/97 オーサリングツールです。VRML は現在インターネットで最も標準的な 3D ファイルフォーマットで、従来のファイルフォーマットようなジオメトリ情報だけではなく、仮想空間にあるオブジェクト動作まで含めた非常にインタラクティブなフォーマットです。現在では、VRML を利用したインタラクティブで楽しい 3D コンテンツが、世界中のホームページで公開されています。

しかし、VRML でこのようなインタラクティブな 3D コンテンツを作成しようとすると、ある程度の VRML についての知識が必要となり、初心者には多少敷居が高いものでした。また、よりインタラクティブで楽しい仮想空間コンテンツを構築しようとすると、必然的に Java や JavaScript などのプログラミング言語を利用してコンテンツを製作しなければならず、プログラミング経験の少ないユーザーには、なかなか良いコンテンツは製作できませんでした。

このようなコンテンツ制作の問題を解決するために、CTB にはプログラミング初心者でも、簡単に仮想空間動作を構築できるような、アイコンベースのビジュアルプログラミング言語が搭載されています。CTB は、従来の VRML オーサリングツールと違い、仮想空間動作をすべて視覚的に定義していく、まったく新しいタイプのツールです。仮想空間動作を定義するのに、もはや専門的な VRML や Java の知識は必要ありません。複雑な仮想空間動作も、積み木を組み立てていくような感覚で、マウスを使い簡単に定義できます。

また CTB には、VRML の基本的編集機能として、ビジュアルなシーングラフエディタやモデラーが付属していますので、CTB だけで一通りの VRML アプリケーションを構築することも可能です。また CTB は他の VRML2.0/97 対応のモデラーで製作したデータを読み込むことができるため、普段使い慣れている VRML2.0/97 ツールとの連携もできます。

2. インストール

必要システム構成

CTB を実行するには、最低限以下の機材が必要です。

Pentium プロセッサ以上を搭載した PC

32MB メモリ / 40MB のディスクスペース

Windows95/98 または WindowsNT

Windows 互換マウス

音源ボード (オプション)

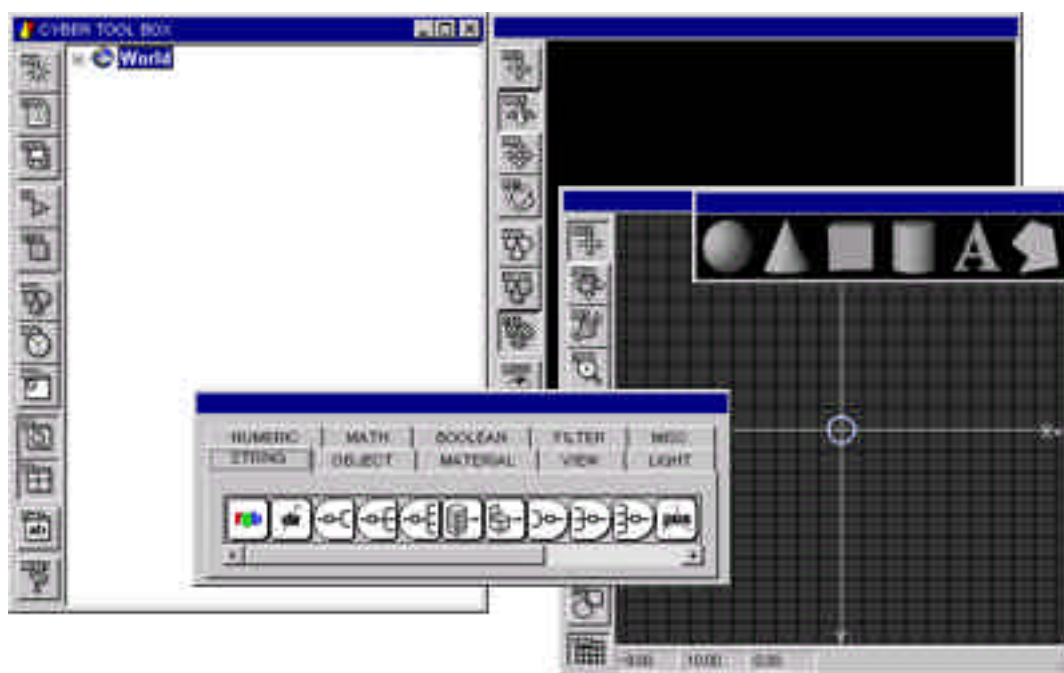
OpenGL 対応グラフィックスボード (オプション)

CTB のインストール

ダウンロードしたパッケージをに含まれる「setup.exe」をダブルクリックしてインストーションプログラムを起動して、メニューに従いインストール作業を進めて下さい。

CTB の起動

スタートメニューから「プログラム CyberToolbox 2.0」を選択しCTB を起動させて下さい。CTB を起動すると、以下のような5種類のウインドウが表示されます。



3.基本操作

ユーザーインターフェイス概要

CTB は、ワールドウィンドウ、透視投影ウィンドウ、正射影ウィンドウ、プリミティブウィンドウ、モジュールウィンドウ、ダイアグラムウィンドウの6種類のウィンドウにより構成されています。これらのウィンドウは独立して表示されており、あるウィンドウでの操作結果は即座に他のウィンドウに反映されます。例えば、仮想空間に車がある場合に、正射影ウィンドウで表示されている車のオブジェクトの移動すると、透視投影ウィンドウでも同じように、そのオブジェクトが移動します。



多くのウィンドウには左にツールバーがあり、各操作を示すアイコンが表示されています。このアイコンをマウスで選択することにより、各種操作を選択します。

ワールドウインドウ

仮想空間に存在するオブジェクトや、その動作を定義しているイベントなどの、全ての情報がツリー構造で表示されます。VRML ファイルを読み込んだり、球などのプリミティブオブジェクトを追加したり、球を回転させる動作を定義するダイアグラムを生成したりすると、その結果がツリー構造で確認できます。

視点透視ウインドウ

仮想空間に存在するオブジェクトが3次的に表示されます。主に仮想空間オブジェクトの3次的な位置関係や、シミュレーション動作の確認を行います。また、マウスで仮想空間をウォークスルーしたり、オブジェクトの選択を行います。

正射影ウインドウ / プリミティブウインドウ

現在の仮想空間に存在するオブジェクトが2次的に表示されます。主に仮想空間オブジェクトの2次的な位置の確認や、オブジェクトの移動/回転などの編集作業を行います。表示するウインドウは、XY/YZ/XZ 平面のいずれかから選択できます。またプリミティブウインドウにあるアイコンをドラッグ&ドロップすることにより、新規のプリミティブオブジェクトを追加できます。

ダイアグラムウインドウ / モジュールウインドウ

仮想空間の動作定義を行います。仮想空間動作は、モジュールウインドウのアイコンを組み合わせで行うビジュアルプログラミング言語にて定義を行います。モジュールは、モジュールウインドウに各クラスに分類されて表示されてます。モジュールには、オブジェクトの位置を設定するものや、数値の足し算や引き算を行うものなど多種多様なものがあります。これらのモジュールの線で接続していくことにより、仮想空間動作の定義を行います。

仮想空間へのオブジェクト追加/保存

仮想空間は、物体形状を表現するジオメトリや、その外観を表現する色やテクスチャなどの複数のオブジェクトから構成されています。例えば、車を運転するアプリケーションの場合には、仮想空間は、車や道路、建物などの色々なオブジェクトにより構成されています。これらの仮想空間にあるオブジェクトは、ワールドウインドウのシーングラフツリーで確認できます。これらのオブジェクトは、VRML ではノードと呼ばれており、これらのオブジェクトはシーングラフと呼ばれる階層構造にて管理されています。

ここでは、仮想空間にオブジェクトを追加する3種類の方法と、仮想空間をファイルに保存する方法を説明します。

オブジェクト追加

CTB の起動直後の仮想空間は、オブジェクトは一つもなく空の状態です。仮想空間にオブジェクトを追加する方法としては以下の3種類の方法があります。

- ・ ワールドウインドウで、VRML ファイルの読み込み
- ・ ワールドウインドウで、シーングラフへのノード追加
- ・ プリビューウインドウから、正射影ウインドウへのプリティブのドラッグ&ドロップ

この3通りの方法で、一つの球オブジェクトを仮想空間に追加する方法を示します。

VRML ファイルの読み込み

CTB では、VRML2.0/97 形式のファイルを読み込み、そのファイルに含まれるシーングラフ情報を仮想空間に追加できます。CTB のモデラーは基本的な機能しかないため、複雑なオブジェクトを仮想空間に追加するには、VRML2.0/97 形式ファイルで出力できる他のモデリングツールでオブジェクトを作成し、CTB でそのファイルを読み込む方法が効率的です。

VRML ファイルを読み込むには、ワールドウインドウの「ファイルロード」ボタンを押して、目的のファイルを表示されるダイアログで選択して下さい。



シーングラフへのノード追加

ワールドウインドウで、直接シーングラフにノードを追加する方法です。この方法は、ある程度のVRML2.0/97 の知識が必要になります。ここでは以下のワールドウインドウのシーングラフツリーに表示されている各ノードにより、球オブジェクトを仮想空間に追加する方法を示します。



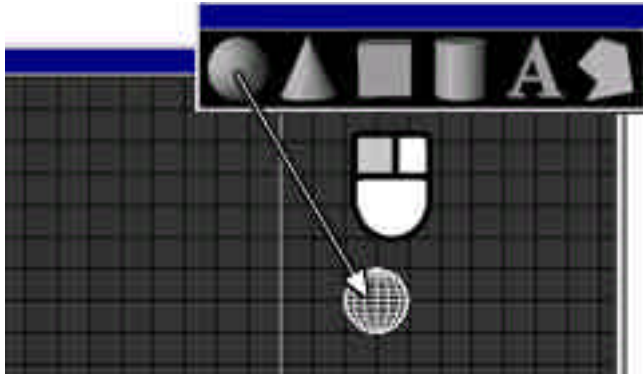
最初に、ワールドウインドウの"SceneGraph"をマウス左ボタンクリックで選択してから、"新規 ノード追加"ボタンを押すとノード追加ダイアログが表示されます。このダイアログからTransform ノードを選択するとシーングラフに Transform ノードが追加されます。



後は同様の手順で、追加した Transform ノードを選択し Shape ノードを追加、追加した Shape ノードに Box ノードを追加して下さい。

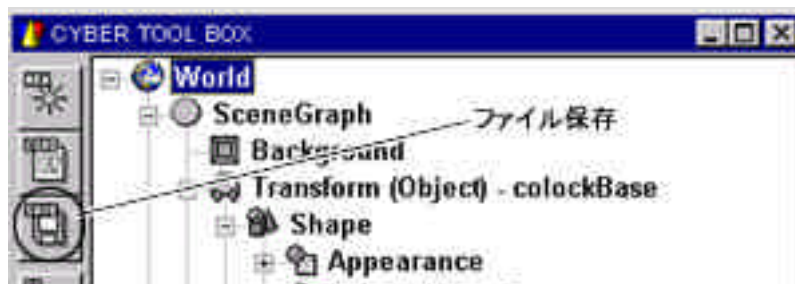
プリミティブウインドウからの追加

CTB では、球や円錐などの基本プリミティブを、仮想空間へマウス操作で簡単に追加することができます。例えば、球オブジェクトを仮想空間に追加するには、プリミティブウインドウにある球アイコンをマウス左ボタンでドラッグし、正射影ウインドウの追加したい位置にドロップして下さい。



オブジェクト保存

仮想空間に追加されたオブジェクトは、その動作定義を含めて1つのVRML2.0/97ファイルに保存できます。仮想空間をファイルに保存するにはワールドウインドウの"ファイル保存"ボタンを押して下さい。

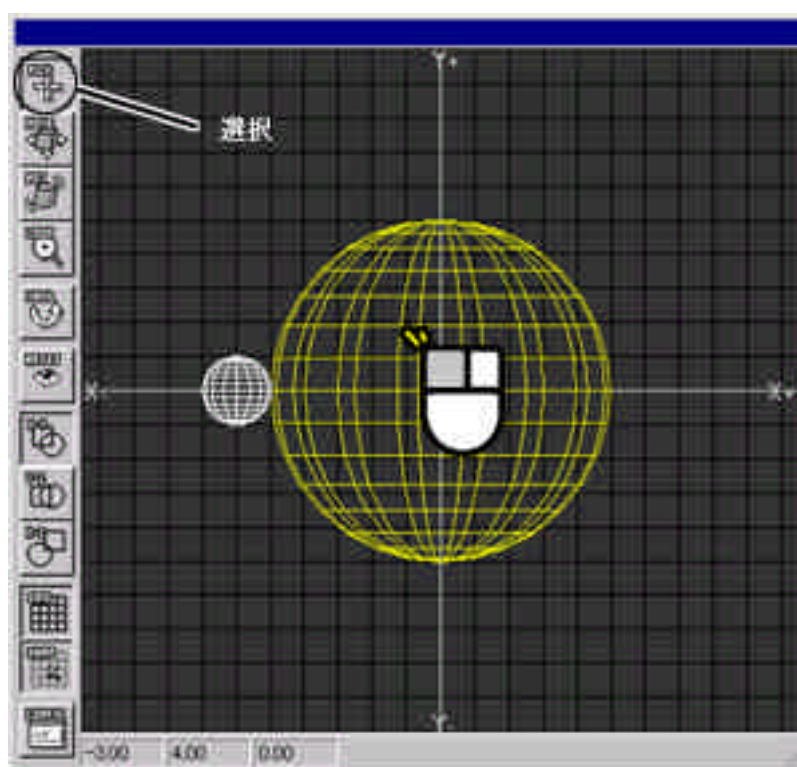


ジオメトリオブジェクト編集

仮想空間に追加されているジオメトリオブジェクトは、正射影ウインドウでその位置や方角などの編集ができます。ここでは、ジオメトリオブジェクトの選択と移動、テクスチャ/マテリアルの設定する方法を説明します。ジオメトリオブジェクトとは、VRML の Shape ノードになります。

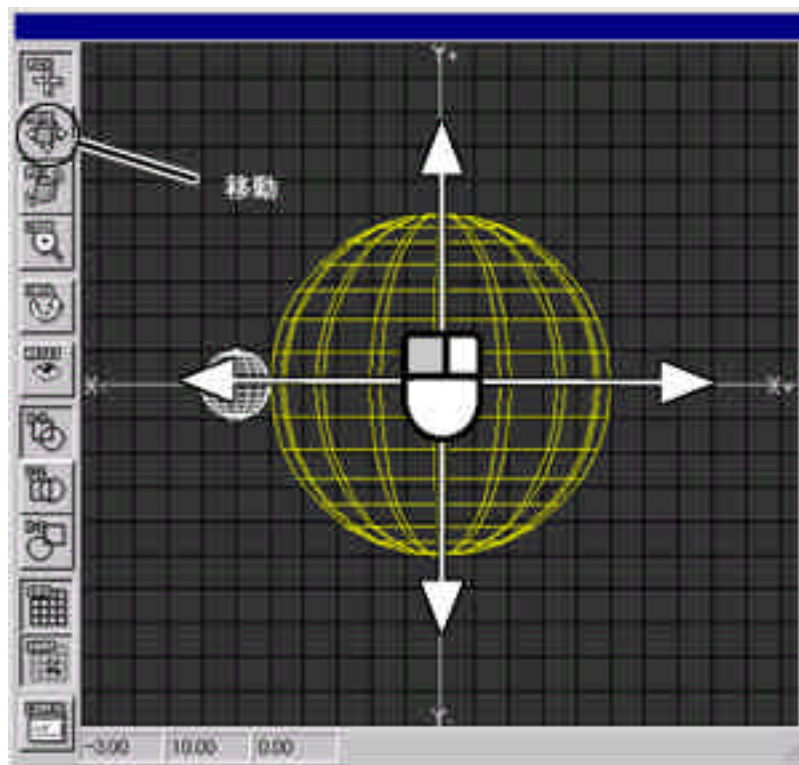
ジオメトリオブジェクト選択

ジオメトリオブジェクトを選択するには、最初に正射影ウインドウの「選択」ボタンを押して下さい。それから、目的のオブジェクトをマウス左ボタンでクリックすると、オブジェクトが選択できます。選択されたオブジェクトは黄色い線で描画されます。



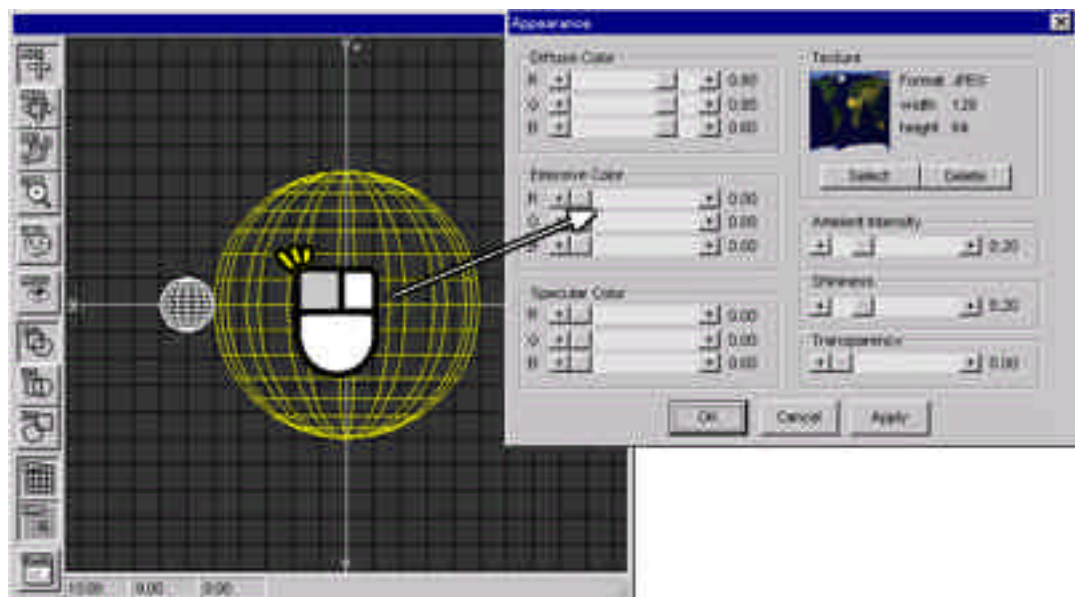
ジオメトリオブジェクト移動

選択されているオブジェクトは、正射影ウインドウで位置を移動することができます。オブジェクトを移動するには、まず正射影ウインドウツールバーの移動ボタンを押して下さい。それから、マウス左ボタンを押しながらマウスカーソルを移動すると、現在正射影ウインドウが表示している平面上をオブジェクトがスライドして移動します。



マテリアル/テクスチャ設定

オブジェクトの色やテクスチャを変更するには、最初に正射影ウィンドウツールバーの選択ボタンを押して下さい。それから、目的のオブジェクトをマウス左ボタンでダブルクリックすると、色やテクスチャを設定するダイアログが表示されます。



仮想空間での移動

透視投影ウインドウでは、マウスを利用してユーザーが仮想空間を自由に移動できます。仮想空間を移動することにより、仮想空間にあるオブジェクトを色々な視点から確認したり、ユーザーの移動にともなうイベントが発生したりします。

仮想空間での移動はマウスで行います。その移動方法には歩行移動、平行移動、回転の3種類があります。移動方法は透視投影ウインドウのツールバーで選択します。一般的には仮想空間をウォークスルーする感じで移動できる歩行移動が用いられます。

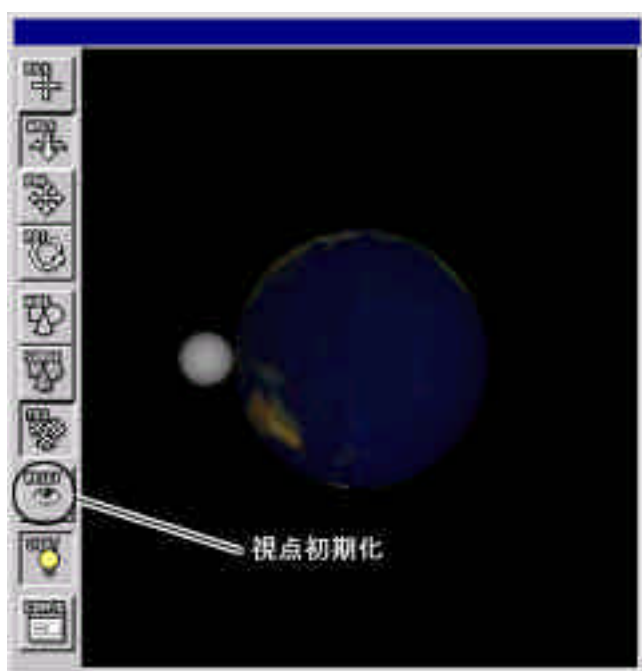


歩行移動

歩行モードでの移動は、マウスの左ボタンを押しながら、マウスポインタを上下方向に移動すると、視点の前進後進、左右方向に移動するとその場で視点が左右に回転します。仮想空間を歩く感覚で移動するのに、最もよく利用される移動モードです。

視点位置の初期化

仮想空間での視点位置を初期化するには、ツールバーの「視点初期化」ボタンを押して下さい。すべての仮想空間オブジェクトが見える位置に、視点が移動します。仮想空間を移動しているうちに、現在自分がどこにいるのかわからなくなった場合などに利用して下さい。



シーングラフの編集

ワールドウィンドウには、現在の仮想空間にあるオブジェクトがシーングラフのノードとして表示されています。これらのノードは、ワールドウィンドウで追加/削除/編集などの操作ができます。

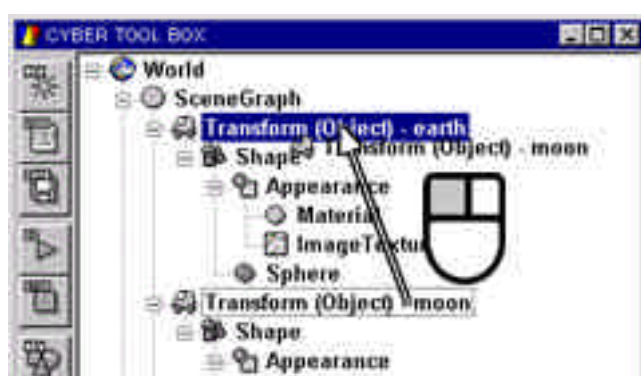
ノードの追加

シーングラフにノードを追加するには、マウスの左ボタンクリックでその親となるノードを選択してから、ツールバーの「新規ノード追加」ボタンを押して下さい。表示されるダイアログには、その親ノードに追加できる子ノードのリスト一覧が表示されます。目的のノードを選択すると、シーングラフにそのノードが追加されます。追加されたノードが Sphere や Box などの可視化できるノードである場合には、透視投影ウィンドウおよび正射影ウィンドウで、その結果を確認できます。



ノードの移動

シーングラフのノードは、マウス操作で現在追加されている親ノードから、他ノードの親ノードへ移動できます。移動するには、目的のノードをマウス左ボタンでドラッグして、移動先の親ノードの上にドロップして下さい。



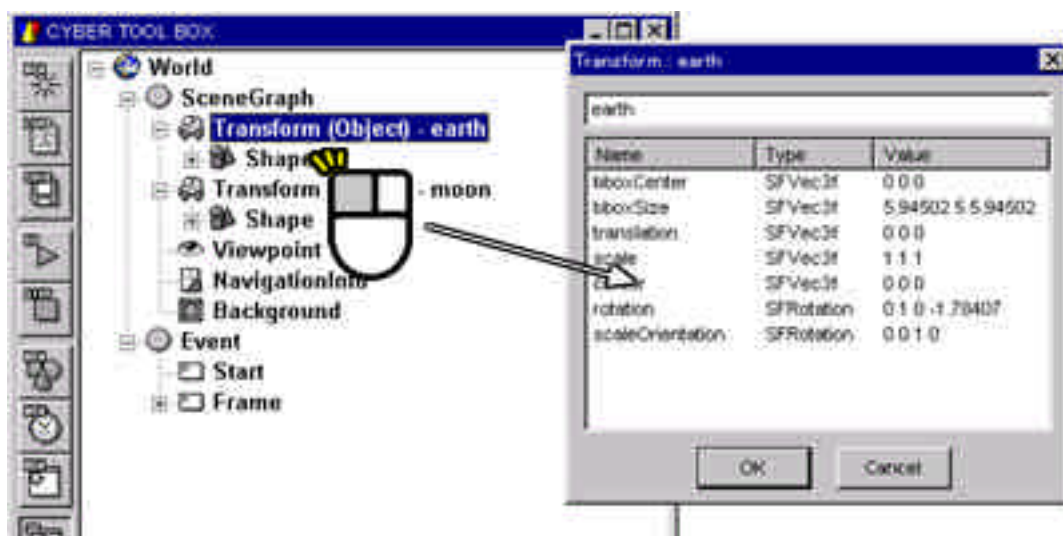
ノードの削除

シーングラフのノードを削除するには、マウス左ボタンで目的のノードをクリックして選択し、キーボードの「削除 (Delete)」キーを押して下さい。削除するノードが子ノードをもっている場合には、その子ノードも一緒に削除されます。



ノードの編集

シーングラフにあるノードをマウスの左ボタンでダブルクリックすると、ノードフィールドの編集用ダイアログが表示されます。このダイアログでノード名の変更や、各フィールドの値の編集を行います。変更したいフィールドをマウスの左ボタンでダブルクリックすると、そのフィールドを編集するダイアログが表示されます。

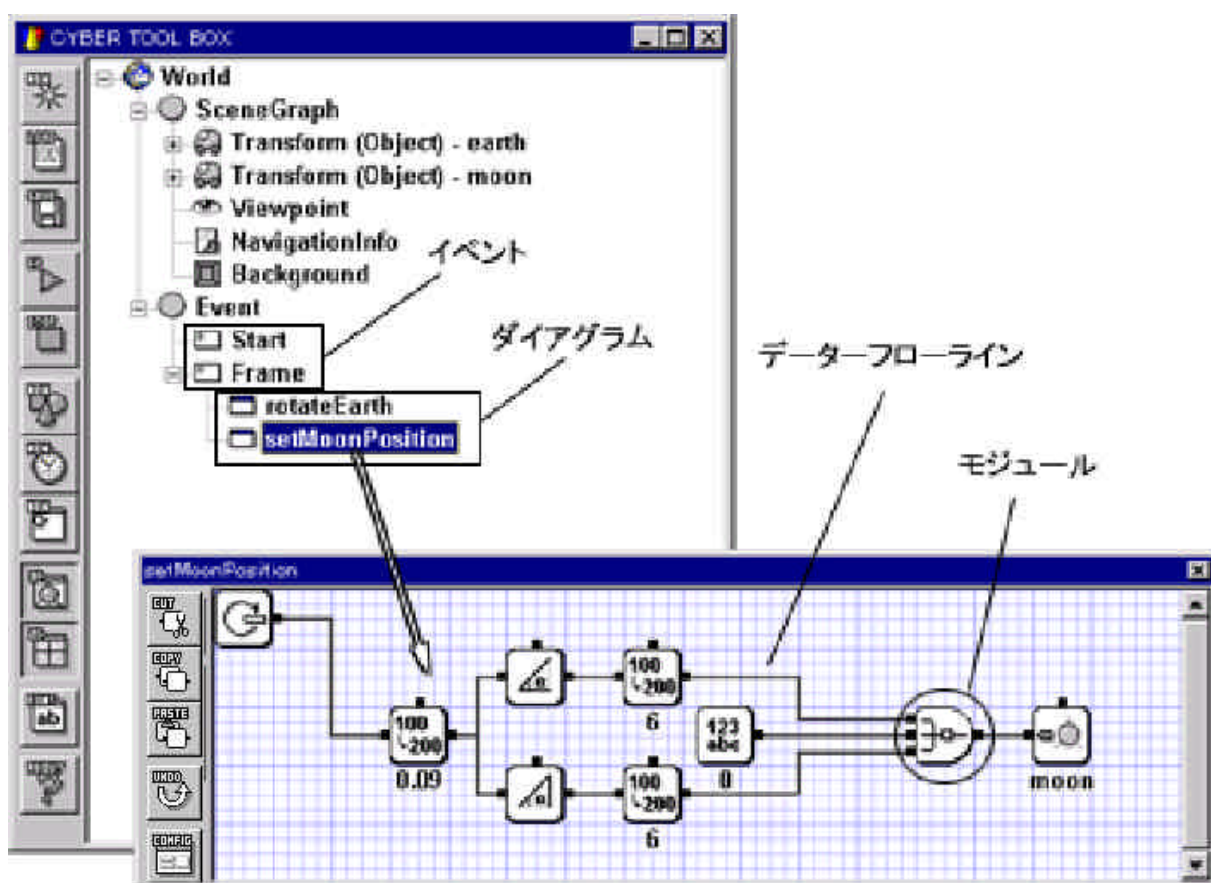


イベントダイアグラム編集

CTB では、アイコンベースのビジュアルプログラミング言語を利用して仮想空間動作を構築していきます。マウス操作でアイコンを線でつないでいく積み木感覚のプログラミング言語ですので、複雑な仮想空間動作であっても誰にでも簡単に構築できます。

CTB の仮想空間動作は、仮想空間で発生したイベントをトリガにして実行されます。この発生するイベントは、VRML2.0/97 のイベントにもある、一定の時間間隔で発生するものや、マウスのクリックによって発生するものなどがあります。

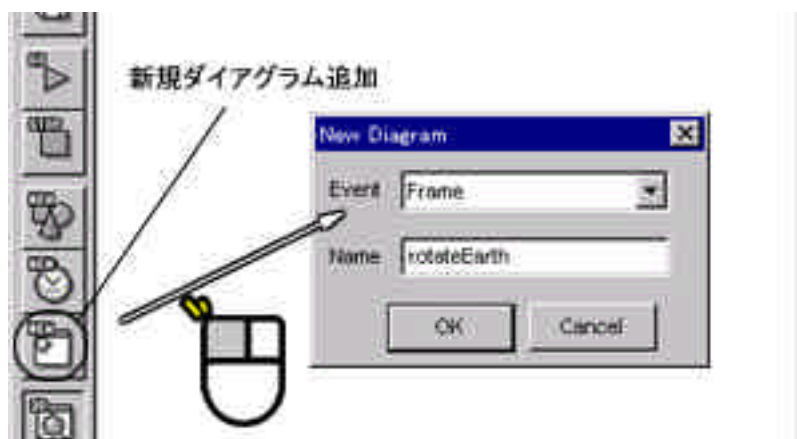
CTB では、あるイベントが発生すると、イベントに関連したダイアグラムが実行されます。このダイアグラムは、アイコンをデータフローラインでつなぐことにより、その動作が定義されており、このアイコンを CTB ではモジュールと呼んでいます。



ここでは、動作定義の基本となるダイアグラムの生成、モジュールの追加、モジュール間のデータフローラインのつなぎ方について説明します。

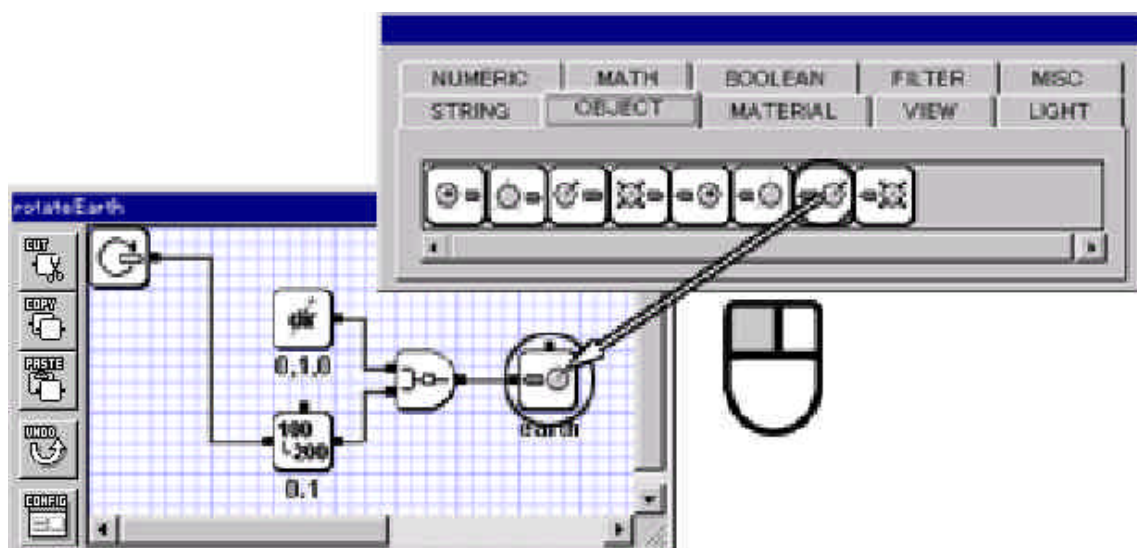
ダイアグラムの追加

ダイアグラムを追加するには、ワールドウインドウの「新規ダイアグラム追加」ボタンを押して下さい。ダイアログでは、ダイアグラムの実行トリガとなるイベントの選択と、ダイアログ名を入力して下さい。ダイアグラムを生成すると、その編集ウインドウが表示されます。



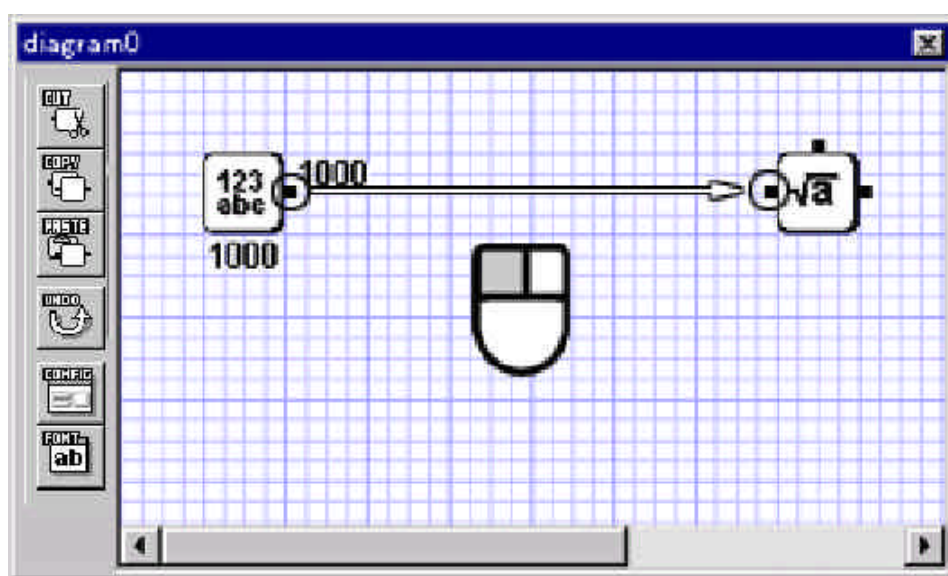
モジュールの追加

ダイアログで動作を定義するには、最初に動作定義に利用するモジュールを追加する必要があります。モジュールを追加するには、モジュールウインドウから目的のモジュールをマウスの左ボタンクリックでドラッグしながら、モジュールを追加したいダイアログウインドウ上でドロップして下さい。



モジュール同士の接続

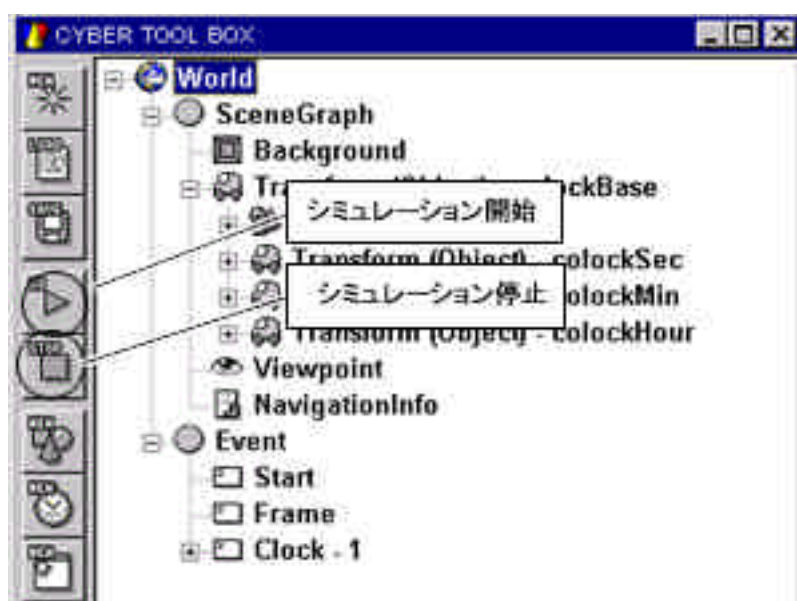
ダイアグラムの動作定義は、モジュールとモジュールを左右にある黒いデータ入出力ノードをデータフローラインで結ぶことにより行います。このラインは、モジュールのノードデータの流れを示しています。ノードデータの流れを定義するには、データを出力するモジュールの右側にある黒い出力ノードを、マウスの左ボタンでドラッグしながら、入力するモジュールの左側にある入力ノードの上でドロップして下さい。



またデータフローラインの接続は、上記の手順とは反対にデータを入力するモジュールから開始することもできます。

シミュレーションの実行/停止

ダイアグラムで定義された動作は、シミュレーションを開始することにより実行されます。シミュレーションを実行するにはワールドウィンドウの「シミュレーション開始」ボタン、シミュレーションを停止させるには「シミュレーション停止」ボタンを押してください。



シミュレーションの実行結果は、透視投影ウィンドウ/正射影ウィンドウ/ダイアグラムウィンドウで確認できます。シミュレーションの実行中は、シーングラフやダイアグラムの編集はできません。編集を行う場合には、シミュレーションを停止させる必要がありますので注意してください。

4.チュートリアル

掛け算

このチュートリアルでは、CTB によるイベントおよびモジュールの動作を学ぶ目的で、ダイアグラムに簡単な掛け算のデータフローを定義して動作を確認してみます。

掛け算動作の定義

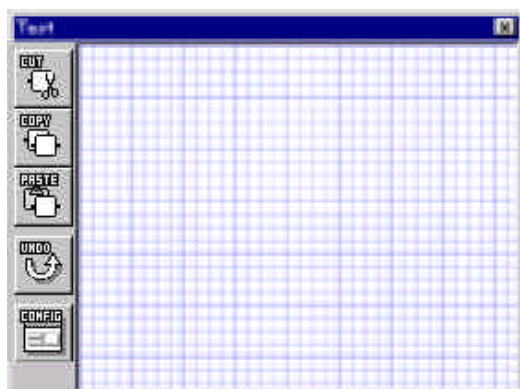
掛け算の動作は、「Start」イベントをトリガとするダイアグラム内で定義します。

ダイアグラムの追加

この動作を定義するには、「Start」イベントをトリガとするダイアグラムを生成する必要があります。このダイアグラムを追加するには、ワールドウィンドウの「新規ダイアグラム追加」ボタンを押すと表示されるダイアログで、イベント(Event)に「Start」を選択し、名前(Name)に「Test」と入力して下さい。






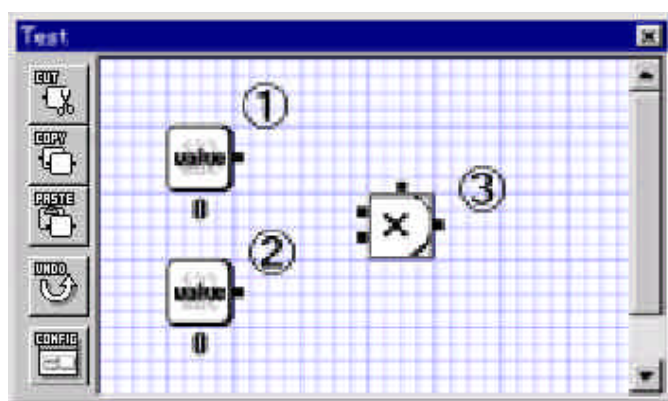
ダイアグラムを追加すると、以下のような空のダイアグラムウィンドウが表示されます。イベントタイプが「Start」のダイアグラム、シミュレーション開始時に 1 回実行されます。



モジュールの追加



生成したダイアログウインドウに動作定義に必要なとなるモジュールを追加します。以下の表に示すモジュールを、モジュールウインドウからマウス左ボタンでドラッグし、ダイアログウインドウへドロップして追加して下さい。

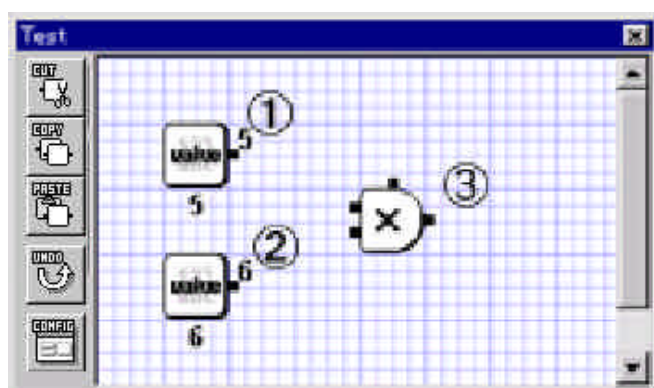
アイコン	クラス名	モジュール名	処理内容
	String	Value	設定されたベクトルを文字列として出力します。
	String	Value	入力された数値文字列のスケール処理を行い出力します。
	String	Multi	入力された2つの文字列を一つに結合します。



モジュール値の設定

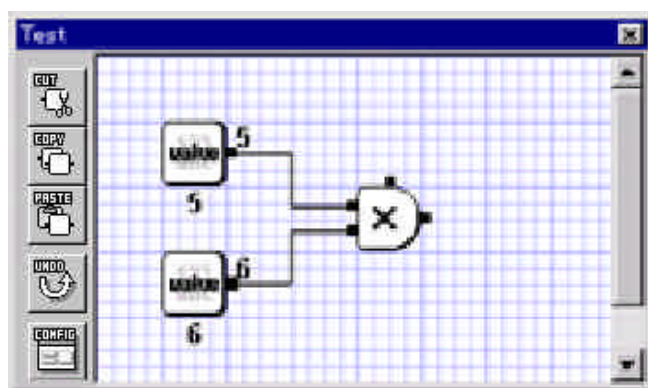
以下の表に示す追加したモジュールの値を設定します。各モジュールをマウス左ボタンでダブルクリックすると、値を設定するためのダイアログが表示されます。以下のモジュールの値を入力または選択し設定して下さい。

アイコン	設定内容
	5
	6



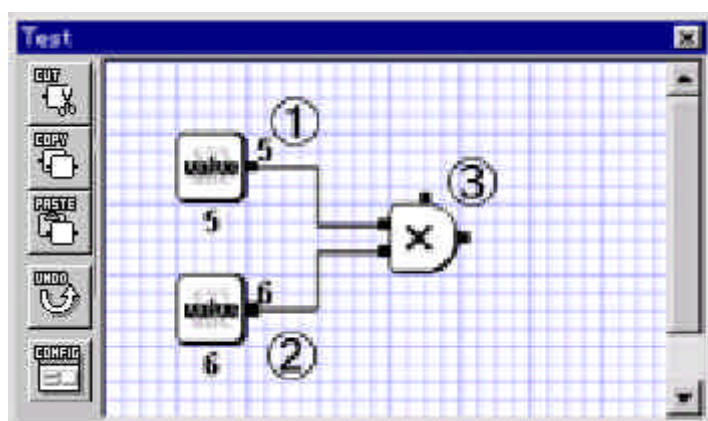
データフローラインの定義

モジュール間のデータの流れを定義するデータフローラインの接続を行います。各モジュールの出力ノードを、マウス左ボタンクリックにてドラッグし、目的の入力モジュールノード上でドロップして接続して下さい。



ダイアグラム動作の説明

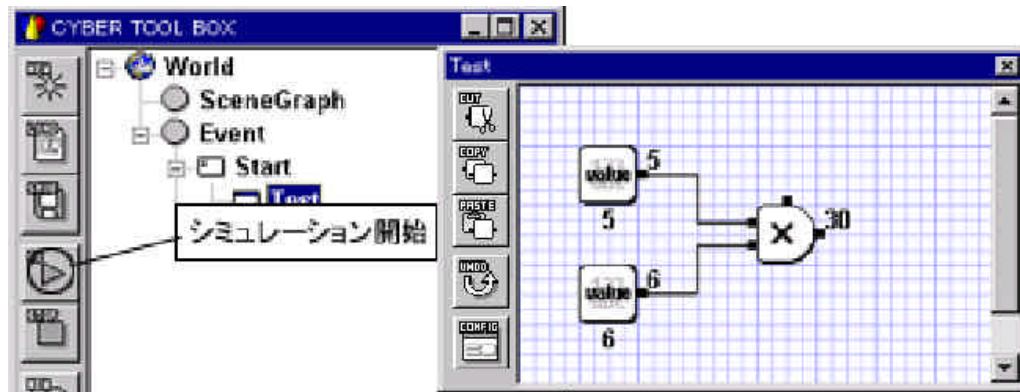
以上で、掛け算を行うために必要な動作定義は完了しました。このダイアグラムで定義されている動作内容を以下の図で説明すると、のモジュールから“5”、のモジュールからは6が出力されています。



のモジュールは、この2つの出力データを掛け合わせた結果を出力します。

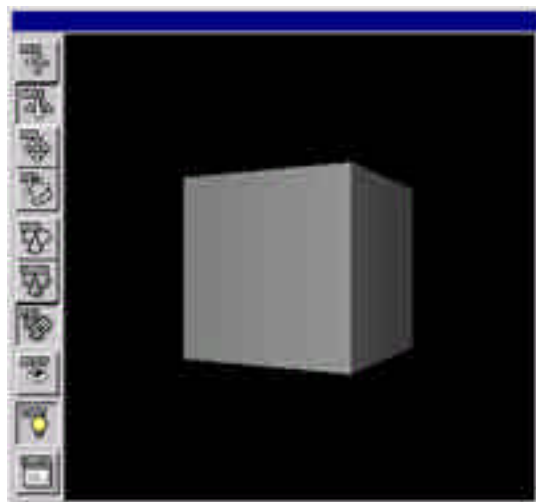
シミュレーションの実行

ワールドウィンドウの「シミュレーション開始」ボタンを押して、立方体オブジェクトが正常に回転するか確認して下さい。結果としてモジュールから「30」が出力されます。



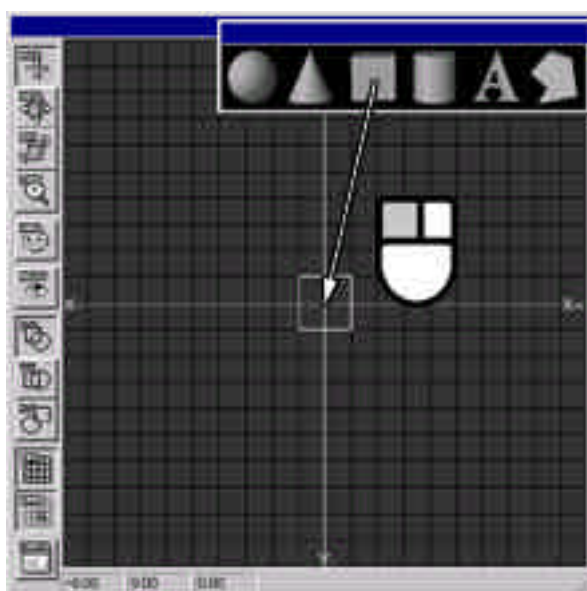
回転する立方体

このチュートリアルでは、仮想空間の基本的な操作を学ぶ目的で、仮想空間にある立方体オブジェクトを回転させるコンテンツを作成します。

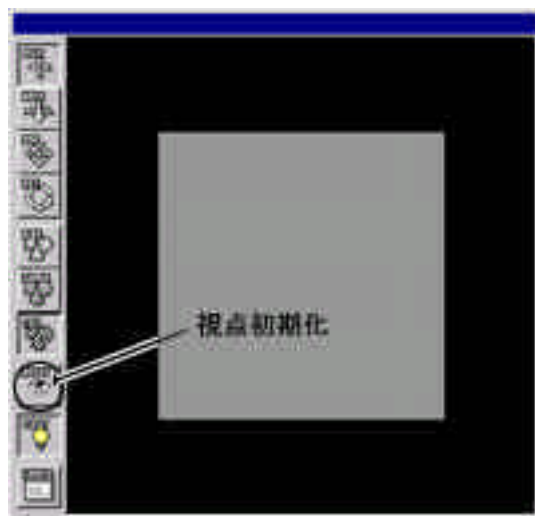


立方体オブジェクトの追加

最初に回転させる立方体オブジェクトを仮想空間に追加します。立方体オブジェクトを追加するには、プリティブウインドウから箱プリティブをマウス左ボタンでドラッグし、正射影ウインドウの追加する位置にドロップして下さい。追加された立方体オブジェクトは、自動的に”box0”と名づけられます。



追加した立方体オブジェクトは、透視投影ウインドウで「視点初期化」ボタンを押して、視点を立方体オブジェクトが見える位置まで移動することにより確認できます。



回転動作の定義

立方体オブジェクトの動作は、ワールド座標系 Y 軸に対してフレーム毎に回転させることにより表現します。

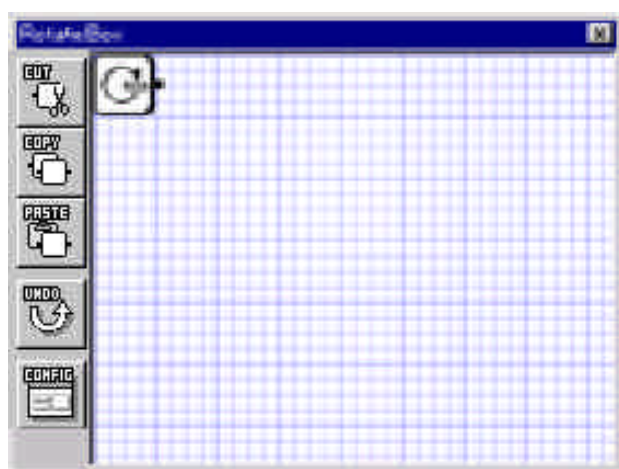
ダイアグラムの追加

この動作を定義するには、「Frame」イベントをトリガとするダイアグラムを生成する必要があります。このダイアグラムを追加するには、ワールドウィンドウの「新規ダイアグラム追加」ボタンを押すと表示されるダイアログで、イベント(Event)に「Frame」を選択し、名前(Name)に「RotateBox」と入力して下さい。







ダイアグラムを追加すると、以下のようなダイアグラムウィンドウが表示されます。イベントタイプが「Frame」のものは、ウィンドウ左上に経過フレーム数を表示するシステムモジュールが自動的に追加されています。このモジュールは、シミュレーション開始からの経過フレーム回数を出力するものです。

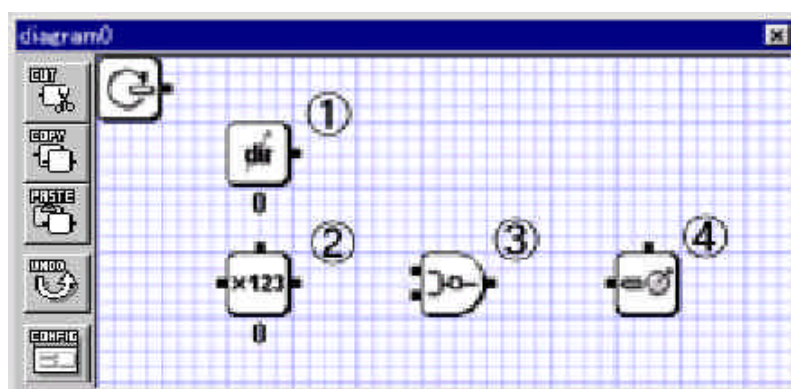
「Frame」イベントは最高で1秒間に10回発生します。



モジュールの追加




生成したダイアログウインドウに動作定義に必要なモジュールを追加します。以下の表に示すモジュールを、モジュールウインドウからマウス左ボタンでドラッグし、ダイアログウインドウへドロップして追加して下さい。

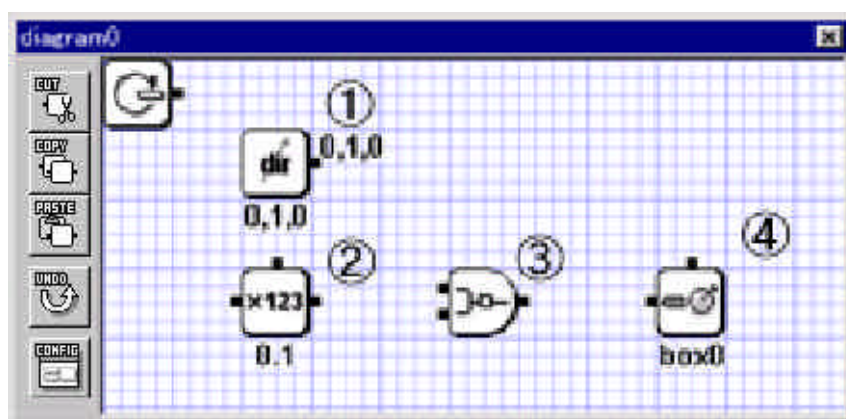
アイコン	クラス名	モジュール名	処理内容
	String	Direction	設定されたベクトルを文字列として出力します。
	Filter	Scale	入力された数値文字列のスケール処理を行い出力します。
	String	Mearge2Values	入力された2つの文字列を一つに結合します。
	Object	SetRotation	オブジェクト方向を設定します。



モジュール値の設定

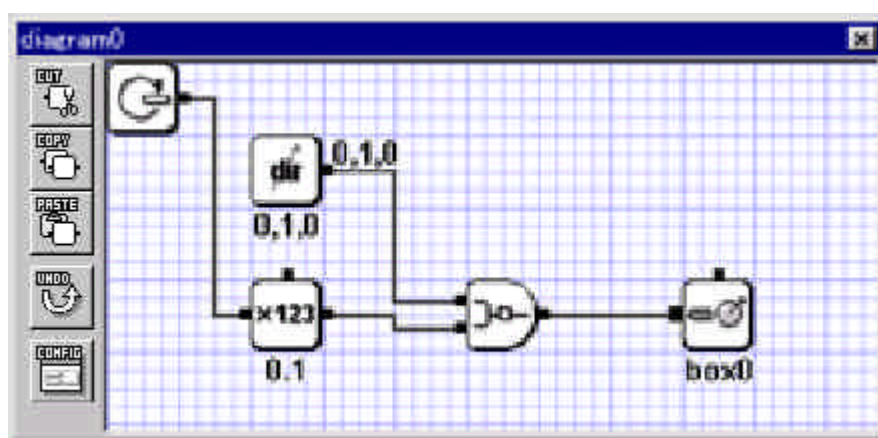
以下の表に示す追加したモジュールの値を設定します。各モジュールをマウス左ボタンでダブルクリックすると、値を設定するためのダイアログが表示されます。以下のモジュールの値を入力または選択し設定して下さい。

アイコン	設定内容
	X=0, Y=1, Z=0 (0,1,0)
	0.1
	box0



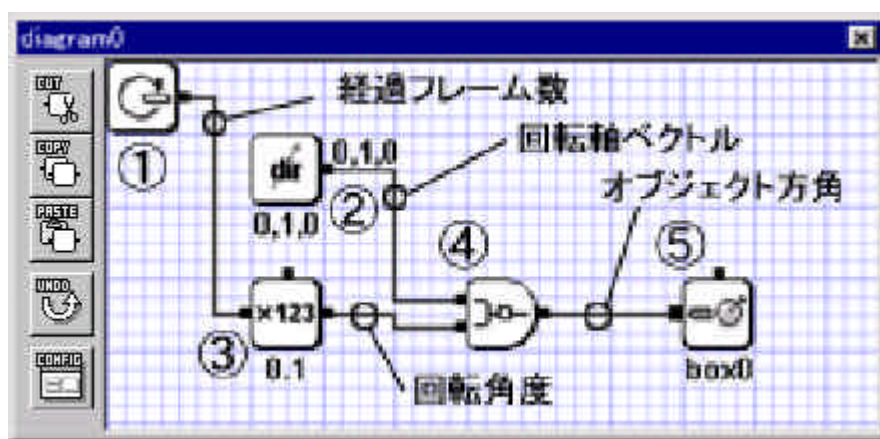
データフローラインの定義

モジュール間のデータの流れを定義するデータフローラインの接続を行います。各モジュールの出力ノードを、マウス左ボタンクリックにてドラッグし、目的の入力モジュールノード上でドロップして接続して下さい。



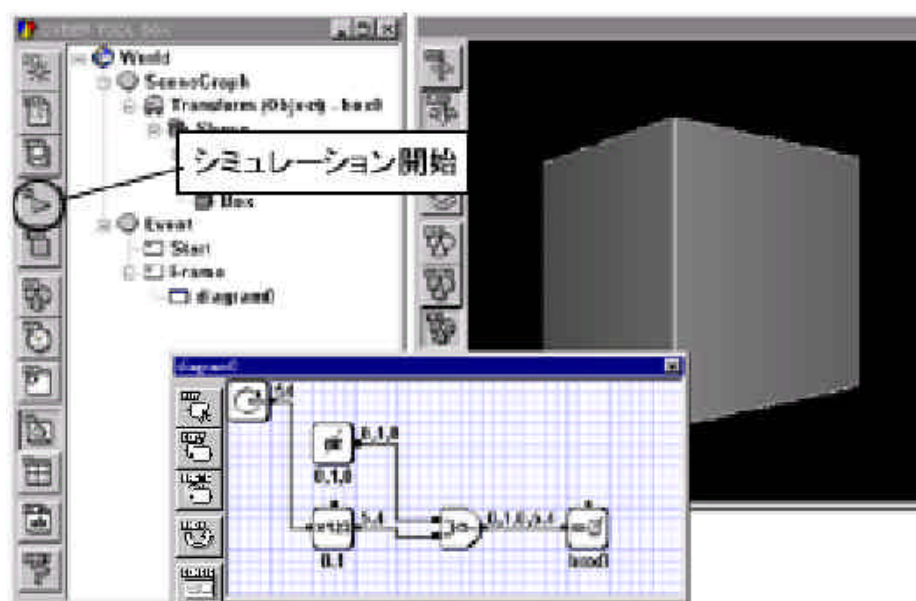
ダイアグラム動作の説明

以上で、立方体オブジェクトを回転させるために必要な動作定義は完了しました。このダイアグラムで定義されている動作内容を以下の図で説明すると、①のモジュールからは経過フレーム数データが出力されています。このデータに、②のモジュールで0.1を掛け合わせることで立方体オブジェクトの回転角度データを生成します。このデータを、④のモジュールから出力される回転軸ベクトルデータと⑤のモジュールで1つの文字列として連結することにより、オブジェクト方角データを生成します。このデータを、③のモジュールへ入力することにより、立方体オブジェクトの方角を変更します。



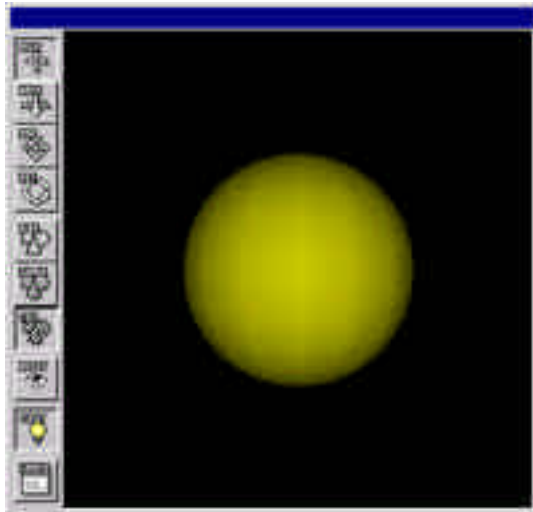
シミュレーションの実行

ワールドウィンドウの「シミュレーション開始」ボタンを押して、立方体オブジェクトが正常に回転するか確認して下さい。もし正射影画面に立方体オブジェクトが表示されない場合には、正射影ウィンドウの視点初期化ボタンを押して視点位置を適切な位置に移動させてみて下さい。



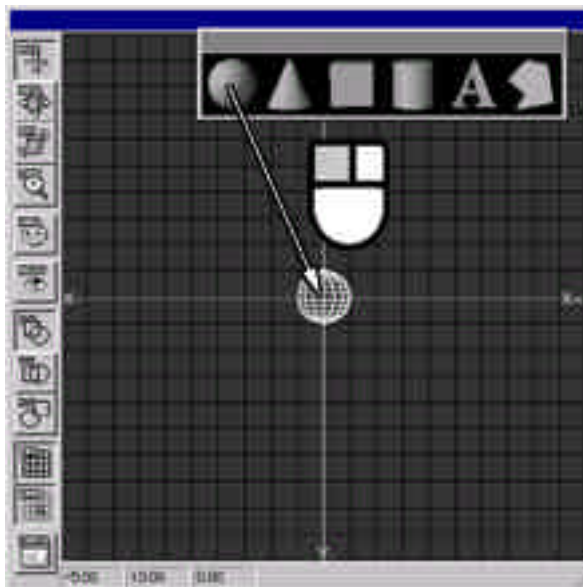
光る球体

このチュートリアルでは、ユーザーイベントの作成と利用方法を学ぶ目的で、仮想空間にある球体オブジェクトをマウスクリックで点滅させるコンテンツを作成します。



球体オブジェクトの追加

最初に点滅させる球体オブジェクトを仮想空間に追加します。球体オブジェクトを追加するには、プリミティブウインドウから球体プリミティブをマウス左ボタンでドラッグし、正射影ウインドウの追加する位置にドロップして下さい。追加された球体オブジェクトは、自動的に”sphere0”と名づけられます。



追加した球体オブジェクトは、透視投影ウインドウで「視点初期化」ボタンを押して、視点を立方体オブ

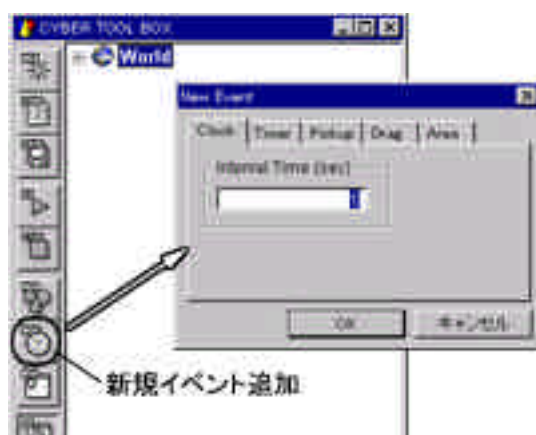
ジェクトが見える位置まで移動することにより確認できます。

点滅動作の定義

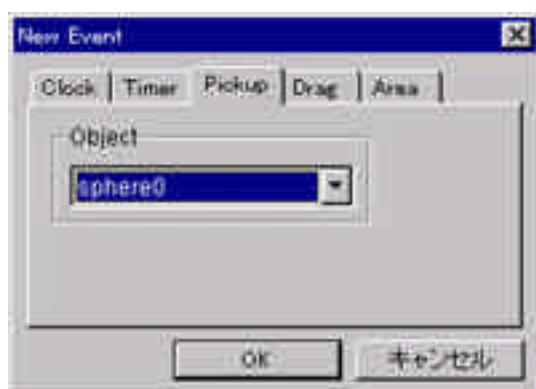
球体オブジェクトの動作は、マウス左ボタンでクリックすること黄色、クリックしたボタンを離すと白色に設定することにより表現します。

クリックイベントの追加

この動作を定義するには、最初に球体オブジェクトをクリックした時に発生するユーザーイベントを生成する必要があります。このユーザーイベントを生成するには、ワールドウインドウの「新規イベント追加」ボタンを押して、ユーザーイベントを追加するダイアログを表示させて下さい。

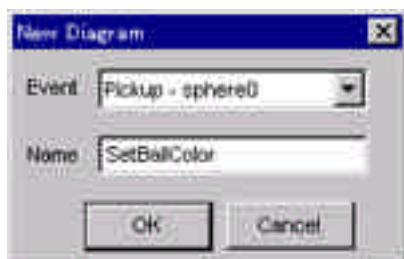


この新規イベント追加ダイアログで、球体オブジェクトをクリックすると発生させるイベントとして「Pickup」タブを選択し、オブジェクトに「sphere0」を選択して追加して下さい。この追加したイベントは、ワールドウインドウのイベントツリーに表示されています。



ダイアグラムの追加

追加したクリックイベントをトリガとするダイアグラムを追加します。このダイアグラムを追加するには、ワールドウィンドウの「新規ダイアグラム追加」ボタンを押すと表示されるダイアログで、イベント(Event)に「Pickup - sphere0」を選択し、名前(Name)に「SetBallColor」と入力して下さい。

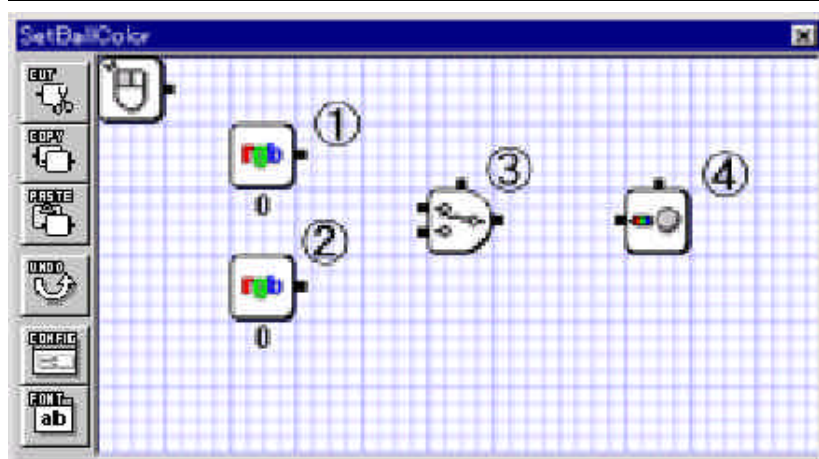


ダイアグラムを追加すると、ダイアグラムウィンドウが表示されます。イベントタイプが「Pickup」のものは、ウィンドウ左上にマウスクリック状態を表示するシステムモジュールが自動的に追加されています。このモジュールは、マウスがオブジェクトをクリックした時に「true」、放した時に「false」を出力します。

モジュールの追加




生成したダイアログウィンドウに動作定義に必要なモジュールを追加します。以下の表に示すモジュールを、モジュールウィンドウからマウス左ボタンでドラッグし、ダイアログウィンドウへドロップして追加して下さい。

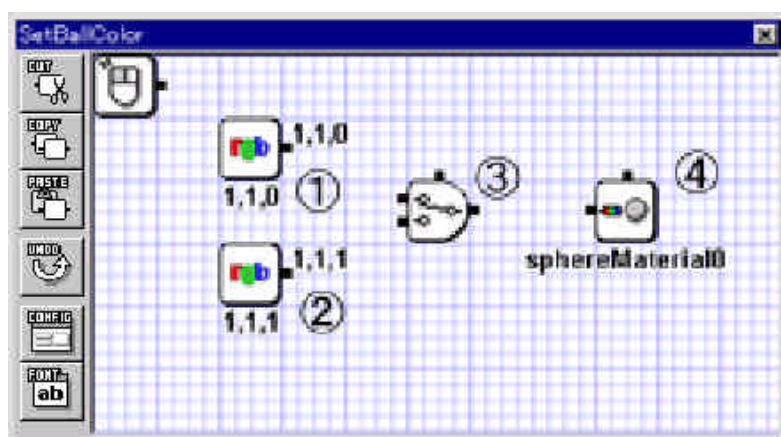
アイコン	クラス名	モジュール名	処理内容
	String	Color	設定された色を文字列として出力します。
	String	Color	設定された色を文字列として出力します。
	String	Selector	制御ノードによる指定により 2 つの入力ノードの値のいずれかを出力します。
	Material	SetDiffuseColor	マテリアルの拡散色を設定します。



モジュール値の設定

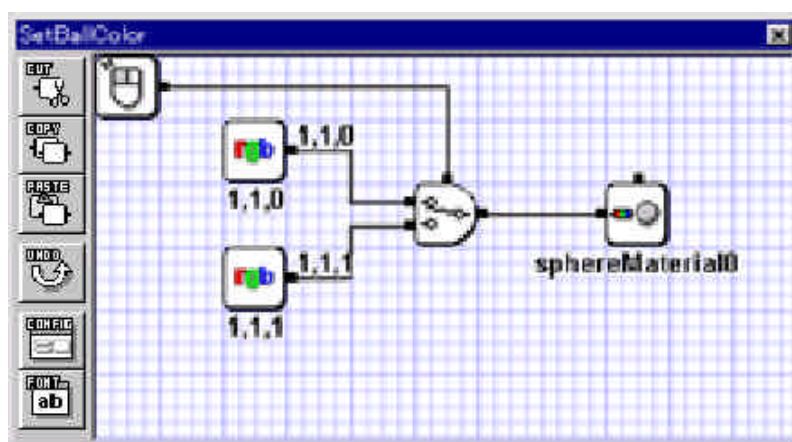
以下の表に示す追加したモジュールの値を設定します。各モジュールをマウス左ボタンでダブルクリックすると、値を設定するためのダイアログが表示されます。以下のモジュールの値を入力または選択し設定して下さい。

アイコン	設定内容
	1,1,0 (黄色)
	1,1,1 (白色)
	boxMaterial0



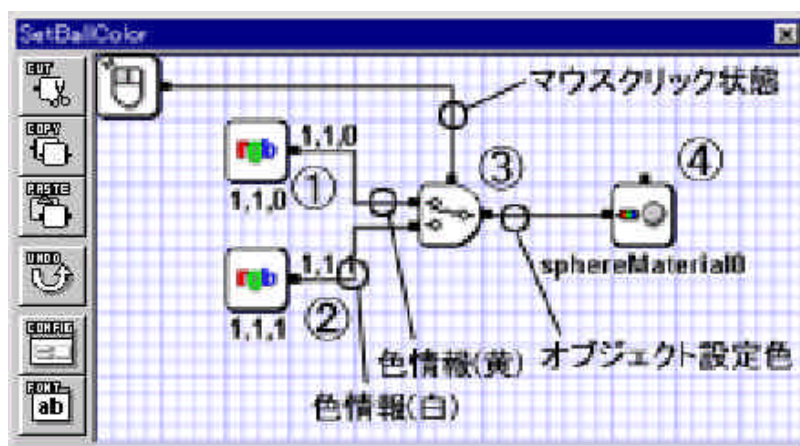
データフローラインの定義

モジュール間のデータの流れを定義するデータフローラインの接続を行います。各モジュールの出力ノードを、マウス左ボタンクリックにてドラッグし、目的の入力モジュールノード上でドロップして接続して下さい。



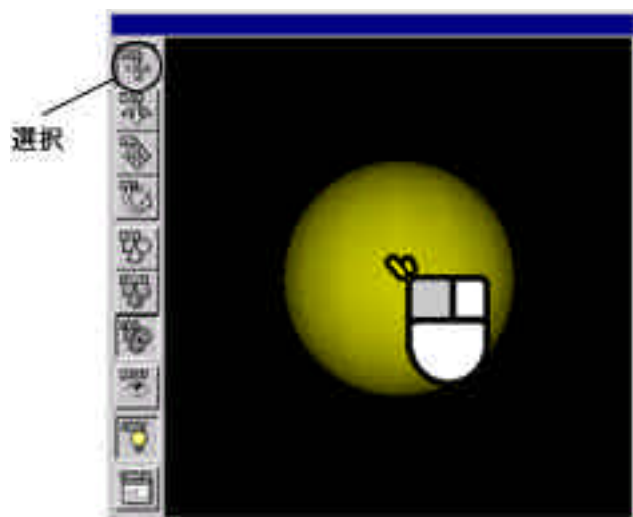
ダイアグラム動作の説明

以上で、球体オブジェクトを点滅させるために必要な動作定義は完了しました。このダイアグラムで定義されている動作内容を以下の図で説明すると、①のモジュールからはマウスのクリック状態を示すデータが出力されています。このデータが“true”、つまりマウスで球体オブジェクトがクリックされた場合、②のモジュールは①のモジュールの色情報データを出力し、“false”、つまりマウスのクリックが開放された時には、③のモジュールは①のモジュールの色情報データを出力します。④のモジュールは、③のモジュールで選択された色情報をマテリアルの拡散色に設定します。



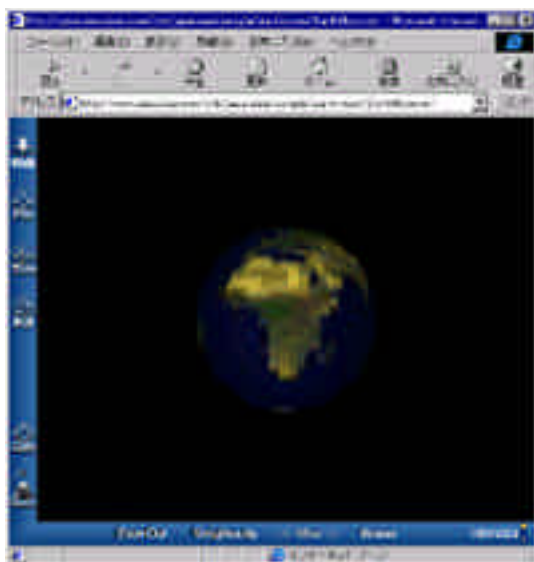
シミュレーションの実行

最初にワールドウインドウの「シミュレーション開始」ボタンを押して下さい。そして正射影ウインドウの「選択」ボタンを押して、マウスで球体オブジェクトがクリックできるようにして下さい。この状態で、球体オブジェクトをマウス左ボタンでクリックすると黄色、放すと白色で表示されます。もし正射影画面に球体オブジェクトが表示されない場合には、正射影ウインドウの視点初期化ボタンを押して視点位置を適切な位置に移動させてみて下さい。



インターネットでのコンテンツ公開

このチュートリアルでは、CTB で作成した VRML コンテンツをインターネットで公開する手順を示します。コンテンツをインターネットの WWW サイトで公開するには、コンテンツ関連ファイルの WWW サイトへのアップロードと、コンテンツを表示するための HTML ファイルの定義が必要になります。



コンテンツ関連ファイルのアップロード

CTB で作成したコンテンツを保存すると、ファイルを保存したディレクトリに、指定したファイルだけではなく、コンテンツの実行に必要な複数の Java クラスファイルやテキストファイルなどが一緒に保存されています。CTB で作成した VRML コンテンツは、一つの VRML ファイルと、複数の Java クラスファイルやテキストファイルによって構成されています。そのため、インターネットで、CTB で作成したコンテンツを公開する場合には、コンテンツが保存されているディレクトリにある全てのファイルを、WWW サイトの同一ディレクトリにアップロードする必要があります。

例えば、チュートリアルの「回転する立方体」を、ファイル名を「RotateBox.wrl」として保存したとすると、保存に設定したディレクトリに以下の 7 つのファイルが保存されています。

- RotateBox.wrl
- FilterScale.class
- ObjectSetRotation.class
- StringConstant.class
- StringMerge2Values.class
- SystemFrame.class
- SystemStart.class

インターネットで、このコンテンツを公開する場合には、上記の全ファイルを WWW サイトの同一ディレクトリにアップロードして下さい。

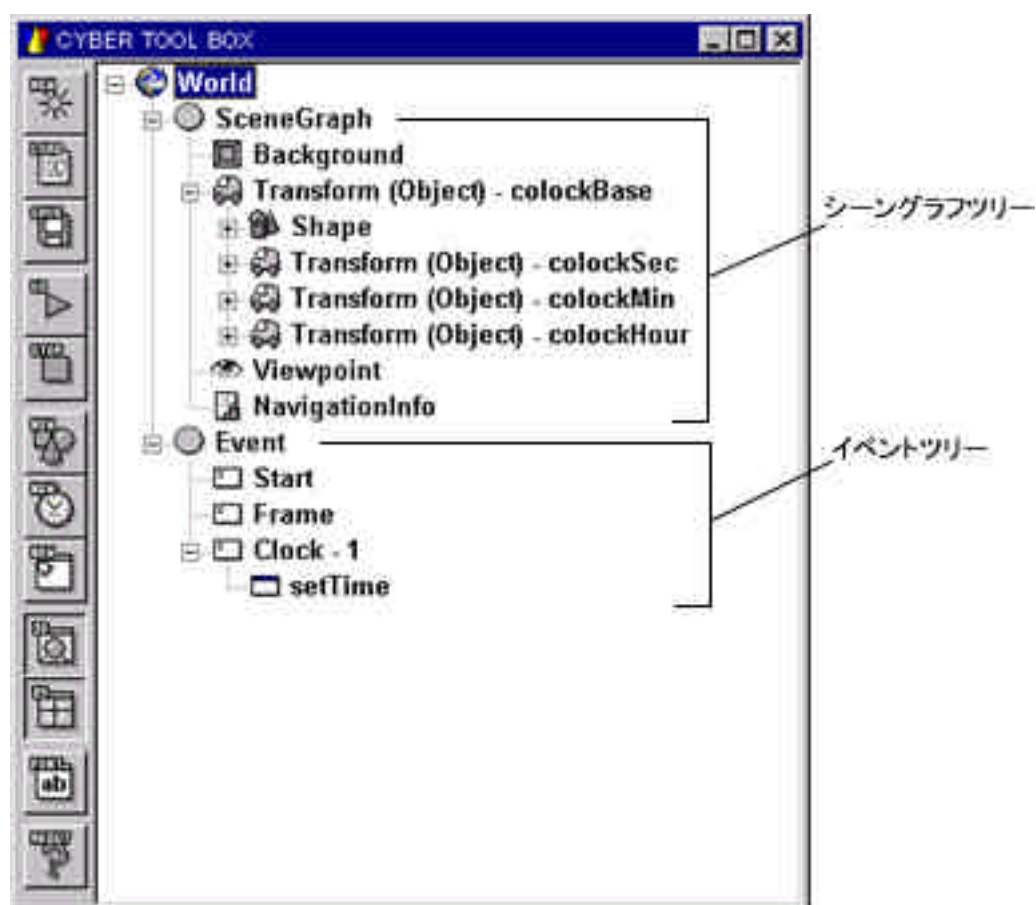
5. 操作説明

ワールドウインドウ

ワールドウインドウは、CTB で最も中心となるウインドウです。ワールドウインドウは、現在の仮想空間情報がツリー構造で表示されています。またウインドウ左側に位置しているツールバーのボタンで、仮想空間への VRML2.0/97 形式ファイルの読み込みおよび保存、シーングラフノード、イベントやダイアグラムなどの追加、シミュレーションの開始/終了などの操作を行います。

ワールドツリー

ワールドウインドウには仮想空間情報を示すワールドツリーが表示されています。ワールドツリーは、現在の仮想空間に追加されているノードオブジェクトを階層的に表示しているシーングラフツリーと、定義されている仮想空間動作を表示しているイベントツリーの2種類から構成されています。



シーングラフツリー

シーングラフツリーは、現在の仮想空間に存在する動作定義以外のオブジェクトを表示しています。シーングラフツリーに表示されているオブジェクトは VRML ノードそのものであり、シーングラフツリーは VRML ノードにより構成される仮想空間をビジュアル的に編集を行うエディタです。

シーングラフツリーを用いて、新規のノードを追加したり、ノードの階層関係を変更したり、ノードの各フィールド値を変更することができるため、通常のエディタで VRML ファイルを編集する方法に比べ直感的かつ簡単に仮想空間の編集が行えます。

ノードの追加

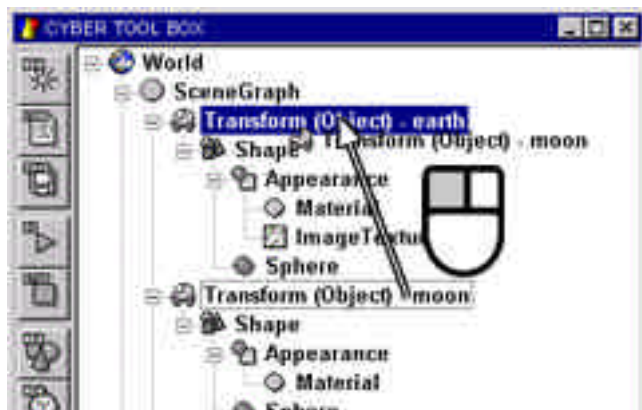
ノードをシーングラフツリーに追加するには、追加するノードの親となるノードを選択してから、ツールバーの「新規ノード追加」ボタンを押して下さい。ノードを、シーングラフの最上位に追加する場合には、シーングラフツリーの「SceneGraph」を選択して下さい。ボタンを押すと、選択されている親ノードに VRML 仕様で追加できるノード一覧のダイアログが表示されますので、追加したいノードを選択して下さい。



追加されたノードが Sphere や Box などの可視化できるノードである場合には、透視投影ウィンドウおよび正射影ウィンドウでも、その結果を確認できます。

ノードの移動

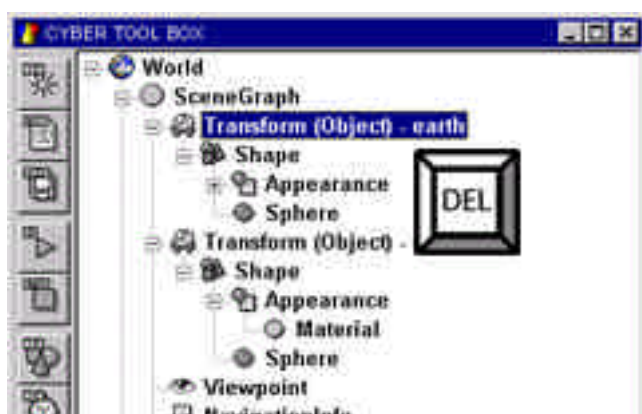
シーングラフツリーに表示されているノードは、現在の親ノードから、他の親ノードに階層を移動することができます。移動するには、目的のノードをマウス左ボタンでドラッグして、移動先の親ノードの上にドロップして下さい。



移動先の親ノードが、ドロップされたノードを VRML 仕様で子ノードとして追加できる場合には、現在の親ノードから移動が行われます。追加できない場合には、移動は行われません。例えば Shape ノードを親ノードとして、Transform ノードをその子ノードとして移動しようとしても無効です。

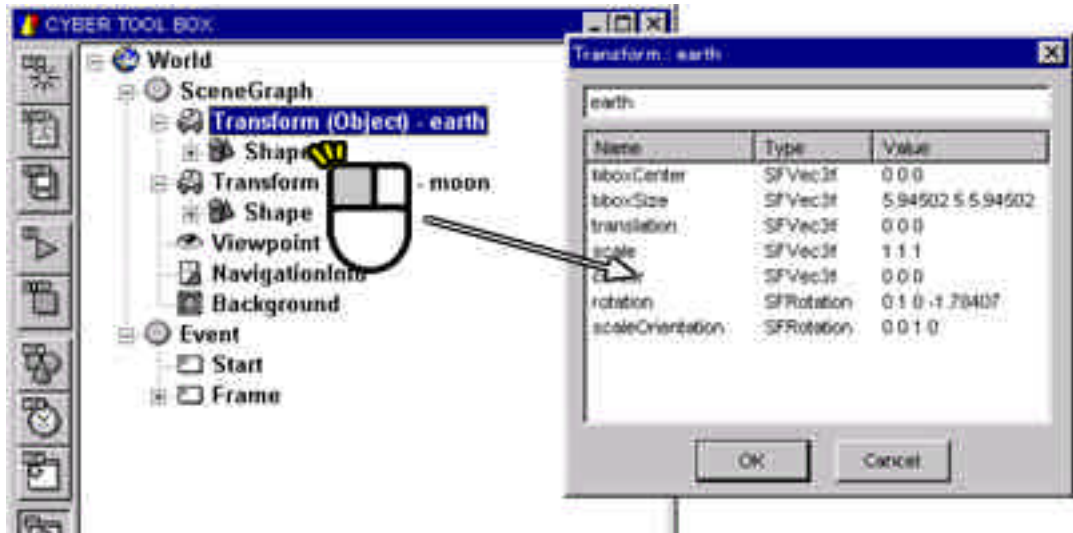
ノードの削除

シーングラフツリーに表示されているノードを削除するには、マウス左ボタンで目的のノードをクリックして選択し、キーボードの削除キーを押して下さい。削除するノードが子ノードをもっている場合には、その子ノードも一緒に削除されます。削除されたノードが Sphere や Box などの可視化できるノードである場合には、透視投影ウィンドウおよび正射影ウィンドウでも、その結果を確認できます。

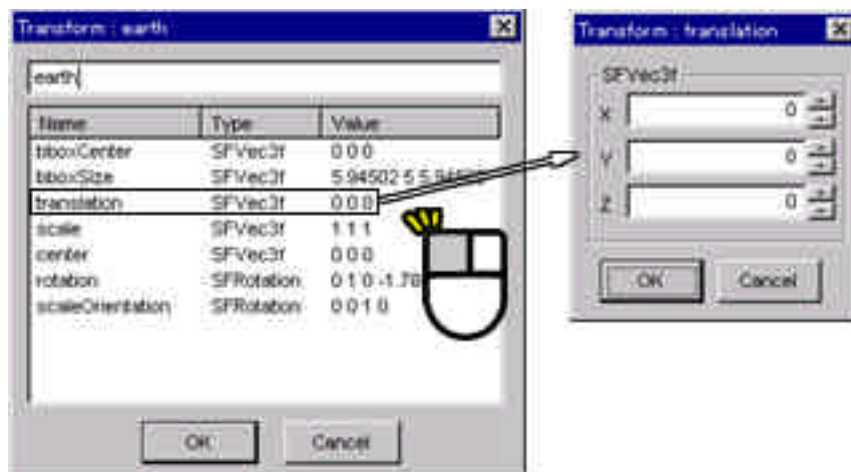


ノードの編集

シーングラフツリーに表示されているノードは、マウス左ダブルクリックにより ノード名とノードフィールド値の確認/修正を行うダイアログが表示されます。この例えば Transform ノードを選択した場合には、以下のようなダイアログが表示されます。

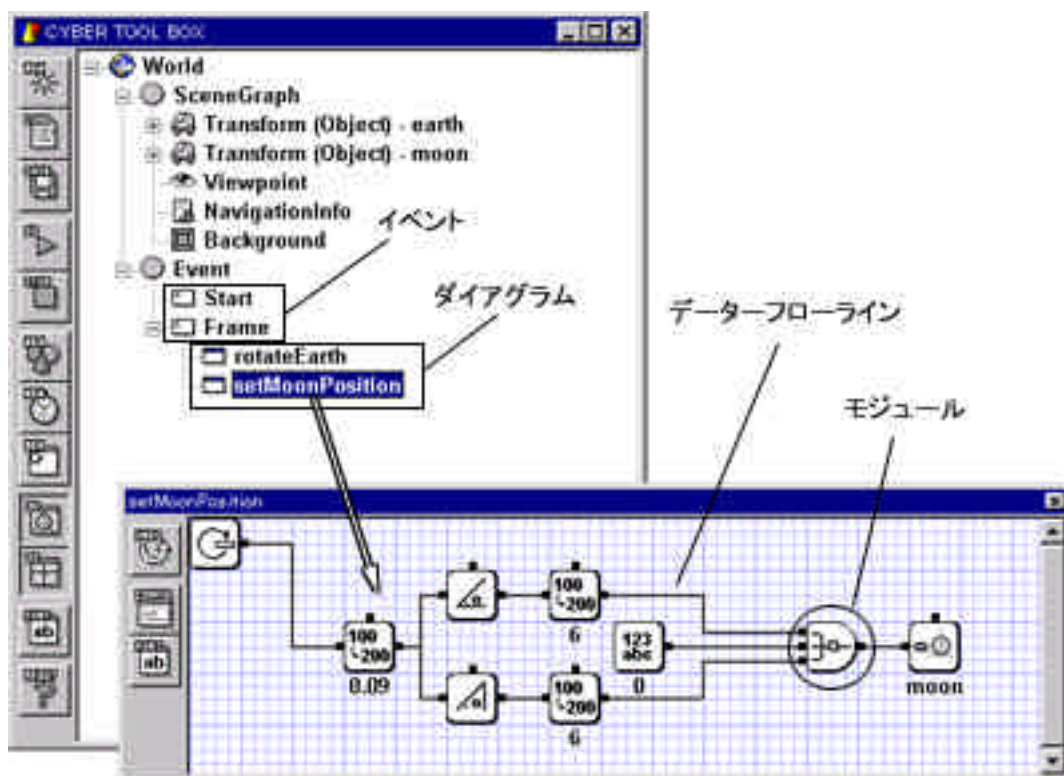


上記ダイアログで、編集したいフィールドをマウス左ボタンでダブルクリックすると、そのフィールド属性に応じたダイアログが表示されます。プロパティの変更は、この表示されるダイアログを利用して下さい。例えば Transform ノードの translation フィールドをダブルクリックした場合には、SFVec3f 型の値を編集するダイアログが表示されます。



イベントツリー

イベントツリーは、現在定義されている仮想空間動作が表示され、仮想空間動作の開始トリガとなる各イベントと、実行されるイベント別に各ダイアグラムが表示されています。ダイアグラムとは、仮想空間動作を定義する一つの単位です。



ダイアグラムは、モジュールと呼ばれるアイコンを、データフローラインと呼ばれる線で結ぶことにより動作定義を行います。ダイアグラムの詳細については、別途ダイアグラムウィンドウや動作定義の説明を参照して下さい。

イベントツリーを用いて、新規のイベントやダイアグラムを追加したり、ダイアグラムを編集するためのダイアグラムウィンドウを開いたりできます。

イベントの追加

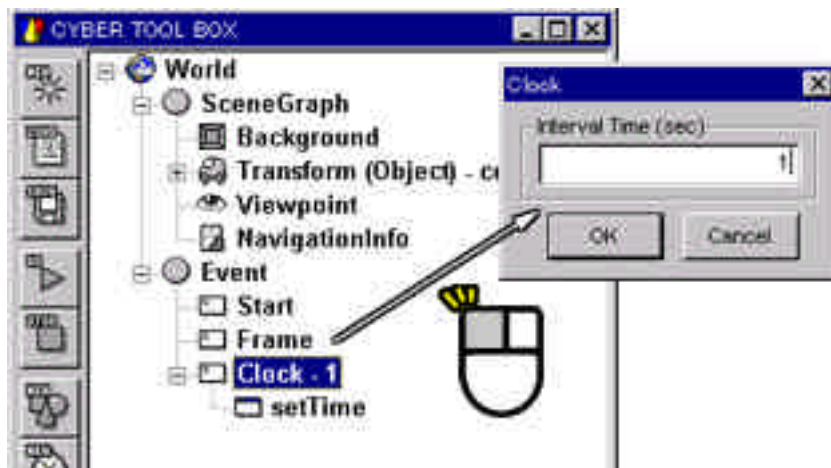
イベントツリーには、標準で2つのシステムイベント「Start」と「Frame」が最初から追加されています。これらのシステムイベントの他に、ユーザー独自のイベントを追加することができます。このユーザー定義のイベントには、指定した時間間隔で発生するイベントや、マウスでオブジェクトをクリックすると発生するイベントなどがあります。ユーザー定義のイベントを追加するには、ツールバーの「新規イベント追加」ボタンを押して下さい。



表示されるダイアログで、追加したいイベントを選択し、適切なイベントパラメータを入力して、新規イベントを追加してください。既に同じパラメータのイベントが仮想空間に存在する場合には、このイベントは追加されません。

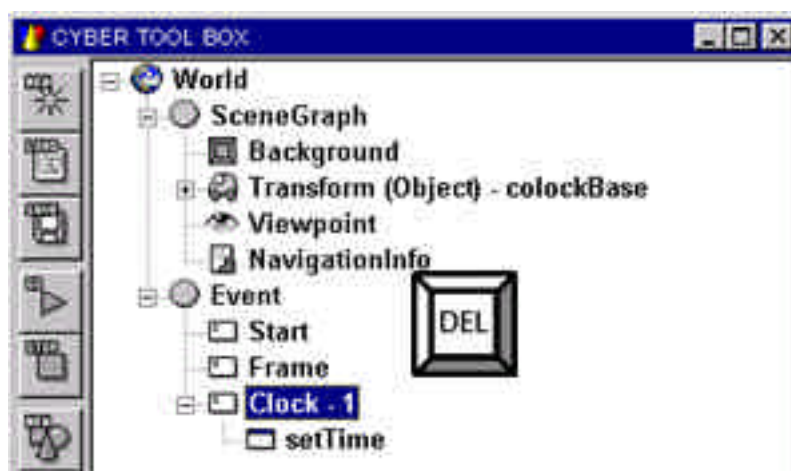
イベントの編集

イベントがユーザーにより定義されたイベントである場合には、イベント項目をマウス左ボタンでダブルクリックすると、パラメータの編集ダイアログが表示されます。



イベントの削除

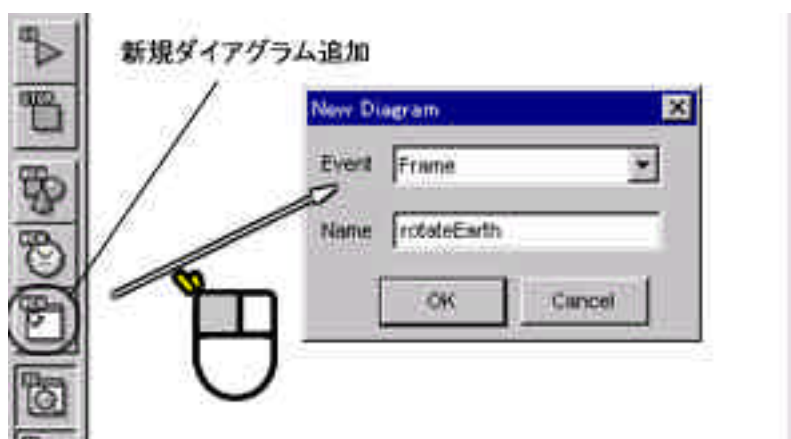
ユーザー定義のイベントを削除するには、マウス左ボタンで削除したいイベントをイベントツリーから選択し、キーボードの削除キーを押して下さい。



イベントを削除した場合には、そのイベントの配下に表示されている、関連した全てのダイアグラムも一緒に削除されます。ただし削除できるのはユーザーにより追加されたイベントのみで、予め追加されているシステムイベントである「Start」と「Frame」は削除できません。

ダイアグラムの追加

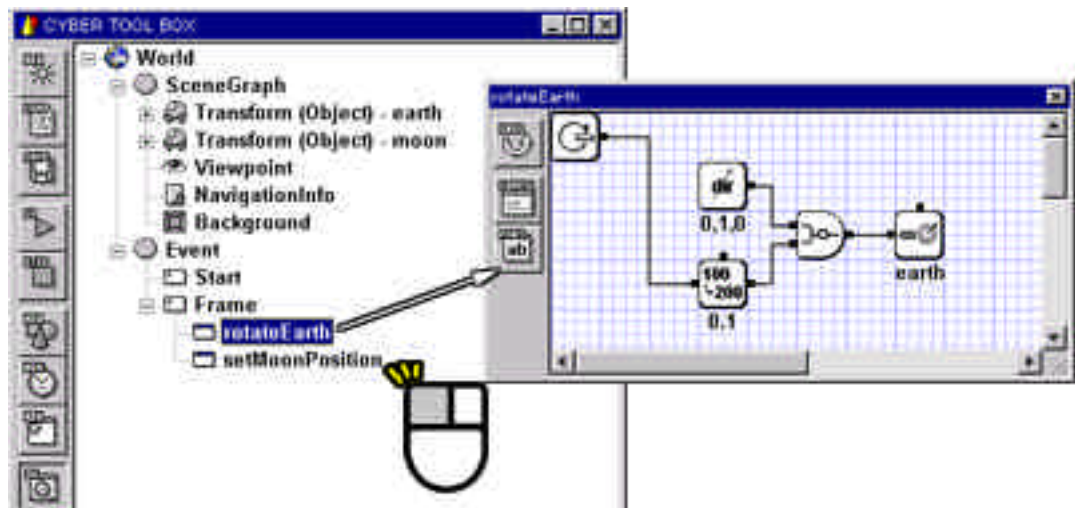
仮想空間動作を定義する新しいダイアグラムを追加するには、ツールバーの「新規ダイアグラム追加」ボタンを押して下さい。



表示されるダイアログで、ダイアグラムが実行されるイベントを選択し、ダイアグラム名を入力して、新しいダイアグラムを追加して下さい。選択できるイベントは、システムイベントである「Start」と「Frame」および、ユーザーで追加したイベントの何れか一つです。既に同イベントおよび名前のダイアグラムが仮想空間に存在する場合は、このダイアグラムは追加されません。

ダイアグラムの編集

ダイアグラムの編集を行うには、イベントツリーに追加されているダイアグラムをマウス左ボタンでダブルクリックして、ダイアグラム編集ウインドウを開いて下さい。



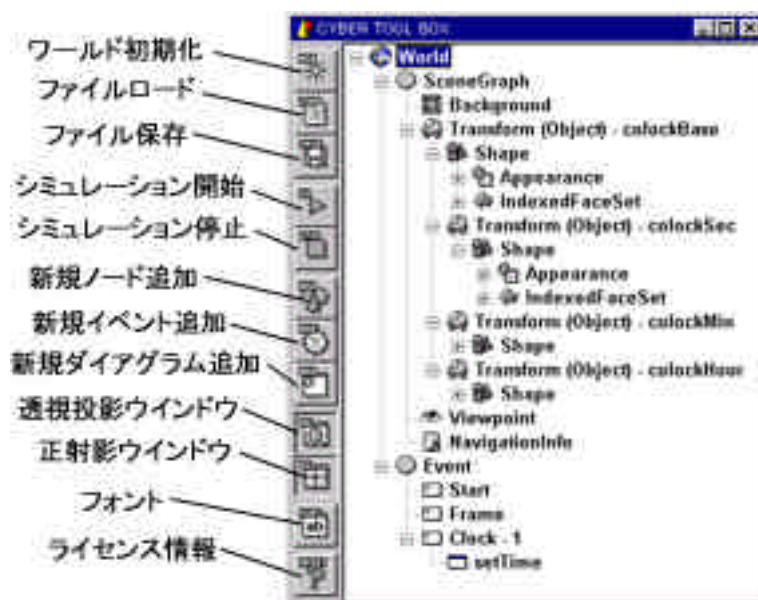
ダイアグラムの削除

ダイアグラムを削除するには、マウスの左ボタンクリックで削除したいダイアグラムをイベントツリーから選択し、キーボードの削除キーを押して下さい。



ツールバー

ワールドウインドウには、左側にツールバーがあります。このツールバーの各ボタンにより VRML ファイルの読み込み/保存、新規 ノード/イベント/ダイアグラムの追加を行います。



シーングラフ初期化

現在仮想空間に存在している、すべてのノード、ユーザー定義イベント、ダイアグラムを削除します。この操作により仮想空間には空の状態になります。仮想空間のすべてのオブジェクトを削除したい場合に、利用して下さい。



ファイルロード

選択したファイルに含まれるシーングラフ情報を、現在の仮想空間に追加します。入力できるファイルは VRML2.0/97 形式のみです。ファイルに文法的なエラーがある場合には、そのエラー行および行数が表示されます。ファイル一つでもエラーがある場合には、そのファイルにあるオブジェクトは何れも仮想空間へ追加されません。



ファイル保存

現在の仮想空間情報を、VRML2.0/97 形式のファイルに出力します。ワールドツリーに表示されている全てのシーングラフおよびイベント情報が VRML2.0/97 形式に変換され出力されます。

仮想空間の動作定義は、最終的にはVRML2.0/97形式の複数のScript ノードとJavaクラスファイルに変換されて出力されます。つまりCTB コンテンツは、一つのVRML ファイルと、複数のJava クラスファイルから構成されます。ファイルを出力する時に、保存する仮想空間動作定義の実行に必要なJava クラスファイルを、出力するファイルと同じディレクトリに生成します。インターネットで、CTB で作成したコンテンツを公開する場合には、出力したVRML2.0/97形式のファイルと、そのディレクトリにあるJava クラスファイル一式を、インターネットサーバーの同じディレクトリにアップロードする必要があります。

またシーングラフに ImageTexture ノードなどの、プロパティにファイル名を含んでいるものがあれば、そのファイルも同じディレクトリに出力されます。CTB コンテンツをインターネットで公開する場合には、同じようにこれらのファイルも、同じディレクトリにアップロードして下さい。



シミュレーション開始

仮想空間のシミュレーションを開始して、イベントダイアグラムにより定義されている仮想空間動作を実行します。シミュレーションの動作結果は、リアルタイムに更新される透視投影ウインドウ/正射影ウインドウ/ダイアグラムウインドウで確認できます。

シミュレーション実行中は、ワールドツリーやダイアグラムウインドウの編集はできませんので注意して下さい。編集を行うには、シミュレーション停止ボタンでシミュレーションを停止させる必要があります。



シミュレーション停止

現在実行されているシミュレーションを停止させます。シミュレーション実行中は、ワールドツリーやダイアグラムウインドウの編集はできませんので、編集を行うには、このボタンでシミュレーションを停止させる必要があります。



新規ノード追加

シーングラフで現在選択されているノードを親ノードとし、新しい子ノードを追加します。このボタンを押して表示されるダイアログには、選択されているノードの子ノードとして追加できるノード一覧が表示されます。この追加できるノードとは、VRML 仕様に準拠し階層構造が認められるものです。



シーングラフの最上部(親ノードがない状態)にノードを追加したい場合には、シーングラフツリーの「SceneGraph」の項目を選択し、ボタンを押して下さい。シーングラフツリーの項目が選択されていない場合に、このボタンを押しても無効です。



新規イベント追加

ユーザー定義のイベントを追加します。CTB では最初から追加されてるイベントとして「System」と「Frame」がありますが、ユーザー定義イベントとはそれ以外の、パラメータを設定できるイベントを示します。

イベントタイプ	パラメータ	概要
Clock	インターバル時間	Frame イベントより詳細なインターバルイベントを定義します。
Timer	タイマー時間	シミュレーションを開始してから、一定時間で発生するイベントを定義します。
PickUp	対象オブジェクト	オブジェクトをマウスでクリックしたときに発生するイベントを定義します。
Drag	対象オブジェクト	オブジェクトをマウスでドラッグしたときに発生するイベントを定義します。
Area	中心位置、範囲	ユーザーがある空間範囲に移動した場合に発生するイベントを定義します。

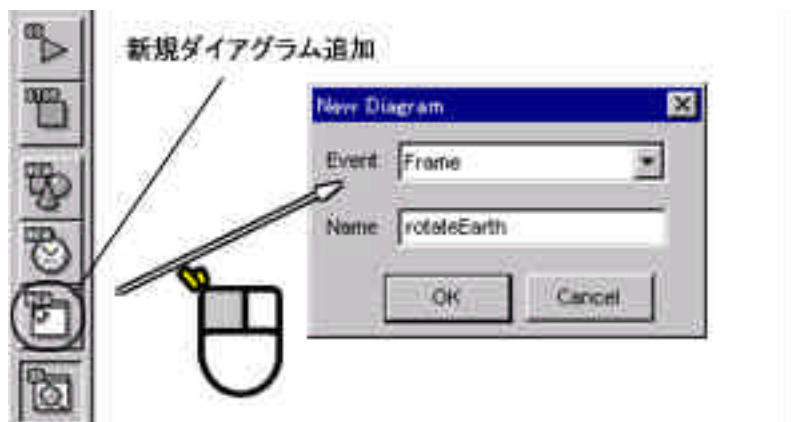
ボタンを押して表示されるダイアログで、追加するイベントタイプをダイアログ上部のタブにて選択し、各イベントパラメータを入力して、新規イベントを生成して下さい。既に同じイベントつまりイベントタイプとパラメータが一致しているイベントが存在している場合には、このイベント追加は無視されます。



イベントが正常に生成されると、生成されたイベントの項目がイベントツリーに追加されます。現状、以下の種類のユーザーイベントが定義できます。各ユーザーイベントの詳細は、動作定義の章を参照して下さい。

新規ダイアグラム追加

仮想空間動作を定義する新しいダイアグラムを追加します。このボタンを押して表示されるダイアログで、ダイアグラムのトガとなるイベントタイプを選択し、適当なダイアログ名を入力して新規ダイアグラムを生成して下さい。既に同じイベントタイプかつ名前であるダイアグラムが仮想空間に存在する場合には、このイベント追加は無視されます。



ダイアグラムが正常に生成されると、生成されたダイアグラムの項目が関連したイベントツリーに追加さ

れます。



視点透視ウインドウ表示

視点透視ウインドウの表示/非表示を切り替えます。視点透視ウインドウを表示する必要がない場合に、このウインドウを閉じるとシミュレーション性能が向上します。



正射影ウインドウ表示

正射影ウインドウの表示/非表示を切り替えます。正射影ウインドウを表示する必要がない場合に、このウインドウを閉じるとシミュレーション性能が向上します。

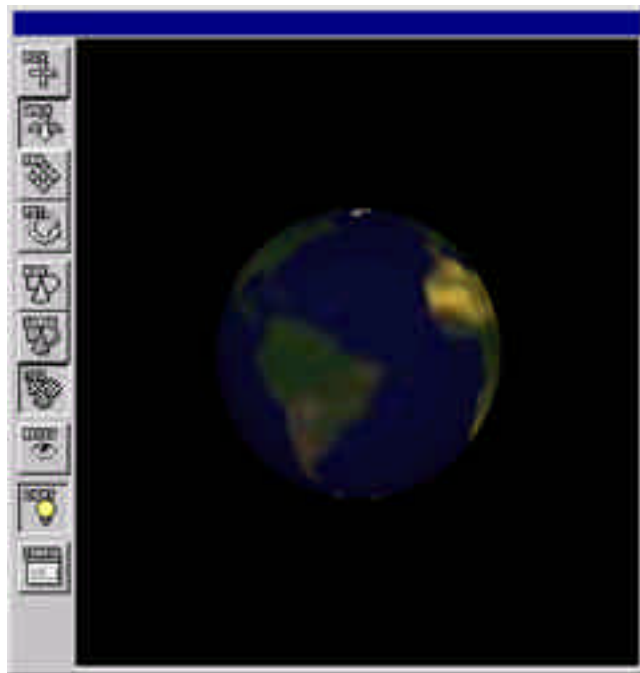


表示フォント選択

ワールドツリーを表示するフォントを選択します。このボタンを押すとフォント選択ダイアログが表示されます。このフォント設定は、CTB 終了時に保存されているため、次回起動時にも有効です。

透視投影ウインドウ

透視投影ウインドウには、仮想空間が3次的に表示されており、主にオブジェクトの3次元的な位置関係や仮想空間動作の確認を行います。この透視投影ウインドウの描画は、OpenGL を利用して行われます。システムに OpenGL のポートが装着されていれば、このウインドウはより高速に描画されます。



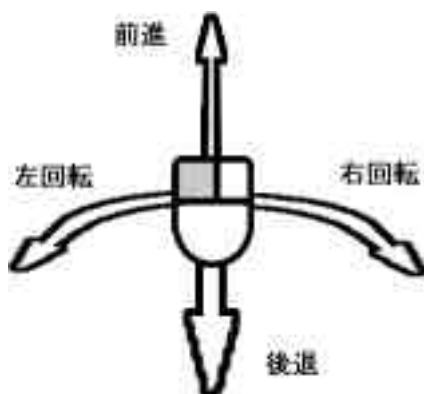
仮想空間の移動

仮想空間の移動はすべてマウスで行います。移動する方法としては、歩行移動、平行移動、回転移動の3種類があります。移動方法はツールバーのボタンで選択します。

これらの移動方法は、マウスカーソルが透視投影ウインドウの中心から離れる距離に比例して移動量が大きくなります。早く移動したい場合には、中心位置から離れた位置で、ゆっくり移動したい場合には中心位置付近でマウスカーソルを移動して下さい。

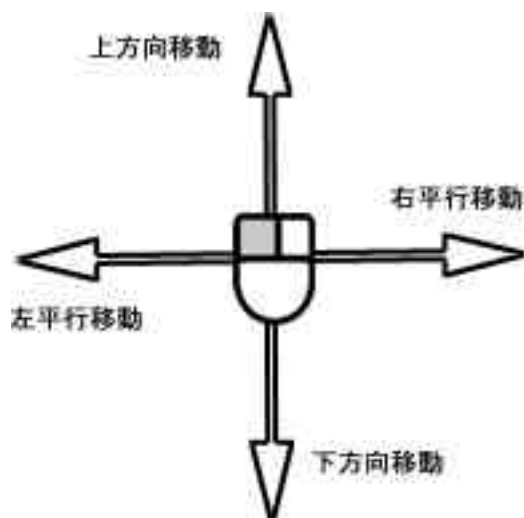
歩行移動

歩行移動は、マウスの左ボタンを押しながら、マウスポインタを上下方向に移動すると視点の前進後進、左右方向に移動するとその場で視点が左右に回転します。人間の動作で言えば、歩く動作に相当します。



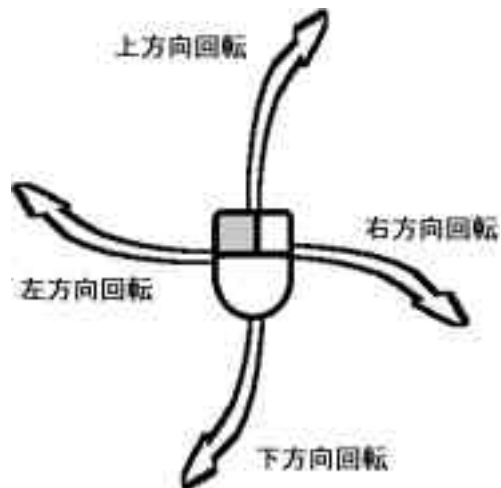
平行移動

平行移動モードでの移動は、マウスの左ボタンを押しながら、マウスポインタを上下方向に移動すると視点が上下、左右方向に移動すると左右にスライドして移動します。人間の動作で言えば、背を伸ばしたりしゃがんだり 左右にステップする動作に相当します。



回転移動

回転モードでの移動は、マウスの左ボタンを押しながら、マウスポインタを上下方向に移動すると、視点が上下方向、左右方向に移動すると視点方向に対して左右に回転します。人間の動作で言えば、上を向いたり、首を左右に傾げる動作に相当します。



オブジェクトの選択

仮想空間でオブジェクトを選択するには、まずツールバーの「選択」ボタンを押して下さい。それからマウス左ボタンでオブジェクトをクリックすると、選択されたオブジェクトが範囲枠を示す白い線で囲まれて表示されます。選択されたオブジェクトは、正射影ウィンドウでも黄色い線で描画されます。

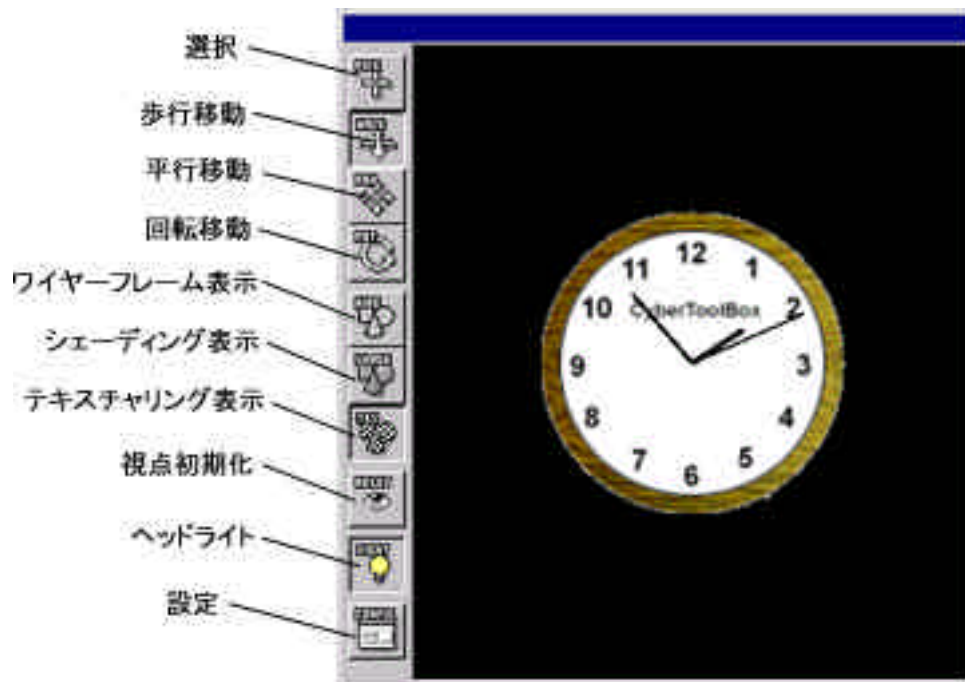


ただしシミュレーション実行中に、オブジェクトをクリックした場合には、オブジェクトの選択は行われずに、

オブジェクトをクリックしたイベントが発生します。

ツールバー

正射影ウインドウには、左側にツールバーがあります。このツールバーの各ボタンにより、マウス動作の選択、表示形式の選択などを行います。またマウス右ボタンのダブルクリックにより、マウス動作および表示形式の変更も可能です。



選択

視点透視ウインドウで仮想空間にあるオブジェクトをマウスで選択したい場合に、このボタンを押して下さい。このオブジェクトとは、VRML では Shape ノードに相当するものです。オブジェクトを選択するには、マウス左ボタンで目的のオブジェクトをクリックして下さい。

オブジェクトの選択動作は、シミュレーションが実行している場合と実行していない場合では、その動作内容は異なります。シミュレーションが実行されていない場合には、単純なオブジェクトの選択に利用されます。オブジェクトが選択されれば、視点透視ウインドウにオブジェクトを囲む白い線の範囲枠が表示され、正射影透視ウインドウでも選択されたオブジェクトは黄色い線で描画されます。シミュレーションが実行されている場合には、仮想空間にあるオブジェクトを選択すると、そのオブジェクトに対する "PickUp" イベントが発生します。



歩行移動

仮想空間の移動方法を、歩行移動にします。歩行移動では、仮想空間を歩くような感じで移動できます。マウス左ボタンを押した状態で、マウスカーソルがウインドウ中心に対して上に位置する場合には、視点が前進、下に位置する場合には後退、左に位置する場合には左に振り向き回転、右に位置する場合には右に振り向き回転を行います。移動量は、視点透視ウインドウ中心位置からのマウスカーソル位置までの距離に比例します。



平行移動

仮想空間の移動方法を、平行移動にします。平行移動では、仮想空間をスライドするような感じで移動できます。マウス左ボタンを押した状態で、マウスカーソルがウインドウ中心に対して上に位置する場合には視点が上昇、下に位置する場合には下降、左に位置する場合には左移動、右に位置する場合には右移動を行います。移動量は、視点透視ウインドウ中心位置からのマウスカーソル位置までの距離に比例します。



回転移動

仮想空間の移動方法を、回転移動にします。歩行移動では、仮想空間を回り込むような感じで移動できます。マウス左ボタンを押した状態で、マウスカーソルがウインドウ中心に対して上に位置する場合には、視点が上を向くように回転、下に位置する場合には下を向くように回転、左に位置する場合には左に傾げるような回転、右に位置する場合には右に傾げるような回転を行います。移動量は、視点透視ウインドウ中心位置からのマウスカーソル位置までの距離に比例します。



表示形式選択

画面の表示形式は、ワイヤーフレーム、シェーディング、テクスチャリングの3種類から選択できます。各表示形式は、作業形態やシステムパフォーマンスにより選択して下さい。例えばテクスチャマッピングがサポートされていないシステムで、CTB を動作させる場合には、表示形式を通常のシェーディングに変更することにより、作業が軽快に行えるでしょう。



視点初期化

視点リセットボタンは、現在仮想空間に追加されているすべてのオブジェクトが見える位置に視点を移動させます。仮想空間オブジェクト全体を見たい場合や、仮想空間を移動しているうちに現在位置がわ

からなくなった場合に、このボタンで視点位置を初期化して下さい。視点の移動アルゴリズムは、最初に仮想空間の中心に移動し、それから仮想空間サイズを参照して Z 軸正方向に移動します。視点方向は (0,0,1,0) に初期化されます。



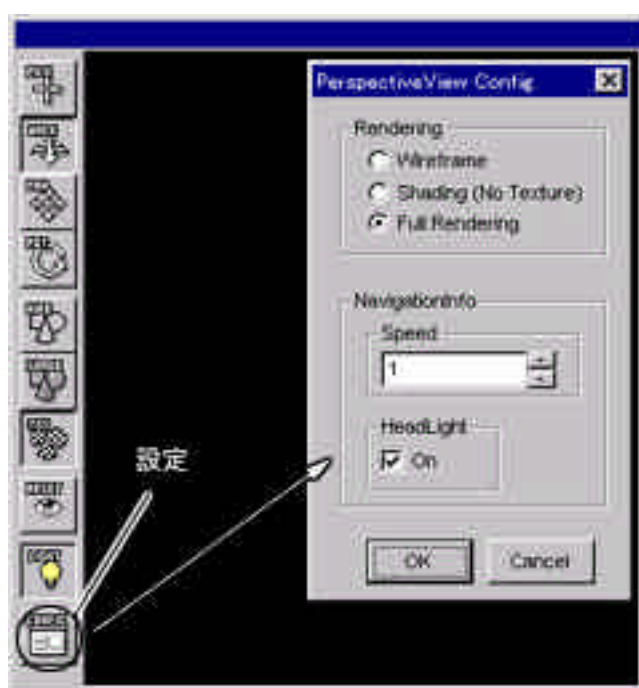
ヘッドライト

ヘッドライトボタンは、仮想空間に視点と一緒に移動する点光源を追加します。仮想空間に適切な光源がない場合には、オブジェクトは表示されません。正確には、オブジェクトは通常どおりレンダリングされていますが、オブジェクトに照射する光源がないため表示されていないように見えます。このような場合に、このボタンを押してヘッドライト光源を利用して下さい。



設定

仮想空間の表示形式や、移動する場合のスケーリング値などを設定します。ボタンを押すと設定用のダイアログが表示されます。

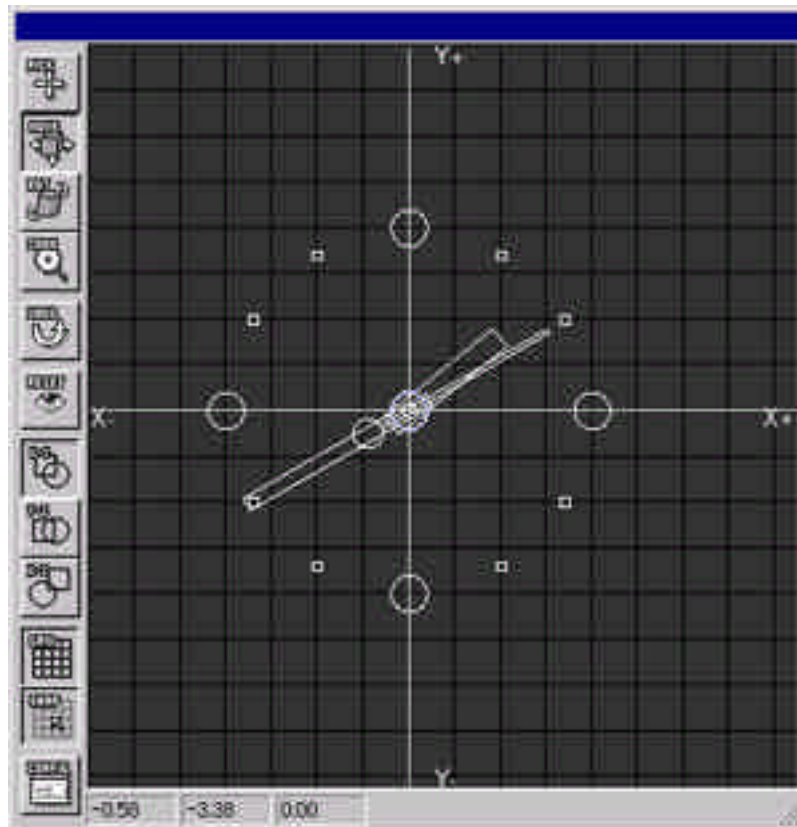


設定項目 Rendering は、画面の表示形式を、ワイヤーフレーム、シェーディング、テクスチャリングから選択します。この選択はツールバーでもできます。

設定項目 NavigationInfo は、マウスで移動する速度とヘッドライトの有無を指定します。ヘッドライトの選択はツールバーでもできます。

正射影ウィンドウ

正射影ウィンドウには、仮想空間が2次元的に表示されており、主にオブジェクトの追加/編集などの作業を行います。この正射影ウィンドウの描画は、OpenGL を利用して行われます。システムに OpenGL のポートが装着されていれば、このウィンドウはより高速に描画されます。



オブジェクト編集

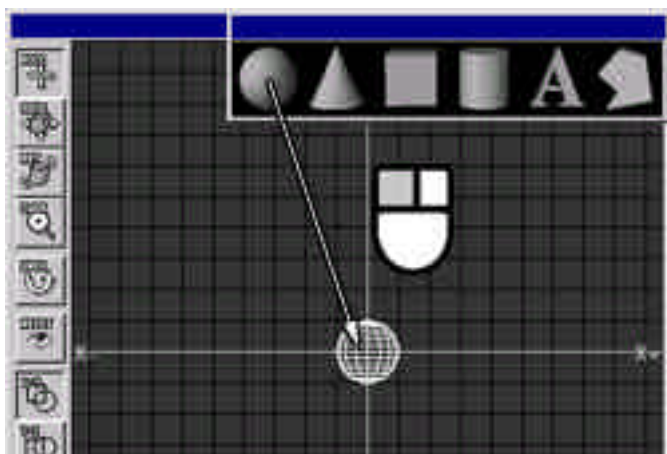
基本的なオブジェクト編集は、すべてマウス操作で行います。新しいプリミティブオブジェクトを仮想空間に追加したり、既に仮想空間に追加されているオブジェクトを移動したり回転することができます。

プリミティブオブジェクトの追加

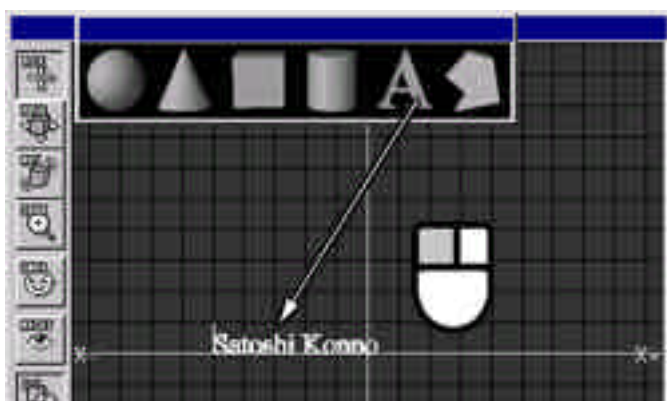
仮想空間にオブジェクトを追加するには、プリミティブウィンドウから正射影ウィンドウに追加したいプリミティブをドラッグ&ドロップして下さい。追加できるプリミティブには、球/円柱/箱/円錐/テキストポリゴンの6種類があります。

球/円柱/箱/円錐プリミティブを追加するには、プリミティブウィンドウから目的のオブジェクトをマウス左ボ

タンでドラッグしながら、正射影ウインドウの追加したい位置にドロップして下さい。プリミティブのサイズは、現在の正射影ウインドウのグリッドサイズを基準に設定されます。例えば、球プリミティブの場合には、このグリッドサイズを半径とした球オブジェクトが追加されます。

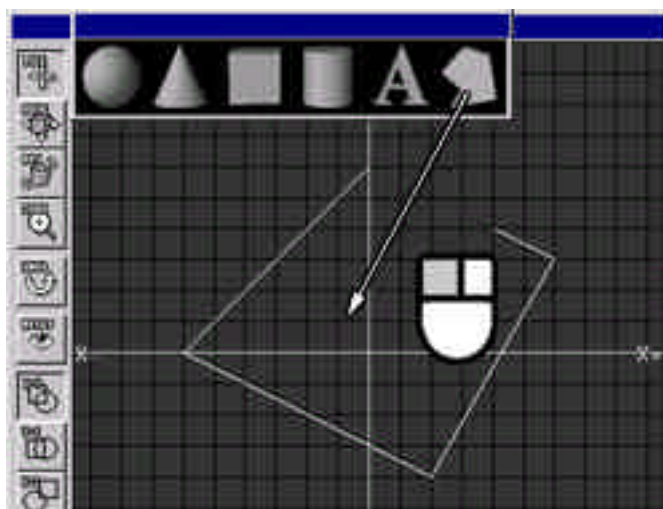


テキストプリミティブを追加する場合には、最初にプリミティブウインドウからテキストプリミティブをマウス左ボタンでドラッグしながら、正射影ウインドウの追加したい位置にドロップして下さい。ドロップした位置に、テキスト入力を示す枠が表示されますので、追加したい文字をキーボードから入力し、リターンキーを押して終了して下さい。後退キーを押すと一つ前に入力した文字を削除、エスケープキーを押すと、テキストプリミティブの追加はキャンセルされます。



ポリゴンプリミティブを追加する場合には、最初にプリミティブウインドウからポリゴンプリミティブをマウス左ボタンでドラッグしながら、正射影ウインドウにドロップして下さい。ポリゴンプリミティブの場合には、ドロップする位置は適当で構いません。ドロップすると、マウスカーソル形状が変化し、ポリゴン頂点の入力状態になります。マウス左ボタンでポリゴン頂点位置を指定していき、キーボードのリターンキー押すか、最初に指定した頂点位置をクリックすると、指定されたポリゴン頂点から、奥行きを現在の正射影ウインドウのグリッドサイズを基準としたポリゴンオブジェクトを生成します。既に追加されているポリゴン稜線と交わるような、ポリゴン頂点は追加できません。キーボードのエスケープキーを押すか、マウスの右

ボタンを押すと、ポリゴンプリミティブの追加はキャンセルされます。



オブジェクト選択

オブジェクトを選択するには、最初にツールバーの「選択」ボタンを押して下さい。それから、目的のオブジェクトをマウス左ボタンでクリックすると、そのオブジェクトが選択されます。選択されたオブジェクトは黄色い線で描画されます。透視投影ウインドウでもオブジェクト範囲を示す白い枠線に囲まれて表示されます。同時に選択できるオブジェクトの数は1つです。



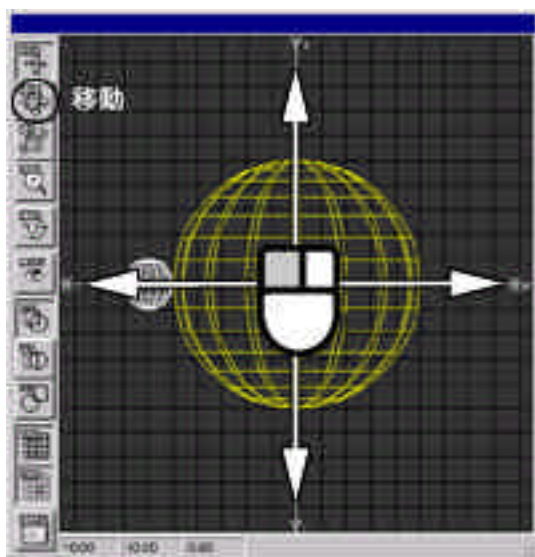
選択されたオブジェクトは、移動、回転、拡大縮小、マテリアル/テクスチャ設定ができます。オブジェクトの選択を解除するには、オブジェクトが何もない位置を選択して下さい。

またシーングラフで選択されたオブジェクトの階層下にあるオブジェクトも、暗い黄色い線で描画されま

す。このオブジェクトは、選択されたオブジェクトの階層下にあるため、親である選択されたオブジェクトへの編集の影響を受けます。例えば、選択されたオブジェクトが平行移動すれば、階層下にあるオブジェクトも一緒に移動します。

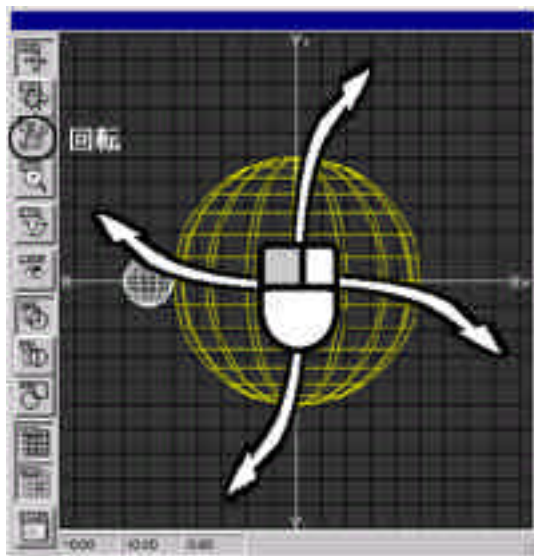
オブジェクト移動

選択されているオブジェクトを移動するには、最初にツールバーの「移動」ボタンを押して下さい。それから、マウス左ボタンを押しながらマウスカーソルを移動すると、現在正射影ウインドウが表示している平面上をオブジェクトがスライドして移動します。この操作は正射影ウインドウのグリッドスナップ設定に影響を受けます。



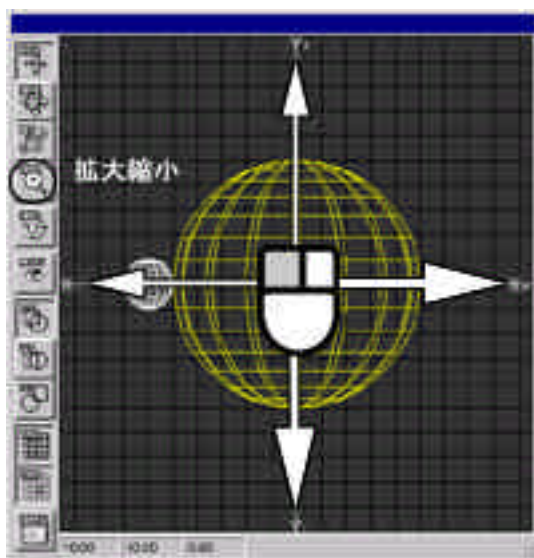
オブジェクト回転

選択されているオブジェクトを回転するには、最初にツールバーの「回転」ボタンを押して下さい。それから、マウス左ボタンを押しながら、マウスカーソルを上下に移動すると、正射影ウインドウが表示している平面に対して水平方向にオブジェクトが回転、マウスカーソルを左右に移動すると、平面上でオブジェクトが回転します。この操作は正射影ウインドウのグリッドスナップ設定に影響を受けます。



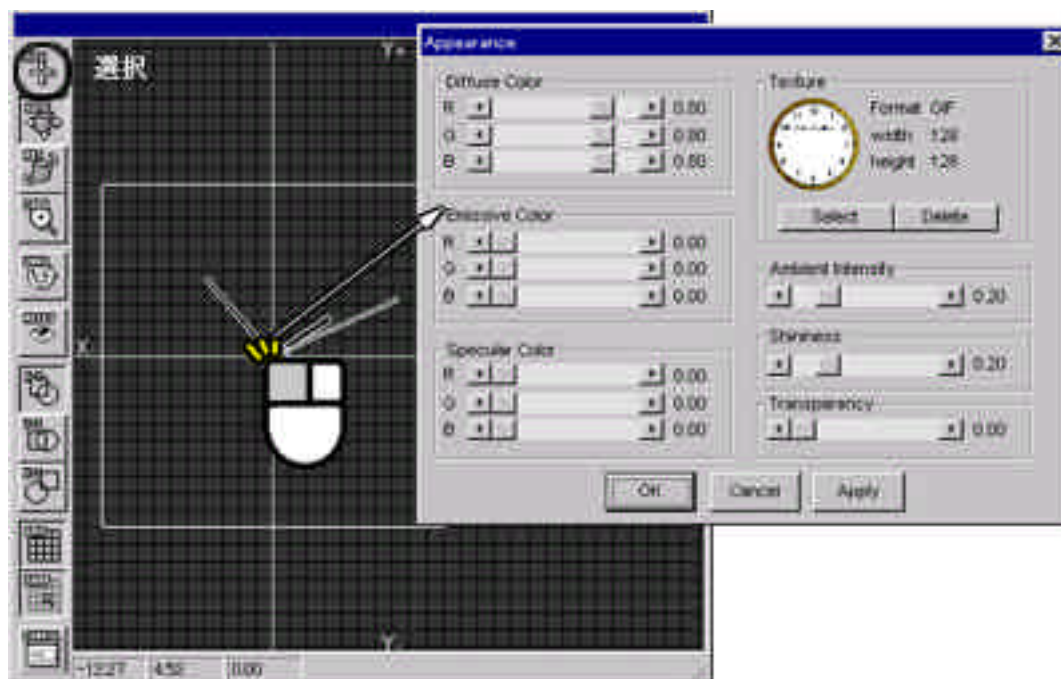
オブジェクト拡大/縮小

選択されているオブジェクトをスケールするには、最初にツールバーの「拡大縮小」ボタンを押して下さい。それから、マウス左ボタンを押しながらマウスカーソルを上下に移動すると、現在正射影ウインドウが表示している平面上でオブジェクトが拡大/縮小します。この操作は正射影ウインドウのグリッドスナップ設定に影響を受けます。



オブジェクトマテリアル/テクスチャ設定

オブジェクトの色やテクスチャを変更するには、最初にツールバーの「選択」ボタンを押して下さい。それから、目的のオブジェクトをマウス左ボタンでダブルクリックすると、色やテクスチャを設定するダイアログが表示されます。



正射影ウインドウは単純なワイヤー フレームでの表示なので、このダイアログでの変更結果は正射影ウインドウには直接的に反映されない点に注意して下さい。変更結果を確認するには、透視投影ウインドウを利用して下さい。また透過(Transparency)指定は、透視投影ウインドウに反映されません。

表示平面/領域

正射影ウインドウは、仮想空間を平面に投影し表示しています。その表示平面や表示領域を任意に変更することができます。

表示平面の選択

正射影ウインドウでは、その表示平面はXY 平面/YZ 平面/XZ 平面の何れから選択できます。表示平面の選択はツールバーのボタンで行います。



XY 平面

XY 平面への正射影の画面を表示します。X 軸正方向は画面の右方向、Y 軸正方向は方向、Z 軸正方向は奥から手前へ方向です。

YZ 平面

YZ 平面への正射影の画面を表示します。X 軸正方向は手前から奥へ方向、Y 軸正方向は方向は上方向、Z 軸正方向は右方向です。

XZ 平面

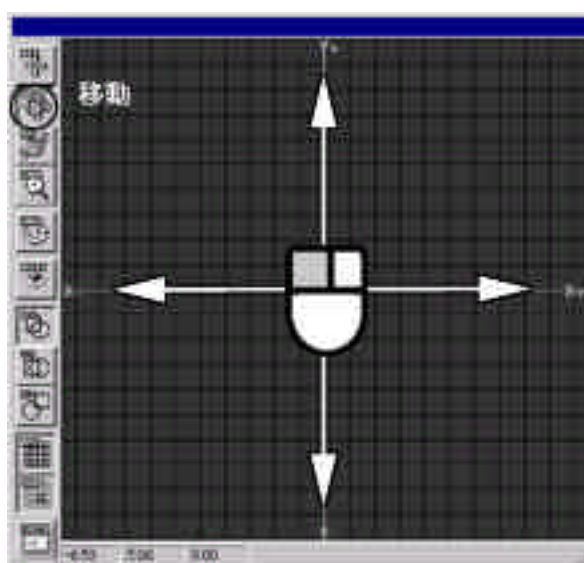
XZ 平面への正射影の画面を表示します。X 軸正方向は画面の右方向、Y 軸正方向は手前から奥へ方向、Z 軸正方向は上方向です。

表示領域の指定

正射影ウインドウの表示領域は、マウス操作により移動、拡大/縮小、範囲の指定を行うことにより 任意の位置を表示させることができます。表示領域の指定を行う前に、オブジェクトが何も選択されていないか確認して下さい。オブジェクトが選択されている場合には、オブジェクトが何も無い位置をマウス左ボタンでクリックして、選択を解除して下さい。

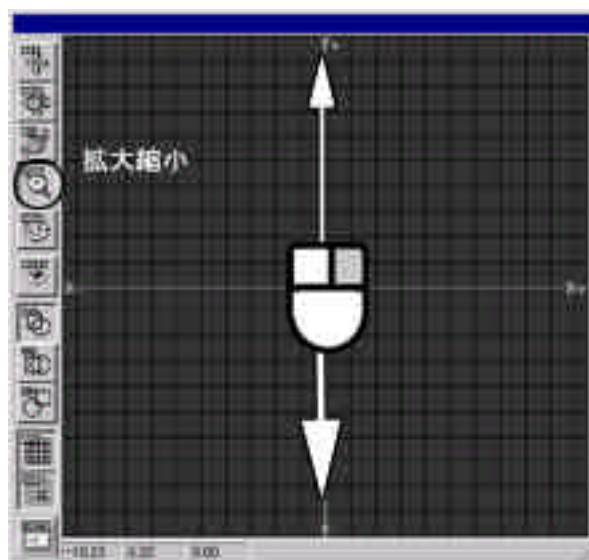
表示領域の移動

正射影ウインドウの表示領域を移動するには、最初にツールバーの「移動」ボタンを押して下さい。マウスの左ボタンを押しながらマウスカーソルを移動させて下さい。マウスカーソルの動きにあわせて、表示領域がスライドして移動します。



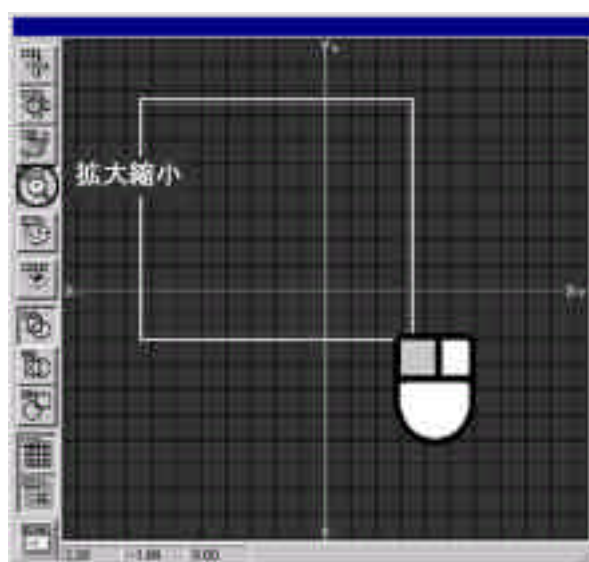
表示領域の拡大/縮小

正射影ウインドウの表示領域の拡大縮小を行うには、最初にツールバーの「拡大縮小」ボタンを押して下さい。マウス右ボタンを押しながら、カーソルを上方向に移動させると表示領域が拡大、下方向に移動すると縮小します。



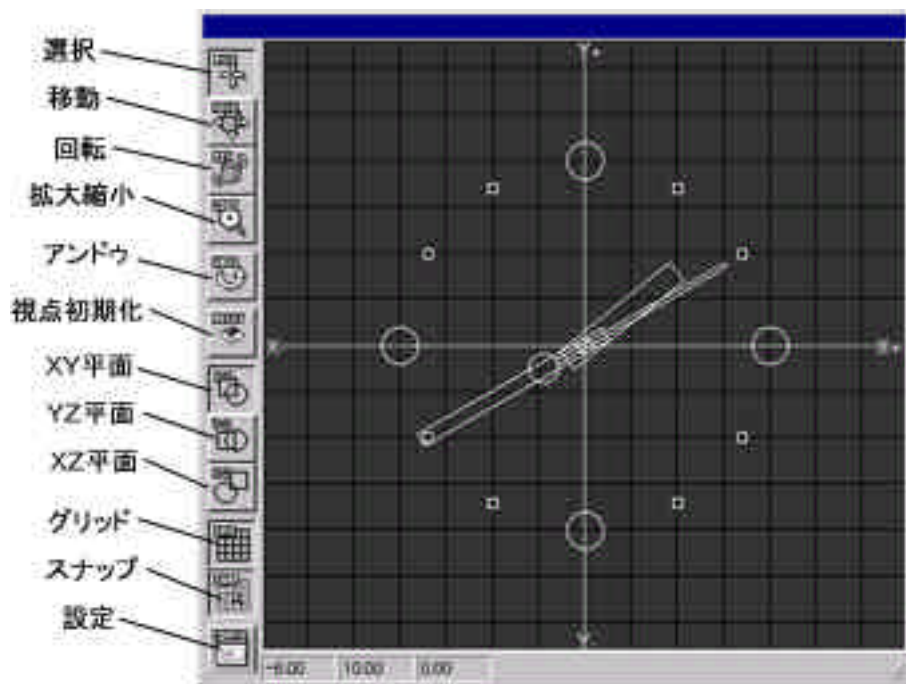
表示領域の範囲指定

正射影ウインドウの表示領域範囲の指定を行うには、最初にツールバーの「拡大縮小」ボタンを押して下さい。マウスの左ボタンを押しながら、ドラッグして表示する範囲の指定をして下さい。指定された領域に合わせて、表示領域が更新されます。



ツールバー

正射影ウインドウには、左側にツールバーがあります。このツールバーの各ボタンにより、オブジェクト操作モードや、表示平面の選択を行います。またマウス右ボタンのダブルクリックにより、マウス動作および表示平面の変更も可能です。



選択

選択モードでは、マウスの左ボタンクリックにより仮想空間にあるオブジェクトの選択ができます。このオブジェクトは、VRML での Shape ノードに該当します。選択されたオブジェクトは黄色で描画され、選択されたオブジェクトの階層下にあるオブジェクトは暗い黄色で描画されます。この選択したオブジェクトに、移動/回転などの操作が行え、この操作は黒い黄色で表示されている選択されたオブジェクトの階層下にあるオブジェクトにも影響します。選択しているオブジェクトを解除するには、オブジェクトが何も無い場所を左クリックして下さい。また選択されたオブジェクトは、透視投影ウインドウでも確認でき、白い範囲枠で囲われています。

移動

移動モードでは、選択されたオブジェクトの平行移動や、表示領域の平行移動を行います。選択モードでオブジェクトが選択されている場合には、現在の表示モードの平面上で、マウスの左ボタンドラッグ方向に対応した軸方向にオブジェクトが移動します。この操作は現在のグリッドスナップ設定に影響を受

けます。オブジェクトが選択されていない場合には、マウスポインタ移動により表示領域がスライドします。



回転

回転モードでは、選択されたオブジェクトの回転を行います。マウスの左ドラッグにより、右方向に移動すれば現在の表示モード平面上で右回転、左方向に移動すれば左回転します。この操作は現在のグリッドスナップ設定に影響を受けます。選択されたオブジェクトがない場合には、何の動作もしません。



拡大縮小

ズームモードでは、現在表示されている画面領域の設定やズーミング、または選択モードで選択されたオブジェクトの拡大縮小を行います。オブジェクトが選択されていない場合には、マウス左ボタンドラッグ操作により、表示領域の設定、マウス右ボタンの上下方向ドラッグにより、表示領域の拡大縮小を行います。上方向の移動により表示領域の縮小、下方向で拡大します。オブジェクトが選択されている場合には、マウスの右ボタンドラッグにより、マウスポインタが移動した方向に対応する各軸方向への拡大縮小処理を行います。マウスポインタが下または右方向に移動した場合にはオブジェクトが拡大、上または左方向に移動した場合には縮小します。この操作は現在のグリッドスナップ設定に影響を受けます。ただし、



アンドゥ

移動/回転/拡大縮小の各操作を取り消します。この機能は、前回の操作だけではなく、数十回前までの操作に対しても有効ですので、ボタンを繰り返し押すことにより、希望するだけ操作を取り消すことができます。



表示平面選択

正射影ウインドウの表示モードを、XY/YZ/XZ 平面から選択します。



グリッド表示

現在設定されているグリッドラインの表示/非表示の選択を行います。ボタンが押されている場合には、現在設定されているグリッド間隔を示す線が描画されます。



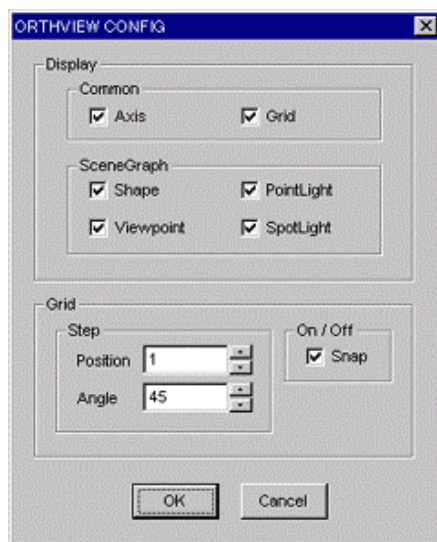
スナップ設定

移動/回転/ズームの各操作を、現在のグリッド設定で拘束するかの設定を行います。ボタンが押されている場合には、グリッド設定が有効になり 移動/回転/ズームの各操作はグリッド単位で行われます。



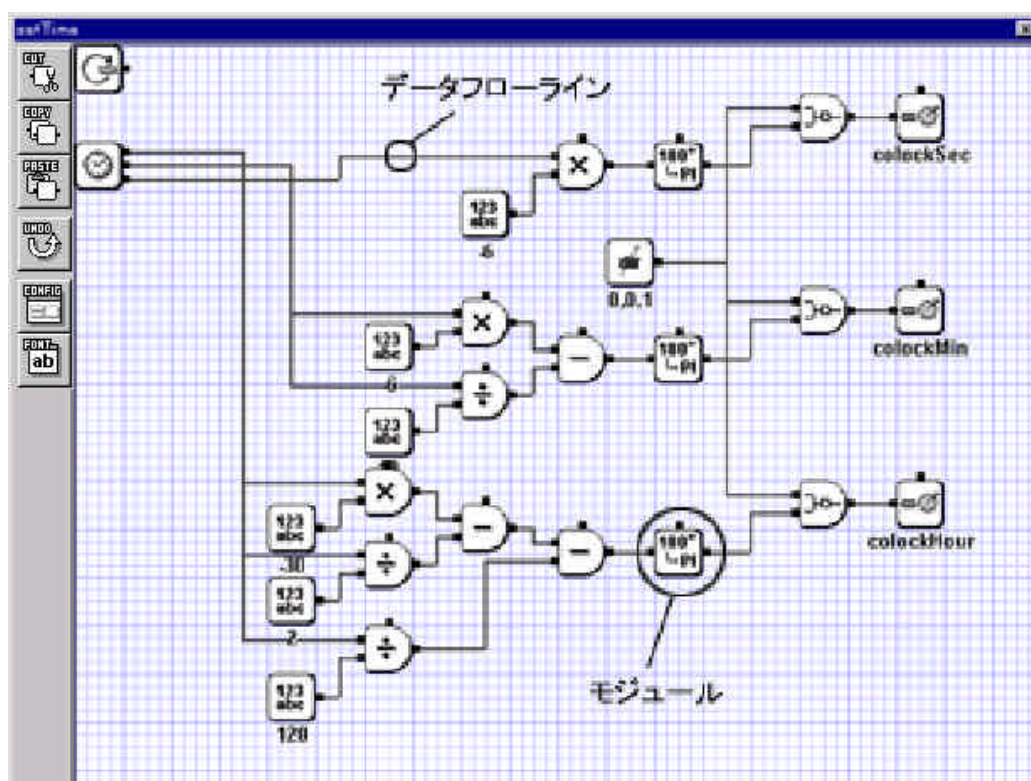
オプション

表示するオブジェクトの設定やグリッド間隔の設定を行います。設定項目 Display では、正射影ウィンドウに表示するオブジェクトを指定します。設定項目 Grid では、グリッドのステップ位置/回転間隔および、グリッド位置にスナップするかの指定をします。



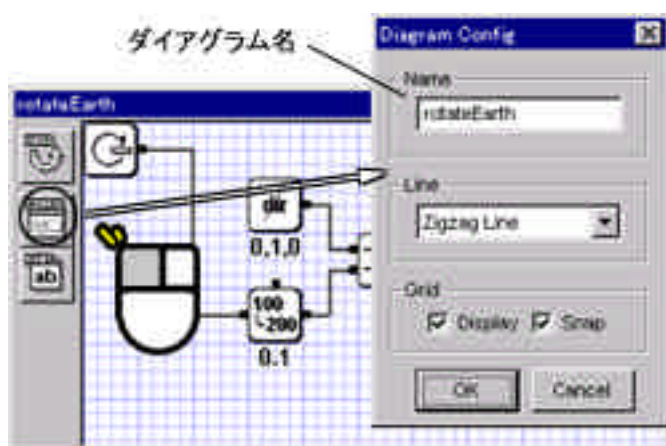
ダイアグラムウィンドウ

ダイアグラムウィンドウは、仮想空間動作を定義するダイアグラムの編集ウィンドウです。ダイアグラムによる仮想動作は複数のモジュールおよび、モジュールのノート間を接続しているデータフローラインにより構成されています。ここでは、このモジュールおよびデータフローラインの操作方法について説明します。



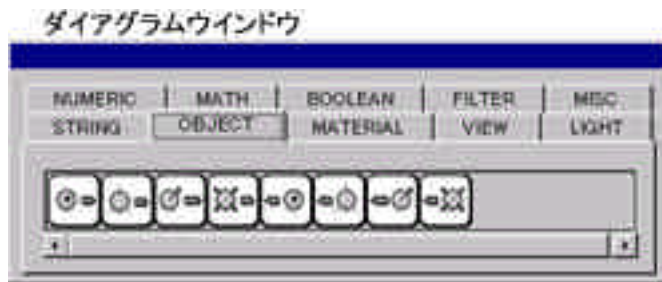
ダイアグラム名

ダイアグラムの名前はダイアグラムウィンドウのタイトルに表示されています。このダイアグラム名を変更するには、ツールバーの「設定」ボタンを押して表示されるダイアログを利用して下さい。



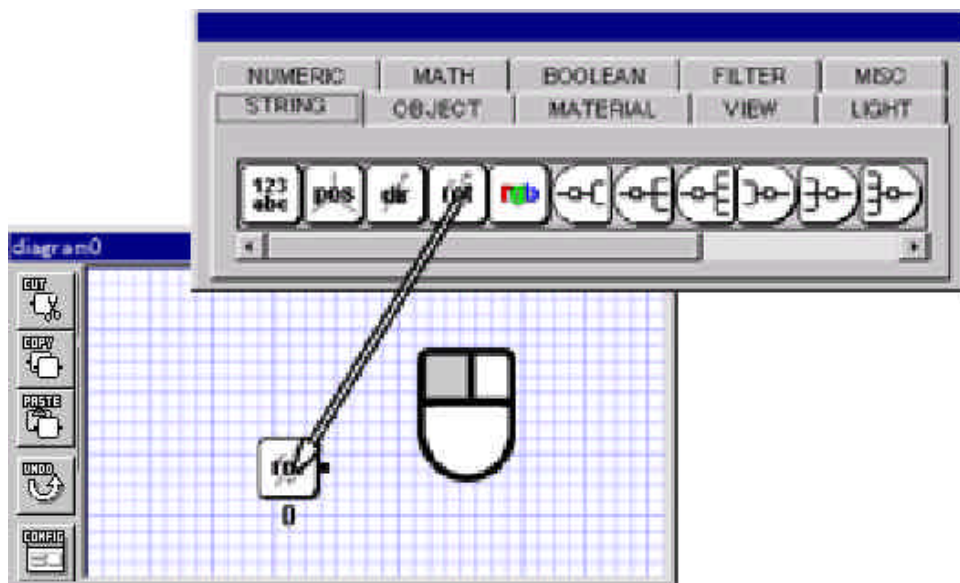
モジュール操作

モジュールは、仮想空間動作を定義する最小の単位で、オブジェクト位置/方角を設定するものや、文字列の加算/減算をおこなうものなど、色々な用途のものがあります。これらのモジュールは、ダイアグラムウインドウに用途別に分類されて表示されています。



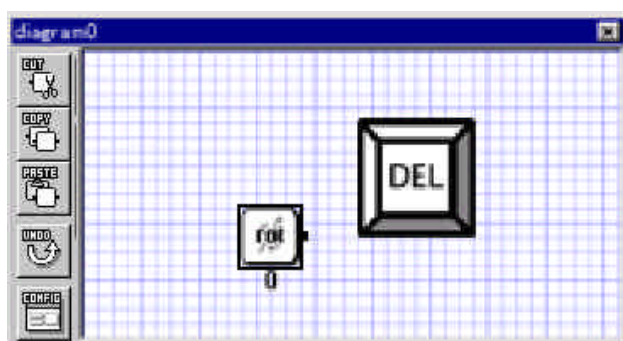
モジュールの追加

モジュールを追加するには、追加したいモジュールをモジュールウインドウからマウス左ボタンでドラッグして、モジュールを追加したいダイアグラムウインドウ上でドロップして下さい。この操作はダイアグラムウインドウのグリッドスナップ設定に影響を受けます。



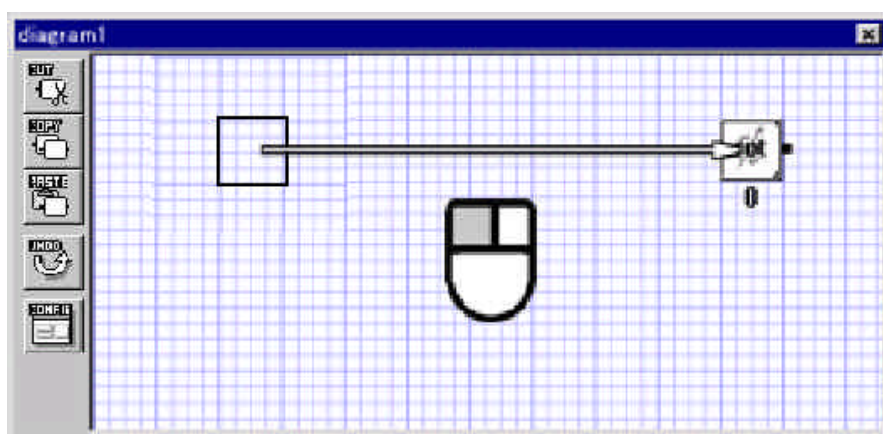
モジュールの削除

モジュールを削除するには、削除したいモジュールをマウス左ボタンクリックで選択し、キーボードの削除キーを押して下さい。モジュールが削除されれば、モジュールの各ノードに接続しているデータフローラインも一緒に削除されます。しかし、ダイアグラム生成時から追加されているシステムモジュールについては削除できません。



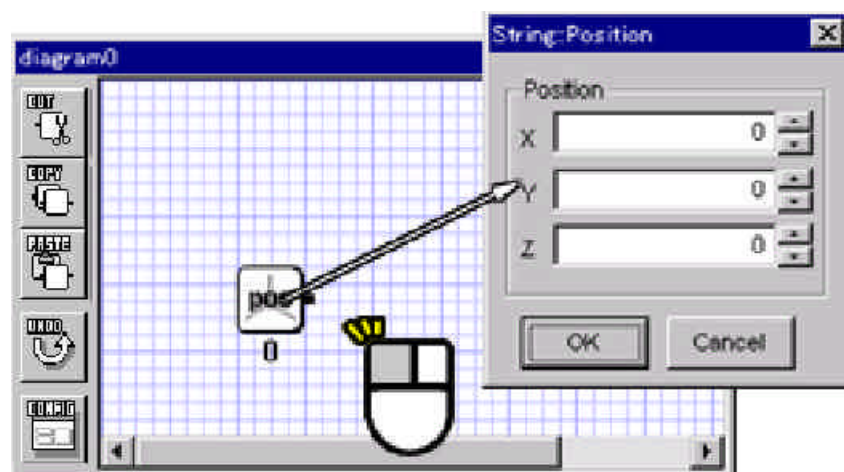
モジュールの移動

モジュールを移動するには、移動したいモジュールをマウスの左ボタンクリックでドラッグしながら、移動して下さい。この操作はダイアグラムウインドウのグリッドスナップ設定に影響を受けます。



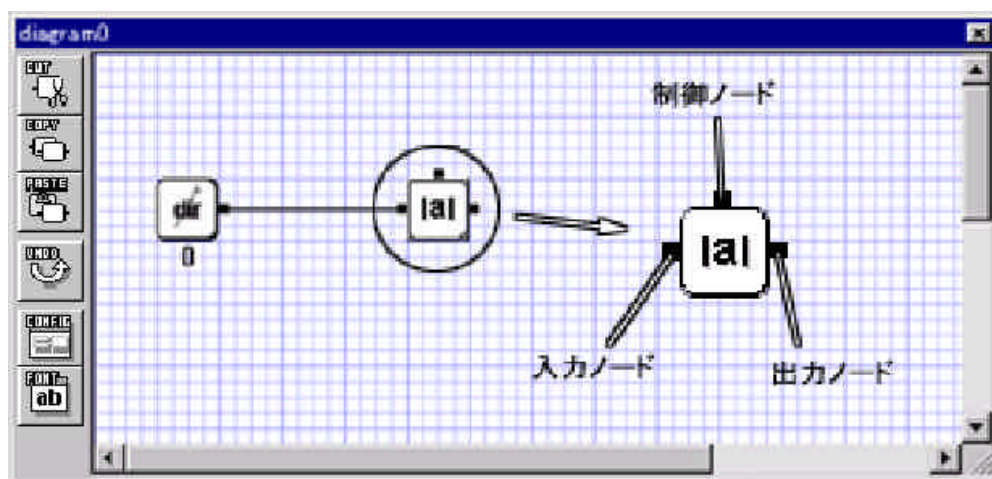
モジュールデータの設定

モジュールの種類によっては、例えば出力する文字列や、設定する対象のオブジェクトなど内部データを設定する必要があります。この内部データを設定するには、マウス左ボタンで設定するモジュールをダブルクリックし、表示されるダイアログを利用して下さい。



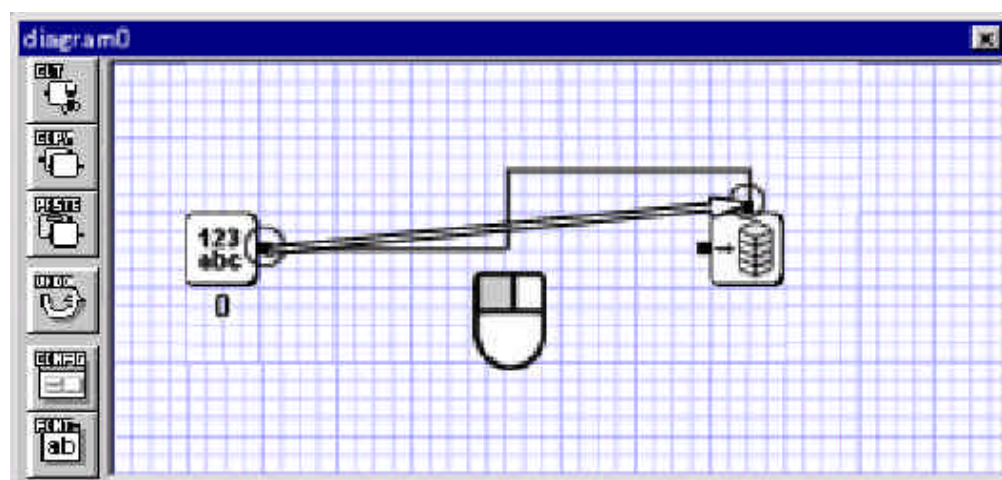
データフローライン操作

モジュールはデータを出力し、その出力したデータを他のモジュールへ入力することができます。このデータ経路を定義するのが、データフローラインです。モジュールにはデータを入力する3種類のノードがあり、モジュール左側にあるのがデータ入力ノード、右側にあるのがデータ出力ノード、上部にあるのが制御ノードがノードになります。



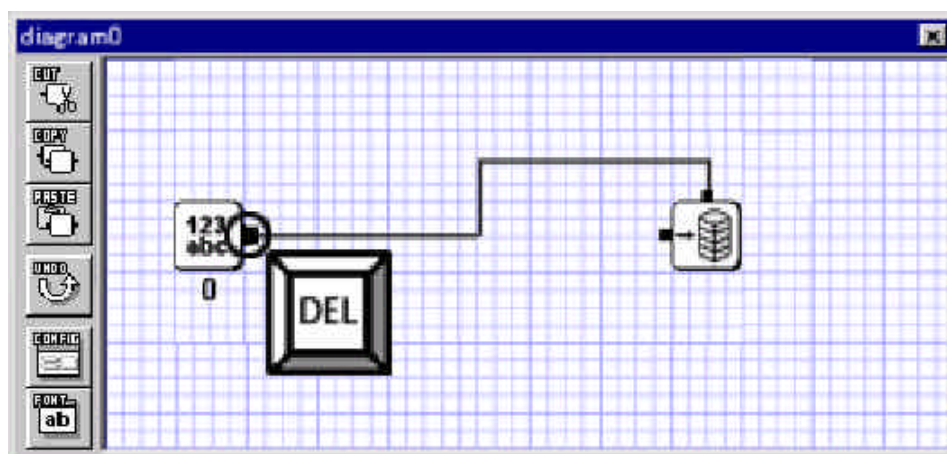
データフローラインの追加

モジュール間をデータフローラインにより接続するには、最初にモジュールの左右にある黒いノードをマウスの左ボタンクリックで選択し、ドラッグしながら他のモジュールのノード上でドロップして下さい。最初にモジュール右側にある出力ノードを選択した場合には、ドロップ先は他モジュールの左側にある入力ノード、または上部にある制御ノードである必要があります。同様に最初に入力ノードまたは制御ノードを選択した場合には、ドロップ先は出力ノードである必要があります。同じモジュール内での、ノード間接続はできません。



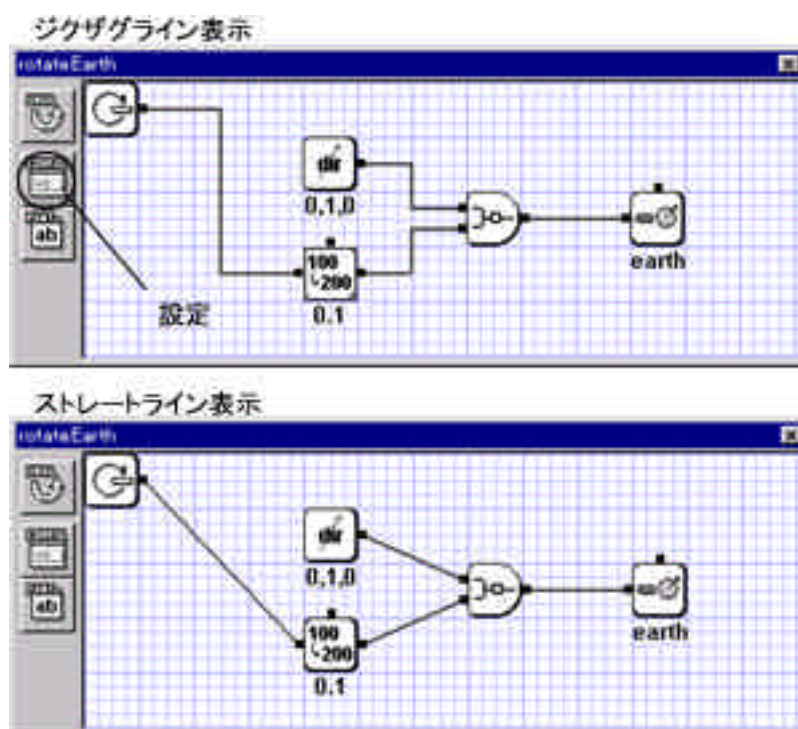
データフローラインの削除

モジュール間のデータフロー接続を削除するには、データフローラインの端点の何れかをマウス左ボタンクリックで選択し、キーボードの削除キーを押して下さい。



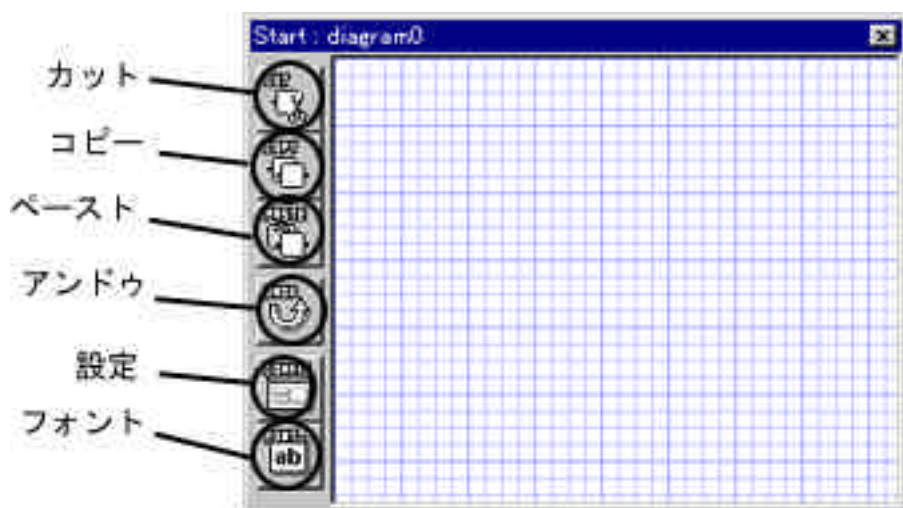
データフローラインの表示形式

ダイアグラムウインドウでの、データフローラインの表示形式は、モジュールのノート間をジグザグの線で結んで表示する方法と、ストレートラインで結ぶ方法があります。この設定はツールバーの「設定」ボタンを押して、表示されるダイログで設定して下さい。状況に応じてデータフローラインが見やすいほうを選択して下さい。



ツールバー

ダイアグラムウインドウには、左側にツールバーがあります。このツールバーの各ボタンにより、アンドゥ操作や、表示形式の設定などを行います。



カット

ダイアグラムで選択された範囲のモジュールおよびデータフローラインをクリップボードにカットします。



コピー

ダイアグラムで選択された範囲のモジュールおよびデータフローラインをクリップボードにコピーします。



ペースト

現在クリップボードにコピーされている範囲のモジュールおよびデータフローラインを現在のダイアグラムにペーストします。



アンドゥ

モジュールの追加/削除/移動や、データフローラインの追加/削除などの操作を取り消します。この機能は、前回の操作だけではなく、数十回前までの操作に対しても有効ですので、ボタンを繰り返し押すことにより、希望するだけ操作を取り消すことができます。



設定

表示されるダイアログで、ダイアグラム名、データフローライン表示するライン形式、グリッドの設定を行います。



フォント

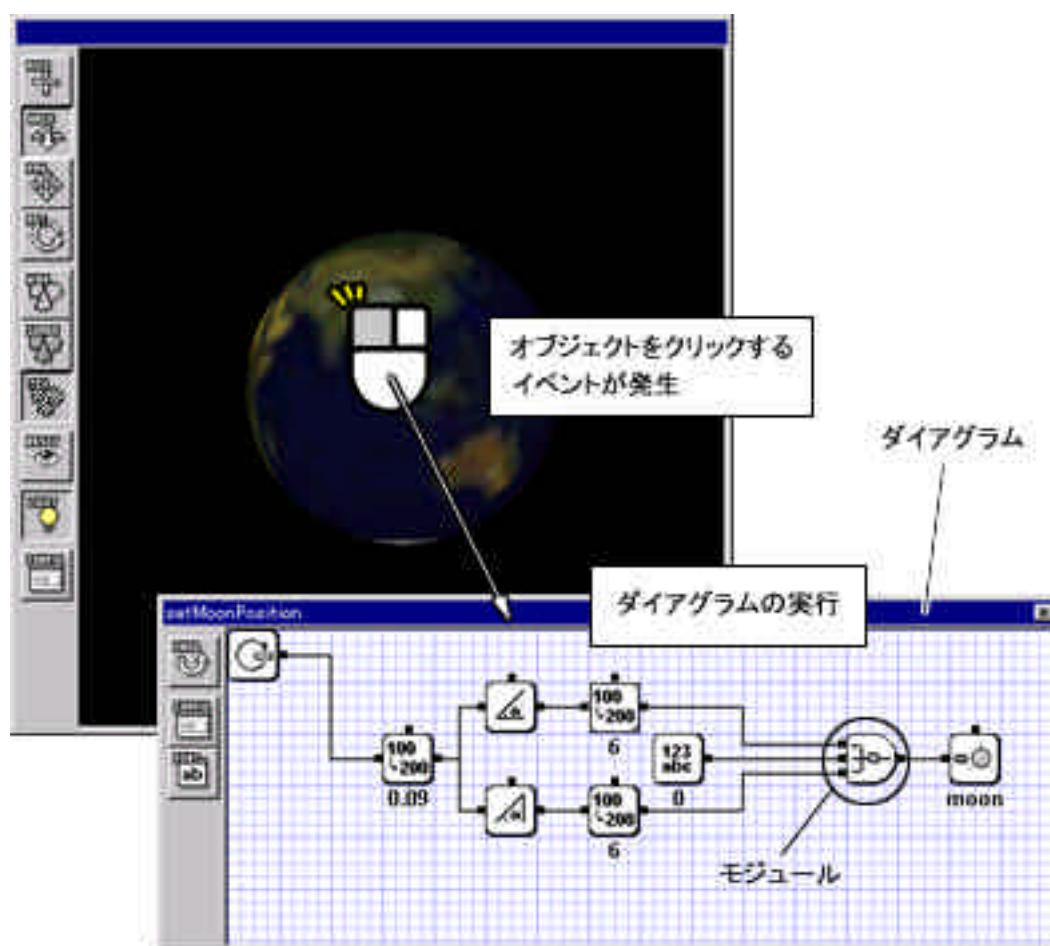
ダイアグラムウインドウで描画される文字のフォントを選択します。このボタンを押すとフォント選択ダイアログが表示されます。このフォント設定は、CTB 終了時に保存されているため、次回起動時にも有効です。

6.動作定義

動作のしくみ

CTB では、仮想空間動作をアイコンベースのビジュアル言語を利用して構築していきます。マウスでアイコン同士を線でつないでいく分かりやすい操作方法でプログラミングを行います、誰にでも簡単に楽しい仮想空間動作が構築できます。

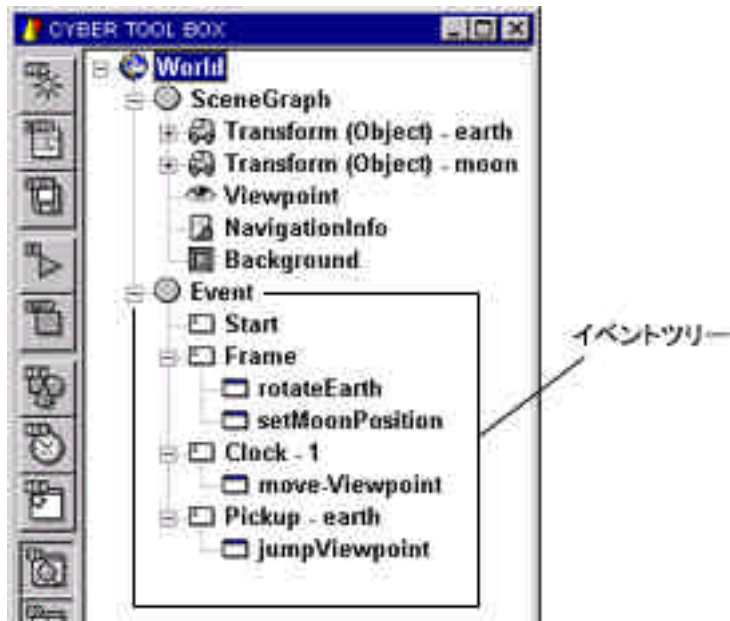
CTB の仮想空間動作は、仮想空間で発生したイベントをトリガにして定義していきます。この発生するイベントは、VRML2.0/97 のイベントと同様な、一定の時間間隔で発生するものや、マウスのクリックによって発生するものなどがあります。CTB では、イベントが発生するとダイアグラムと呼ばれるビジュアル言語により定義された動作が実行されます。このダイアグラムは、アイコンを線でつなぐことにより動作が定義されており、このアイコンをCTB ではモジュールと呼んでいます。



イベント

CTB では、シミュレーションを開始した時や、マウスでオブジェクトをクリックしたときにイベントが発生します。仮想空間動作はダイアグラムと呼ばれるウインドウでモジュールを線でつなぎあわせて定義していきますが、イベントはダイアグラムに定義された仮想空間動作を開始する トガとなるものです。ダイアグラムには、生々時に実行のトガとなるイベントを必ず選択します。そしてこの実行のトガとなるイベントが発生した時に、ダイアグラムが実行される仕組みになっています。

イベントには、CTB で予め定義されているシステムイベントと、ユーザーにより追加できるユーザー定義イベントの 2 種類があります。現在の仮想空間で定義されているイベントについては、ワールドウインドウのイベントツリーで確認でき、各イベントが発生した時に実行されるダイアグラムは、そのイベント項目の配下に整理されて表示されています。



CTB のイベントは、VRML で発生するイベントをベースに作成されたものです。そのため、CTB で発生するイベントは、VRML で発生するイベントと互換性があります。

システムイベント

システムイベントは、CTB で最初から定義されているイベントで、仮想空間シミュレーションの基本になるイベントです。イベントは CTB システム内部より発生する固定的なもので、設定できるパラメータはありません。システムイベントは、以下の 2 種類です。

イベントタイプ	概要
Start	シミュレーション開始時に発生するイベント
Frame	一秒間に 10 回発生するインターバルイベント

Start

仮想空間のシミュレーションが開始される時に、一回だけ発生するイベントです。シミュレーション開始時の何らかの初期化が必要な場合に利用します。

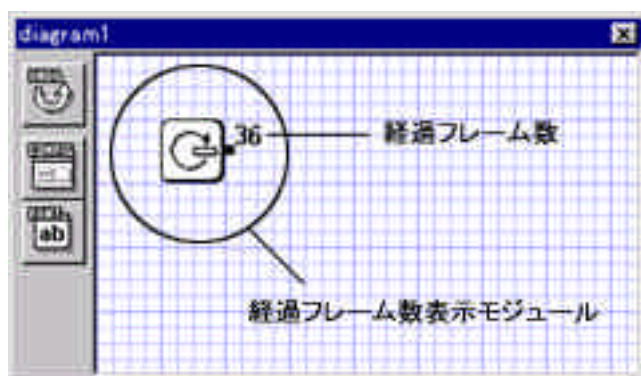
例えば車などのオブジェクトのスタート位置を設定したい場合には、このイベントにより実行されるダイアグラムを車をスタート位置に移動するように定義をして下さい。

このイベントは VRML2.0/97 の TimeSensor ノードのイベントから生成されています。

Frame

仮想空間のシミュレーションが更新されるたびに発生するイベントです。もう少し正確に言えば、一秒間に最高 10 回発生する定期的なイベントです。例えば車や飛行機などの仮想空間を連続的に移動するオブジェクト位置を更新する場合に利用して下さい。

このイベントに関連したダイアグラムには、現在の経過フレーム数を出力するモジュールが生成時に追加されています。経過フレーム数は 0 から開始され、フレームイベントが発生するたびに 1 増加します。



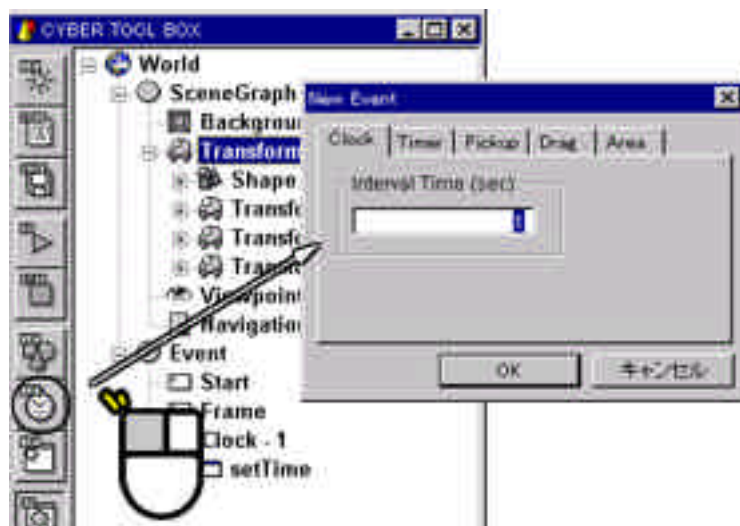
このイベントは VRML2.0/97 の TimeSensor ノードのイベントから生成されています。

ユーザー定義イベント

ユーザー定義イベントは、ユーザー自身によりパラメータを設定し追加されるイベントです。多くのイベントが仮想空間でのユーザーの操作や移動によって、インタラクティブに発生します。ユーザーが定義できるイベントは、以下の5種類です。

イベントタイプ	パラメータ	概要
Clock	インターバル時間	Frame イベントより詳細なインターバルイベントを定義します
Timer	タイマー時間	シミュレーション開始から、指定した経過時間に発生するイベントを定義します
PickUp	対象オブジェクト	オブジェクトをマウスでクリックしたときに発生するイベントを定義します
Drag	対象オブジェクト	オブジェクトをマウスでドラッグしたときに発生するイベントを定義します
Area	中心位置、範囲	ユーザーがある空間範囲に移動した場合に発生するイベントを定義します

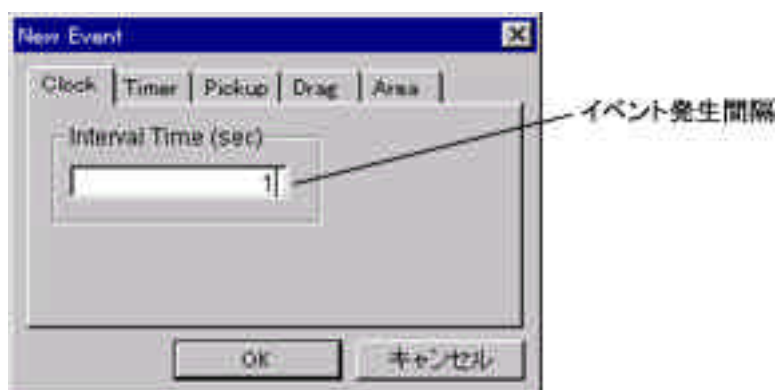
ユーザー定義イベントの追加は、ワールドウィンドウの「新規イベント追加」ボタンで行います。追加したいイベントの種類を、表示されるダイアログのタブから選択し、適切なパラメータを入力または選択して新規イベントを生成して下さい。



ユーザーで定義したイベントは、イベントツリーにある項目をマウス左ボタンでダブルクリックして表示されるダイアログで、そのパラメータの変更ができます。

Clock

Frame イベントより自由度のある、定期的に発生するイベントです。イベントを追加するダイアログでは、イベントの発生間隔を秒単位で整数値を指定して下さい。例えば、2 秒を指定すると10 秒間に5 回発生するイベントを定義したことになります。Frame イベントは、一秒間に 10 回、つまり0.1 秒間隔のイベントです。



このイベントもFrame イベントと同様に、現在のイベント実行回数を出力するモジュールが生成時に追加されています。イベント実行回数は 0 から開始され、フレームイベントが発生するたびに 1 増加します。このイベントは VRML2.0/97 の TimeSensor ノードのイベントから生成されています。

Timer

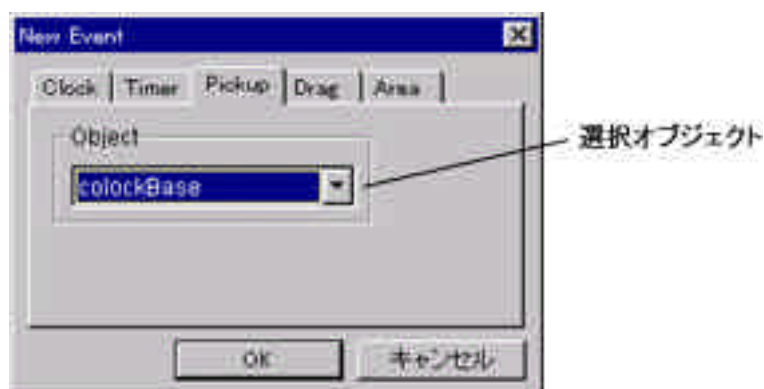
シミュレーションが開始されてから、指定された経過時間で発生するイベントです。ダイアログではイベントが発生するシミュレーション開始時からの時刻を秒単位で整数値を指定して下さい。例えば、10 秒を指定すると シミュレーション開始から 10 秒後に発生するイベントを定義したことになります。



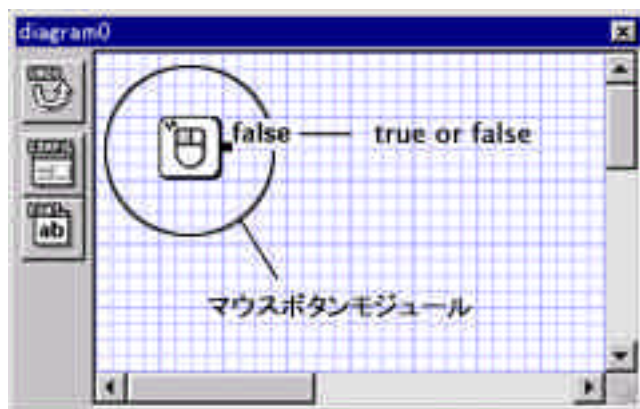
このイベントには、標準で追加されるモジュールはありません。このイベントは VRML2.0/97 の TimeSensor ノードのイベントから生成されています。

PickUp

オブジェクトがマウス左ボタンクリックで選択された場合と、放された場合に発生するイベントです。ダイアログでは選択するオブジェクトをリストから選択して下さい。このオブジェクトは Transform ノードを対象にしており、仮想空間内の名前がつけられた Transform ノードのみが選択リストに表示されます。



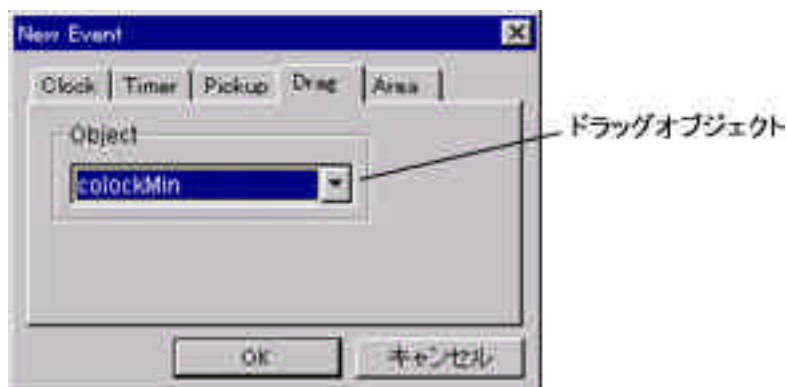
このイベントに関連したダイアグラムには、マウスのクリック状態を示すモジュールが生成時に追加されています。このモジュールは、オブジェクトをマウスでクリックした場合に true、放された場合に false を、イベント発生と同時に出力します。



このイベントは VRML の TouchSensor ノードのイベントから生成されています。

Drag

オブジェクトがマウスでドラッグされた時に発生するイベントです。ダイアログではドラッグするオブジェクトをリストから選択して下さい。このオブジェクトは Transform ノードを対象にしており、仮想空間内の名前がつけられた Transform ノードのみが選択リストに表示されます。



このイベントに関連したダイアグラムには、マウスのドラッグ位置を示すモジュールが生成時に追加されています。このモジュールは、オブジェクトローカル座標系でのドラッグ位置を出力します。



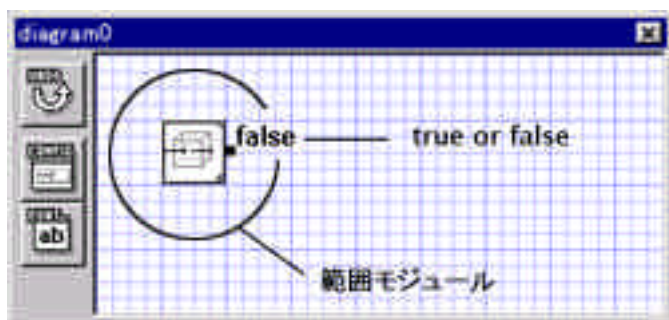
このイベントは VRML の PlaneSensor ノードイベントから生成されています。

Area

ユーザーが、ある範囲内に移動した場合に発生するイベントです。ユーザーが設定範囲内に移動した場合と、設定範囲外に移動した場合にイベントが発生します。ダイアログでは、この範囲枠を示す中心位置とサイズを整数値で指定して下さい。



このイベントに関連したダイアグラムには、ユーザーの範囲移動状態を示すモジュールが生成時に追加されています。このモジュールは、ユーザーが設定範囲内に入った場合に true、設定範囲外に移動した場合に false が出力します。

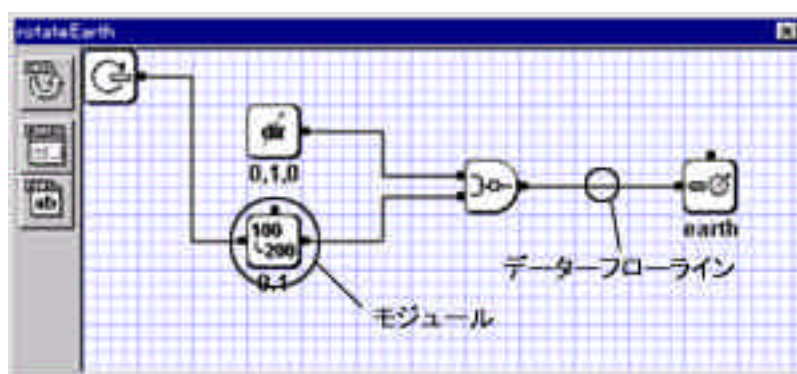


このイベントは VRML の ProximitySensor ノードイベントから生成されています。

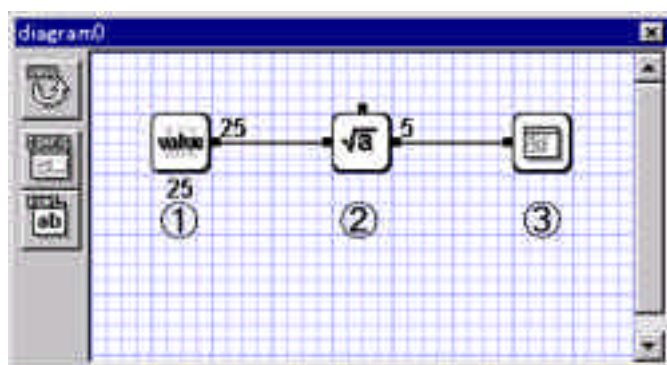
ダイアグラム

ダイアグラムは、仮想空間動作定義の中心になる部分です。ダイアグラムでは、モジュールを線で接続して仮想空間動作を定義していく、データフロー型のビジュアル言語にてプログラミングを行います。

ダイアグラムは、アイコンとして表示されているモジュールと、モジュール同士を接続しているデータフローラインから構成されます。ダイアグラムはイベントに関連づけられて生成されており、そのイベントが発生するとダイアグラムが実行されます。



ダイアグラムでは、データフローラインの上位から順番に、モジュール内部の処理が実行され、接続されている下位モジュールにデータが伝達されていきます。以下にその例を示します。

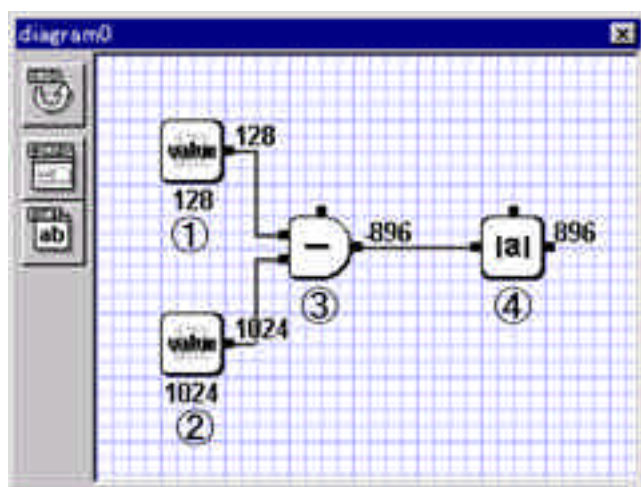


上記ダイアグラムは3つのモジュールと、その間を結ぶデータフローラインから構成される、以下に示す1つのデータフローが定義されています。

このダイアグラムでは、モジュールがこのデータフローでの最上位のモジュールになり、モジュールはモジュールより先に実行されます。モジュールが上位、モジュールが下位のデータフロー関係

になります。モジュールの演算処理が終了し、その結果である出力ノードの値が、モジュールのモジュールの入力ノードへ伝達されていきます。同様に、モジュールはより先に実行されます。つまりデータフローの順序に従って、連続的にモジュールが実行され、その結果であるデータがデータフローラインに沿って伝達されることによって、ダイアグラムの処理が行われます。

ダイアグラムでは、一般的に複数のデータフローを定義することにより複雑な動きをプログラミングしていきます。以下にこのような複数のデータフローが定義されたダイアグラムでの、若干の注意を示します。



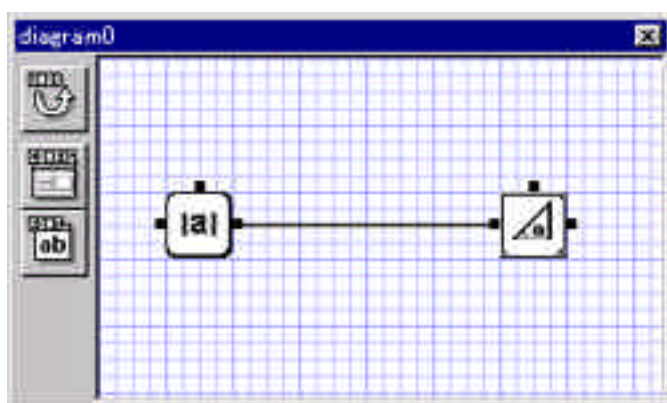
このダイアグラムは5つのモジュールとデータフローラインから構成されており、以下に示す二つのデータフローが定義されています。

- -
- -

この場合にも、各データフロー内でのモジュールの実行順序は保証されていますが、厳密には2つのデータフローがどちらが先に実行されるかは不定です。ただし、入力ノードが複数あるモジュールは、ノードへのデータ伝達がどちらかが先であっても、結果が一緒になるような設計になっています。そのため、一般的にはどちらのデータフローが先に実行されても結果は一緒であり、通常は気にする必要はありません。CTBでのデータフロー定義は最終的にVRMLのROUTE定義に変換されていますが、VRMLブラウザによっては、このデータフロー、つまりROUTE定義の処理が特殊な場合も予想されます。この場合には、CTBで作成したアプリケーションが期待通りの動作をしない可能性があります。詳細は互換性の章を参照して下さい。

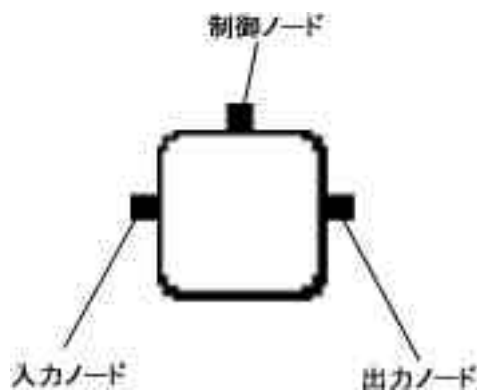
またモジュールには自立型とそうでないものがあります。自立型のモジュールとは、入力ノードが1つもなく、そのモジュール単体で出力ノードにデータが出力されるものです。自立型ではないモジュールとは、

入力ノードを1つ以上もち、入力ノードから入力されるデータから内部演算を行い、その結果を出力ノードに出力するものです。もしデータフロー最上位のノードが、自立型ではないモジュールである場合には注意が必要です。ダイアグラム内のモジュール実行は、データフローの最上位ノードから開始されますが、最上位モジュールが自立型ではなく、そのモジュールの入力ノードにデータが入力されていない、つまりはデータフローラインが接続されていない場合には、そのデータフローは実行されません。結果的に、データフローの最上位モジュールが自立型ではなく、入力ノードにデータフローラインが接続されていないと、このデータフローは実行されないことに注意して下さい。以下に、定義はされているが実行されないデータフロー例を示します。



モジュール概要

モジュールは、仮想空間動作を定義する最小の単位で、属するダイアグラムが実行されるイベントが発生した時点で処理が行われます。モジュールには以下の図に示す 3 種類のノードがあり、各ノード間をデータフローラインで接続することによりデータの流れを定義します。



入力ノードは、他のモジュールからのデータを入力するノードで、出力ノードは、入力ノードからのデータなどを用いてモジュール内部で演算した結果を出力するノードです。入力ノードと出力ノードの個数は、モジュールによって異なります。

例えば、以下のモジュールは、入力ノードを2つ、出力ノードを1つもち、2つの入力ノードの値を加算して、出力ノードに出力するものです。

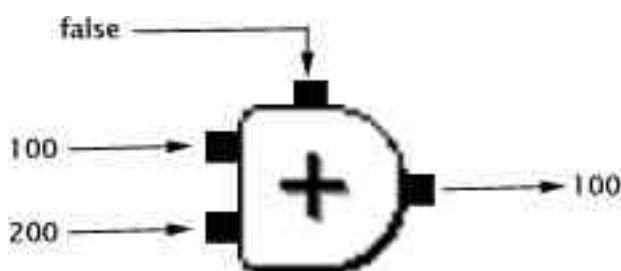


モジュールには入力ノードが1つもないものがあり、モジュールに設定されている値や内部演算処理結果を出力ノードに出力します。このようなモジュールを、入力ノードによるデータを必要としないことから、自立型のモジュールと呼びます。ダイアグラムでデータフローを定義するには、自立型のモジュールが最上位のモジュールである必要があります。最上位のモジュールが自立型でない場合には、そのデータフローは実行されない点に注意して下さい。

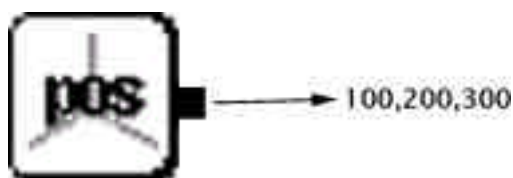
制御ノードは、一般的にモジュールの演算処理を実行するかどうかの設定を行います。制御ノードに他のモジュールの出力ノードからのデータフローラインが接続されていないければ、モジュールの内部演算処理は通常通り実行されます。データフローラインが接続されている場合にも、その入力されるデータが"TRUE"または"true"である場合には、同様に内部演算処理は実行されます。入力されるデータ

が"FALSE"または"false"である場合には、モジュールの演算処理は実行されず、一般的に、モジュール左最上部の入力ノード(第1入力ノード)のデータ文字列がそのまま演算結果として出力ノードへ出力されます。この制御ノードによるモジュールの処理方法は、モジュールにより異なります。制御ノードは、モジュールによってない場合もあります。

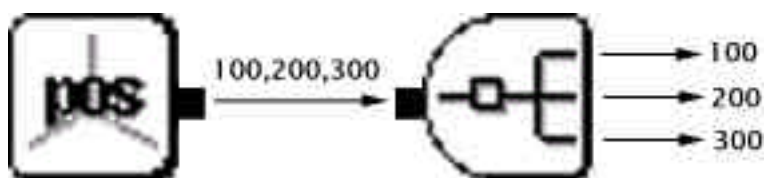
例えば、以下のモジュールは、入力ノードを2つ、出力ノードを1つもち、通常は2つの入力ノードの値を加算して出力ノードに出力しますが、制御ノードに"TRUE"または"true"以外のデータが、他モジュールから入力された場合には、2つの入力ノードの値は加算されず、左上部の入力ノードの値がそのまま出力ノードに出力されます。



各ノードのデータは、すべて文字列として取り扱われており、数値演算の必要がある場合には、モジュール内部で可能な限り数値に変換した後に、各種演算が行われます。複数のデータはカンマ(,)にて連結することにより表現しています。例えば以下のモジュールは、仮想空間内の位置を出力するものですが、その出力されるデータはカンマで位置(100, 200, 300)をあらわすカンマで連結された文字列です。



CTB では、このようにカンマで連結された文字列の取り扱いを容易にするように、複数の値をカンマで連結して1つの文字列にするモジュールや、カンマで連結された文字列を分解して値を取り出すモジュールが幾つか用意されています。例えば上記のモジュールのように3つの値がカンマで連結されている文字列は、以下の図の右側にあるモジュールで3つの値に分解できます。



すべてのデータを文字列で処理する理由は、各モジュール間のノード接続を容易に処理するためです。

VRML では、各種のデータ型が存在していますが、各データ型間のキャストはできないような仕様になっています。この仕様を緩和するために、モジュールの内部にて文字列から各種 VRML データ型への変換を行うことにより、一つの型データに統一して処理しています。

シミュレーション実行中には、モジュールの各出力ノードには出力データが表示されます。このデータが数値の場合には小数点以下はの 2 桁で丸められたものが表示される点に注意して下さい。モジュールの演算精度は通常の単精度または倍精度で行われています。一部のモジュール下部に表示されるモジュールデータも、同様に丸められたものが表示されます。

モジュールは、VRML2.0/97 仕様に準拠した JSAI(Java script authoring interface)のクラスファイルとして記述されています。そのため、CTB で作成したアプリケーションが動作する VRML ブラウザの条件としては、最低限 JSAI に対応していることが必要です。しかしながら、JSAI に対応しているブラウザでも、高速化の目的などで CTB アプリケーションが動作しない場合もありますので注意して下さい。詳細については、互換性の章を参照して下さい。

モジュール詳細

モジュールは、以下に示すクラスに分類されてモジュールウインドウに表示されています。下記分類にあてはまらないモジュールについては、Misc クラスに分類されています。

クラス	モジュール処理概要
String	文字列の生成、連結、分割を行います。
Numeric	視点の位置/方向などを設定/取得を行います。
Math	$\sin(x)$ や $\cos(x)$ などの数学演算を行います。
Filter	スケーリングや範囲制限を行います。
Geometry	ベクトル関連の演算を行います。
Boolean	一致や比較を行い、真偽値を出力します。
Object	オブジェクトの位置/方向などを設定/取得を行います。
Material	マテリアルの色情報の設定/取得を行います。
Light	光源の位置/輝度などの設定/取得を行います。
View	視点の位置/方向などを設定/取得を行います。
Interpolator	Interpolator ノードの fraction 値を設定します。

String クラス

String クラスのモジュールは、数値や位置ベクトル、方向ベクトルの文字列を生成したり文字列の連結や分割を行います。例えば方向ベクトル文字列である”1,0,1”を、”1”, “0”, “1”と各軸別に分割したり方向ベクトル文字列”1,1,1”と角度文字列”3.14”を連結して、方角ベクトル文字列”1,1,1,3.14”を生成したりします。



Value

入力ノード	-
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = ダイアログ設定値
制御ノード	-
設定ダイアログ	O

ダイアログで設定された文字列を出力します。ダイアログで設定した値が数値である場合にも、文字列に変換されて出力されます。



Position

入力ノード	-
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = ダイアログ設定位置
制御ノード	-
設定ダイアログ	O

ダイアログで設定された3次元位置を文字列に変換して出力します。同様の文字列は Value モジュールでも生成できます。



Direction

入力ノード	-
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = ダイアログ設定ベクトル
制御ノード	-
設定ダイアログ	O

ダイアログで設定された3次元ベクトルを文字列に変換して出力します。同様の文字列は Value モジュールでも生成できます。



Rotation

入力ノード	-
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = ダイアログ設定方角
制御ノード	-
設定ダイアログ	O

ダイアログで設定された3次元方角を文字列に変換して出力します。同様の文字列は Value モジュールでも生成できます。

**Bool**

入力ノード	-
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = ダイアログ設定真偽値
制御ノード	-
設定ダイアログ	O

ダイアログで設定された真偽値(true または false)を文字列に変換して出力します。同様の文字列は Value モジュールでも生成できます。

**Color**

入力ノード	-
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = ダイアログ設定色
制御ノード	-
設定ダイアログ	O

ダイアログで設定された色を文字列に変換して出力します。同様の文字列は Value モジュールでも生成できます。

**PI**

入力ノード	-
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = PI
制御ノード	-
設定ダイアログ	-

円周率 PI を示す文字列を出力します。



E

入力ノード	-
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = e
制御ノード	-
設定ダイアログ	-

自然対数基数 e を示す文字列を出力します。



Divide2Values

入力ノード	InValue1,InValue2
出力ノード	OutValue1 OutValue2
対象ノード	-
出力結果	OutValue1 = InValue1 OutValue2 = InValue2
制御ノード	-
設定ダイアログ	-

カンマ(,)で連結された文字列を、2 つに分割します。例えば入力データ文字列が”abcd,1.0”である場合には、OutValue1 に”abcd”、OutValue2 に 1.0 が出力されます。入力データ文字列にカンマがない場合には分割は行われず、OutValue1 とOutValue2 には空文字が出力されます。文字列にカンマが複数存在する場合には、最初のカンマを分割位置とします。例えば、入力データ文字列が”abcd,1.0,efgh”である場合には、OutValue1 に”abcd”、OutValue2 に”1.0,efgh”が出力されます。



Divide3Values

入力ノード	InValue1,InValue2,InValue3
出力ノード	OutValue1 OutValue2 OutValue3
対象ノード	-
出力結果	OutValue1 = InValue1 OutValue2 = InValue2 OutValue3 = InValue3
制御ノード	-
設定ダイアログ	-

カンマ(,)で連結された文字列を、3 つに分割します。例えば入力データ文字列

が“abcd,1.0,efgh”である場合には、OutValue1 に“abcd”、OutValue2 に 1.0、OutValue3 に“efgh”が出力されます。入力データ文字列にカンマがない場合には分割は行われず、OutValue1、OutValue2、OutValue3 には空文字が出力されます。文字列にカンマが複数存在する場合には、最初の 2 つのカンマを分割位置とします。例えば、入力データ文字列が“abcd,1.0,efgh,2.0”である場合には、OutValue1 に“abcd”、OutValue2 に“1.0”、OutValue3 に“efgh,2.0”が出力されます。



Divide4Values

入力ノード	InValue1,InValue2,InValue3,InValue4
出力ノード	OutValue1 OutValue2 OutValue3 OutValue4
対象ノード	-
出力結果	OutValue1 = InValue1 OutValue2 = InValue2 OutValue3 = InValue3 OutValue4 = InValue4
制御ノード	-
設定ダイアログ	-

カンマ(',')で連結された文字列を、4 つに分割します。例えば入力データ文字列が“abcd,1.0,efgh,2.0”である場合には、OutValue1 に“abcd”、OutValue2 に 1.0、OutValue3 に“efgh”、OutValue4 に“2.0”が出力されます。入力データ文字列にカンマがない場合には分割は行われず、OutValue1、OutValue2、OutValue3、OutValue4 には空文字が出力されます。文字列にカンマが複数存在する場合には、最初の 3 つのカンマを分割位置とします。例えば、入力データ文字列が“abcd,1.0,efgh,2.0,ijkl”である場合には、OutValue1 に“abcd”、OutValue2 に“1.0”、OutValue3 に“efgh”、OutValue4 には“2.0,ijkl”が出力されます。



Merge2Values

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1,InValue2
制御ノード	-
設定ダイアログ	-

InValue1、InValue2 をカンマで連結しOutValue に出力します。例えば InValue1 の入力データ文字列が“x”、InValue2 が“y”である場合には、OutValue には“x,y”が出力されます。



Merge2Values

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1,InValue2
制御ノード	-
設定ダイアログ	-

InValue1、InValue2 をカンマで連結しOutValue に出力します。例えば InValue1 の入力データ文字列が"x"、InValue2 が"y"である場合には、OutValue には"x,y"が出力されます。



Merge3Values

入力ノード	InValue1 InValue2 InValue3
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1,InValue2, InValue3
制御ノード	-
設定ダイアログ	-

InValue1、InValue2、InValue3 をカンマで連結しOutValue に出力します。例えば InValue1 の入力データ文字列が"x"、InValue2 が"y"、InValue3 が"z"である場合には、OutValue には"x,y,z"が出力されます。



Merge4Values

入力ノード	InValue1 InValue2 InValue3 InValue4
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1,InValue2, InValue3,InValue4
制御ノード	-
設定ダイアログ	-

InValue1、InValue2、InValue3、InValue4 をカンマで連結し OutValue に出力します。例えば InValue1 の入力データ文字列が"x"、InValue2 が"y"、InValue3 が"z"、InValue4 が"angle"である場合には、OutValue には"x,y,z,angle"が出力されます。

**Selector**

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1 または InValue2
制御ノード	
設定ダイアログ	-

制御ノードにデータフローラインが接続されていない場合には、OutValue には InValue1 が出力されます。データフローラインが接続されていて制御ノードに”TRUE”または”true”の文字列が入力されている場合には OutValue には InValue1 が出力され、それ以外の場合には OutValue には OutValue2 が出力されます。

**SetValue**

入力ノード	InValue
出力ノード	-
対象ノード	-
出力結果	-
制御ノード	
設定ダイアログ	

ダイアログで入力または選択したグローバル変数に、InValue の値を保存します。制御ノードにデータフローラインが接続されていない場合には、InValue の値は保存されます。データフローラインが接続されていて制御ノードに”TRUE”または”true”の文字列が入力されている場合にも InValue の値は保存され、それ以外の場合には保存されません。このモジュールで保存した値は、GetValue モジュールで取得できます。

**SetArrayValue**

入力ノード	InValue
出力ノード	-
対象ノード	-
出力結果	-
制御ノード	
設定ダイアログ	

ダイアログで入力または選択したグローバル変数配列に、InValue の値を保存します。制御ノード

には、保存したい配列位置を示すの 1 以上の番号を入力します。制御 ノードに 1 以上の値が入力されている場合には、InValue の値はグローバル変数配列の指定された位置に保存され、それ以外の場合には保存されません。このモジュールで保存した値は、GetArrayValue モジュールで取得できます。



GetValue

入力 ノード	-
出力 ノード	OutValue
対象 ノード	-
出力結果	-
制御 ノード	
設定ダイアログ	

ダイアログで選択したグローバル変数の値を、OutValue に出力します。



GetArrayValue

入力 ノード	-
出力 ノード	OutValue
対象 ノード	-
出力結果	-
制御 ノード	
設定ダイアログ	

ダイアログで入力または選択したグローバル変数配列の値を、制御 ノードに入力された配列番号に従って OutValue に出力します制御 ノードには、保存したい配列位置を示すの 1 以上の番号を入力してください。

Numeric クラス

Numeric クラスのモジュールは、2 つの入力 ノードに入力された文字列を数値に変換して、2 つの数値を盛り込んで加算/減算などの演算を行い、その結果を出力 ノードに出力します。

このクラスの全モジュールは制御 ノードをもちます。制御 ノードにデータフローラインが接続されていて、入力データ文字列が"FALSE"または"false"である場合には、内部演算は行われず、モジュール左側上部の入力 ノードの文字列がそのまま出力されます。



入力 ノード	InValue1 InValue2
出力 ノード	OutValue
対象 ノード	-
出力結果	OutValue = InValue1 + InValue2 (または InValue1)
制御 ノード	O
設定ダイアログ	-

入力された 2 つの文字列を、数値に変換して加算します。カンマ(,)で連結された入力データ文字列も演算の対象となります。ただし演算を行うには、InValue1 と InValue2 の連結された文字列は同じ数である必要があります。

InValue1、InValue2 共に、カンマで連結されていない単独の文字列である場合には、InValue1 と InValue2 を数値に変換し、その加算結果を OutValue に出力します。InValue1 および InValue2 の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。

InValue1、InValue2 共に、カンマで同じ数の文字列が連結されている場合には、その文字列毎に加算を行い、その結果を OutValue に出力します。例えば、InValue1 が"1,2,3"、InValue2 が"10,20,30"である場合には、OutValue には"11,22,33"が出力されます。InValue1 および InValue2 の文字列の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。



Minus

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1 – InValue2 (または InValue1)
制御ノード	-
設定ダイアログ	-

入力された 2 つの文字列を、数値に変換して減算します。カンマ(,)で連結された入力データ文字列も演算の対象となります。ただし演算を行うには、InValue1 と InValue2 の連結された文字列は同じ数である必要があります。

InValue1、InValue2 共に、カンマで連結されていない単独の文字列である場合には、InValue1 と InValue2 を数値に変換し、InValue1 から InValue2 の値を減算した結果を OutValue に出力します。InValue1 および InValue2 の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。

InValue1、InValue2 共に、カンマで同じ数の文字列が連結されている形式である場合には、その文字列毎に減算を行い、その結果を OutValue に出力します。例えば、InValue1 が”11,22,33”、InValue2 が”10,20,30”である場合には、OutValue には”1,2,3”が出力されます。InValue1 および InValue2 の文字列の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。



Multi

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1 * InValue2 (または InValue1)
制御ノード	-
設定ダイアログ	-

入力された 2 つの文字列を、数値に変換して掛け算を行います。

一方の入力文字列が 4 つの文字列がカンマで連結されいて、もう一方の文字列が 3 つの文字列が連結されている場合には、前者の入力文字列を方角データ、後者を 3 次元ベクトルデータとして数値に変換し、回転演算を行います。例えば InValue1 が”0,1,0,1.54”、InValue2 が”0,0,1”の場合

合には、Z 軸方向ベクトルを Y 軸に対して 90 度回転させる演算を行う結果となり OutValue に”は 1,0,0”が出力されます。

一方の入力文字列が 3 つの文字列がカンマで連結されいて、もう一方の文字列がカンマで連結されていない文字列である場合には前者を 3 次元ベクトルデータとして、後者を倍率データとしてスケーリング演算を行います。例えば InValue1 が”1,2,3”、InValue2 が”0.5”の場合には、OutValue に”は 0.5,1,1.5”が出力されます。

両方の入力文字列がカンマで連結されていない場合には、InValue1 とInValue2 を数値に変換し、InValue1 に InValue2 を掛け合わせた結果を OutValue に出力します。

これらの演算で、InValue1 および InValue2 の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。



Divide

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1 / InValue2 (または InValue1)
制御ノード	-
設定ダイアログ	-

入力された 2 つの文字列を、数値に変換して割り算を行います。

一方の入力文字列が 3 つの文字列がカンマで連結されいて、もう一方の文字列がカンマで連結されていない文字列である場合には前者を 3 次元ベクトルデータとして、後者を倍率データとしてスケーリング演算を行います。例えば InValue1 が”1,2,3”、InValue2 が”0.5”の場合には、OutValue に”は 0.5,1,1.5”が出力されます。

両方の入力文字列がカンマで連結されていない場合には、InValue1 とInValue2 を数値に変換し、InValue1 を InValue2 で割った結果を OutValue に出力します。

これらの演算で、InValue1 および InValue2 の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。



Mod

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1 / InValue2 (または InValue1)
制御ノード	-
設定ダイアログ	-

入力された2つの文字列を、数値に変換して割り算を行います。カンマ(,)で連結された入力データ文字列は演算の対象となりません。

InValue1 と InValue2 を数値に変換し、InValue1 を InValue2 で割った余りを OutValue に出力します。InValue1 および InValue2 の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。



And

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1 & InValue2 (or InValue1)
制御ノード	0
設定ダイアログ	-

入力された2つの文字列を、ビットごとの AND 演算を行います。カンマ(,)で連結された入力データ文字列も演算の対象となります。ただし演算を行うには、InValue1 と InValue2 の連結された文字列は同じ数である必要があります。

InValue1、InValue2 共に、カンマで連結されていない単独の文字列である場合には、InValue1 と InValue2 を数値に変換し、ビットごとの AND 演算結果を OutValue に出力します。InValue1 および InValue2 の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。

InValue1、InValue2 共に、カンマで同じ数の文字列が連結されている場合には、その文字列毎にビットごとの AND 演算を行い、その結果を OutValue に出力します。例えば、InValue1 が”1,0,1,0”、InValue2 が”1,1,0,0”である場合には、OutValue には”1,0,0,1”が出力されます。InValue1 および InValue2 の文字列の何れかが数値に変換できない場合には、OutValue に

NULL 文字が出力されます。



Or

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1 InValue2 (or InValue1)
制御ノード	O
設定ダイアログ	-

入力された 2 つの文字列を、ビットごとの OR 演算を行います。カンマ(,)で連結された入力データ文字列も演算の対象となります。ただし演算を行うには、InValue1 と InValue2 の連結された文字列は同じ数である必要があります。

InValue1、InValue2 共に、カンマで連結されていない単独の文字列である場合には、InValue1 と InValue2 を数値に変換し、ビットごとの OR 演算結果を OutValue に出力します。InValue1 および InValue2 の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。

InValue1、InValue2 共に、カンマで同じ数の文字列が連結されている場合には、その文字列毎にビットごとの OR 演算を行い、その結果を OutValue に出力します。例えば、InValue1 が”1,0,1,0”、InValue2 が”1,1,0,0”である場合には、OutValue には”1,1,1,0”が出力されます。InValue1 および InValue2 の文字列の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。



Xor

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue1 InValue2 (or InValue1)
制御ノード	O
設定ダイアログ	-

入力された 2 つの文字列を、ビットごとの XOR 演算を行います。カンマ(,)で連結された入力データ文字列も演算の対象となります。ただし演算を行うには、InValue1 と InValue2 の連結された文字列は同じ数である必要があります。

InValue1、InValue2 共に、カンマで連結されていない単独の文字列である場合には、InValue1 と InValue2 を数値に変換し、ビットごとの XOR 演算結果を OutValue に出力します。InValue1 および InValue2 の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。

InValue1、InValue2 共に、カンマで同じ数の文字列が連結されている場合には、その文字列毎にビットごとの XOR 演算を行い、その結果を OutValue に出力します。例えば、InValue1 が”1,0,1,0”、InValue2 が”1,1,0,0”である場合には、OutValue には”1,0,0,1”が出力されます。InValue1 および InValue2 の文字列の何れかが数値に変換できない場合には、OutValue に NULL 文字が出力されます。

Math クラス

Math クラスのモジュールは、1つまたは2つの入力ノードに入力された文字列を数値に変換して、絶対値や Sin/Cos などの数値演算を行い、その結果を出力ノードに出力します。

このクラスの全モジュールは制御ノードをもちます。制御ノードにデータフローラインが接続されていて、入力データ文字列が"FALSE"または"false"である場合には、内部演算は行われず、モジュール左側上部の入力ノードの文字列がそのまま出力されます。



Increment

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = \text{InValue} + 1$
制御ノード	O
設定ダイアログ	-

入力された1つの文字列に1を加算した結果を出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その値に1を加算してOutValue に出力します。InValue が数値に変換できない場合には、OutValue にNULL文字が出力されます。例えば InValue が"12"である場合には、OutValue には"13"が出力されます。



Decrement

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = \text{InValue} - 1$
制御ノード	O
設定ダイアログ	-

入力された1つの文字列に1を減算した結果を出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その値に1を減算してOutValue に出力します。InValue が数値に変換できない場合には、OutValue にNULL文字が出力されます。例えば InValue が"12"である場合には、OutValue には"11"が出力されます。



Abs

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = \text{InValue} $
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列の絶対値を出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その絶対値を OutValue に出力します。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。例えば InValue が“-123”である場合には、OutValue には“123”が出力されます。



Negative

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = -\text{InValue}$
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列の符号を反転し出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その符号を反転した値を OutValue に出力します。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。例えば InValue が“-123”である場合には、OutValue には“123”が出力されます。



Pow

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = (\text{InValue1})^{\text{InValue2}}$
制御ノード	O
設定ダイアログ	-

InValue1 を InValue2 乗した結果を OutValue に出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue1 または InValue2 が数値に変換できない場合には、OutValue に NULL 文字が出力されます。例えば InValue1 が“2”、InValue2 が“10”である場合には、

OutValue には”1024”が出力されます。



Sqrt

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = sqrt(InValue)
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列の平方根を出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その平方根を OutValue に出力します。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。例えば InValue が”9”である場合には、OutValue には”3”が出力されます。



Min

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = min(InValue1, InValue2)
制御ノード	O
設定ダイアログ	-

入力された 2 つの文字列の小さいほうを出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue1 と InValue2 を数値に変換し、小さいほうの値を OutValue に出力します。InValue1 と InValue2 が数値に変換できない場合には、OutValue に NULL 文字が出力されます。例えば InValue1 が”10”、InValue2 が”20”である場合には、OutValue には”10”が出力されます。



Max

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = max(InValue1, InValue2)
制御ノード	O
設定ダイアログ	-

入力された 2 つの文字列の大きいほうを出力しますカンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue1 とInValue2 を数値に変換し、大きいほうの値を OutValue に出力します。InValue とInValue2 が数値に変換できない場合には、OutValue に NULL 文字が出力されます。例えば InValue1 が”10”、InValue2 が”20”である場合には、OutValue には”20”が出力されます。



Log

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = log(InValue)
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列の対数を出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その対数を OutValue に出力します。InValue の単位はラジアンです。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。



Exp

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = log(InValue)
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列の指数を出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その指数を OutValue に出力します。InValue の単位はラジアンです。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。

**Sin**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = \sin(\text{InValue})$
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列のサインを出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、そのサインを OutValue に出力します。InValue の単位はラジアンです。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。

**Cos**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = \cos(\text{InValue})$
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列のコサインを出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、そのコサインを OutValue に出力します。InValue の単位はラジアンです。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。

**Tan**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = \tan(\text{InValue})$
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列のタンジェントを出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、そのタンジェントを OutValue に出力します。InValue の単位はラジアンです。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。

**ASin**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = asin(InValue)
制御ノード	O
設定ダイアログ	-

入力された1つの文字列のアークサインを出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、そのアークサインをOutValue に出力します。InValue の単位はラジアンです。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。

**ACos**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = acos(InValue)
制御ノード	O
設定ダイアログ	-

入力された1つの文字列のアークコサインを出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、そのアークコサインをOutValue に出力します。InValue の単位はラジアンです。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。

**ATan**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = atan(InValue)
制御ノード	O
設定ダイアログ	-

入力された1つの文字列のアークタンジェントを出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、そのアークタンジェントを OutValue に出力します。InValue の単位はラジアンです。InValue が数値に変換できない場合には、

OutValue に NULL 文字が出力されます。



Degree2Radiun

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = \text{InValue} / 180 * \text{PI}$
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列を、度数角から弧度角に変換します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その値を度数角から弧度角に変換します。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。



Radiun2Degree

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = \text{InValue} / \text{PI} * 180$
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列を、弧度角から度数角に変換します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その値を弧度角から度数角に変換します。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます。

Filter クラス

Filter クラスのモジュールは、入力された 1 つの文字列から数値に変換し、モジュールに設定されているフィルター演算を行い、その結果を出力します。このクラスの全てのモジュールは制御 ノードをもち、モジュールによってはフィルター値を設定するダイアログが用意されています。

このクラスの全モジュールは制御 ノードをもちます。制御 ノードにデータフローラインが接続されていて、入力データ文字列が"FALSE"または"false"である場合にはフィルター演算は行われず、入力 ノードの文字列がそのまま出力されます。



Scale

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = InValue x Scaling value
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列をスケールリングして出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その数値をダイアログで設定されている倍率でスケールリングして OutValue に出力します。例えば InValue が"2"、ダイアログによる設定値が"100"である場合には、OutValue には"200"が出力されます。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます



Ceil

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = ceil(InValue)
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列を切り上げた値を出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その数値以上の整数のうち、最も小さい整数値 OutValue に出力します。例えば InValue が"12.3"である場合には、OutValue には"13"が出力されます。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます

**Floor**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = floor(InValue)
制御ノード	O
設定ダイアログ	-

入力された 1 つの文字列を切り捨てた値を出力します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その数値以下の整数のうち、最も大きい整数値 OutValue に出力します。例えば InValue が”12.3”である場合には、OutValue には”12”が出力されます。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます

**High**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = high(InValue)
制御ノード	O
設定ダイアログ	O

入力された 1 つの文字列が、ダイアログで設定されている上限値以下であるか確認します。カンマ(,)で連結された入力データ文字列は演算の対象となりません。InValue を数値に変換し、その数値が設定されている上限値以下であれば InValue をそのまま OutValue に出力し、上限値以上であれば OutValue に上限値を出力します。例えば InValue が”12.3”で、モジュールに設定されている上限値が”10”である場合には、OutValue には”10”が出力されます。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます

**Low**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = low(InValue)
制御ノード	O
設定ダイアログ	O

入力された 1 つの文字列がダイアログで設定されている下限値以下であるか確認します。カンマ(,)で連結された入力データ文字列は演算の対象なりません。InValue を数値に変換し、その数値が設定されている下限値以上であれば InValue をそのまま OutValue に出力し、下限値以下であれば OutValue に下限値を出力します。例えば InValue が”12.3”で、モジュールに設定されている下限値が”20”である場合には、OutValue には”20”が出力されます。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます



Range

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = range(InValue)
制御ノード	O
設定ダイアログ	O

入力された 1 つの文字列がダイアログで設定されている範囲内にあるか確認します。カンマ(,)で連結された入力データ文字列は演算の対象なりません。InValue を数値に変換し、その数値が設定されている範囲内であれば InValue をそのまま OutValue に出力し、範囲外で範囲下限値以下であれば OutValue に下限値、範囲外で範囲上限値以上であれば OutValue に上限値を出力します。例えば InValue が”12.3”で、設定されている範囲が”5 ~ 10”である場合には、OutValue には”10”が出力されます。InValue が数値に変換できない場合には、OutValue に NULL 文字が出力されます



ScalarInterpolator

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = (to value - from value) * InValue + from value
制御ノード	O
設定ダイアログ	O

入力された 1 つの文字列からダイアログで設定されている 2 つの数値の補間を行い、その結果を出力します。InValue には範囲 0.0 ~ 1.0 の数値文字列を入力する必要があり、ダイアログに設定されている From 項目の数値を 0.0、To 項目を 1.0 として補間された結果を OutValue に出力します。例えば From 項目が”0”、To 項目が”100”、InValue が 0.5 であった場合には、OutValue に”50”が出力されます。

**Position2DInterpolator**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = (\text{to value} - \text{from value}) * \text{InValue} + \text{from value}$
制御ノード	O
設定ダイアログ	O

入力された 1 つの文字列からダイアログで設定されている 2 つの 2 次元座標値の補間を行い、その結果を出力します。InValue には範囲 0.0 ~ 1.0 の数値文字列を入力する必要があり、ダイアログに設定されている From 項目の数値を 0.0、To 項目を 1.0 として補間された結果を OutValue に出力します。例えば From 項目が(0.0, 0.0)、To 項目が(1.0, 1.0)、InValue が 0.5 であった場合には、OutValue に”0.5, 0.5”が出力されます。

**Position3DInterpolator**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = (\text{to value} - \text{from value}) * \text{InValue} + \text{from value}$
制御ノード	O
設定ダイアログ	O

入力された 1 つの文字列からダイアログで設定されている 2 つの 3 次元座標値の補間を行い、その結果を出力します。InValue には範囲 0.0 ~ 1.0 の数値文字列を入力する必要があり、ダイアログに設定されている From 項目の数値を 0.0、To 項目を 1.0 として補間された結果を OutValue に出力します。例えば From 項目が(0.0, 0.0, 0.0)、To 項目が(1.0, 1.0, 1.0)、InValue が 0.5 であった場合には、OutValue に”0.5, 0.5, 0.5”が出力されます。

**OrientationInterpolator**

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	$\text{OutValue} = (\text{to value} - \text{from value}) * \text{InValue} + \text{from value}$
制御ノード	O
設定ダイアログ	O

入力された 1 つの文字列からダイアログで設定されている 2 つの方角値の補間を行い、その結果を出力します。InValue には範囲 0.0 ~ 1.0 の数値文字列を入力する必要があり、ダイアログに設定

されているFrom 項目の数値を 0.0、To 項目を 1.0 として補間された結果を OutValue に出力します。例えば From 項目が(0.0, 0.0, 1.0, 0.0)、To 項目が(0.0, 0.0, 1.0, 3.14)、InValue が 0.5 であった場合には、OutValue に”0.0, 0.0, 1.0, 1.57”が出力されます。

Geometry クラス

Geometry クラスのモジュールは、入力された文字列のベクトル長の取得、ベクトルの正規化、平面ベクトルの取得などを行います。



Normalize

入力ノード	InVector (x, y, z)
出力ノード	OutVector (x, y, z)
対象ノード	-
出力結果	OutVector = normalize(InVector)
制御ノード	-
設定ダイアログ	-

入力された InVector ベクトルを正規化して OutVector に出力します。InVector がベクトル数値に変換できない場合には NULL が出力されます。



Inverse

入力ノード	InVector (x, y, z)
出力ノード	OutVector (x, y, z)
対象ノード	-
出力結果	OutVector = InVector
制御ノード	-
設定ダイアログ	-

入力された InVector ベクトルを反転して OutVector に出力します。nVector がベクトル数値に変換できない場合には NULL が出力されます。



GetDistance

入力ノード	InPosition1 (x, y, z) InPosition2 (x, y, z)
出力ノード	OutVector
対象ノード	-
出力結果	OutValue = 座標間の距離
制御ノード	-
設定ダイアログ	-

入力された2つの InPosition1, InPosition2 座標間の距離を OutVector に出力します。InPosition1, InPosition2 の何れかがベクトル数値に変換できない場合には NULL が出力されます。



GetVector

入力ノード	InPosition1 (x, y, z) InPosition2 (x, y, z)
出力ノード	OutVector (x, y, z)
対象ノード	-
出力結果	OutVector = InPosition2 - InPosition1
制御ノード	-
設定ダイアログ	-

入力された InPosition1 座標から InPosition2 座標へのベクトルを OutVector に出力します。InPosition1, InPosition2 の何れかがベクトル数値に変換できない場合には NULL が出力されます。



GetLength

入力ノード	InVector (x, y, z)
出力ノード	OutLength
対象ノード	-
出力結果	OutLength = InVector ベクトルの長さ
制御ノード	-
設定ダイアログ	-

入力された InVector ベクトルの長さを OutLength 出力します。InVector がベクトル数値に変換できない場合には NULL が出力されます。



GetDot

入力ノード	InVector1 (x, y, z) InVector2 (x, y, z)
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = (InVector1 · InVector2)
制御ノード	-
設定ダイアログ	-

入力された InVector1, InVector2 の内積を OutValue に出力します。InVector1, InVector2 の何れかがベクトル数値に変換できない場合には NULL が出力されます。

**GetAngle**

入力ノード	InVector1 (x, y, z) InVector2 (x, y, z)
出力ノード	OutAngle
対象ノード	-
出力結果	OutAngle = ベクトルの成す角
制御ノード	-
設定ダイアログ	-

入力された2つの InVector1, InVector2 ベクトルの成す角度を OutAngle に出力します。

**GetCross**

入力ノード	InVector1(x, y, z) InVector2(x, y, z)
出力ノード	OutVector
対象ノード	-
出力結果	OutVector = 直行ベクトル
制御ノード	-
設定ダイアログ	-

入力された2つの InVector1, InVector2 ベクトルに直行するベクトルを OutVector に出力します。
InVector1, InVector2 の何れかがベクトル数値に変換できない場合には NULL が出力されます。

**Rotate**

入力ノード	InVector(x, y, z) InRotation(x, y, z, angle)
出力ノード	OutVector
対象ノード	-
出力結果	OutVector = 回転したベクトル
制御ノード	-
設定ダイアログ	-

入力された InVector のベクトルを、InRotation の角度で回転させ、その結果を OutVector に出力します。InVector あるいは InRotation が正常な数値に変換できない場合には NULL が出力されます。

Boolean クラス

Boolean クラスのモジュールは、入力された 1 つまたは 2 つの文字列から比較などの論理演算を行い、その結果を出力します。出力される値は文字列を示す真偽値で、“true”または“false”です。

一般的に Boolean クラスのモジュールは、2 つの入力文字列の比較を行う場合には、最初に文字列としての比較を行います。その結果が等しい場合には“true”が出力され、等しくない場合には 2 つの入力文字列が数値に変換し再度比較を行います。これらの文字列が数値に変換でき、数値的に等しい場合に“true”が出力されます。例えば入力文字列が“1.0”と“1”であった場合には、文字列としては異なりますが、数値的には等しいと判断され“true”が出力されます。



Equal

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = (InValue1 == InValue2)
制御ノード	-
設定ダイアログ	-

入力された 2 つの文字列を比較し、それが等しいかどうかの真偽値を出力します。InValue1 と InValue2 が等しい場合には OutValue に“true”が、等しくない場合には“false”が出力されます。カンマ(,)で連結された入力データ文字列も比較の対象にはなりません。



NotEqual

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = (InValue1 != InValue2)
制御ノード	-
設定ダイアログ	-

入力された 2 つの文字列を比較し、それが等しくないかどうかの真偽値を出力します。InValue1 と InValue2 が等しくない場合には OutValue に“true”が、等しい場合には“false”が出力されます。カンマ(,)で連結された入力データ文字列も比較の対象にはなりません。

**Greater**

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = (InValue1 > InValue2)
制御ノード	-
設定ダイアログ	-

入力された2つの文字列を数値に変換し、その大きさを比較します。InValue1 とInValue2 を数値に変換し、InValue1 がInValue2 より大きければ”true”を出力し、そうでなければ”false”を出力します。カンマ(,)で連結された入力データ文字列は比較の対象にはなりません。

**Less**

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = (InValue1 < InValue2)
制御ノード	-
設定ダイアログ	-

入力された2つの文字列を数値に変換し、その大きさを比較します。InValue1 とInValue2 を数値に変換し、InValue1 がInValue2 より小さければ”true”を出力し、そうでなければ”false”を出力します。カンマ(,)で連結された入力データ文字列は比較の対象にはなりません。

**Equal Greater**

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = (InValue1 >= InValue2)
制御ノード	-
設定ダイアログ	-

入力された2つの文字列を数値に変換し、その大きさを比較します。InValue1 とInValue2 を数値に変換し、InValue1 とInValue2 が等しいか、InValue1 がInValue2 より大きければ”true”を出力し、そうでなければ”false”を出力します。カンマ(,)で連結された入力データ文字列は比較の対象にはなりません。



Equal Less

入力ノード	InValue1 InValue2
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = (InValue1 <= InValue2)
制御ノード	-
設定ダイアログ	-

入力された2つの文字列を数値に変換し、その大きさを比較します。InValue1 とInValue2 を数値に変換し、InValue1 とInValue2 が等しいか、InValue1 がInValue2 より小さければ”true”を出力し、そうでなければ”false”を出力します。カンマ(,)で連結された入力データ文字列は比較の対象にはなりません。



Not

入力ノード	InValue
出力ノード	OutValue
対象ノード	-
出力結果	OutValue = !InValue
制御ノード	O
設定ダイアログ	-

入力された文字列の真偽値を反転します。InValue が”true”の場合には OutValue に”false”を、そうでなければ OutValue に”true”を出力します。

ただし、制御ノードにデータフローラインが接続されていて、入力データ文字列が”false”である場合には、内部演算は行われず、OutValue には InValue1 の文字列がそのまま出力されます。

Object クラス

Object クラスのモジュールは、仮想空間のオブジェクトの位置/方向などの設定/取得を行います。オブジェクトは VRML2.0/97 の Transform ノードを対象としており、モジュールを左マウスボタンでダブルクリックすることにより表示されるダイアログで対象の Transform ノードを選択できます。ただし選択の対象となるのは名前がつけられている Transform ノードに限られ、名前がつけられていない Transform ノードは選択の対象にはなりません。希望するノードがダイアログの選択ボックスに表示されない場合には、その Transform ノードに名前がつけられているか確認して下さい。

オブジェクトの1つの属性に対して、データを設定するモジュールとデータを取得するモジュールがセットで用意されています。例えばオブジェクトの位置を設定するモジュールには SetLocation モジュール、取得するモジュールには GetLocation モジュールがあります。データを設定するタイプのモジュールには制御ノードがあり、制御ノードにデータフローラインが接続されていて、“false”が入力されている場合には、データの設定は行われません。



SetLocation

入力ノード	location
出力ノード	-
対象ノード	Transform
出力結果	Transform::translation = location
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたオブジェクトの位置を設定します。location には“x,y,z”形式の数値文字列を入力して下さい。オブジェクトが選択されていない場合や、入力データ文字列が不正な場合には位置は更新されません。



SetRotation

入力ノード	rotation
出力ノード	-
対象ノード	Transform
出力結果	Transform::rotation = rotation
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたオブジェクトの方角を設定します。rotation には“x,y,z,angle”形式の数値文字列を入力して下さい。オブジェクトが選択されていない場合や、入力データ文字列が不正な

場合には方角は更新されません。



SetScale

入力ノード	scale
出力ノード	-
対象ノード	Transform
出力結果	Transform::scale = scale
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたオブジェクトのスケール倍率を設定します。scale には"x,y,z"形式の数値文字列を入力して下さい。オブジェクトが選択されていない場合や、入力データ文字列が不正な場合にはスケール倍率は更新されません。



SetCenter

入力ノード	center
出力ノード	-
対象ノード	Transform
出力結果	Transform::center = center
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたオブジェクトの旋回位置を設定します。center には"x,y,z"形式の数値文字列を入力して下さい。オブジェクトが選択されていない場合や、入力データ文字列が不正な場合には旋回位置は更新されません。



GetLocation

入力ノード	-
出力ノード	location
対象ノード	Transform
出力結果	location = Transform::translation
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたオブジェクトの現在位置を出力します。location には"x,y,z"形式の数値文字列が出力されます。オブジェクトが選択されていない場合には NULL 文字を出力します。

**GetRotation**

入力ノード	-
出力ノード	rotation
対象ノード	Transform
出力結果	rotation = Transform::rotation
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたオブジェクト方角を出力します。rotation には"x,y,z,angle"形式の数値文字列が出力されます。オブジェクトが選択されていない場合には NULL 文字を出力します。

**GetScale**

入力ノード	-
出力ノード	scale
対象ノード	Transform
出力結果	scale = Transform::scale
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたオブジェクトのスケール倍率を出力します。scale には"x,y,z"形式の数値文字列が出力されます。オブジェクトが選択されていない場合には NULL 文字を出力します。

**GetCenter**

入力ノード	-
出力ノード	center
対象ノード	Transform
出力結果	center = Transform::center
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたオブジェクトの旋回位置を出力します。center には"x,y,z"形式の数値文字列が出力されます。オブジェクトが選択されていない場合には NULL 文字を出力します。

Material クラス

Material クラスのモジュールは仮想空間内のマテリアルの色情報の設定/取得を行います。マテリアルは VRML2.0/97 の Material ノードを対象としており、モジュールを左マウスボタンでダブルクリックすることにより対象の Material ノードをダイアログから選択できます。ただし選択の対象となるのは名前がつけられている Material ノードに限られ、名前がつけられていないノードは選択の対象にはなりません。希望するノードがダイアログの選択ボックスに表示されない場合には、そのノードに名前がつけられているか確認して下さい。

マテリアルの1つの属性に対して、データを設定するモジュールとデータを取得するモジュールがセットで用意されています。例えばマテリアルの拡散色を設定するモジュールには SetDiffuseColor モジュール、取得するモジュールには GetDiffuseColor モジュールがあります。データを設定するタイプのモジュールには制御ノードがあり、制御ノードにデータフローラインが接続されていて、“false”が入力されている場合には、データの設定は行われません。



SetAmbientIntensity

入力ノード	ambientIntensity
出力ノード	-
対象ノード	Material
出力結果	Material:: ambientIntensity = ambientIntensity
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの環境輝度を設定します。ambientIntensity は、0.0 ~ 1.0 の範囲の数値文字列を入力して下さい。マテリアルが選択されていない場合や、入力データ文字列が不正な場合には環境輝度は更新されません。



SetDiffuseColor

入力ノード	diffuseColor
出力ノード	-
対象ノード	Material
出力結果	Material:: diffuseColor = diffuseColor
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの拡散色を設定します。diffuseColor は、“r,g,b”形式の数値文字列を入力して下さい。r,g,b は 0.0 ~ 1.0 の範囲の中で指定して下さい。マテリアルが選択されてい

ない場合や、入力データ文字列が不正な場合には拡散色は更新されません。



SetEmissiveColor

入力ノード	emissiveColor
出力ノード	-
対象ノード	Material
出力結果	Material:: emissiveColor = emissiveColor
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの放射色を設定します。emissiveColor は、"r,g,b"形式の数値文字列を入力して下さい。r,g,b は 0.0 ~ 1.0 の範囲の中で指定して下さい。マテリアルが選択されていない場合や、入力データ文字列が不正な場合には放射色は更新されません。



SetSpeculatColor

入力ノード	speculatColor
出力ノード	-
対象ノード	Material
出力結果	Material:: speculatColor = speculatColor
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの鏡面色を設定します。speculatColor は、"r,g,b"形式の数値文字列を入力して下さい。r,g,b は 0.0 ~ 1.0 の範囲の中で指定して下さい。マテリアルが選択されていない場合や、入力データ文字列が不正な場合には鏡面色は更新されません。



SetShininess

入力ノード	shininess
出力ノード	-
対象ノード	Material
出力結果	Material:: shininess = shininess
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの鏡面係数を設定します。shininess は、0.0 ~ 1.0 の範囲の数値文字列を指定して下さい。マテリアルが選択されていない場合や、入力データ文字列が不正な場合には鏡面係数は更新されません。



GetAmbientIntensity

入力ノード	-
出力ノード	ambientIntensity
対象ノード	Material
出力結果	ambientIntensity = Material:: ambientIntensity
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの環境輝度を ambientIntensity に出力します。マテリアルが選択されていない場合には NULL 文字を出力します。



GetDiffuseColor

入力ノード	diffuseColor
出力ノード	-
対象ノード	Material
出力結果	Material:: diffuseColor = diffuseColor
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの拡散色を出力します。diffuseColor には”r,g,b”形式の数値文字列が出力されます。マテリアルが選択されていない場合には NULL 文字を出力します。



GetEmissiveColor

入力ノード	-
出力ノード	emissiveColor
対象ノード	Material
出力結果	emissiveColor = Material:: emissiveColor
制御ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの放射色を出力します。emissiveColor には”r,g,b”形式の数値文字列が出力されます。マテリアルが選択されていない場合には NULL 文字を出力します。



GetSpeculatColor

入力ノード	-
出力ノード	speculatColor
対象ノード	Material

出力結果	speculatColor = Material:: speculatColor
制御 ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの鏡面色を出力します。speculatColor には”r,g,b”形式の数値文字列が出力されます。マテリアルが選択されていない場合には NULL 文字を出力します。



GetShininess

入力 ノード	-
出力 ノード	shininess
対象 ノード	Material
出力結果	shininess = Material:: shininess
制御 ノード	-
設定ダイアログ	O

ダイアログで選択されたマテリアルの鏡面係数を shininess に出力します。マテリアルが選択されていない場合には NULL 文字を出力します。

Light クラス

Light クラスのモジュールは、仮想空間の光源の On/Off の設定や色などの設定/取得を行います。光源は VRML2.0/97 の DirectionalLight / PointLight / SpotLight ノードのいずれかを対象としており、モジュールを左マウスボタンでダブルクリックすることにより表示されるダイアログで対象の光源 ノードを選択できます。ただし選択の対象となるのは名前がつけられている光源 ノードに限られ、名前がつけられていない光源 ノードは選択の対象にはなりません。希望するノードがダイアログの選択ボックスに表示されない場合には、その光源 ノードに名前がつけられているか確認して下さい。ただしモジュールによっては、すべての種類のノードがダイアログで選択できる訳ではありません。例えば SetDirection / GetDirection モジュールは光源方向属性に関連したモジュールで、ダイアログにはその属性をもつ DirectionalLight / SpotLight ノードは表示されますが、属性をもたない PointLight ノードは表示されません。

光源の1つの属性に対して、データを設定するモジュールとデータを取得するモジュールがセットで用意されています。例えば視点位置を設定するモジュールには SetPosition モジュール、取得するモジュールには GetPosition モジュールがあります。データを設定するタイプのモジュールには制御 ノードがあり、制御 ノードにデータフローラインが接続されていて、“false”が入力されている場合には、データの設定は行われません。



SetOn

入力ノード	on
出力ノード	-
対象ノード	DirectionalLight / PointLight / SpotLight
出力結果	Light::on= on
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の ON/OFF 状態を設定します。on には“true”、“false”いずれかの文字列を入力して下さい。光源が選択されていない場合や、入力データ文字列が不正な場合には状態は変更されません。



SetColor

入力ノード	color
出力ノード	-
対象ノード	DirectionalLight / PointLight / SpotLight
出力結果	Light::color = color
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の色を設定します。color は、"r,g,b"形式の数値文字列を入力して下さい。r,g,b は 0.0 ~ 1.0 の範囲内で指定して下さい。光源が選択されていない場合や、入力データ文字列が不正な場合には色は更新されません。



SetIntensity

入力ノード	intensity
出力ノード	-
対象ノード	DirectionalLight / PointLight / SpotLight
出力結果	Light::intensity = intensity
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の輝度を設定します。intensity には、0.0 ~ 1.0 の範囲の数値文字列を入力して下さい。光源が選択されていない場合や、入力データ文字列が不正な場合には輝度は更新されません。



SetLocation

入力ノード	location
出力ノード	-
対象ノード	PointLight / SpotLight
出力結果	Light::location = location
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の位置を設定します。location は、"x,y,z"形式の数値文字列を入力して下さい。光源が選択されていない場合や、入力データ文字列が不正な場合には位置は更新されません。



SetDirection

入力ノード	direction
出力ノード	-
対象ノード	DirectionalLight / SpotLight
出力結果	Light::direction = direction
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の方向を設定します。direction は、"x,y,z"形式の数値文字列を入

力して下さい。光源が選択されていない場合や、入力データ文字列が不正な場合には方向は更新されません。



SetRadius

入力ノード	radius
出力ノード	-
対象ノード	PointLight / SpotLight
出力結果	Light::radius = radius
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の半径を設定します。radius は、0.0 以上の数値文字列を入力して下さい。光源が選択されていない場合や、入力データ文字列が不正な場合には方向は更新されません。



GetOn

入力ノード	
出力ノード	on-
対象ノード	DirectionalLight / PointLight / SpotLight
出力結果	on = Light::on
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の ON/OFF 状態を”true”または”false”で出力します。光源が選択されていない場合には NULL 文字を出力します。



GetColor

入力ノード	-
出力ノード	color
対象ノード	DirectionalLight / PointLight / SpotLight
出力結果	color = Light::color
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の色を出力します。color は、”r,g,b”形式の数値文字列が出力されます。光源が選択されていない場合には NULL 文字を出力します。

**GetIntensity**

入力ノード	-
出力ノード	intensity
対象ノード	DirectionalLight / PointLight / SpotLight
出力結果	intensity = Light::intensity
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の輝度を出力します。intensity には、0.0 ~ 1.0 の範囲の数値文字列が出力されます。

**GetLocation**

入力ノード	-
出力ノード	location
対象ノード	PointLight / SpotLight
出力結果	location = Light::location
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の位置を出力します。location は、"x,y,z"形式の数値文字列が出力されます。光源が選択されていない場合には NULL 文字を出力します。

**GetDirection**

入力ノード	-
出力ノード	direction
対象ノード	DirectionalLight / SpotLight
出力結果	direction = Light::direction
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の方向を出力します。direction は、"x,y,z"形式の数値文字列が出力されます。光源が選択されていない場合には NULL 文字を出力します。



GetRadius

入力ノード	radius
出力ノード	-
対象ノード	PointLight / SpotLight
出力結果	Light::radius = radius
制御ノード	-
設定ダイアログ	O

ダイアログで選択された光源の半径を出力します。光源が選択されていない場合には NULL 文字を出力します。

Viewpoint クラス

Viewpoint クラスのモジュールは、仮想空間の視点の位置/方向などの設定/取得を行います。視点は VRML2.0/97 の Viewpoint ノードを対象としており、モジュールを左マウスボタンでダブルクリックすることにより表示されるダイアログで対象の Viewpoint ノードを選択できます。ただし選択の対象となるのは名前がつけられている Viewpoint ノードに限られ、名前がつけられていない Viewpoint ノードは選択の対象にはなりません。希望するノードがダイアログの選択ボックスに表示されない場合には、その Viewpoint ノードに名前がつけられているか確認して下さい。

視点の1つの属性に対して、データを設定するモジュールとデータを取得するモジュールがセットで用意されています。例えば視点位置を設定するモジュールには SetPosition モジュール、取得するモジュールには GetPosition モジュールがあります。データを設定するタイプのモジュールには制御ノードがあり、制御ノードにデータフローラインが接続されていて、“false”が入力されている場合には、データの設定は行われません。



SetPosition

入力ノード	position
出力ノード	-
対象ノード	Viewpoint
出力結果	Viewpoint:: position = position
制御ノード	-
設定ダイアログ	O

ダイアログで選択された視点の位置を設定します。position には“x,y,z”形式の数値文字列を入力して下さい。視点が選択されていない場合や、入力データ文字列が不正な場合には位置は更新されません。



SetOrientation

入力ノード	orientation
出力ノード	-
対象ノード	Viewpoint
出力結果	Viewpoint::orientation = orientation
制御ノード	-
設定ダイアログ	O

ダイアログで選択された視点の方角を設定します。orientation には“x,y,z,angle”形式の数値文字列を入力して下さい。視点が選択されていない場合や、入力データ文字列が不正な場合には方

角は更新されません。



SetFOV

入力ノード	fov
出力ノード	-
対象ノード	Viewpoint
出力結果	Viewpoint::fieldOfView = fov
制御ノード	-
設定ダイアログ	O

ダイアログで選択された視点の視野角を設定します。fov には 0 以上 PI 以下の数値文字列を入力してください。視点が選択されていない場合や、入力データ文字列が不正な場合には旋回位置は更新されません。



GetPosition

入力ノード	-
出力ノード	position
対象ノード	Viewpoint
出力結果	position = Viewpoint:: position
制御ノード	-
設定ダイアログ	O

ダイアログで選択された視点の位置を出力します。position には"x,y,z"形式の数値文字列が出力されます。視点が選択されていない場合には NULL 文字を出力します。



GetOrientation

入力ノード	-
出力ノード	orientation
対象ノード	Viewpoint
出力結果	orientation =Viewpoint::orientation
制御ノード	-
設定ダイアログ	O

ダイアログで選択された視点の方角を出力します。orientation には"x,y,z,angle"形式の数値文字列が出力されます。視点が選択されていない場合には NULL 文字を出力します。

**GetFOV**

入力ノード	-
出力ノード	fov
対象ノード	Viewpoint
出力結果	fov=Viewpoint::fieldOfView
制御ノード	-
設定ダイアログ	O

ダイアログで選択された視点の視野角を出力します。fov には”x,y,z,angle”形式の数値文字列が出力されます。視点が選択されていない場合には NULL 文字を出力します。

Interpolator クラス

Interpolator クラスのモジュールは、アニメーションなどに利用される Interpolator ノードの fraction 値を設定するものです。対象となる Interpolator ノードは、モジュールを左マウスボタンでダブルクリックすることにより表示されるダイアログで選択できます。



Scalar

入力ノード	fraction
出力ノード	-
対象ノード	ScalarInterpolator
出力結果	ScalarInterpolator :: set_fraction = fraction
制御ノード	O
設定ダイアログ	O

入力された fraction を数値に変換し、ダイアログで選択された ScalarInterpolator ノードの set_fraction に設定します。fraction が数値に変更できない場合には無効です。



Position

入力ノード	fraction
出力ノード	-
対象ノード	PositionInterpolator
出力結果	PositionInterpolator:: set_fraction = fraction
制御ノード	O
設定ダイアログ	O

入力された fraction を数値に変換し、ダイアログで選択された PositionInterpolator ノードの set_fraction に設定します。fraction が数値に変更できない場合には無効です。



Normal

入力ノード	fraction
出力ノード	-
対象ノード	NormalInterpolator
出力結果	NormalInterpolator:: set_fraction = fraction
制御ノード	O
設定ダイアログ	O

入力された fraction を数値に変換し、ダイアログで選択された NormalInterpolator ノードの set_fraction に設定します。fraction が数値に変更できない場合には無効です。

**Orientation**

入力ノード	fraction
出力ノード	-
対象ノード	OrientationInterpolator
出力結果	OrientationInterpolator:: set_fraction = fraction
制御ノード	O
設定ダイアログ	O

入力された fraction を数値に変換し、ダイアログで選択された OrientationInterpolator ノードの set_fraction に設定します。fraction が数値に変更できない場合には無効です。

**Scalar**

入力ノード	fraction
出力ノード	-
対象ノード	ColorInterpolator
出力結果	ColorInterpolator:: set_fraction = fraction
制御ノード	O
設定ダイアログ	O

入力された fraction を数値に変換し、ダイアログで選択された ColorInterpolator ノードの set_fraction に設定します。fraction が数値に変更できない場合には無効です。

Misc クラス

Misc クラスには、時間を出力したり ランダムな値を出力したりするモジュールがあります。



SetSwitch

入力ノード	whichChoice
出力ノード	-
対象ノード	Switch
出力結果	Switch:: whichChoice = whichChoice
制御ノード	O
設定ダイアログ	O

入力された whichChoice を数値に変換し、ダイアログで選択された Switch ノードの whichChoice に設定します。whichChoice が数値に変更できない場合には無効です。



SetSkyColor

入力ノード	color (r, g, b)
出力ノード	-
対象ノード	Background
出力結果	Background:: skyColor[0] = color
制御ノード	O
設定ダイアログ	O

入力された color を色に変換し、ダイアログで選択された Background ノードの skyColor フィールドの先頭値に設定します。color が色に変更できない場合には無効です。



SetText

入力ノード	text
出力ノード	-
対象ノード	Text
出力結果	Text:: string[0] = text
制御ノード	O
設定ダイアログ	O

入力された text 文字列を、ダイアログで選択された Text ノードの string フィールドの先頭値に設定します。

**GetTime**

入力ノード	-
出力ノード	hour minute second
対象ノード	-
出力結果	-
制御ノード	-
設定ダイアログ	-

現在の時刻を、hour/minute/second に出力します。もし出力される時刻がおかしい場合には、環境変数 TZ を正常な値を設定して下さい。例えば日本の場合には、TZ には”JST-9”を設定する必要があります。

**Random**

入力ノード	-
出力ノード	random value
対象ノード	-
出力結果	-
制御ノード	-
設定ダイアログ	-

数値の範囲が 0～1 までのランダムな文字列を出力します。表示される文字列は、モジュール出力データ文字列表示の仕様により小数点以下は 2 桁の精度しかありませんが、実際には単精度の文字列が出力されています。

**PlaySound**

入力ノード	-
出力ノード	-
対象ノード	AudioClip
出力結果	-
制御ノード	O
設定ダイアログ	O

ダイアグラムで選択されている AudioClip ノードの音源ファイルを再生します。ダイアログの選択リストに表示される AudioClip ノードは、仮想空間にある名前のついたものだけです。

音源再生の制御はモジュールの制御ノードで行ないます。制御ノードに true が入力されれば音源の再生を開始します。false が入力された場合には無視されます。



Beep

入力ノード	-
出力ノード	-
対象ノード	AudioClip
出力結果	-
制御ノード	O
設定ダイアログ	-

Beep 音を発生させます。Beep 音再生の制御はモジュールの制御ノードで行ないます。制御ノードに true が入力されれば Beep 音を発生させます。false が入力された場合には無視されます。



JavaConsole

入力ノード	value
出力ノード	-
対象ノード	-
出力結果	-
制御ノード	-
設定ダイアログ	-

CTB を作成したコンテンツを VRML プラグインで動作させる場合に、モジュールに入力された文字列をブラウザの Java コンソール画面に出力します。CTB コンテンツを VRML プラグインでデバッグする場合に活用して下さい。

7. 互換性

VRML ブラウザ/プラグイン

CTB で作成されたアプリケーションは、通常の VRML2.0/97 形式として特別な拡張なしにファイルに落とされます。そのため、CTB で作成されたアプリケーションは、一般的な VRML2.0/97 に対応したブラウザまたはプラグインであれば閲覧可能です。しかしながら、VRML 仕様にすべて準拠したものは少なく、またその実装にも若干の差があり CTB で製作したコンテンツと異なる動作をする場合がありますので注意して下さい。

Java サポート

CTB では動的な動作定義を実現するのに、多くの Java でプログラミングされたモジュールを利用しています。複数の入出力 ノードをもち、アイコンとして画面に表示されているダイグラムのモジュールは、最終的に VRML の Script ノードに変換されています。Script ノードのスクリプト言語として Java を利用した、VRML 仕様の JSAI(java script authoring interface)に準拠して作成されています。そのため、CTB で製作したコンテンツを実行するには、ブラウザでこの JSAI がサポートされている必要があります。

CTB では、モジュール動作を明確かつ迅速に処理するように VRML ブラウザ/プラグインに指示するため、Script ノードに変換する時に directOutput/mustEvaluate フィールドによる設定を行います。directOutput を TRUE に設定することにより Script ノードが保持しているノード情報を確実に更新されるように指示しています。mustEvaluate も TRUE に設定しており Script ノードに迅速にイベントを送るように指示しています。このようなフィールドが VRML 仕様に存在する理由は、VRML ブラウザ/プラグイン側での Script ノード処理のボトルネックを避けるためです。一般的に Script ノードの処理は VRML ブラウザ/プラグインの実装言語(C/C++など)から、スクリプト言語(Java など)への変換インターフェイスが必要となるため、処理的な負荷が多くなります。しかしながら VRML ブラウザ/プラグインの実装によっては、主にスクリプト処理の軽減による高速化を目的として、上記のようなdirectOutput/mustEvaluate フィールドの設定が無視されている場合があります。このような場合には、CTB で作成されたアプリケーションは正確には動作しませんので注意して下さい。例えば、directOutput 設定が無視されている場合には、オブジェクトや光源の位置/回転の設定などが全く更新されなかったり mustEvaluate 設定が無視されている場合には、オブジェクトや光源の位置/回転の更新間隔が遅くなるなどの状況が発生します。

また CTB の一部のモジュールは JDK1.1 の機能を利用して作成されているため、ブラウザは JDK1.1 に対応しているほうが望ましい結果が得られます。例えば Misc クラスの Beep モジュールは、Beep 音の発生などが JDK1.1 の機能を利用して作られているモジュールに該当します。

ROUTE 処理順序

ダイアグラムのモジュール間のデータフローラインは、最終的には VRML の ROUTE 定義に変換されています。この ROUTE の処理順序が、VRML ブラウザ/プラグインの実装によって異なる場合があります。例えば以下のような ROUTE 定義がある場合に、

```
ROUTE TimeSensor.cycleTime TO A.inField -  
ROUTE C.outField TO D.inField -  
ROUTE B.outField TO C.inField -  
ROUTE A.outField TO B.inField -
```

実装の方法としては、大別すると、発生したイベントを各ノート間の伝達順序を考慮してと実行する場合と、と ROUTE が定義されている行順に実行する方法があります。一般的な実装方法としては前者の場合が多く、CTB 付属ブラウザもこの方式にて実装されています。しかし、VRML ブラウザ/プラグインによっては、ROUTE 処理を簡略化するなどの目的で、後者の方式にて実装される場合も考えられます。このような VRML ブラウザ/プラグインでは、データフローのモジュール実行順序が不定になるため、CTB アプリケーション動作に延滞などの状況が発生しますので注意して下さい。