ようこそ、マリオネットの世界へ

マリオネットは Vectorworks を使うデザイナーのためのビジュアルプログラミング環境です。

この入門書をきっかけに、ぜひ新しいデザインの世界を体験してください。

マリオネット入門

Marionette Primer

20160115

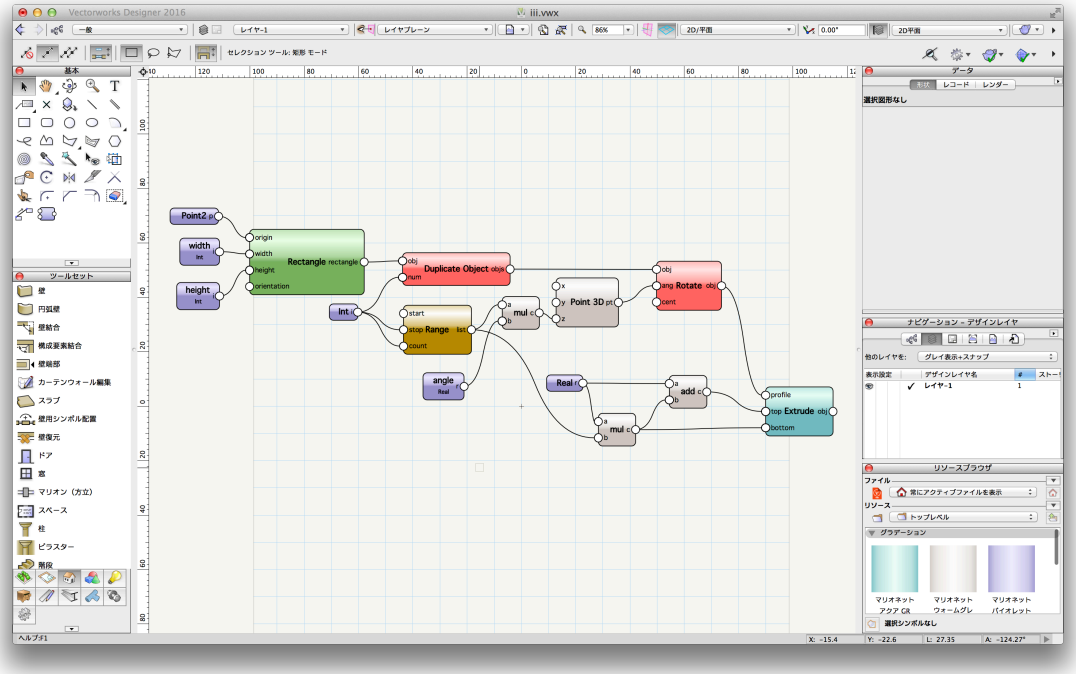
マリオネット入門

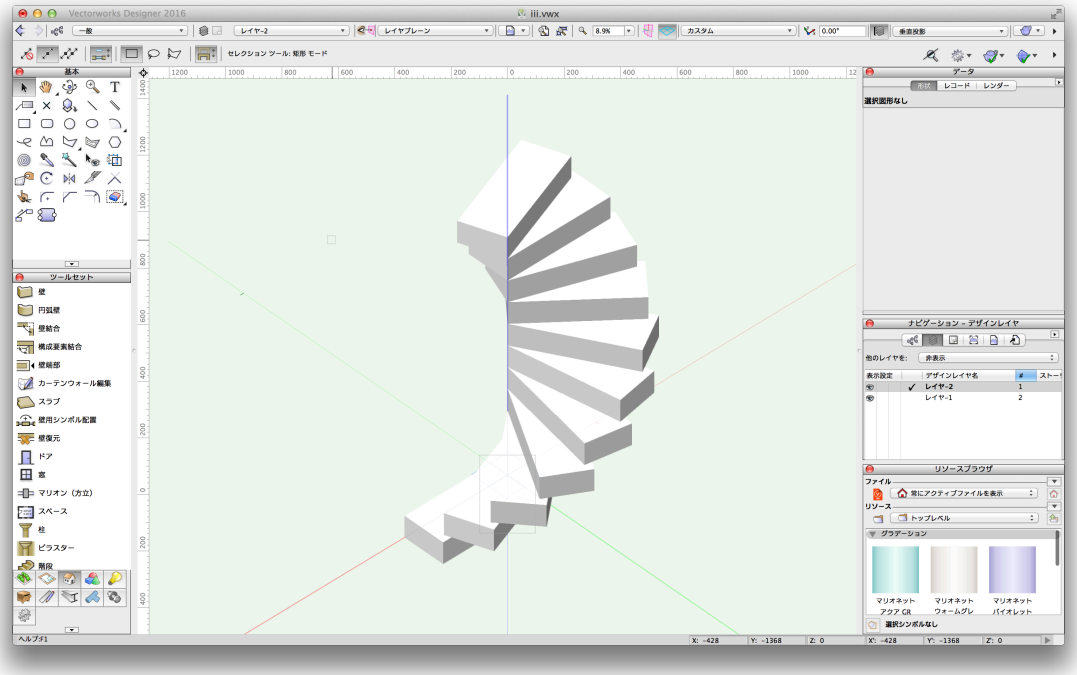
目次

マリオネットとは

マリオネットは Vectorworks に組み込まれた、ビジュアルプログラミングの環境です。たったひとつの「マリオネットツール」からその世界は始まります。

コンピューテーショナルデザインやパラメトリックデザインなど、プログラミングで様々なデザインの可能性が開けます。そんなプログラミングが、ぱちぱちテキストを打ち込むことなくできてしまいます。「ノード」と「ワイヤー」という図形をつなげるだけでプログラミングができるのです。

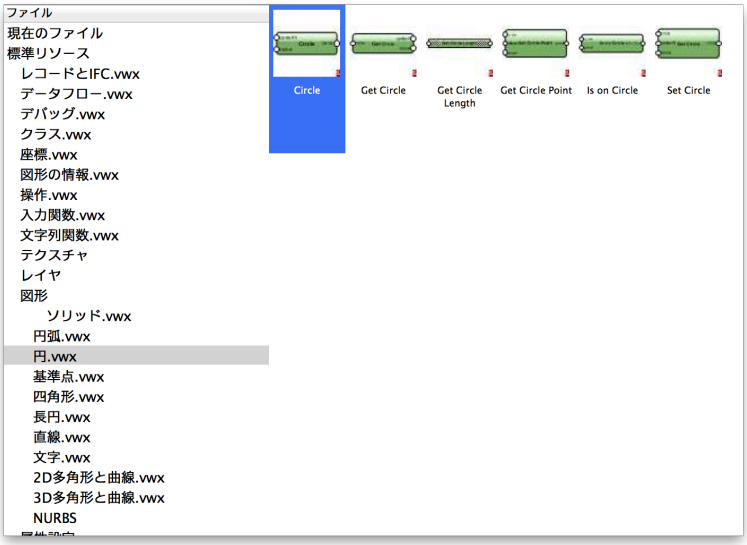




マリオネットをはじめる

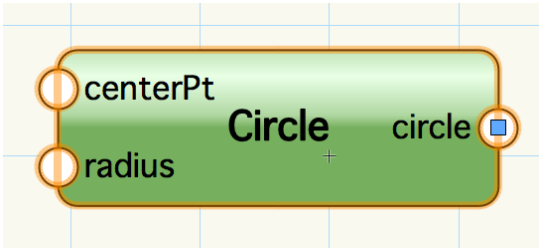
・ ノードの選択

マリオネットツールを使って、Vectorworks にあらかじめ用意されたノード使用することができます。マリオネットツールには、図形を描画するもの、図形の属性を編集するもの、数学の演算を行うもの、データの流れを管理するものなど、500 種以上のノードが用意されています。



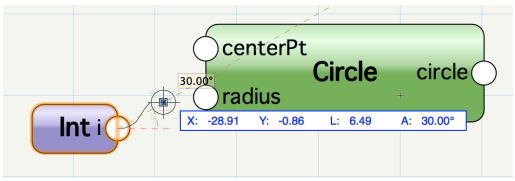
・ ノードを書類に配置する

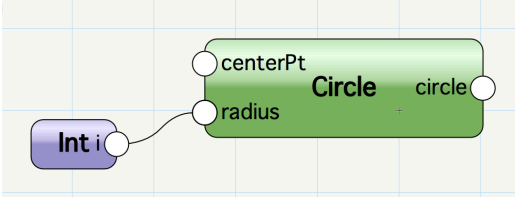
マリオネットツールがアクティブになっていると、選択したノードのシルエットが画面上のカーソルに追従し、クリックした位置に、選択したノードが配置されます。



・ ノードを繋いでマリオネットネットワークを作成する

ノードの出力ポートからワイヤーを伸ばして他のノードの入力ポート上でクリックすることで、ノードとノードを繋ぐことができます（※ビューは 2D/平面に設定しておく）。ワイヤーで繋がれたノードの集合をマリオネットネットワークといいます。





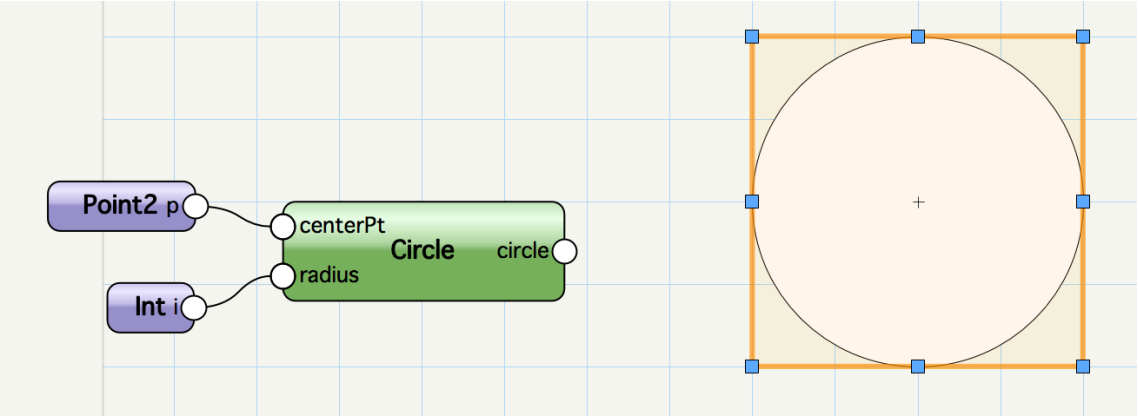
・ マリオネットスクリプトの実行

作成したマリオネットネットワークは一連の動作をするスクリプトになっており、次のいずれかの操作で

実行できます。

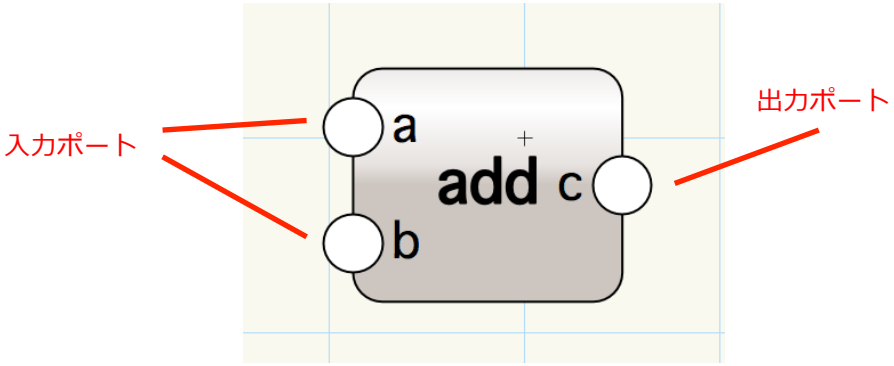
- ノードを右クリック > コンテキストメニュー > マリオネットスクリプトを実行
- ノードを選択 > データパレット > 実行ボタン

スクリプトの実行により新たに作成されたオブジェクトはグループ化されます。

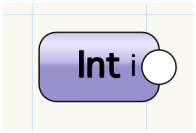


ノード

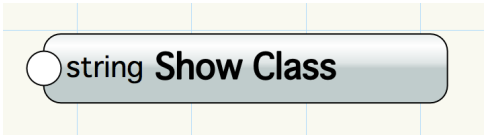
・ データを左側にある入力ポートから受け取り、ノード内のスクリプトを実行して、右側にある出力ポートから結果を出力すます。



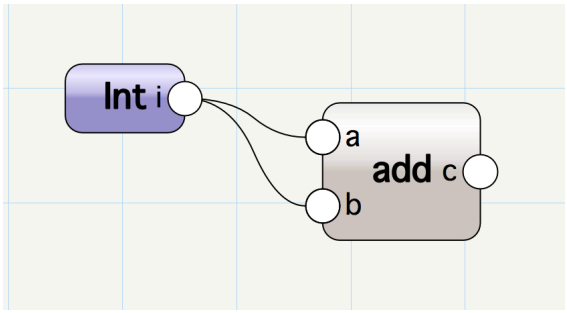
出力のみのノード：データを受け取らず、結果の出力のみ行います。



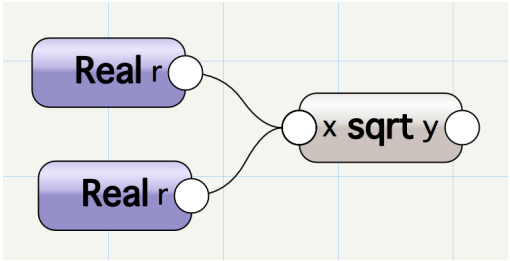
入力のみのノード：データを受け取り、スクリプトを実行するが、結果は出力されません。



ひとつの出力ポートから複数のワイヤーを伸ばせます。

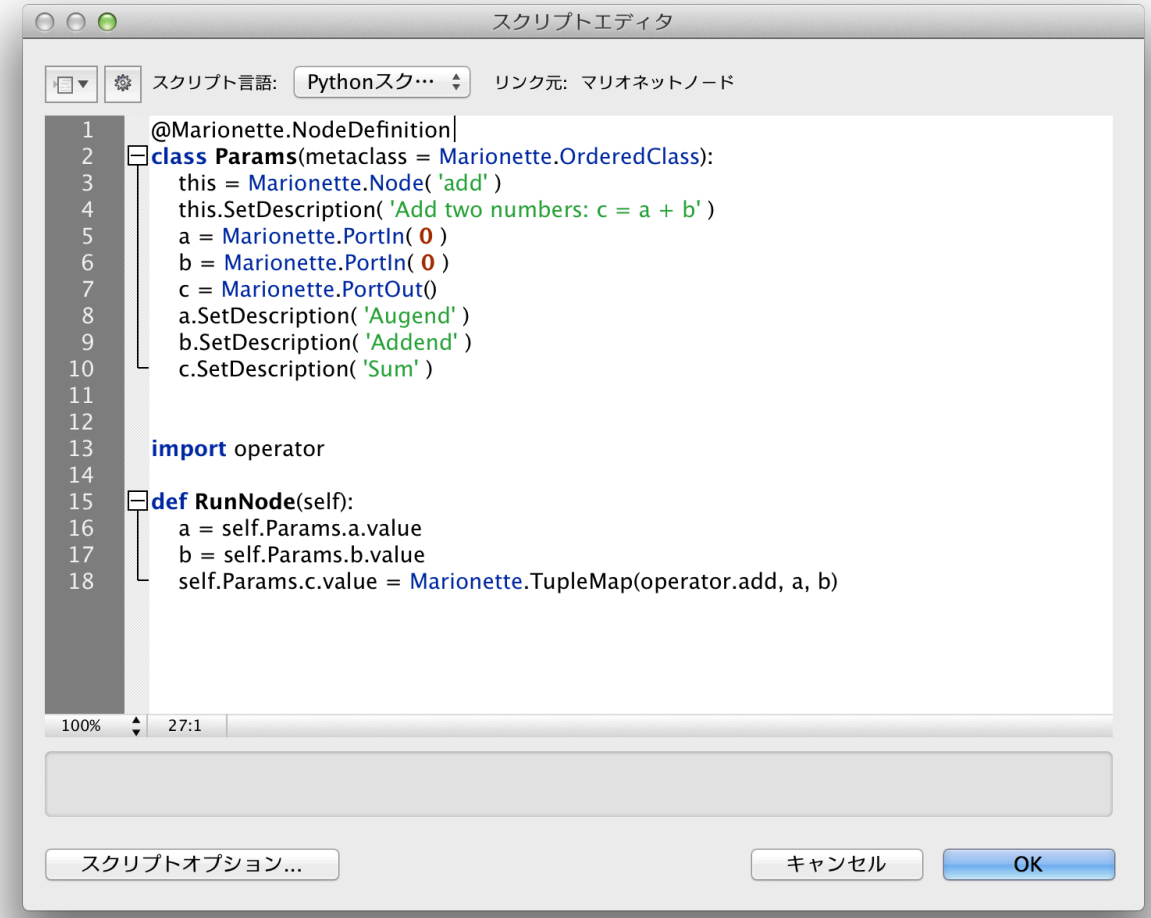


ひとつの入力ポートに複数のワイヤーを接続できます。



ノードのスクリプトの編集

ノードのスクリプトは Python で記述されており、スクリプトエディタで編集できます。Python の知識があれば、既存のノードを編集することで、オリジナルのノードを作成できます。

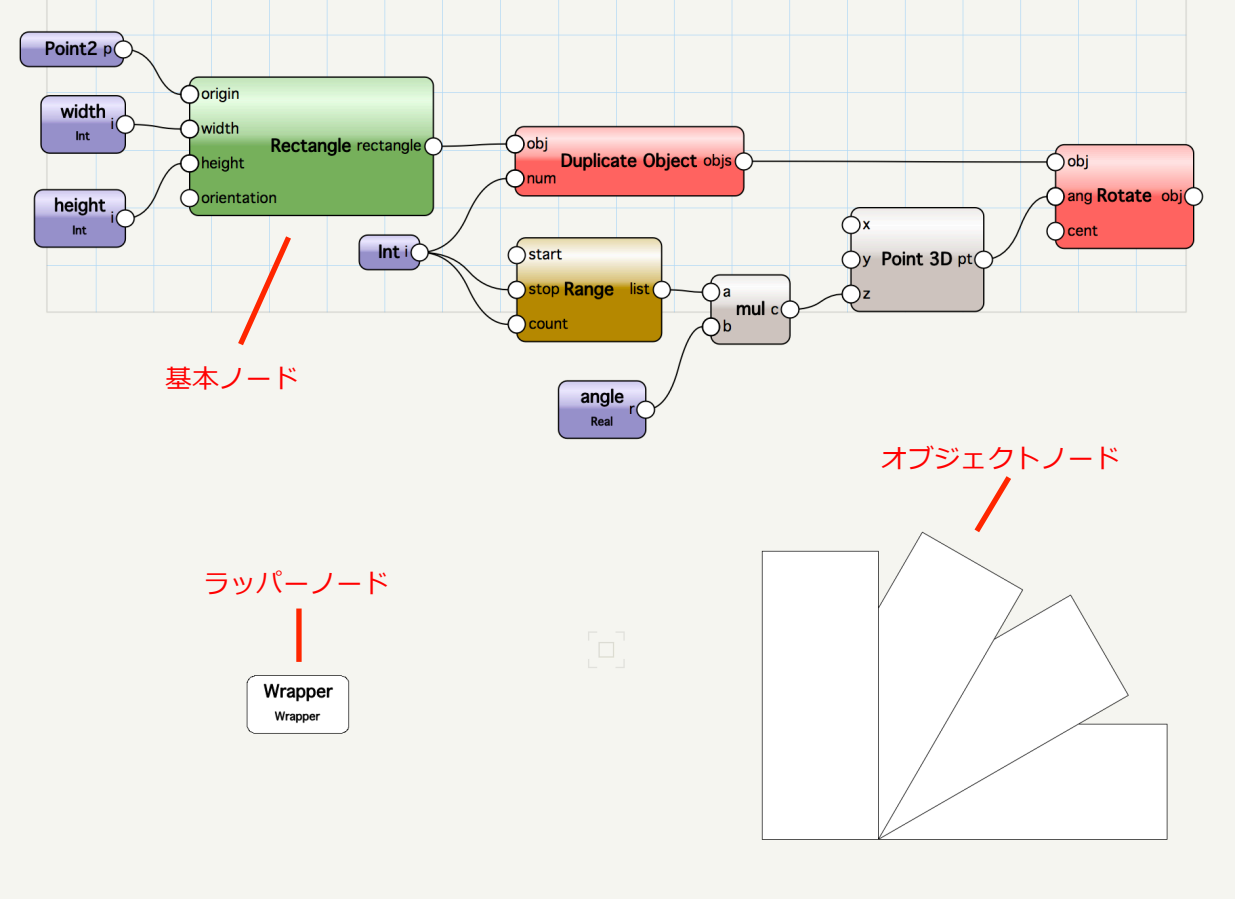


<スクリプトエディタの開き方(3 パターン)>

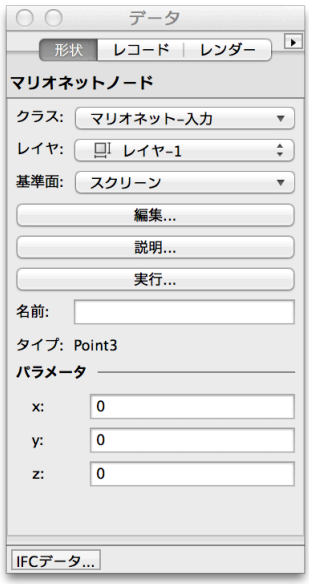
- ノードを選択 > データパレット > 編集ボタン
- ノードを右クリック > コンテキストメニュー > スクリプトの編集
- ノードをダブルクリックする

ノードの種類

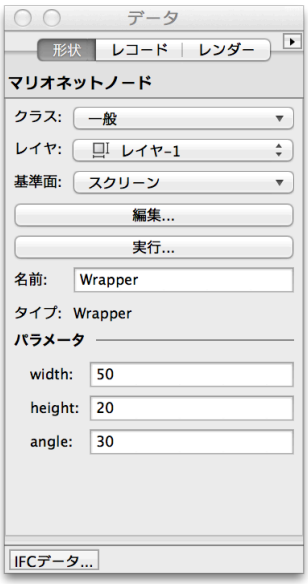
マリオネットのノードは、3 種の型（基本ノード・ラッパーノード・オブジェクトノード）があります。



それぞれデータパレットの表示はつぎのとおりです。



基本ノード

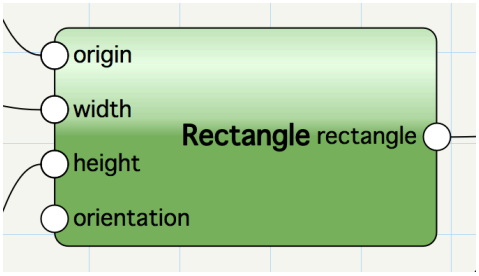


ラッパーノード

オブジェクトノード

基本ノード

ノードの基本の型であり、スクリプトを内包します。



<基本ノードのデータパレット>

- 編集ボタン：スクリプトを編集するエディタを開きます。
- 説明ボタン：ノードの動作の概要を表示します。
- 実行ボタン：ノードが接続されているマリオネットネットワークのスクリプトを実行します。
- 名前：選択中のノードに名前を設定します。
- タイプ:ノード種類を示します。
- パラメータ：ノードの動作に影響する値を入力します。

ラッパーノード

複数のノードを繋いだネットワークを、単一のノードに統合したノードの型。マリオネットネットワークの整理に用います。

<ラッパーノードの作成手順>

- ネットワークを右クリック ＞ コンテキストメニュー ＞ マリオネットネットワークをラップします。

<ラッパーノードの解除手順>

- ラッパーノードを右クリック ＞ コンテキストメニュー ＞ マリオネットネットワークのラップを解除します。

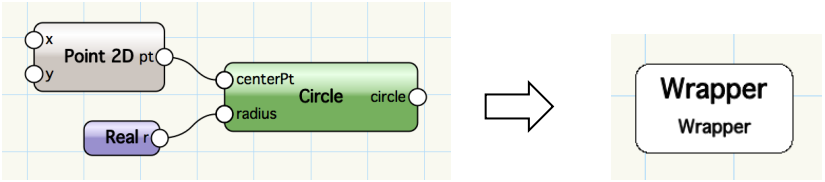
<ラッパーノードのデータパレット>

- 編集ボタン：内包しているマリオネットネットワークの編集画面を開きます。
- 実行ボタン：ノードが接続されているマリオネットネットワークのスクリプトを実行します。
- 名前：選択中のラッパーノードに名前をつけます。
- パラメータ：内包するノードのうち名前をつけたノードのパラメータが表示します。

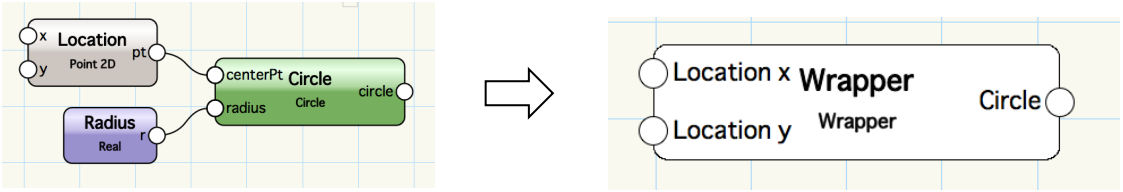
<ラッパーノードに入出力ポートを表示する>

- マリオネットネットワークをラッパーノードに変換する際、各ノードに名前を設定しておくと、名前をつけたノードの未接続の入出力ポートはラッパーノードの入出力ポートとして、パラメータはラッパーノードのデータパレットにフィールドとして表示されます。

  “ノードに名前をつけない場合”



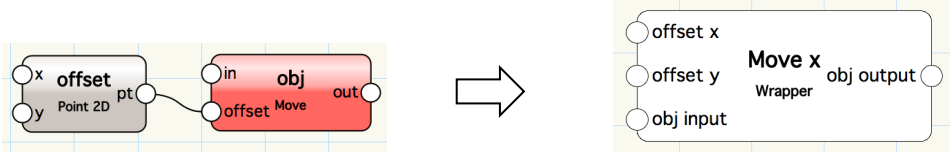
  “ノードに名前をつけた場合”



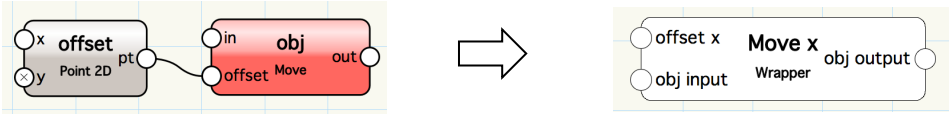
<未接続のポートを非表示にする>

- 名前をつけたノードの未接続の入出力ポートは、ポートの位置に 2D 基準点を配置すると、ラッパーノードでは非表示になります。

  “未接続のポートに 2D 基準点を配置しない場合”

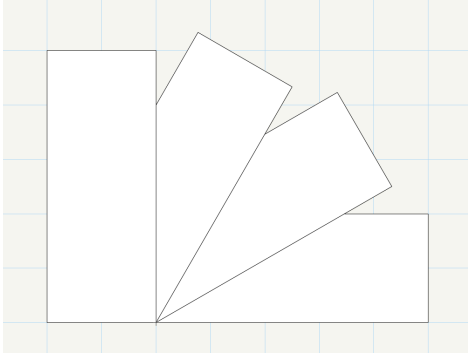


  “未接続のポート(y ポート)に 2D 基準点を配置した場合”



オブジェクトノード

ラッパーノード同じく、マリオネットネットワークをひとつに統合したノード。オブジェクトノードは内包するマリオネットスクリプトの実行結果の図形として表示され、スクリプトの実行を必要としません。ネットワークの変更やパラメータの変更はすぐに図形の表示に反映されます。



<オブジェクトノードの作成手順>

- ラッパーノードを右クリック ＞ コンテキストメニュー ＞ オブジェクトノードに変換

<オブジェクトノードの解除手順>

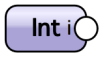
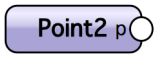
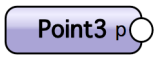
- オブジェクトノードを右クリック ＞ コンテキストメニュー ＞ ラッパーノードに変換

<オブジェクトノードのデータパレット>

- パラメータ：内包するノードのうち名前をつけたノードのパラメータが表示されます。

入力関数ノード

・ マリオネットネットでは、数値、座標、真偽、図形など様々な型のデータを扱うことができます。マリオネットツールにはこれらのデータを入力データとしてネットワークに接続するノードが用意されています。

	Python の eval 関数を使用して任意の値を作成する。
	データパレットのチェックボックスから、True または False の値を生成する。
	マリオネットオブジェクトの定義ネットワークで使われる時、コントロールジオメトリグループの最初のオブジェクトを返す。オブジェクトの外で使用できない。
	単位を持つ数値を設定する。
	整数値を返す。
	ファイル上の指定した名前のオブジェクトを返す。
	指定した検索条件に合致するオブジェクトのリスト返す。
	データパレットで定義した 2D 座標を返す。
	データパレットで定義した 3D 座標を返す。
	データパレットから定義する実数値を返す。 ファイルの単位が反映される。
	データパレットで定義する文字列を返す。
	2 次元ベクトルを作成する。
	3 次元ベクトルを作成する。

マリオネットネットワーク

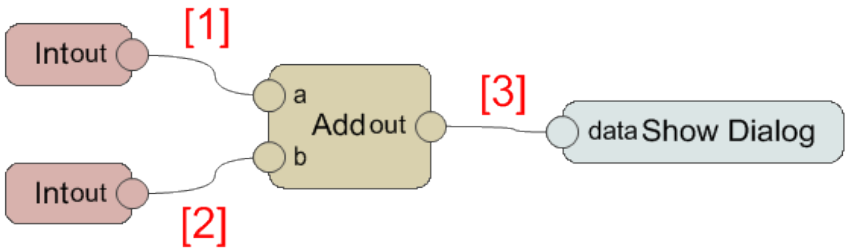
・ マリオネットネットワークは相互にデータのやりとりをするノードが集まったもので、繋がった順に各ノードで定義されたプログラムが実行されます。実行の順番はデータフローに依存し、データは一方向にしか流れません。データフローは、各ノードの入出力ポートを繋ぐワイヤーにより定義されます。

・ データフロー

ネットワークに接続されたノードは、入力ポートが受け取る値のタイプによって 4 パターンの方法で処理を行います。

(1) 単一の値を受け取る場合

それぞれのポートで受け取った単一の値を用いて、1 度だけスクリプトを実行します。



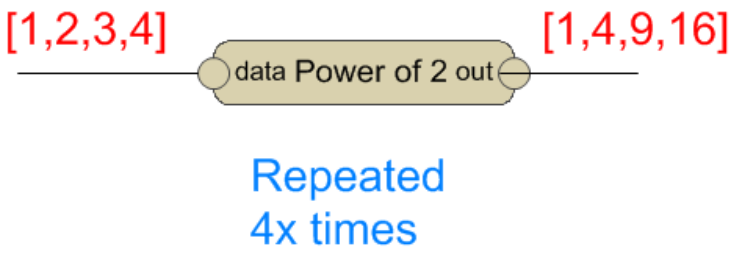
赤い文字はワイヤーを通過する値を示します。

<実行される結果>

1 + 2 = 3

(2) 値のリストを受け取る場合

ノードは複数の値を出力する場合があります。このデータは値のリストとし入力ポートに渡ります。一般的なノードはリストを受け取ると、リストの要素をひとつずつ順番に取得しながら、要素の数だけスクリプトを繰り返し実行します。



<実行される結果>

1 * 1 = 1

2 * 2 = 4

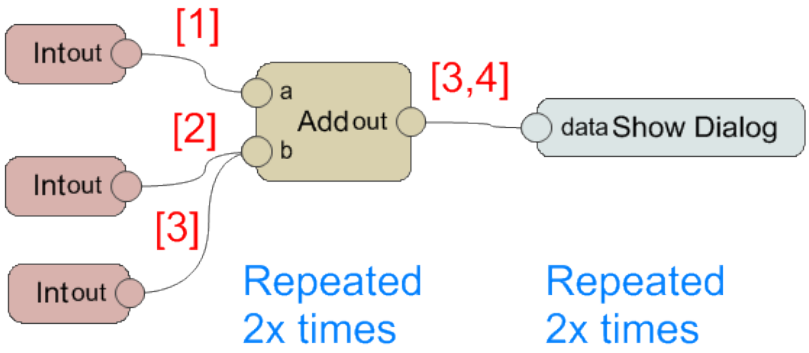
3 * 3 = 9

4 * 4 = 16

(3) 要素数の違うリストが混合する場合

複数の入力ポートを持つノードで、各ポートに要素数の違うリストを受け取ると、通常は最も長いリ

ストの要素数だけスクリプトが実行されます。入力されるリストがすべて同じ長さになるように、要素数が少ないリストは、最後の値で補完されます。



<使用するデータ>

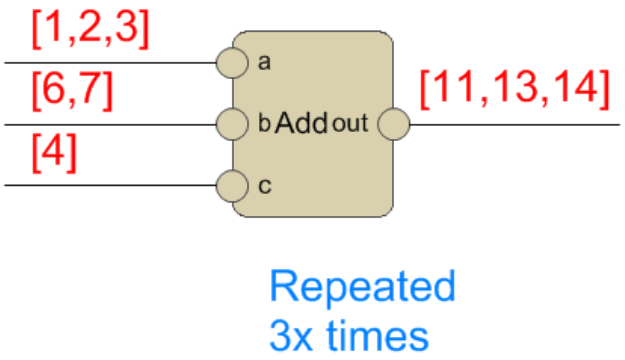
a ポート ：[1, 1]

b ポート ：[2, 3]

<実行される結果>

1 + 2 = 3

1 + 3 = 4



<使用するデータ>

a ポート ：[1, 2, 3]

b ポート ：[6, 7, 7]

c ポート ：[4, 4, 4]

<実行される結果>

1 + 6 + 4 = 11

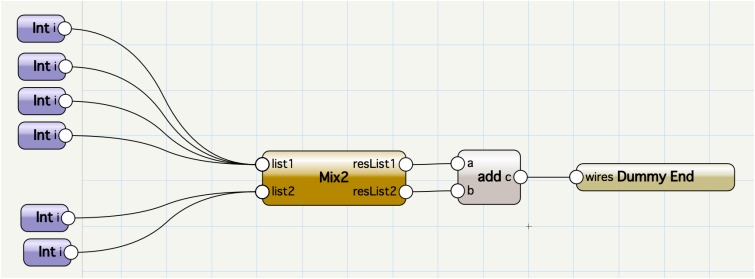
2 + 7 + 4 = 13

3 + 7 + 4 = 14

<長さの違うリストを任意の方式で揃える>

長さの違うリストを目的のノードに渡す前に、Mix2 ノード(データフローカテゴリ)を通過させること

で、リストの長さの揃え方を４パターンから選べます。



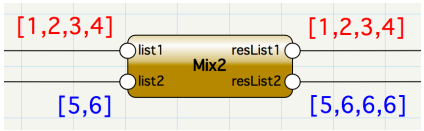
揃える方法は Mix2 ノードの List Matching パラメータから選択します。

[Short List]

最も短いリストの要素数に揃える。長いリストではみ出す値は切り捨てられます。

[Longest List]

最も長いリストの要素数に揃える。短いリストは最後の値で補間する。通常のノードはこの処理を行います。

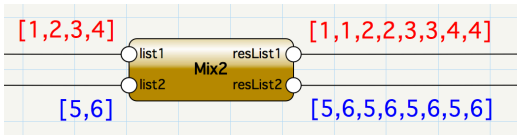


[Cycle]

最も長いリストの要素数に揃える。短いリストは要素数を満たすまで最初の値から順に補完されます。

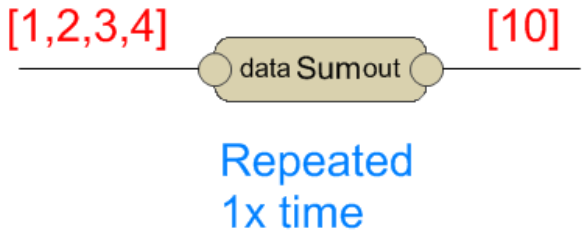
[Cross Reference]

要素数に関係なく、すべての値の組み合わせを取得できるリストを作成します。



(4) ノードの List Mix の設定が Absorb になっている場合

ノードが List Absorb として定義されている場合、受け取ったデータはすべてひとつのリストに連結されます。このノードではリスト自体をひとつの値として取得する。そのため、受け取るデータのタイプに関わらずスクリプトは 1 度だけ実行されます。



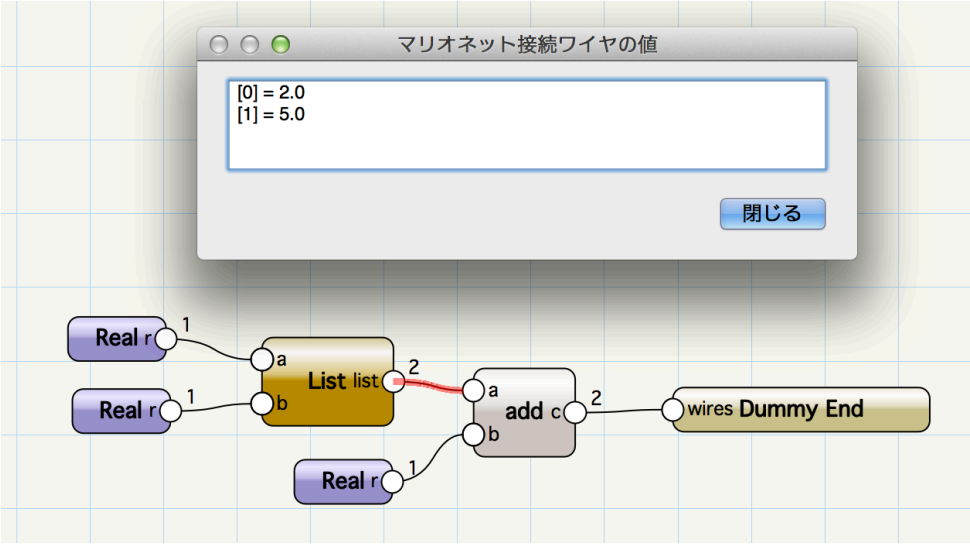
<実行される結果>

1 + 2 + 3 + 4 = 10

ネットワークのデバッグ

マリオネットツールのデバッグモードを用いて、ネットワークのトラブルシューティングが行えます。

デバッグモードでは、各出力ポートの隣に数字が表示され、その出力ポートから接続された入力ポートに流れる値の数が表示されます。ワイヤーを選択するとワイヤーを通過する値がダイアログに表示されます。ダイアログ内で値の変更を行うことができます。ワイヤーを通過する値がハンドル型の場合、図形のタイプ番号が表示されます。ダイアログを閉じるとスクリプトが実行されます。



マリオネットスクリプトで作成される図形の処理の注意点

- Name ノード

  指定した名前の図形そのものを使用するか、複製した図形を使用するか選択できます。

- 図形を新しく作成する

  新しく作成された図形はすべてグループ化されます。

- 図形を変換する

  変換系のノードには、入力した図形そのものを変換するノードと、複製を変換するノードがあります。

- 不要な図形を削除する

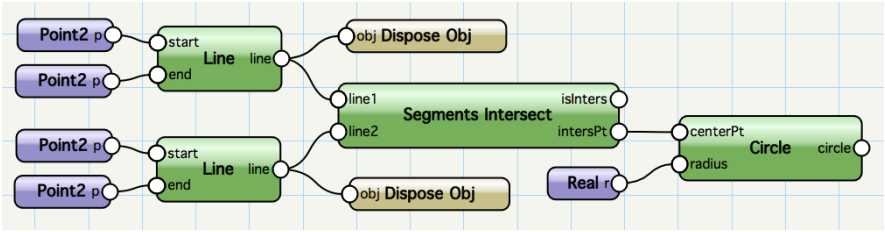
  マリオネットスクリプト内で作成された不要な図形を削除にするには、ネットワーク内で不要な図形のハンドルを”vs.Marionette_DisposeObj()”関数に渡す必要があります。この関数に渡された図形は、スクリプト実行後にまとめて削除されます。

  マリオネットツールに標準搭載されているノード群に、”vs.Marionette_DisposeObj()”を実行できるものが用意されていないため、自作する必要があります。

  例えば、次のようなノードを作ります。

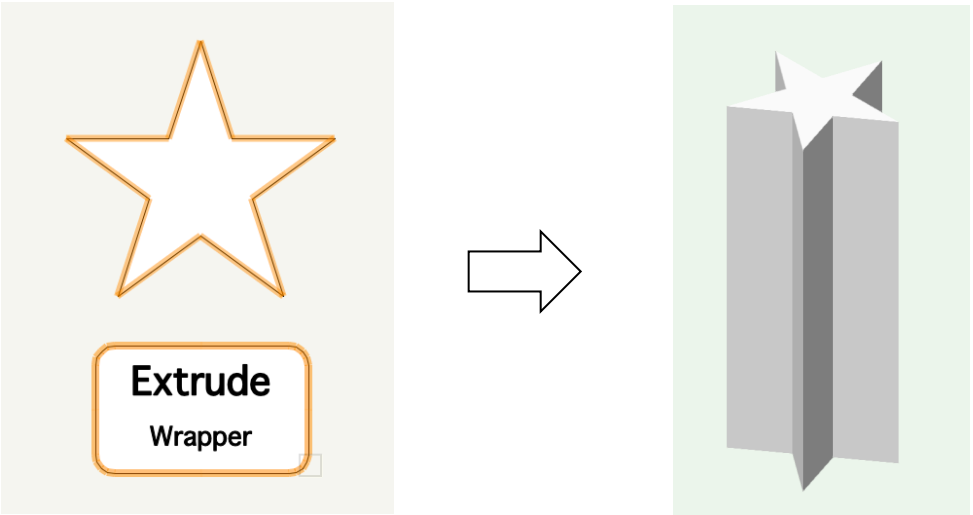
```
@Marionette.NodeDefinition
class Params(metaclass = Marionette.OrderedClass):
        this = Marionette.Node( 'Dispose Obj' )
        this.SetDescription('オブジェクトをスクリプト実行後に削除します')
        obj = Marionette.PortIn( None )
        obj.SetDescription('オブジェクトのハンドル')

def RunNode(self):
        obj = self.Params.obj.value
        if not obj is None:
                vs.Marionette_DisposeObj(obj)
```

<使用例> 途中の 2 つの直線を削除します。



コントロールジオメトリ

・ オブジェクトノードでは、パスオブジェクトをコントロールジオメトリとして持つことができます。



<コントロールジオメトリの設定>

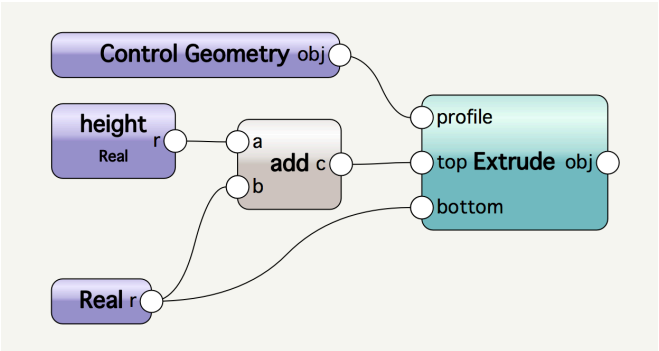
- ラッパーノードと単一のオブジェクトを選択する ＞ コンテキストメニュー ＞ オブジェクトノードに変換

<コントロールジオメトリの編集>

- オブジェクトノード ＞ コンテキストメニュー ＞ コントロールジオメトリの編集

<コントロールジオメトリの利用>

Control Geometry ノード（入力関数カテゴリ）を用いて、パスオブジェクトのハンドルを取得します。



マリオネット関連情報

1　Vectorworks 社ユーザーフォーラム
https://techboard.vectorworks.net/ubbthreads.php?ubb=cfrm&c=10

Marionette 全般
https://techboard.vectorworks.net/ubbthreads.php?ubb=postlist&Board=46&page=1

Marionette リソース集
https://techboard.vectorworks.net/ubbthreads.php?ubb=postlist&Board=45&page=1

2　Marionette Tutorials
http://kbase.vectorworks.net/questions/1350/Marionette+Tutorials

3　Developer Wiki Marionette　詳細
http://developer.vectorworks.net/index.php/Marionette

Marionette Basics
http://developer.vectorworks.net/index.php/Marionette_Basics

Marionette ノードの作り方（詳解）
http://developer.vectorworks.net/index.php/Marionette_Implement_a_Node

ノード一覧（作成中）
http://developer.vectorworks.net/index.php/Marionette_Node_Reference