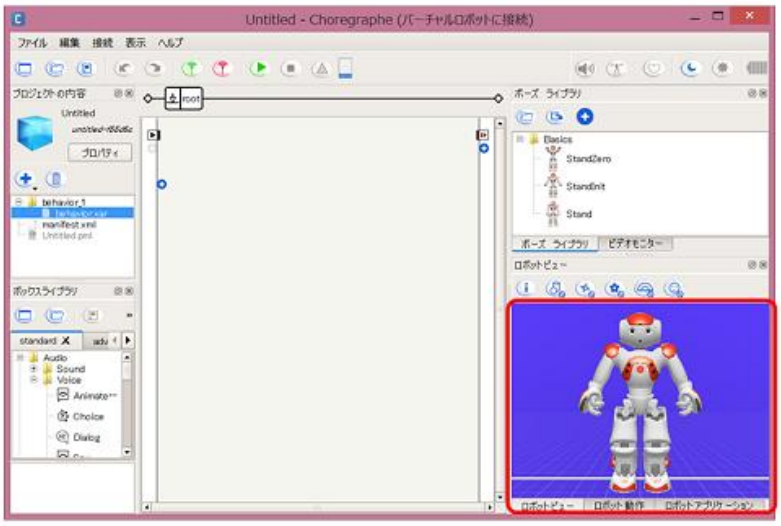
# Pepper チュートリアル (1):SDK インストールとアプリケーションの作成/実行

## Choregraphe の実行

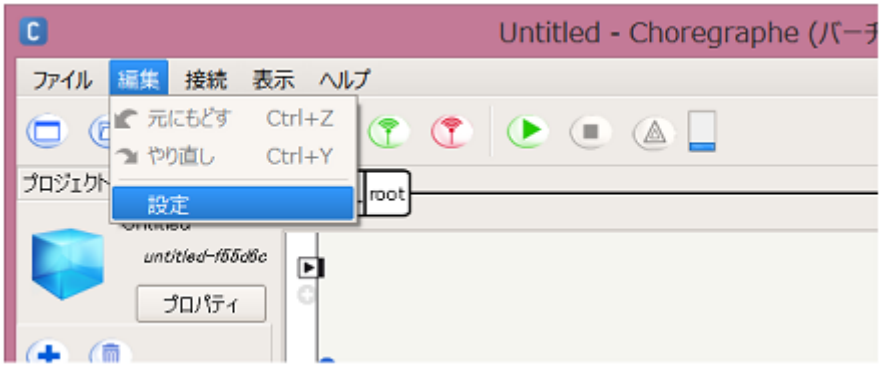
SDK をインストールすると、Choregraphe のアイコンがスタートメニューなどに登録されますので、ここから Choregraphe を起動してください。


[ようこそ]ダイアログが表示されたら、まずは **[新規プロジェクト...]** をクリックしてください。Choregraphe の画面が起動します。

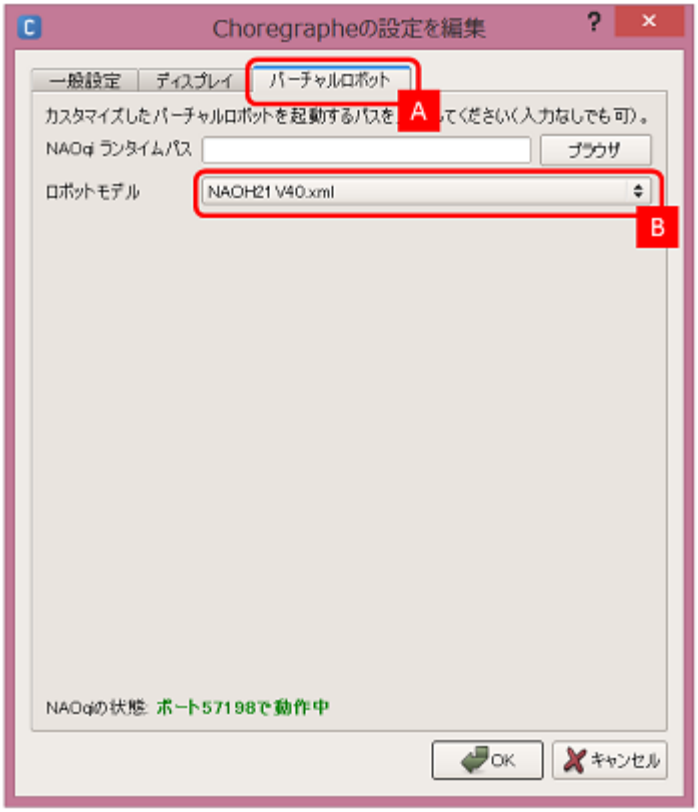



Choregraphe では、Pepper の実機がなくても、バーチャルロボットを用いて動作確認をおこなうことができますが、デフォルトではバーチャルロボットは **NAO** になっています。これを、以下の手順で Pepper に変更します。

1. [編集]メニューの **[設定]** を選択します

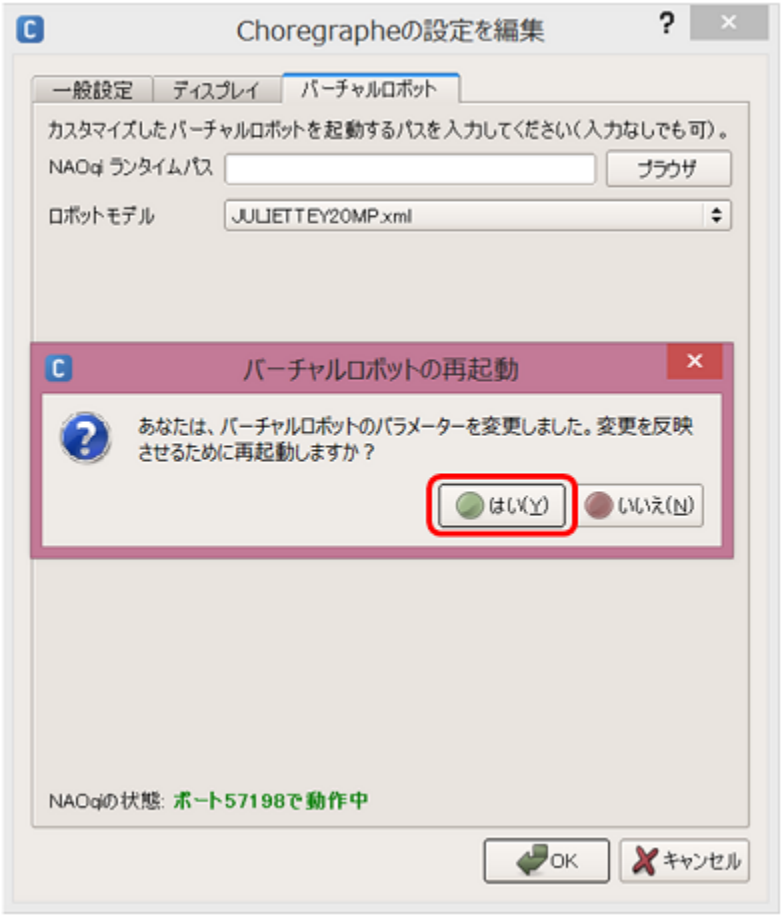



2. **[バーチャルロボット]タブを選択[A]** し、**[ロボットモデル]リストを選択[B]** します

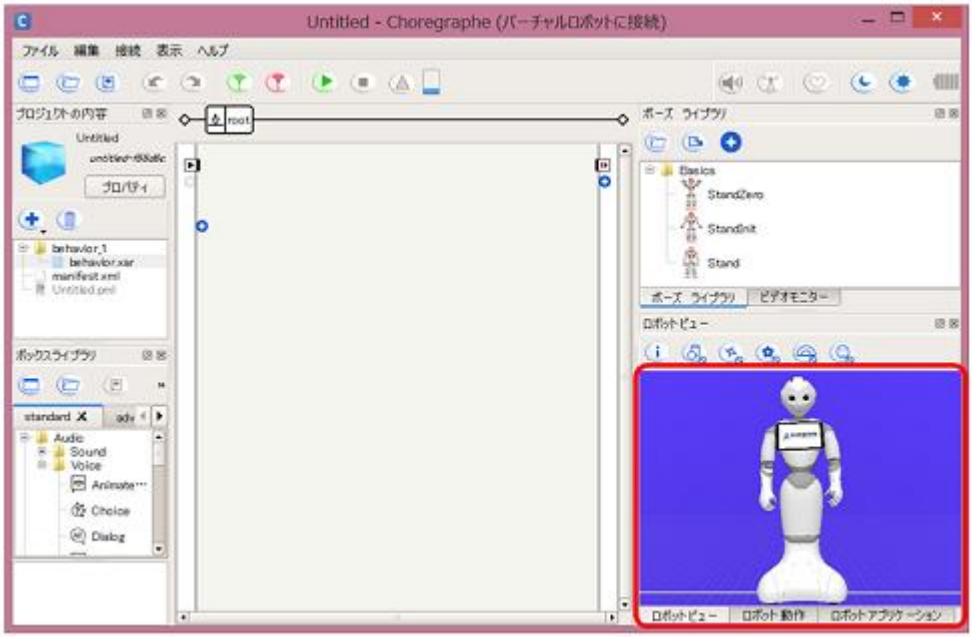



3. ロボットモデルとして **JULIETTEY20MP.xml を選択 [A]** し、**[OK]ボタンをクリック [B]**します


4. バーチャルロボットを再起動してよいか問い合わせるダイアログが表示されますので、**[はい]をクリック**します

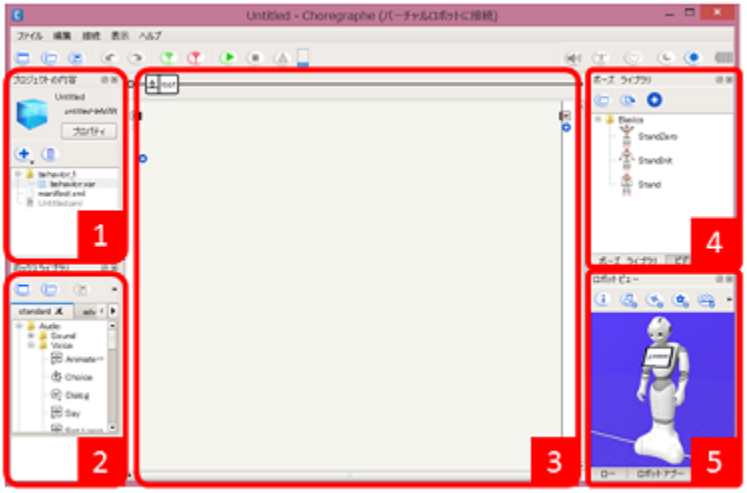



これで、バーチャルロボットが **Pepper** になりました。それでは、例を通して実際にプロジェクトを作成していきましょう。



(参考)SDK の各部説明

Choregraphe は複数のパネルから構成されます。よく使うパネルは以下の 5 つです。




1. プロジェクトの内容
2. ボックスライブラリ
3. フローダイアグラム
4. ポーズライブラリ
5. ロボットビュー

また、メニューには以下の項目があります。

- ファイル
- 編集
- 接続
- 表示
- ヘルプ

ツールバーとして、以下のようなボタンがあります。

他にも、 [表示] メニューから以下のようなパネルを選択することができます。

- ビデオモニター
- ダイアログ
- スクリプトエディタ
- ロボットアプリケーション
- ロボット動作
- リソースビューアー
- メモリウォッチャー
- ログビューアー
- スタックのやり直し

それぞれの使い方については別のチュートリアルで説明していきます。

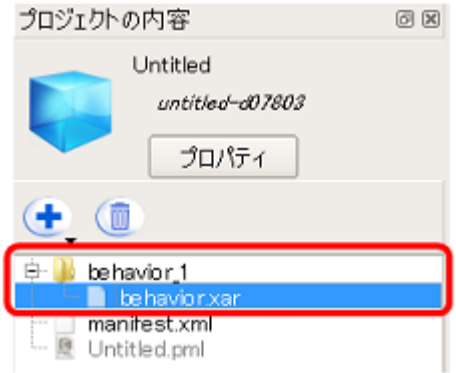
# アプリケーションの作成と実行

## プロジェクトの作成

1. 新たに プロジェクト (アプリケーションの生成に必要なファイルの集まり) を作成します
    - **Choregraphe 起動直後** →[ようこそ！]ダイアログから **[新規プロジェクト… ]** を選択
    - **それ以外** →[ファイル]メニューから **[新規プロジェクト…]** を選択
2. フローダイアグラムに 空のフローが表示されることを確認します

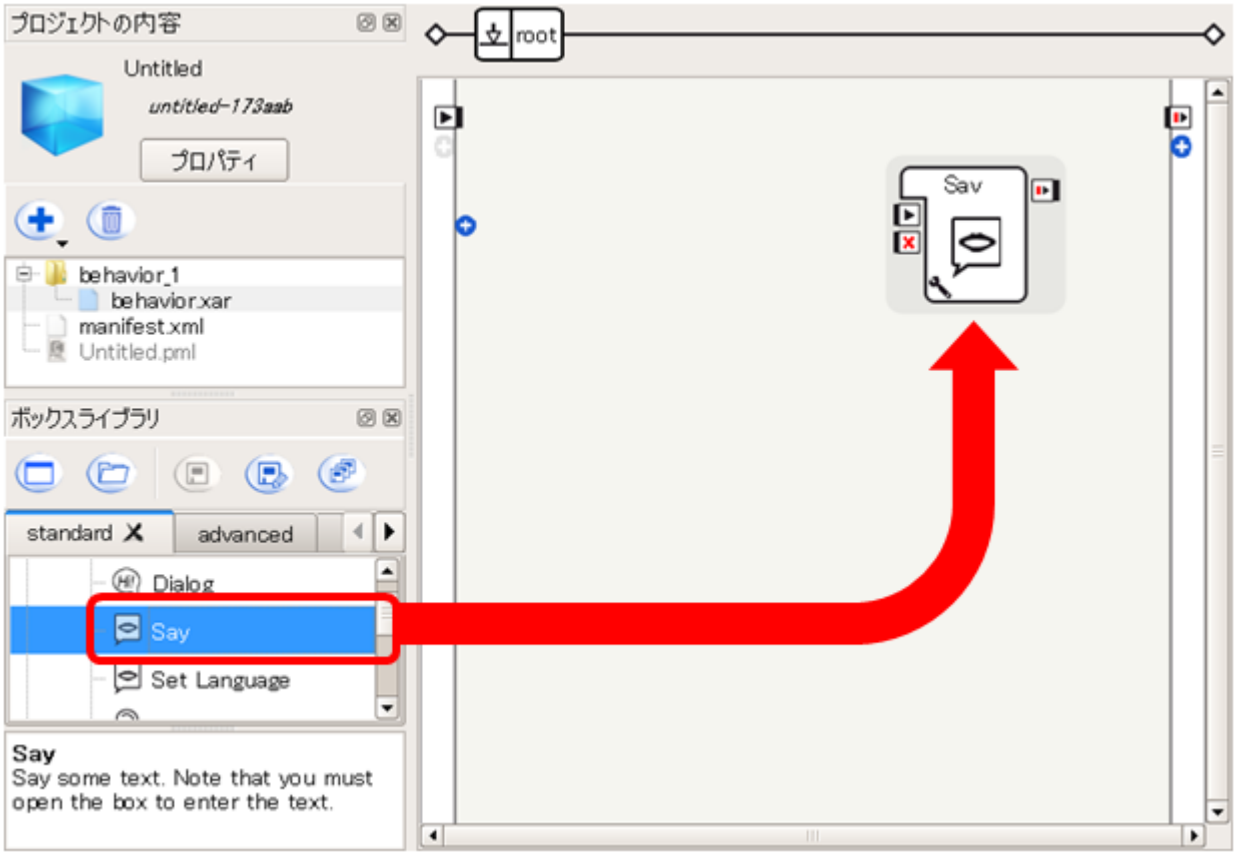

プロジェクト/ビヘイビア/ボックス

- 新規プロジェクトには１つだけビヘイビア **(振る舞いを定義する単位)** が用意されます
- ビヘイビアは複数のボックス **(ビヘイビアの機能単位：“Say”など)** によって構成されます
    - これらのボックスは順序実行/同時実行されます
- プロジェクトに含まれるビヘイビアは [プロジェクトの内容] パネルで確認できます

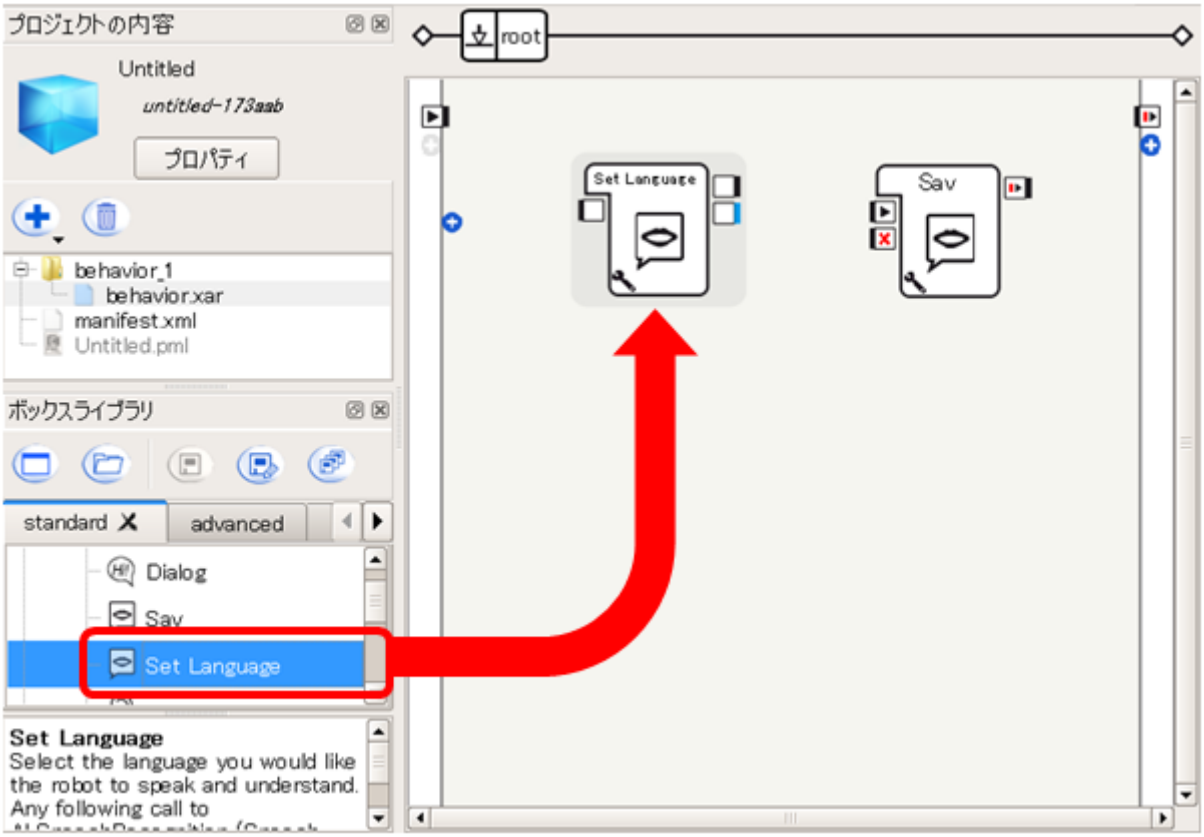


## ボックスの配置

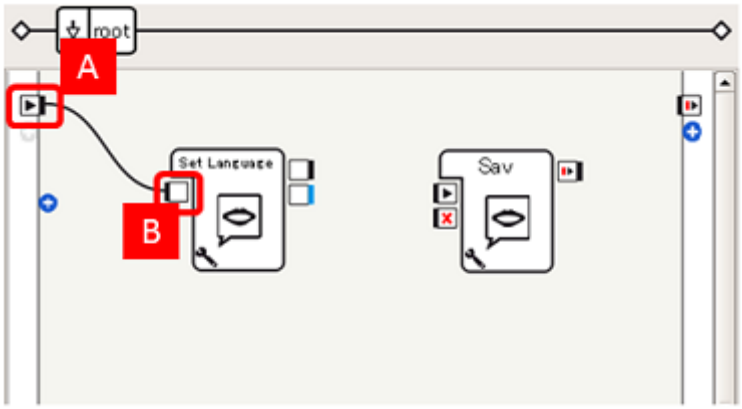
1. [ボックスライブラリ]パネルの **"Say" ボックス** を **[フローダイアグラム]パネル** へとドラッグ &ドロップします



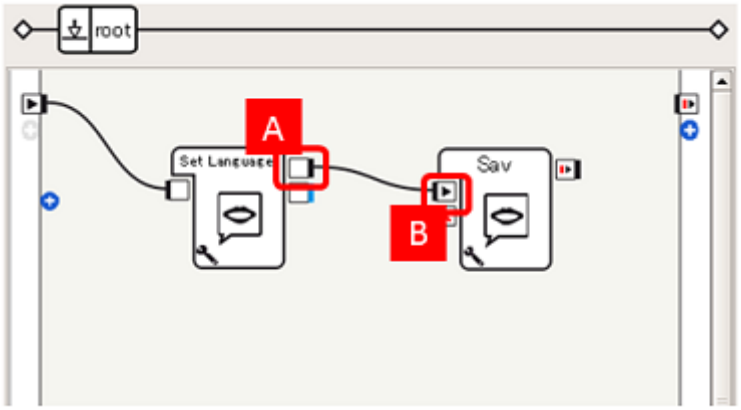

2. [ボックスライブラリ]パネルの **"Set Language" ボックス** を **[フローダイアグラム]パネル** へとドラッグ &ドロップします



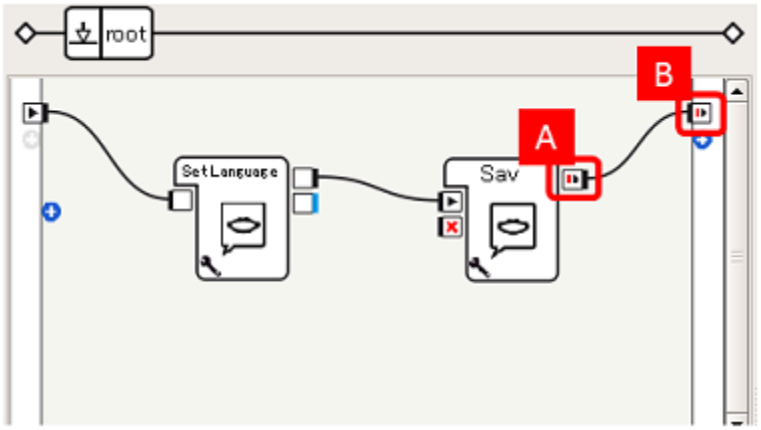

3. **フローダイアグラム左端の[onStart 入力][A]** から **Set Language ボックスの[onStart 入力][B]** をマウスドラッグでつなぎます




4. **Set Language ボックスの[onReady 出力][A]** から **Say ボックスの[onStart 入力][B]** をマウスドラッグでつなぎます




5. **Say ボックスの[onStopped 出力][A]** から **フローダイアグラム右端の[onStopped 出力][B]**をマウスドラッグでつなぎます
   基本的には、フローダイアグラム右端の onStopped 出力をフローの最後に実行されるボックスとつなぐようにしてください。このようにすることで、このボックスの停止とともにアプリケーションが停止することを明示することができます。




6. **Set Language ボックスのパラメータボタン** をクリックします


7. Language 変数に Japanese を設定 **[A]** し、**[OK]ボタン[B]** をクリックします


8. [ファイル]メニューの **[プロジェクトを名前をつけて保存...]** を選択します
    - 現在編集しているビヘイビアなどの情報を保存することができます

これで、アプリケーションの準備は完了です。次に、ロボットでアプリケーションを実行してみましょう。

## バーチャルロボットでの実行

まずは、SDK にあるバーチャルロボットを利用して動作確認をしてみます。

1. [接続]メニューの **[バーチャルロボットに接続]** をクリックします
    - [ロボットビュー]パネルにバーチャルロボットが表示されます


2. [表示]メニューの **[ダイアログ]** をクリックします
    - Choregraphe の画面に [ダイアログ]パネル が表示されます


3. ツールバーの **[ロボットにアップロードして再生]** をクリックします

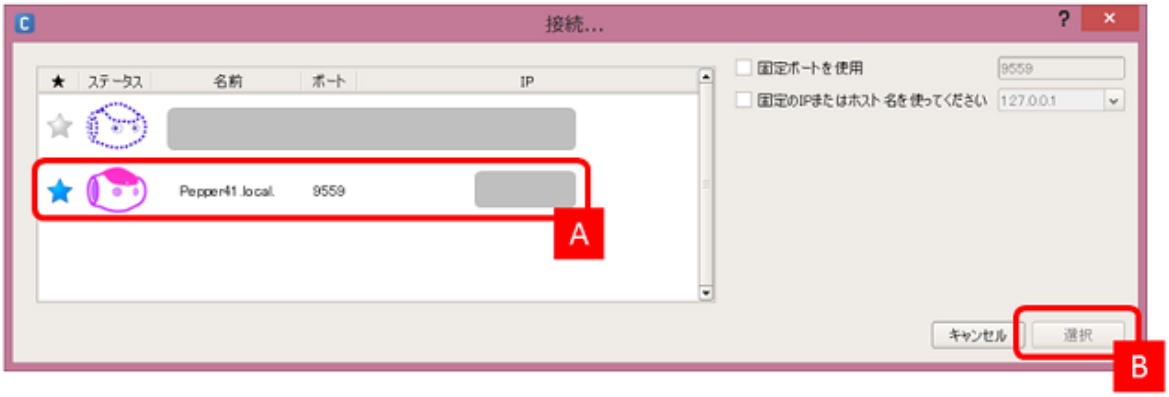
4. [ダイアログ]パネルに **ロボット: こんにちは** と表示されれば成功です


## Pepper での実行

次に、実際に Pepper にしゃべらせてみましょう。

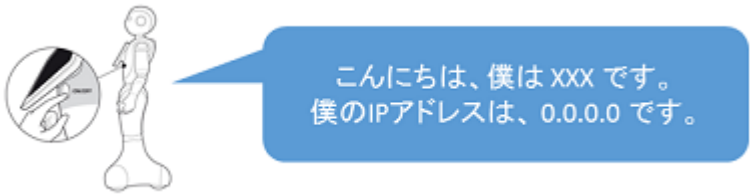
1. ツールバーの **[接続…]** をクリックします

2. 接続可能な Pepper の一覧が表示されます
   - 自分の Pepper をクリック **[A]** し、**[選択][B]** ボタンをクリックしてください

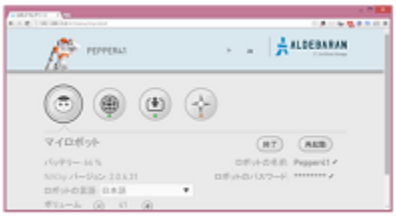



(Tips)Pepper の IP アドレスを知る方法

Pepper のチェストボタンを押すと、自分の情報をしゃべります




ブラウザでこの IP アドレスを開くと、より詳細な情報を得たり、停止、再起動などを行うことができます



3. **[ロボットにアップロードして再生]** をクリックします

4. Pepper が **こんにちは** としゃべれば成功です


簡単なアプリケーションですが、この、ボックスを置く、ボックスをつなぐ、が Pepper アプリケーション開発の基本になります。

# Pepper チュートリアル (2):ボックスの考え方

## このチュートリアルの内容

チュートリアル(1) では、ボックスをつないでいくという操作で、Pepper にしゃべらせるということをおこないました。

このチュートリアルでは、Choregraphe での開発の中心となる **ボックス** について、それがどのような考え方で構成されているのかを具体的な例を通じて説明します。

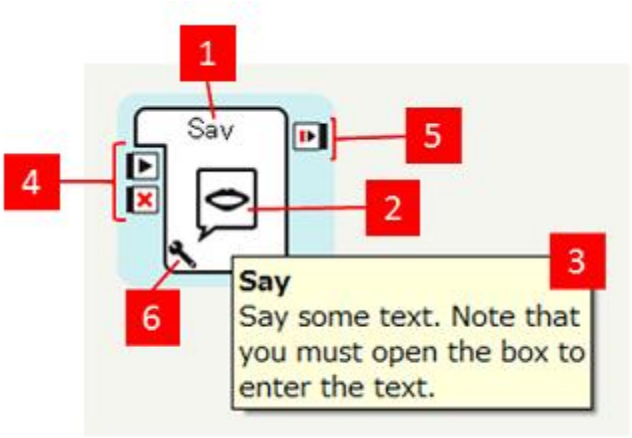
1. ボックスの構成要素
2. 標準ボックスの利用
3. 独自ボックスの作成

## ボックスの構成要素

ボックスはアプリケーションの振る舞いの定義(フロー)の基本構成要素で、Say のような単純な機能から、部屋を探索するような複雑な機能まで幅広い機能まで、さまざまな機能を表現する部品です。

ボックスは、標準でボックスライブラリとして使用できるものもありますし、利用者がボックスを独自に定義したり、第三者が定義したボックスをボックスライブラリにインポートして利用することもできます。

ボックスは以下のような要素から構成されます。




1. ボックスの名前
2. ボックスイメージ
3. ボックスの説明 ... ボックスでマウスオーバーすることで確認することができます
4. 入力 ... ボックスを起動したり停止したりするためのシグナルや、データを受け取るためのコネクタです
5. 出力 ... ボックスからのデータや、終了を示すシグナルを出力するためのコネクタです
6. パラメータボタン ... ボックスに関連するデータを設定することができます

## 入力と出力

**ボックスの入力と出力には以下のようなものがあります。**

### アイコンによる種類

**入力、出力にはアイコンがついています。**

- 入力
  - ボックスの開始(onStart)。この入力にシグナルが送られると、ボックスは開始状態になります。開始状態での振る舞いはボックスの種類により異なります。
  - ボックスの停止(onStop)。この入力にシグナルが送られると、ボックスは停止状態になります。
  - 開始/停止以外の入力(onEvent)。ボックスによりこの入力に対する挙動は異なります。
- 出力
  - ボックスの停止(onStopped)。この出力からシグナルが送られた場合、ボックスが停止したことを意味します。
  - ボックスからの出力(punctual)。ボックスによりこの出力の意味は異なります。

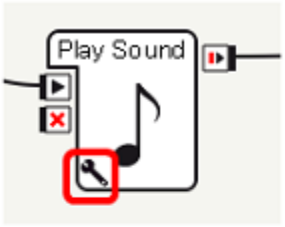
### 色による種類

入力、出力ともに色がついています。色により、この入力がどのようなデータを受け取るものなのか、あるいは出力がどのようなデータを引き渡すものなのかがわかります。

- 単純イベント(Bang)。この入出力ではデータはともないません。
- 数値(Number)。この入出力では数値(小数点値もしくは整数値)が引き渡されます。
- 文字列(String)。この入出力では文字列が引き渡されます。
- 動的(Dynamic)。単純イベント(値なし)あるいは何らかの値あり。値がある場合は、数値、文字列、数値の配列、文字列の配列のいずれかとなる。

## パラメータ

ボックスの設定のうち、変数として定義されているものはパラメータボタンで確認、変更することができます。
パラメータボタンを押すと、以下のようなダイアログが現れます。





設定可能な内容はボックスにより異なります。具体的にはチュートリアルを通じて説明していきます。

## ボックスの内容

ボックスの構成方法により、以下の 4 種類のボックスがあります。

1. Python ボックス：Python によって記述されたボックスです
2. フローダイアグラムボックス：ボックスとそれらの接続の組み合わせをひとつのボックスにまとめたボックスです
3. ダイアログボックス：QiChat script という、Pepper との対話を定義するスクリプト言語によって記述されたボックスです
4. タイムラインボックス：1 つの時間軸に沿ってポーズやフローなどの動きを並べたボックスです

このチュートリアルでは、まずはフローダイアグラムボックス、タイムラインボックスについて説明します。Python ボックス、ダイアログボックスについては別のチュートリアルで説明していきます。

# 標準ボックスの利用

ボックスでできることを、標準のボックスライブラリを使って説明していきます。

# ボックスライブラリの種類

標準のボックスライブラリには以下のようなカテゴリが用意されています。

- standard ... 基本的なボックス群
- advanced ... 高度な機能を実現するために必要なボックス群
- tablet ... タブレットを持つ(Pepper のような)ロボット用ボックス群

このチュートリアルでは、standard なボックス群を利用して説明していきます。

# フローの組み立て

具体的なフローの組み立てを通じて、どんなボックスがあるのか、どんなボックスのつなぎ方があるのかを説明していきます。それぞれ、Pepper の実機がなくても動作確認できるよう、バーチャルロボットでの確認方法をあわせて説明します。

## 順序実行

ボックスを複数、順番につなぐことで、あるボックスの処理が終わってから、別のボックスの処理に移ることができます。
ここでは、**前に進ませてから、「こんにちは」としゃべらせる** ことをやってみます。

### つくってみる

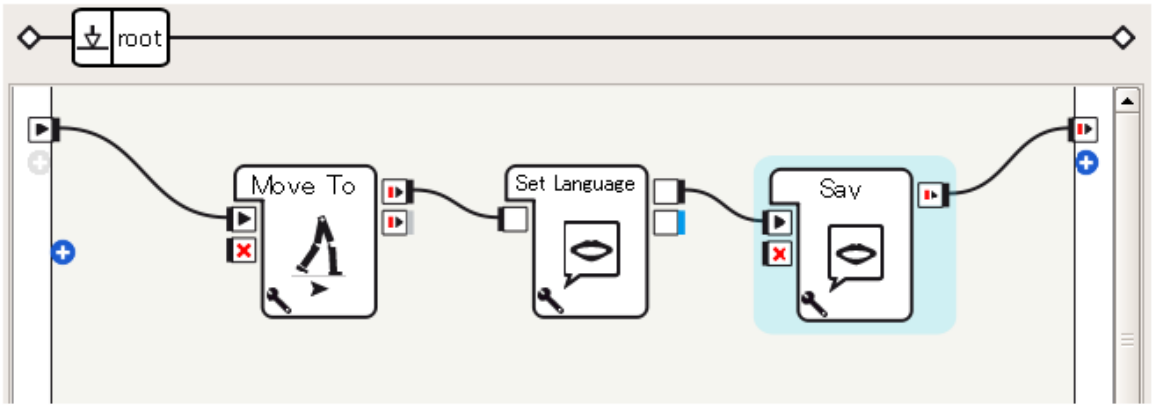
1. 利用するボックスの準備

ボックスライブラリの standard タブから、以下のボックスをフローダイアグラムにドラッグ &ドロップしてください。

- Motions > Move To ... 現在位置から指定座標だけ移動する
- Audio > Voice > Set Language ... 言語設定を変更する
- Audio > Voice > Say ... こんにちはと言う

ボックスライブラリは階層構造になっており、Motions, Audio など、ボックスの種類によって分類されています。

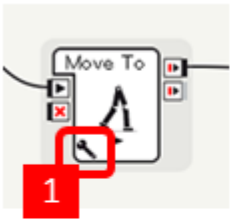
2. ボックスを順番につなげる

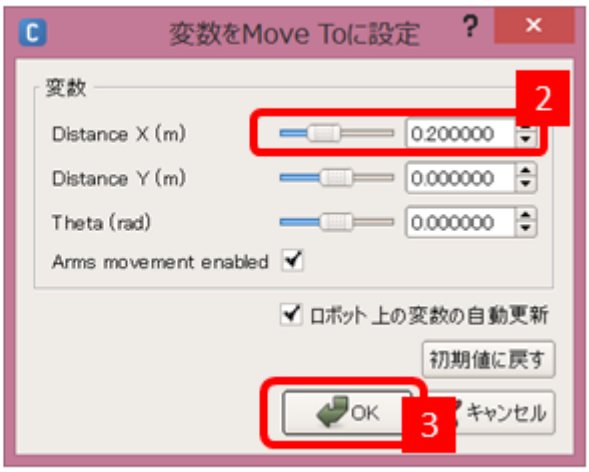



3. 言語を設定する
   - Set Language ボックスによって言語設定を日本語へと変更します。パラメータボタンをクリック[1] し、言語に Japanese を設定 [2]、[OK]ボタンをクリック [3] します。



4. 移動する距離を設定する
   - Move To ボックスでどの程度移動するかをパラメータダイアログで変更します。パラメータボタンをクリック[1] し、X 方向の移動距離に 0.2 (m) を設定 [2]、[OK]ボタンをクリック[3] します。

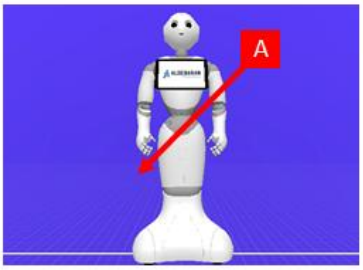


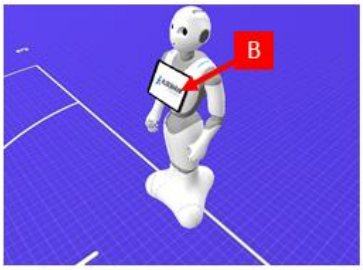



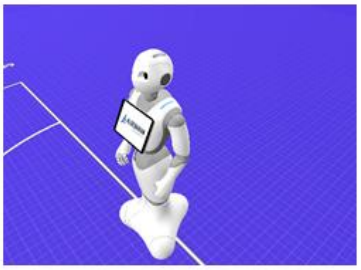

このように、onStart と onStopped を接続することで、 アプリケーションの開始→Move To が開始→Move To が終了(20cm 前に移動することが完了)→Say が開始 という順序実行が実現できます。

## バーチャルロボットでの動作確認

1. [接続]メニューの [バーチャルロボットに接続] をクリックします
   - [ロボットビュー]パネルにバーチャルロボットが表示されます
2. **ロボットビュー上で マウス右ボタン を押しながら [A] のようにマウスをドラッグしたり、左ボタンを押しながら [B] のようにドラッグすることで、ロボットビューの視点を移動します**
   - このようにすることで、移動の様子がわかりやすくなります

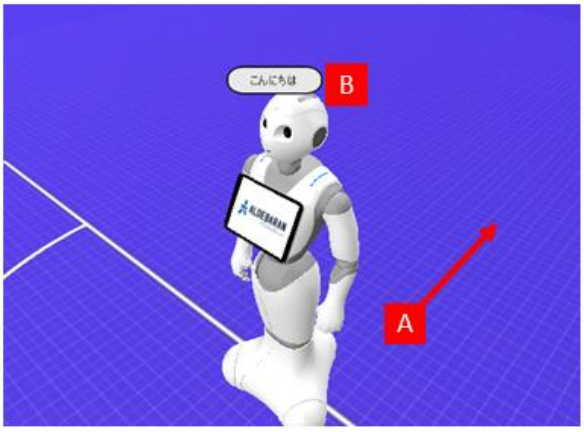


3. [表示]メニューの [ダイアログ] をクリックします
   - Choregraphe の画面に [ダイアログ]パネル が表示されます
4. ツールバーの [ロボットにアップロードして再生] をクリックします

5. バーチャルロボットが前面に移動した [A] (ロボットは画面中央に固定されるので、相対的に床が後ろに動く)後、ロボットビューとダイアログパネルに こんにちは と表示され [B] れば成功です





Pepper での動作確認

1. ツールバーの **[接続…]** をクリックし、しゃべらせたい Pepper と接続します

2. ツールバーの［ロボットにアップロードして再生］をクリックします

3. Pepper が前面に動いてきて、「こんにちは」と言えば成功です

[Tips]動かないときは

周りに障害物などがあり、Move To が正常に終了しなかった場合(つまり、20cm より短い距離で止まってしまった)に、次のボックスが起動されないことがあります。
このような場合は、以下のような点を疑ってみてください。

**Rest(おやすみ)中**
Pepper が以下のような状態になっているときは、おやすみ中です。

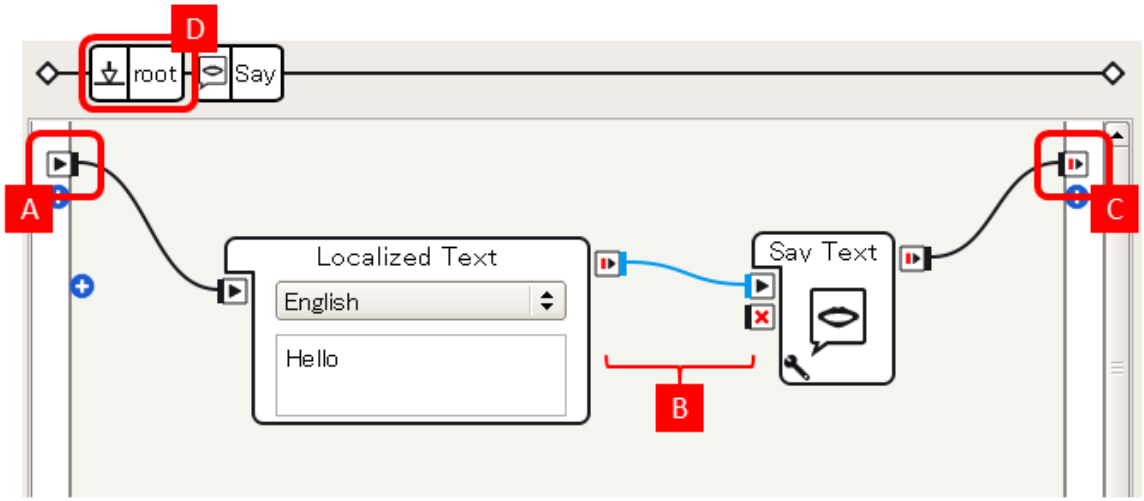
このような場合は、[Wake Up]をクリックして起こしてあげてください。

**センサーによるセーフティ**
Pepperは足元にレーザーとソナーによるセンサーを持っており、移動先に障害物などがある場合は移動を行わないなどのセーフティ機構がはたらきます。
このような場合、障害物を取り除くなどの対応をしてください。

## ボックスをカスタマイズしてみる

**ここまでで利用してきたSayボックスですが、実際には、いくつかの基本的な機能を持つボックスを組み合わせたものです。**
**これを編集して、「やあ」としゃべらせてみます。**

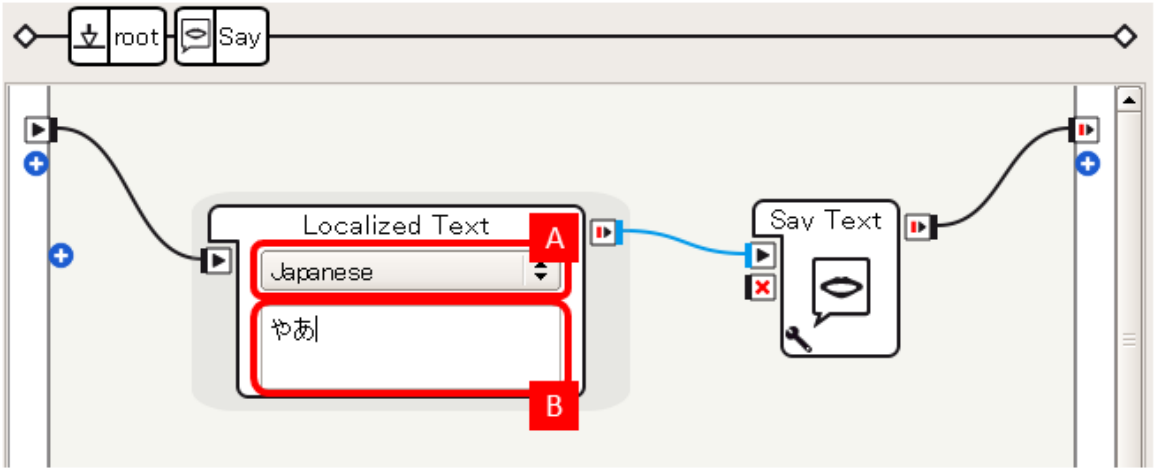
1. フローダイアグラム上のSayボックス(入出力、パラメータボタン以外)をダブルクリックします
2. **Localized Textというボックスと Say Textというボックスが現れます。これは、Sayボックスは複数のボックスを組み合わせた フローダイアグラムボックス であるということを意味しています**




- Sayボックスに onStart が入力されると、Sayボックス内部が開始状態となり、ボックス内の onStart **[A]** からシグナルが送られます
- Localized Textボックスは、現在の言語設定(この例では Set Language によって設定されたもの)にしたがって入力された文字列を、Say Textボックスに送ります **[B]**
- Say Textボックスが停止すると、そのシグナルはボックス内の onStopped **[C]** にシグナルが送られ、Sayボックス全体が停止状態になります

**※なお、フローダイアグラム上の [root] [D] をクリックすると元のフロー(ビヘイビア)の表示に戻れます**

3. **Localized Textの言語を Japaneseに切り替え、「こんにちは」となっていた文字列を「やあ」に変更します**




4. ［ロボットにアップロードして再生］ で、しゃべる内容が変わったことを確認します

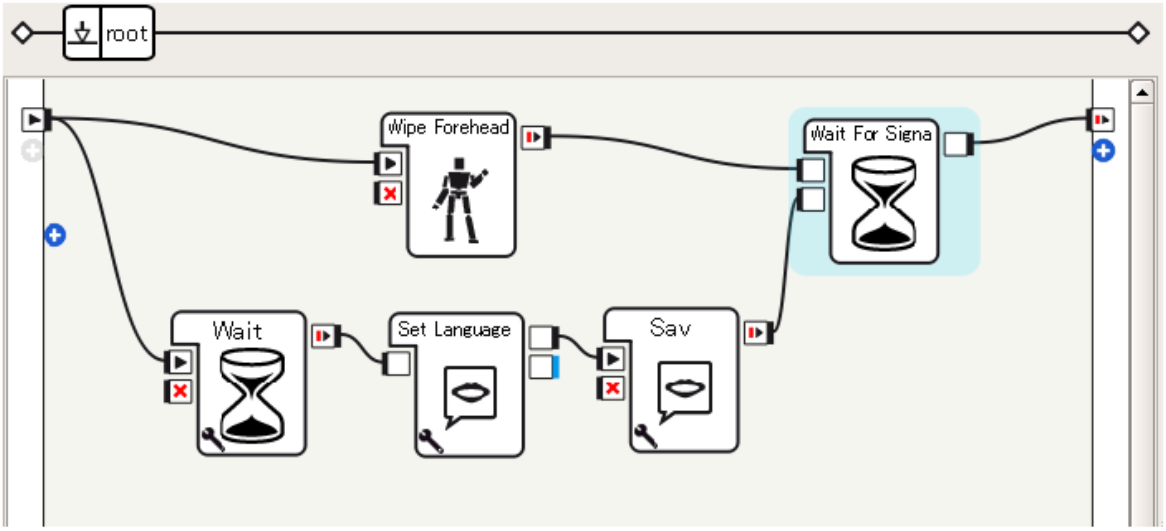
ボックスはコピーをして増やすことができますが、ペーストしたボックスはコピー元のボックスとは異なるボックスになるので、それぞれを自由に変更することができます。

## 並列実行

シグナルを二又に複数の onStart に入力させると、複数のボックスを同時に実行することができます。ここでは、**汗をぬぐうモーションを再生しつつ、モーション開始後 3 秒後に「ふう」としゃべらせる** ことをやってみます。

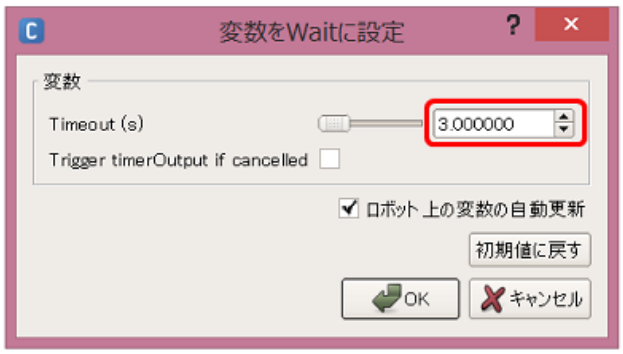
### つくってみる

1. 利用するボックスの準備
   - Motions > Animations > Wipe Forehead ... 汗をぬぐうモーション
   - Flow Control > Time > Wait ... 一定時間待つ
   - Audio > Voice > Set Language ... 言語設定を変更する
   - Audio > Voice > Say ... こんにちはと言う
   - Flow Control > Wait For Signals ... 2 つのシグナルが両方とも入力されるまで待つ
2. ボックスをつなげる
   - onStart と同時に、Wipe Forehead ボックスと Wait ボックスを開始し、Wait ボックス側は 3 秒待ったら Say ボックスに処理を渡します。Wipe Forehead と Say ボックスの両方が終わったら、このビヘイビアの終了とします
   - **Wait For Signals ボックス**は、2 つの入力両方にシグナルが入力されると、自身からシグナルを出力しますので、これを停止条件の制御に利用しています



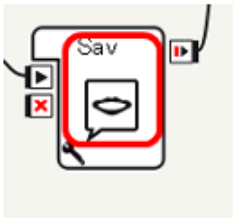

3. パラメータを設定する
   - Wait ボックスの Timeout には 3 (秒)を、Set Language ボックスには Japanese を、それぞれ設定します

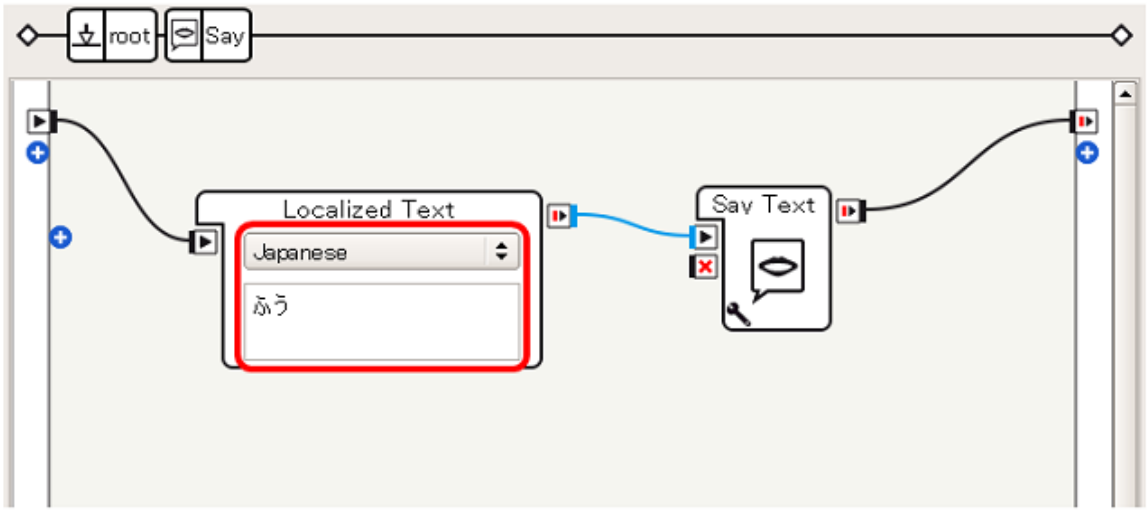






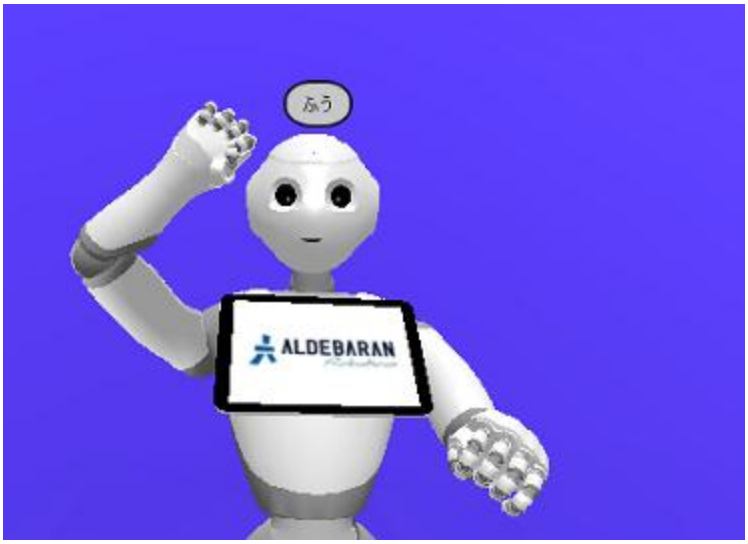
4. Say の文字列を設定する
   - Say ボックスをダブルクリックし、Localized Text を Japanese にし、内容は「ふう」と入力します




このようにすることで、モーション開始の onStart と同時に Wait ボックスが開始され、3 秒後に onStopped シグナル発生、Say ボックスが開始されるようになります。

動作確認方法は先のチュートリアルと同様です。汗をぬぐうようなモーションが実行され、指定されたタイミングで「ふう」と言うことを確認してみてください。

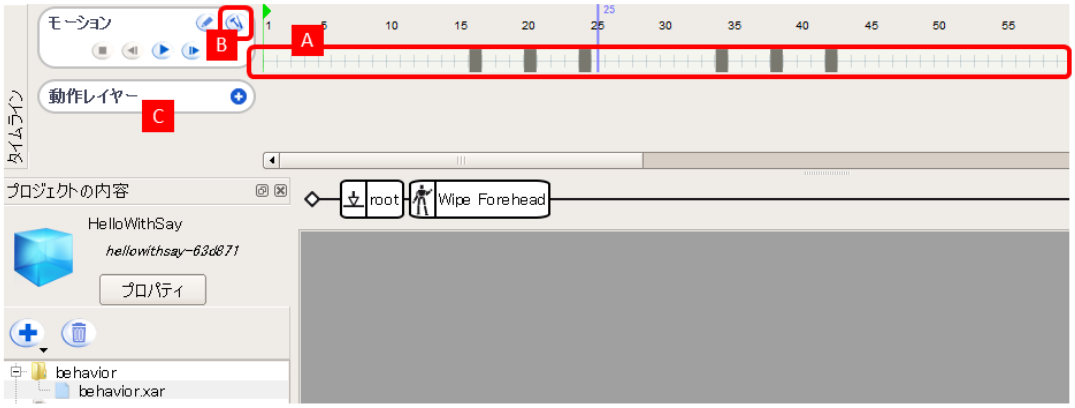



## カスタマイズしてみる

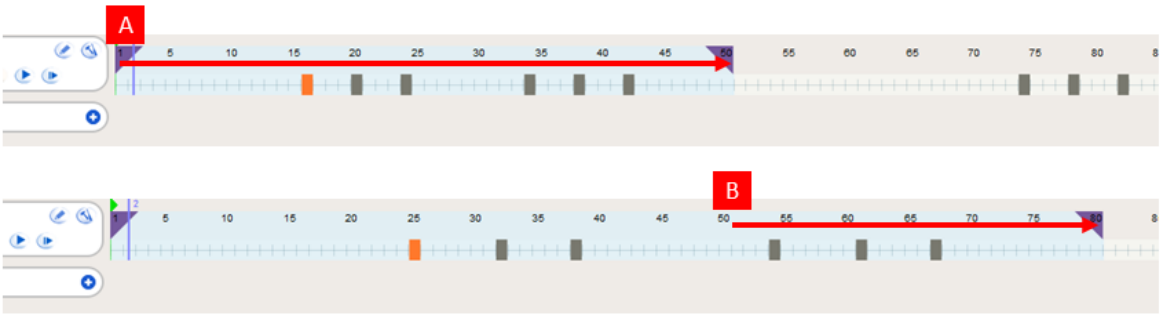
Wipe Forehand ボックスは **タイムラインボックス** という形式のボックスで、時間ごとにモーションやビヘイビアを組み合わせた形で表現されています。このタイムラインに修正を加えて、汗をぬぐうはじめのモーションを少し遅くしてみましょう。

1. Wipe Forehand ボックスをダブルクリックします

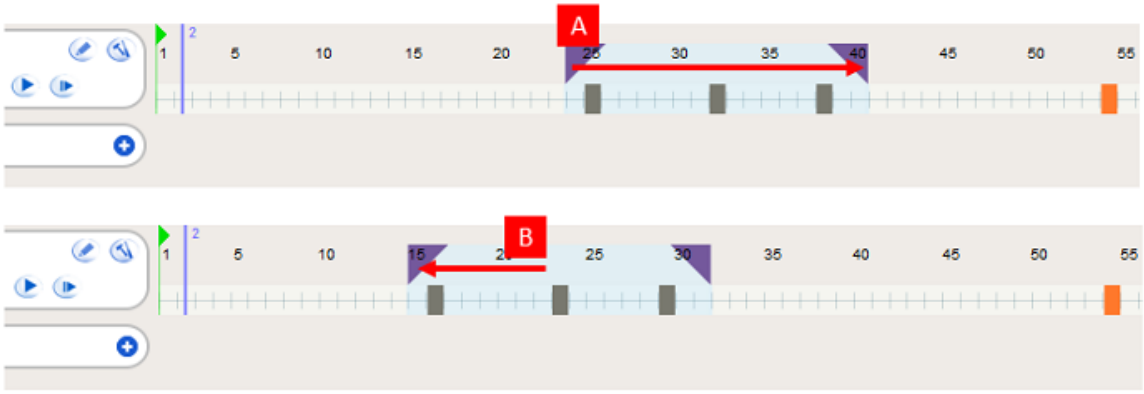
2. 以下のようなタイムラインパネルがあらわれます
    - タイムライン [A] はモーション(ポーズの変化)の時間進行をあらわします。タイムライン中のグレーの箇所は、モーションキーフレームと呼び、それぞれに、その時点で Pepper がとるべきポーズが記録されています。(キーフレーム間の動きは、なめらかにつながるように自動的に補間されます)
    - タイムラインの単位はフレームで、このタイムラインにおいて 1 秒間に何フレームを実行するかは、プロパティ [B] により設定できます。通常は 25 フレーム/秒です
    - **動作レイヤー [C] は、(このボックスにはありませんが、)タイムラインとあわせて動作させたいビヘイビアを定義したいときに利用します**
        - プロジェクトのビヘイビア(ボックスをつなぎ合わせたフロー)はプロジェクトの再生とともに開始状態になりますが、タイムラインの動作レイヤーとして定義することで、特定のフレームに到達したら開始状態にするフローを定義することができます



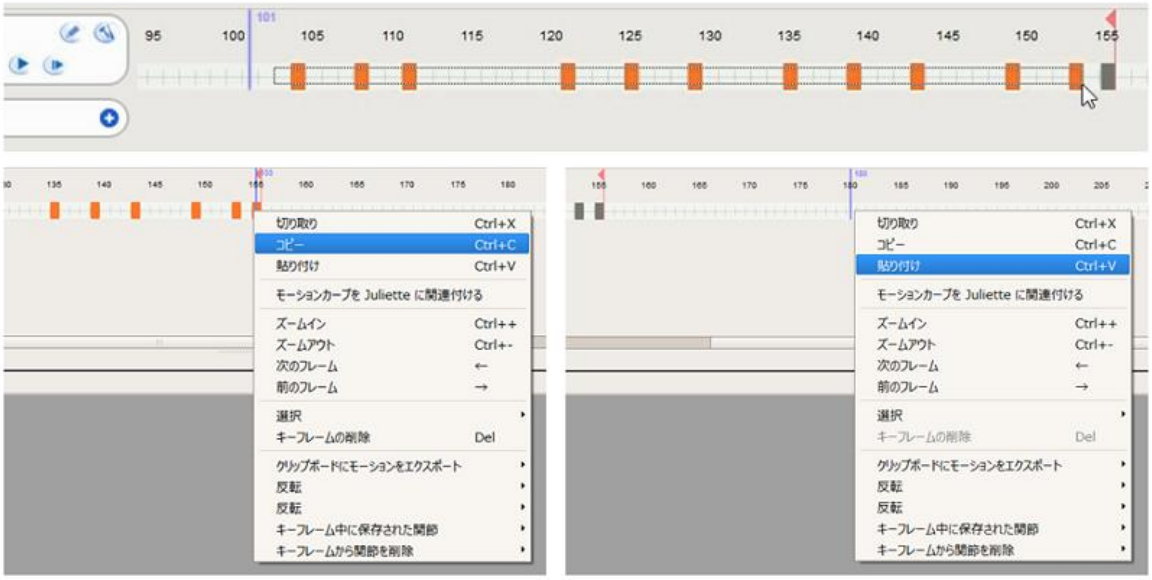

3. タイムラインの一部をマウス左ボタンドラッグにより選択し [A]、選択領域右端をマウスでドラッグ [B] します。すると、タイムラインが引き延ばされます




4. また、タイムラインの一部をマウス左ボタンドラッグにより選択し [A]、選択領域左端をマウスでドラッグ [B] します。すると、タイムラインを移動することができます




5. タイムライン上、特にキーフレームが配置される領域上を範囲選択することも可能です。以下のようにマウスドラッグすることで、複数キーフレームを選択することができます。この後、右クリックするとコピーすることができ、他の領域に貼り付けなどおこなうことができます。



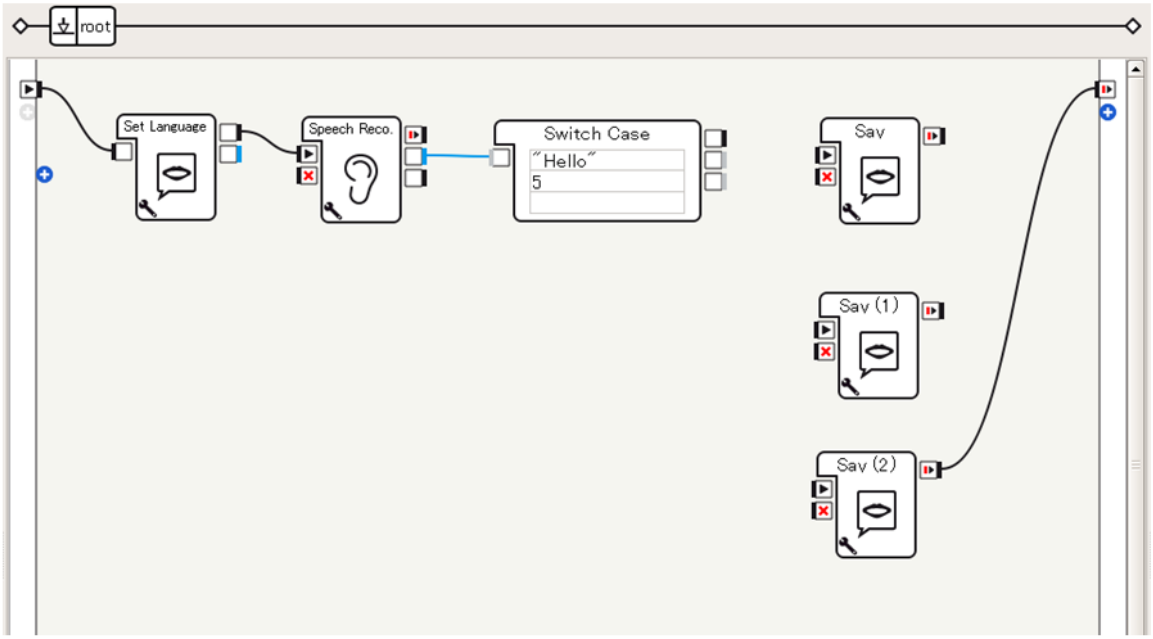

動作確認をしてみると、モーション初期の動きがすこしゆっくりになったことが確認できるはずです。このように、モーションはタイムラインの形で編集、管理します。

## 制御構造

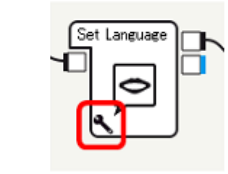
値を出力するボックスの場合、その値によって続くフローを切り替えることができます。
ここでは、**言うことを聞き取り、「いちご」ならば「あかい」と言い、「ばなな」ならば「きいろい」と言う** ことをやってみます。なお、このアプリケーションは、**「おしまい」と言われたら「ばいばい」と言って停止する** ことにします。

つくってみる

1. **利用するボックスの準備**
   - Audio > Voice > Set Language ... 言語設定を変更する
   - Audio > Voice > Speech Reco. ... 音声認識をおこない、パラメータで与えられた単語候補のいずれかを判別する
   - Flow Control > Switch Case ... 入力された値に対して、条件に応じて異なる出力をおこなう
   - Audio > Voice > Say ... こんにちはと言う (3 回ドラッグ＆ドロップ)
2. **ボックスをつなぐ**



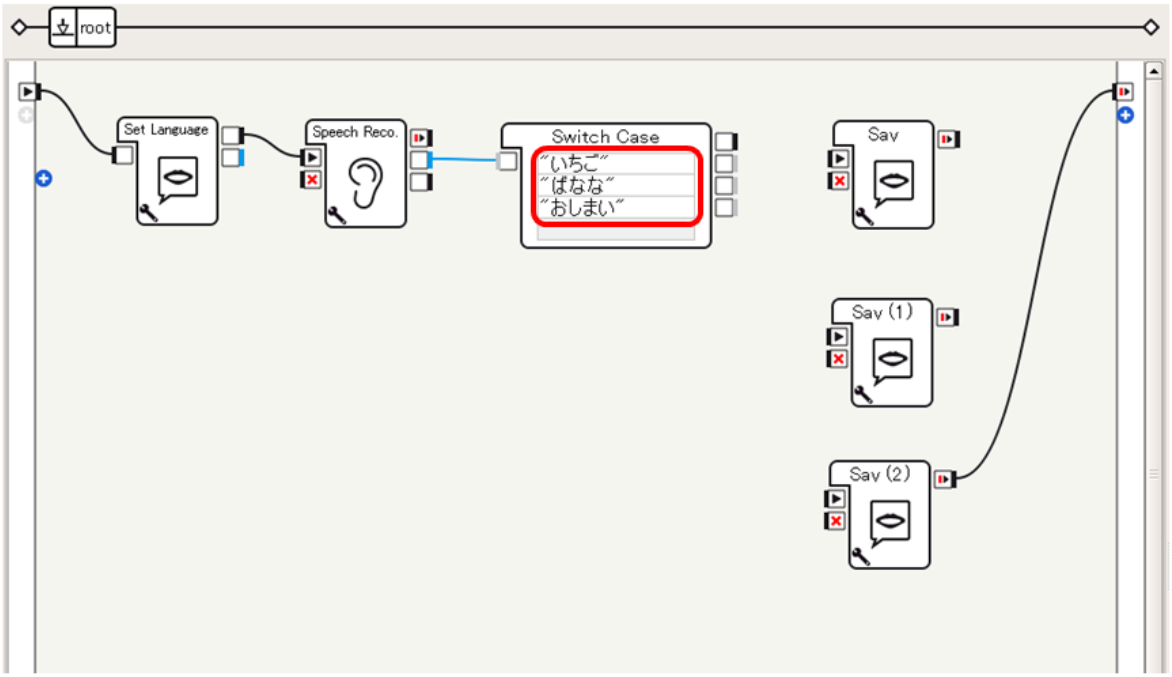

   - Set Language ボックスで言語設定をしたのち、Speech Reco.ボックスへと接続します
   - Speech Reco.ボックスは音声認識処理を実装するボックスで、話しかけられると候補の単語から最も候補としてふさわしいものを選択し、文字列として出力します
   - この出力を、条件分岐機能を提供する Switch Case ボックスで分岐させ、「あかい」と言う Say ボックス、「きいろい」と言う Say ボックス、「ばいばい」と言う Say ボックスへと接続することで、話者への応答を実現します。これらのボックスの設定、接続はこの先の手順でおこないます
   - 3 つめの Say ボックス(「ばいばい」と言う Say ボックスに変更する予定)が停止したらアプリケーションが停止するように、Say ボックスの onStopped 出力をビヘイビアの onStopped 出力に接続します
3. **パラメータを設定する**
   - Set Language ボックスにはいつも通り Japanese を、Speech Reco.ボックスには いちご;ばなな;おしまい (セミコロンで区切る)を設定します。これにより、Speech Reco.ボックスは、話しかけられた内容に対して「いちご」か「ばなな」か「おしまい」かのいずれかを判断するように設定されます



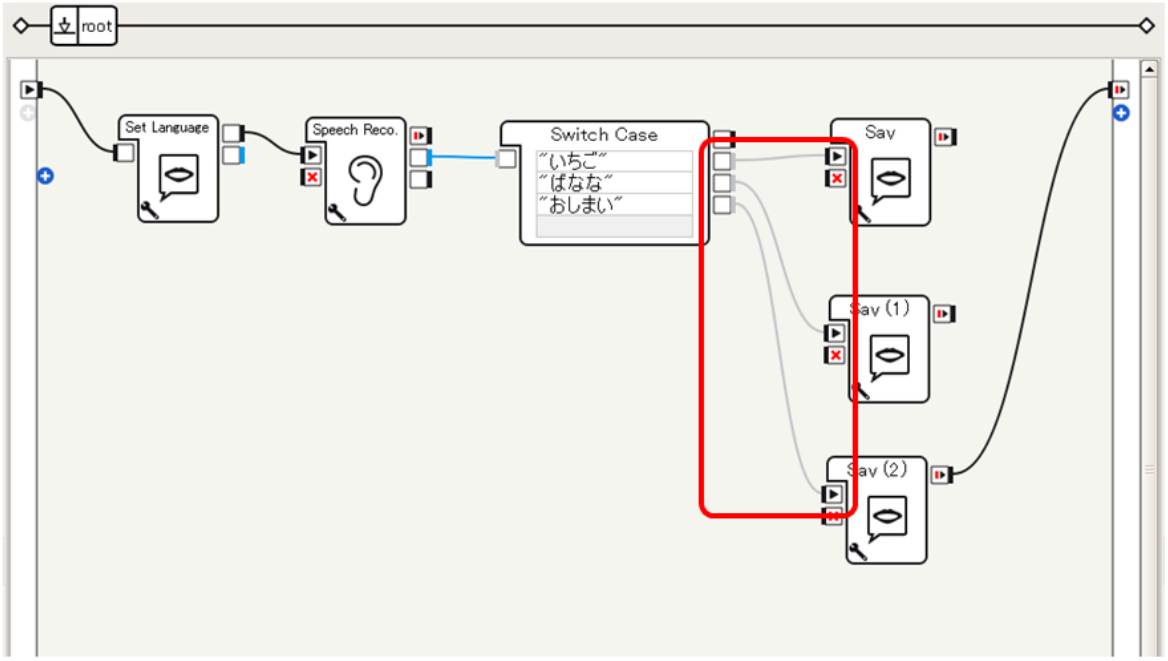




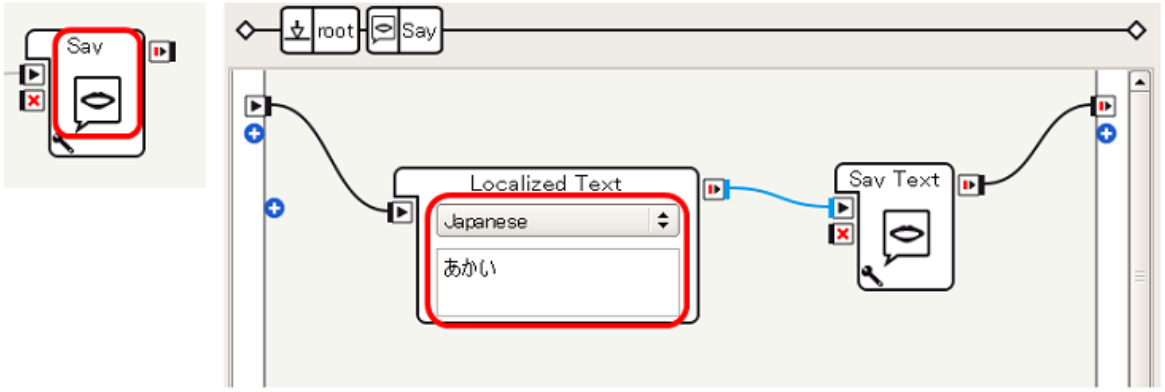
4. 条件を設定する
   Switch Case に "いちご", "ばなな", "おしまい" (ダブルクォーテーションで囲む)を設定します。

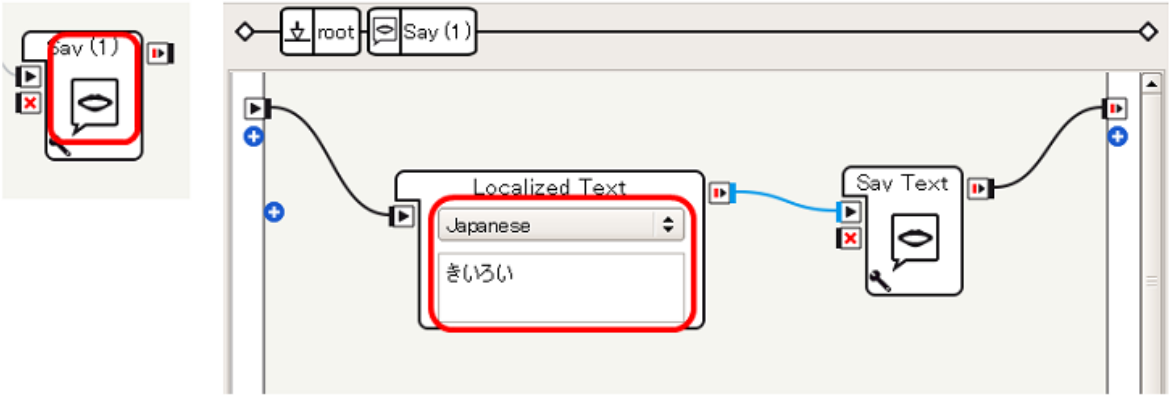


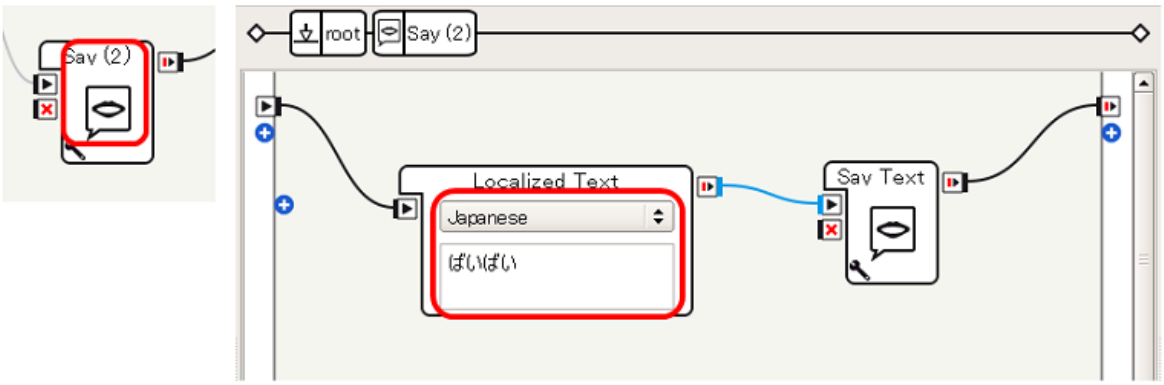

   すると、Switch Case ボックスのグレー(Dynamic)出力が 2 つから 3 つに増えます。Speech Reco. から渡された文字列が「いちご」であれば output_1 出力に、「ばなな」であれば output_2 出力に、「おしまい」であれば output_3 出力にシグナルを送るように Switch Case ボックスが設定された状態になります。
5. Switch Case ボックスの 3 つの出力をそれぞれ以下のように 3 つの Say ボックスへと接続します




6. Say の内容を変更する
   - Switch Case の output_1 につないだ Say は、Localized Text に Japanese を設定したうえで あかい とし、output_2 に接続された Say は きいろい とし、output_3 に接続された Say は ばいばい とします



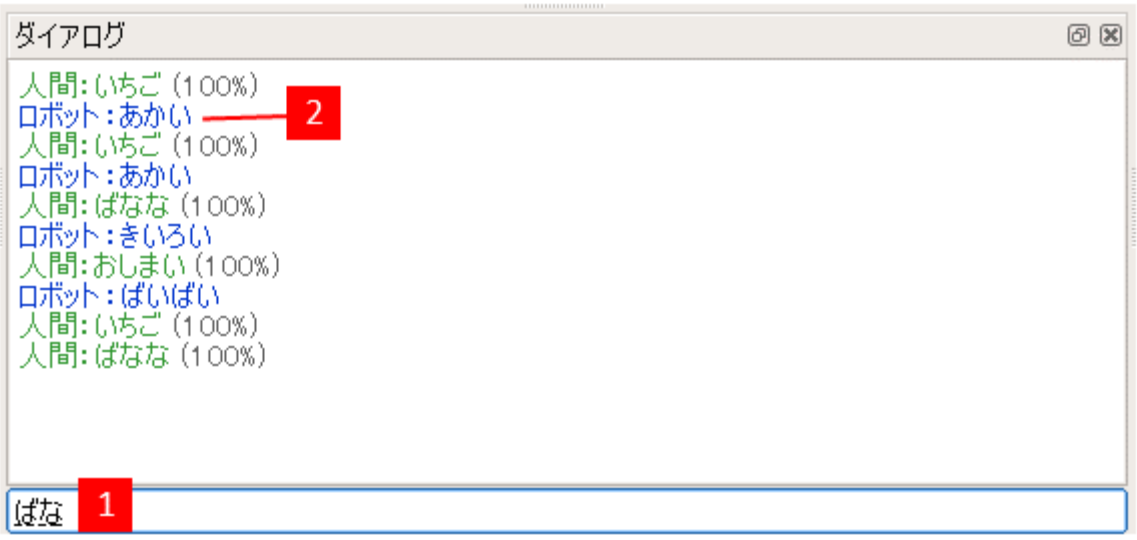



これで、単純な **音声の認識→条件分岐→音声による応答** を実現することができます。

### バーチャルロボットでの動作確認

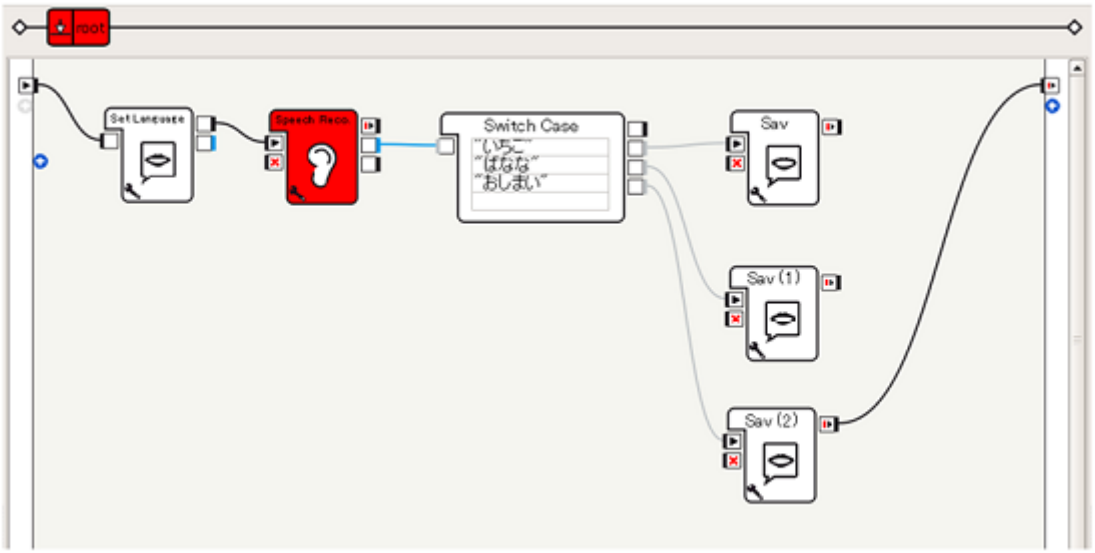
バーチャルロボットは音声出力と同様、ダイアログパネルによって文字列ベースで対話することができます。

先のチュートリアルと同様にバーチャルロボットで再生したのち、ダイアログパネルに単語を入力する [1] と、フローダイアグラムにしたがった応答が確認できます。[2]
また、「おしまい」と入力すると「ばいばい」と応答し、それ以降は応答しなくなります。これはビヘイビア全体が停止状態になったためです。




### [参考]エラーの表示とログビューア

バーチャルロボットで Speech Reco.ボックスを再生すると、ボックスが赤く表示されます。




これは、ボックスの処理でエラーが発生したことを示していて、具体的な内容は [表示]メニューの[ログビューア]で確認することができます。


ここでエラーが発生しても、機能が一部無効化されるだけで、基本機能はバーチャルロボットでも確認することができます。

### Pepper での動作確認

先のチュートリアルと同様にして Pepper に接続、再生すると、Pepper と音声で対話することができます。

Pepper の場合、バーチャルロボットではエラーで動作しなかった部分、たとえば話を聞いているようなジェスチャーをすることを確認することができます。このように、入出力をおこなうようなボックスには、入出力機能そのもの以外に、その機能と関連づいたジェスチャーなど、細かい振る舞いが設定されているものがあります。

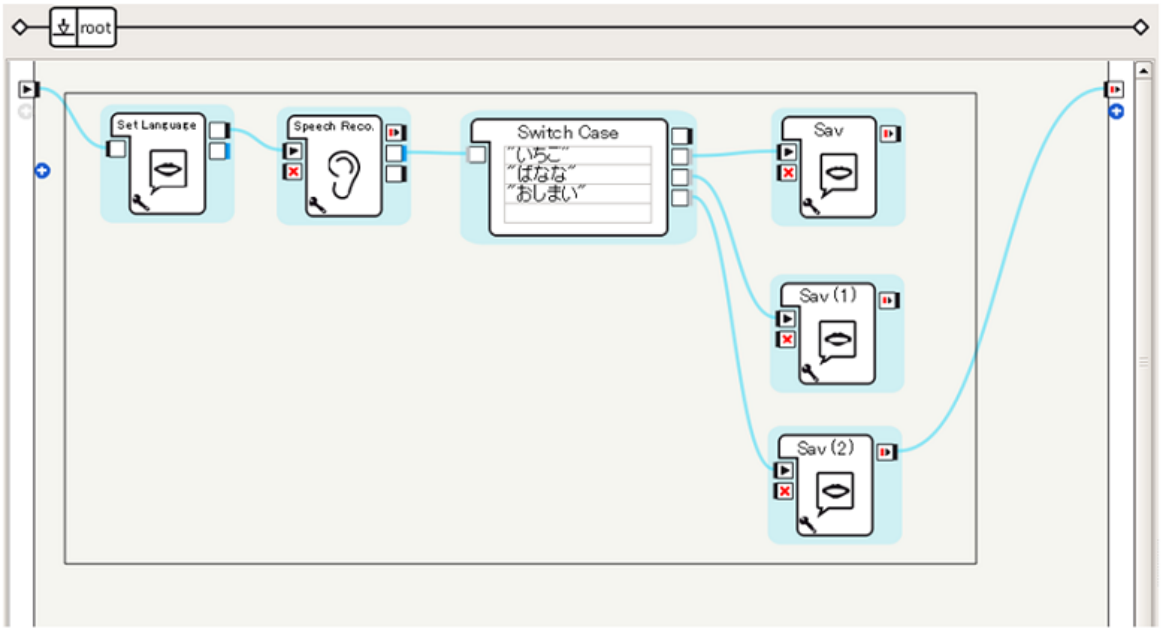
## 独自ボックスの作成

ここまでで、標準で用意されているボックスを使っていくつかの機能を実現してきましたが、
Choregraphe では自分でボックスを作ることもできます。
よく使う処理をボックスにすることで、コピー＆ペーストで簡単に再利用したり、処理の一部をパラメータに置き換えることで、処理の形だけを固定しておいて、中の値だけを自由に変更したりなどができるようになります。

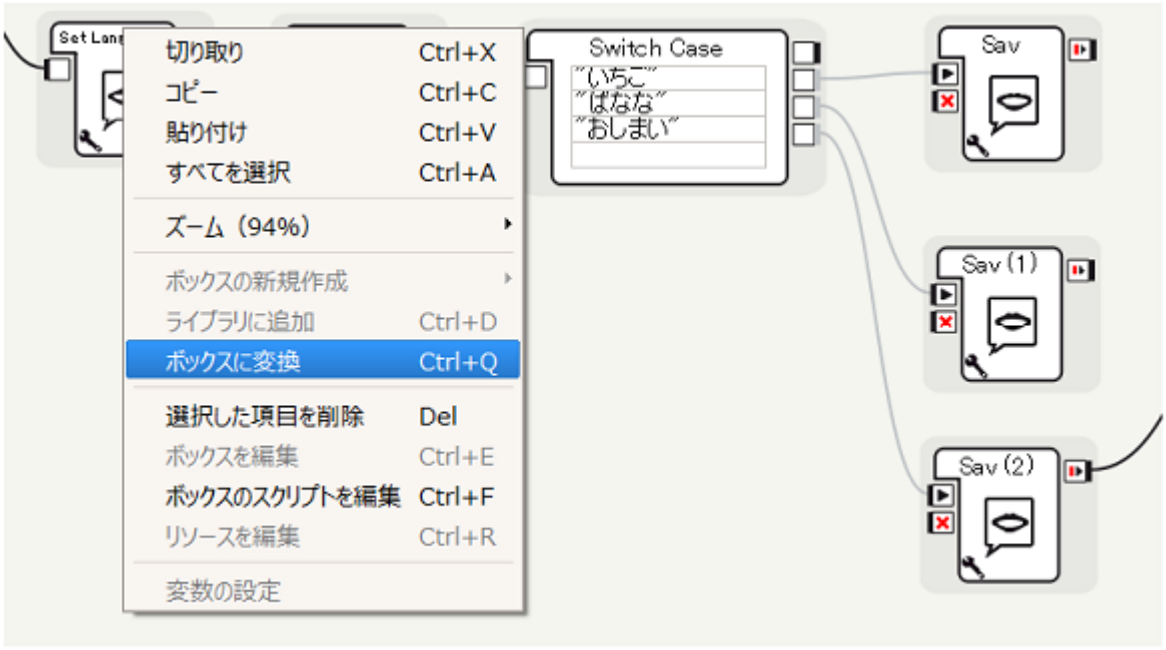
## 複数のボックスを 1 つにまとめる

ここでは例として、先につくった音声認識と応答処理をひとつのボックスにまとめてみます。

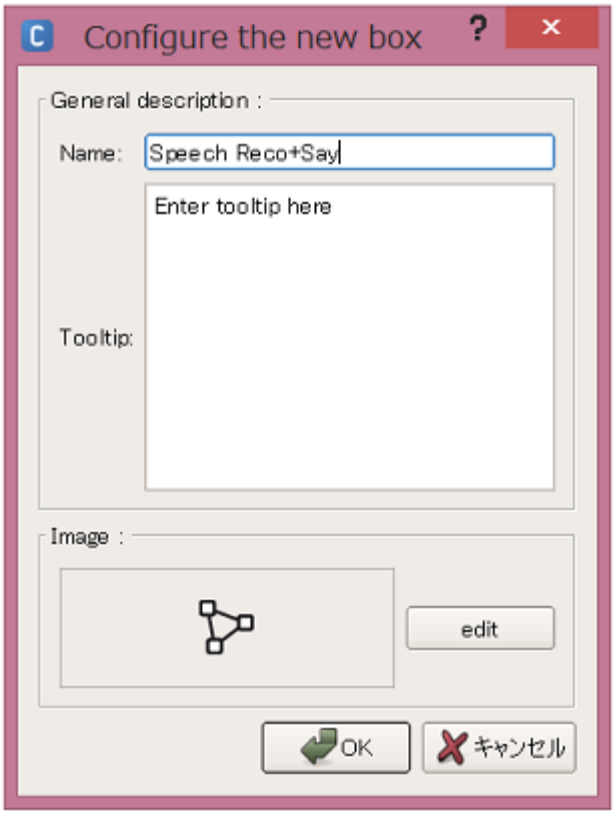
1. マウスドラッグにより、新たにボックスとしてまとめたいボックスをまとめて選択します



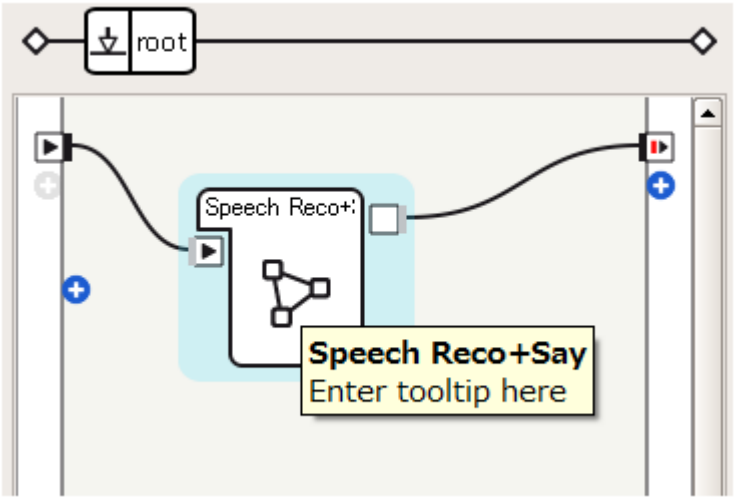

2. 右クリックし、**[ボックスに変換]** をクリックします



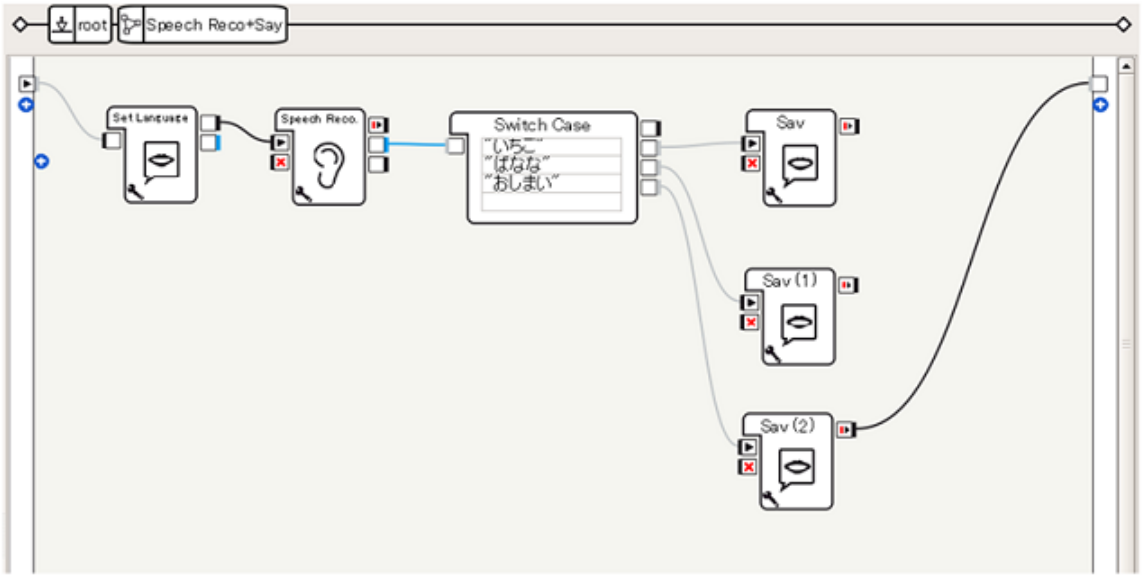

3. 新たに作るボックスの情報を問い合わせるダイアログが表示されるので、好きな名前や説明を設定します




4. 新たなボックスとして 1 つにまとめられます



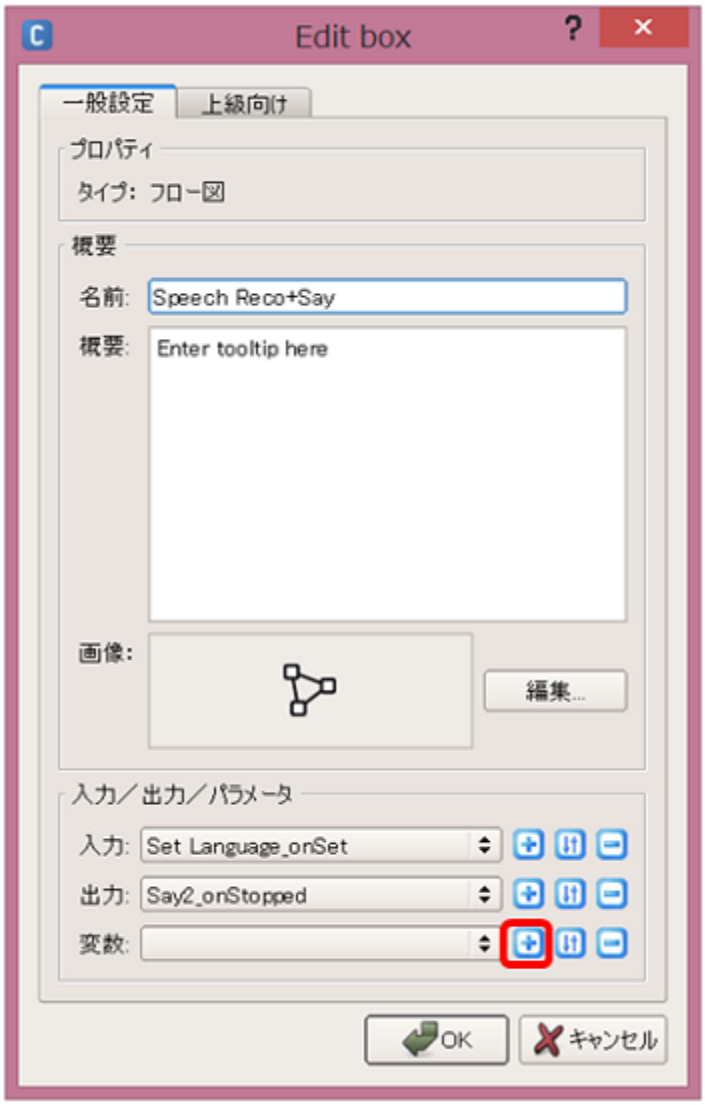

このボックスをダブルクリックすると、選択したボックス群がこのボックスの中に入っていることがわかります。つまり、新たなフローダイアグラムボックスを定義できたわけです。



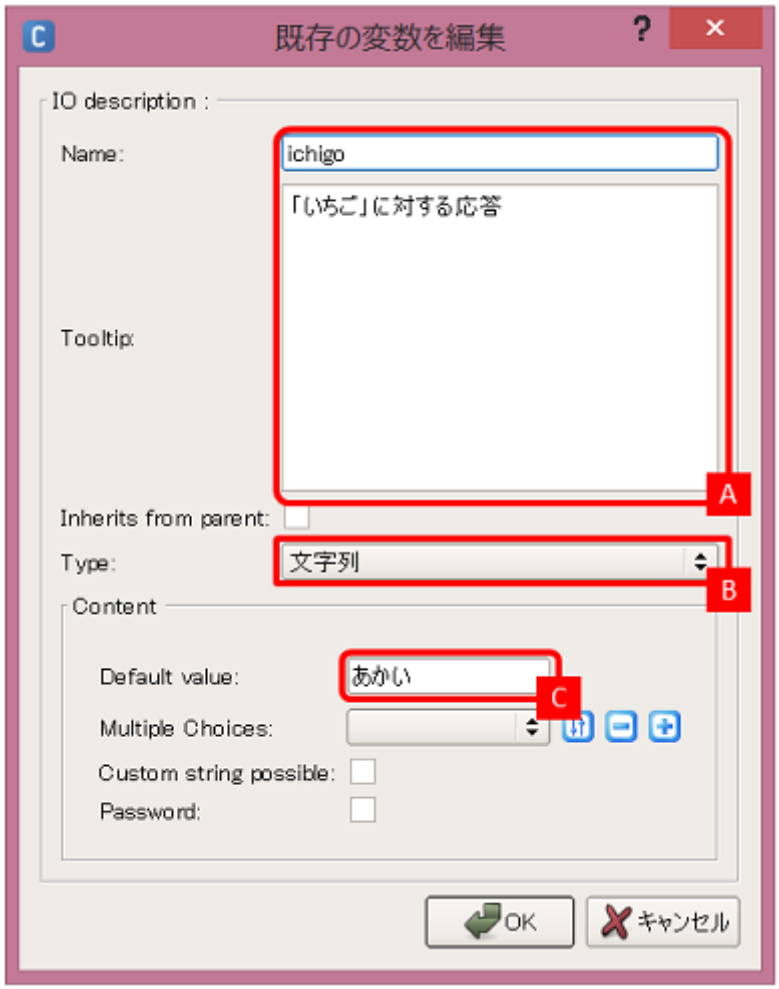

## フローダイアグラムボックスのパラメータ化

新たに作ったボックスには独自にパラメータを追加することもできます。ここでは、簡単な例として、いちご、ばななに対する応答(あかい、きいろい)を、それぞれ変数 ichigo と banana のパラメータとして変更できるようにします。

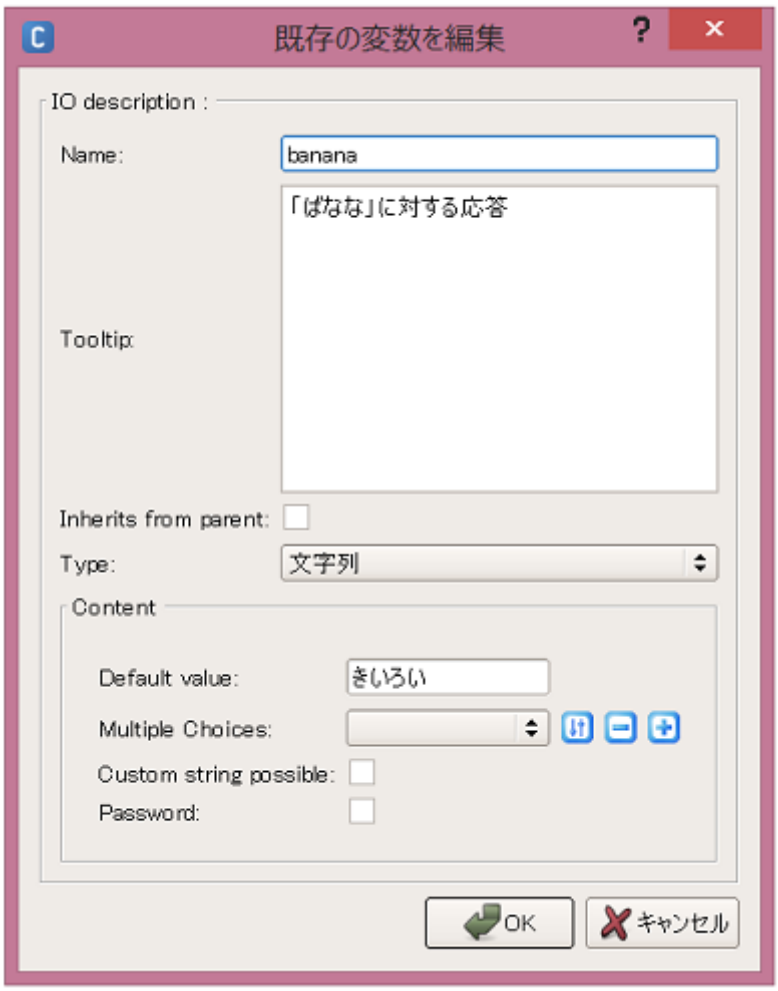
1. 独自ボックスにパラメータ定義を追加する
    1. 先に作った Speech Reco+Say ボックスで右クリックし、**[ボックスを編集]** をクリックします
    2. ボックスの編集ダイアログが表示されるので、変数の **追加(+)** ボタンをクリックします




3. 変数の編集ダイアログがあらわれるので、変数 ichigo に関する説明 [A] 、変数の型 [B]、デフォルト値 [C] を設定し、[OK]ボタンをクリックします

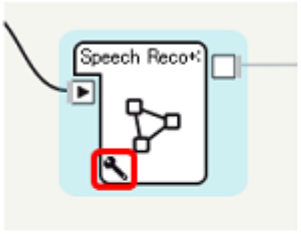



4. 同じ要領で、変数 banana を追加します




5. ボックスの編集ダイアログで [OK]ボタンをクリックしボックスの編集を完了します

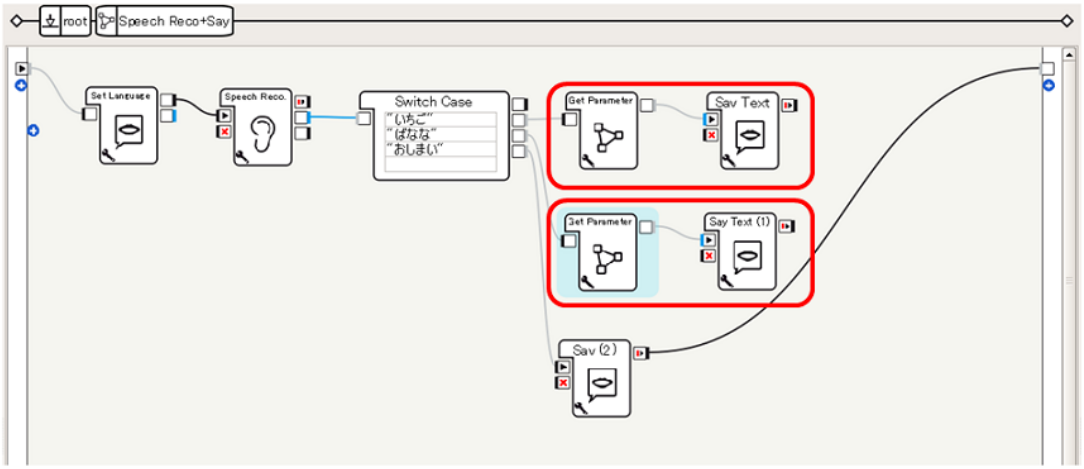
これで、このボックス Speech Reco+Say に変数が 2 つ定義できました。ボックスのパラメータボタンを押すことで、変数が設定されていることが確認できます。





2. ボックス内部を、ボックスに設定された変数を参照するような形に修正します

1. パラメータの取得には、**Flow Control** > **Flow Diagram** > **Get Parameter** ボックスを利用します。Say ボックスで構成されていた箇所を、以下のように Get Parameter ボックスと Say Text ボックスで置き換えます




2. Get Parameter ボックスのパラメータを以下のように設定します。これにより、いちごに対する応答として変数 ichigo の値を、ばななに対する応答として変数 banana の値が Say Text ボックスに入力されるようになります



これで、いちご、ばななに対する応答を独自の変数 ichigo, banana で定義できるようになります。実際にそれぞれの変数の値を変えてみて、挙動に反映されることを確認してみてください。

このようにして、Choregraphe ではボックスの組み合わせで様々な振る舞いを表現することができ、ボックスを独自に定義していくことでより複雑な振る舞いも定義できるようになっています。ぜひ、いろいろなボックスを使って(作って)、Pepper にいろいろなことをやらせてあげてください。

# Pepper チュートリアル (3):ポーズを作る

## このチュートリアルの内容

このチュートリアルでは、Pepper の動き、特に上半身の動きをつくることを通じて、Pepper の関節の仕様や、制御方法について説明します。

1. 各種アクチュエータ仕様
2. ポーズの作成
3. 作成したポーズの実行

## 各種アクチュエータ仕様

Pepper のポーズに関するアクチュエータには以下のようなものがあります。

- 頭: 首(2 軸:Pitch,Yaw)
- 腕(左右共通): 肩(2 軸:Roll,Pitch), 肘(2 軸:Yaw,Roll), 腕(1 軸:Yaw), 手(開く→閉じる)
- 下半身: 腰(2 軸:Pitch,Roll), ひざ(1 軸:Pitch)

Pitch,Roll,Yaw の説明は、ヨーイング – Wikipedia などを参考にしてください。

## ポーズに関するパネル

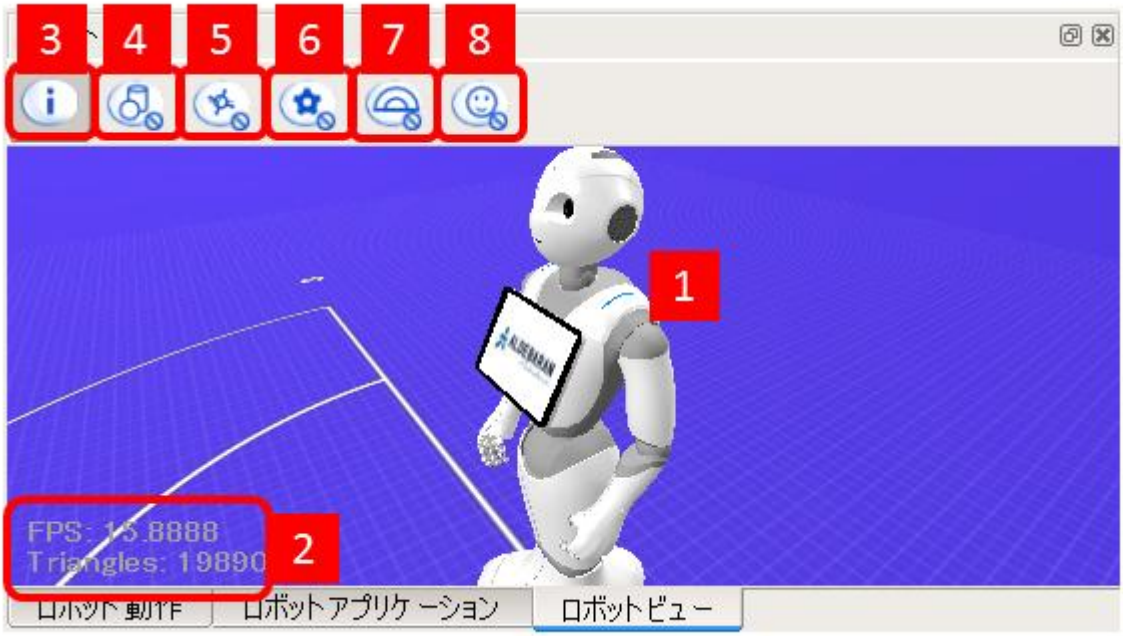
## ポーズ ライブラリパネル

ポーズを管理するためには、ポーズ ライブラリパネルを利用します。
もし現在の Choregraphe のウィンドウにポーズ ライブラリパネルが表示されていない場合は、[表示]メニューの[ポーズ ライブラリ]をクリックします。


1. ポーズ一覧 ... 右クリックでコンテキストメニュー(削除, プロパティ, フォルダ作成)を開けます。ポーズをダブルクリックすると、接続されているロボットがそのポーズをとります
2. ポーズ ライブラリを開く
3. 現在のポーズ ライブラリをエクスポートする
4. ポーズを追加する ... 現在ロボットビューに表示されているポーズ(接続しているバーチャルロボット/Pepper のポーズ)をポーズ ライブラリに追加できます

## ロボットビュー

現在接続しているバーチャルロボットや Pepper の状態を確認したり、関節の操作を行うことができます。
また、Pepper が認識している顔の位置や、人を認識する領域などの補助情報も表示されます。
もし現在の Choregraphe のウィンドウにロボットビューが表示されていない場合は、[表示]メニューの[ロボットビュー]をクリックします。




1. ロボット
2. 3D 情報表示 ... 描画に関する情報です。
3. 3D 情報表示のオン/オフ
4. ALWorldRepresentation に記憶されているオブジェクト情報表示のオン/オフ
5. ロボット位置を更新する/しない
6. モーターの位置を更新する/しない
7. ゾーン表示のオン/オフ
8. 認知した人の表示のオン/オフ

また、ロボットの表示領域では、以下の操作が可能です。

- 表示領域移動 ... マウス左ボタンを押しながらドラッグ
- ズームイン・ズームアウト ... マウスのスクロールホイール操作
- 視点移動 ... マウス右ボタンを押しながらドラッグ

# ポーズの作成

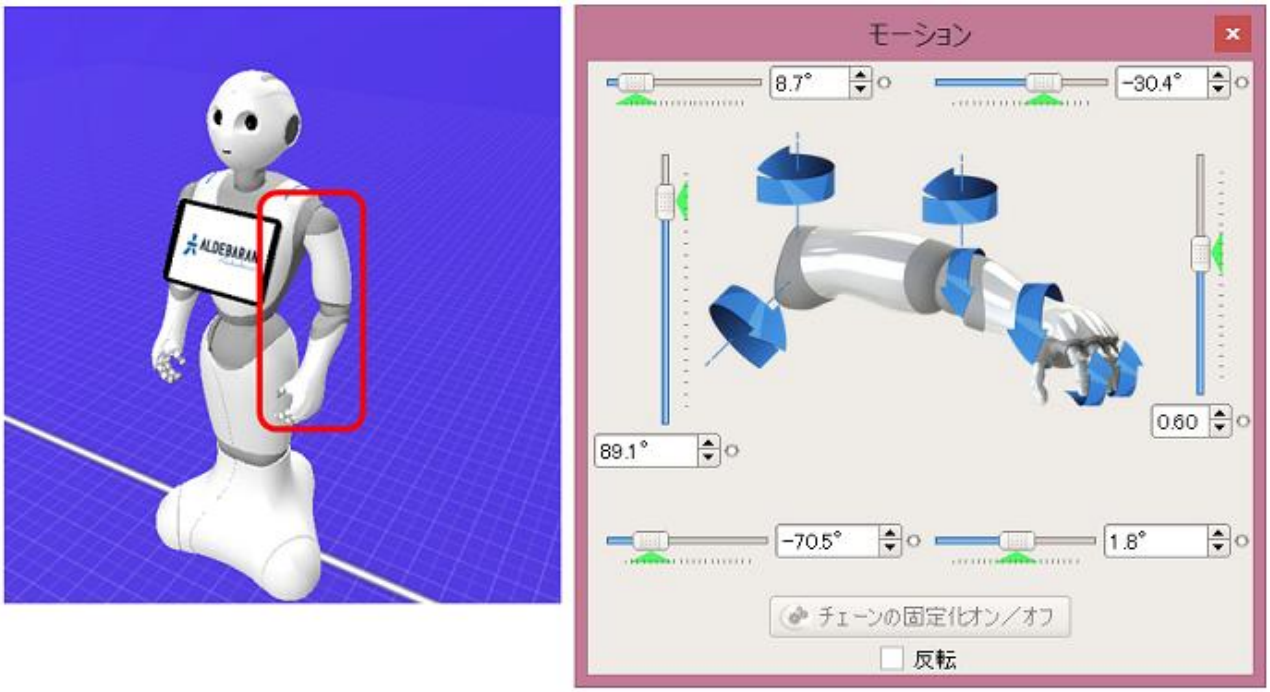
まずは止めのポーズを作ってみます。ポーズを作るには、Choregraphe 上で行う方法と、Pepper を手で動かしながらポーズをつけていく方法の 2 種類があります。

## Choregraphe によるポーズ作成

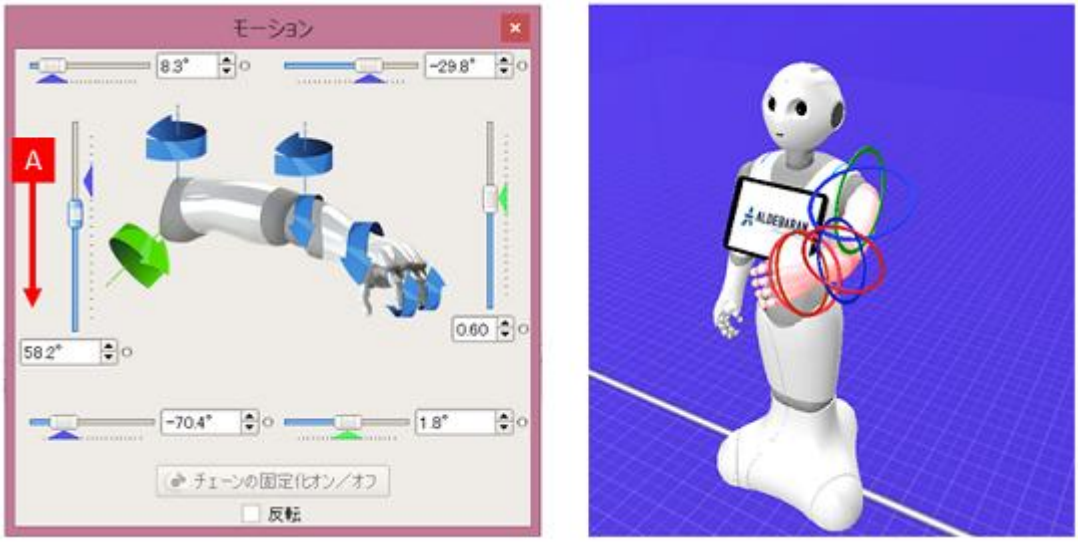
Choregraphe を使うと、接続しているバーチャルロボット、もしくは Pepper に対して関節の制御をおこなうことができます。

## モーション ダイアログボックスによる編集

ロボットビュー上のロボットに対して、変更をおこないたい部位をクリックするとモーション ダイアログが開きます。



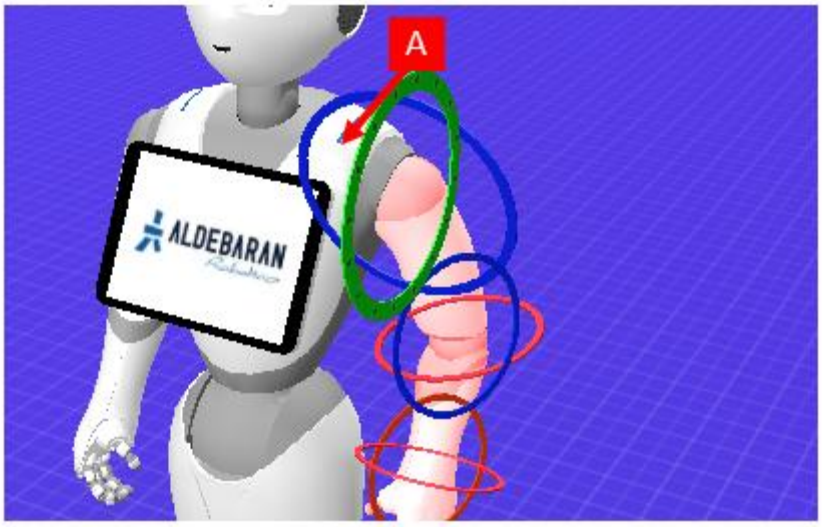

モーションダイアログ上のスライダーを移動させる [A] ことで、関節を回転させることができます。




ロボットは、自分の体に自身の部品をぶつけないように、Anti-self collision といった機構を持っています。そのため、関節を動かした際に意図しない動きをする場合があります。

### ローテーションハンドルによる編集

ロボットをクリックしたときにあらわれるリング(ローテーションハンドル)によっても関節を制御することができます。
操作したいローテーションハンドル上にマウスオーバーするとハンドル幅が広がって表示されます。この状態でマウスドラッグ [A] すると、対象の関節を制御することができます。

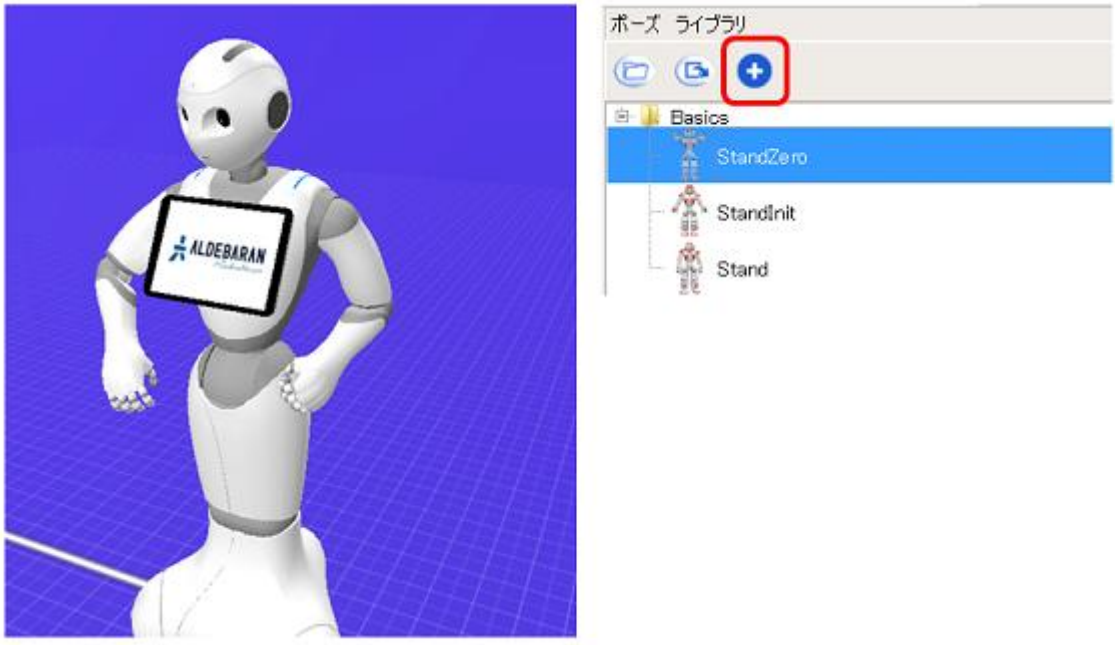



場面に応じて、モーションダイアログとローテーションハンドルの使いやすいほうを利用することができます。

## ポーズの記憶

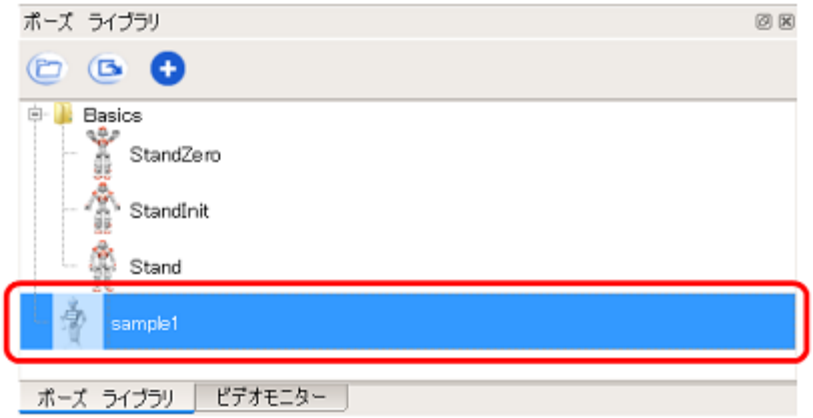
バーチャルロボットのポーズを、ポーズライブラリに保存しておいて、好きな時に呼び出すことができます。

1. ロボットが保存したいポーズをしている状態で、**[ポジションを追加]ボタン** をクリックします




2. [ポジションを新規作成]ダイアログで、ポーズの情報を記入します
    - ポーズの名前を入力 [A] したり、ポーズのイメージ画像を登録することができます。[スクリーンショット]ボタン [B]をクリックすることで、現在のロボットビューの内容をイメージ画像 [C] にすることができます


3. ポーズが登録されます




ポーズライブラリ中のポーズは、ダブルクリックすることで現在接続しているロボットに反映することができます。

また、ポーズ ライブラリの **[ポジションライブラリをエクスポート]ボタン**と**[ポジションライブラリを開く]ボタン** を使って編集したポジションライブラリを保存することも可能です。

[Tips]動かないときは

Pepper に接続している場合、ポーズを選択しても、うなだれたまま何もしない場合があります。Pepper が以下のような状態になっているときは、おやすみ中です。

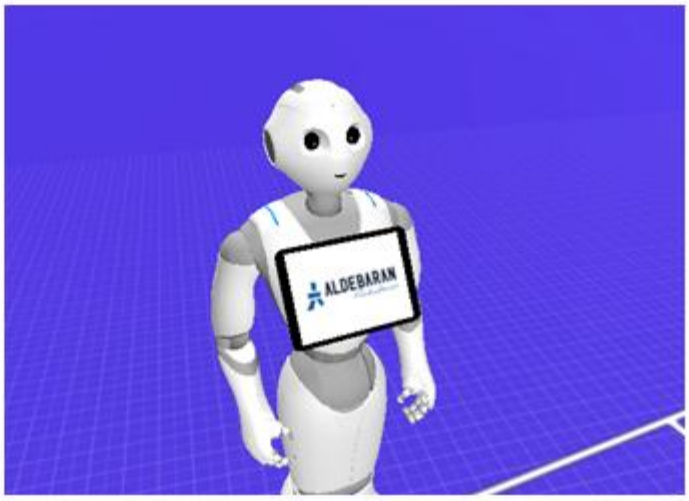
このような場合は、[Wake Up]をクリックして起こしてあげてください。

## Pepper によるポーズ作成

アニメーションモードを使うと、Pepper をさわりながらポーズをつくることもできます。
アニメーションモードに入るためには、Pepper に接続した状態で、ツールバーの **[アニメーションモード]ボタン** をクリックします。

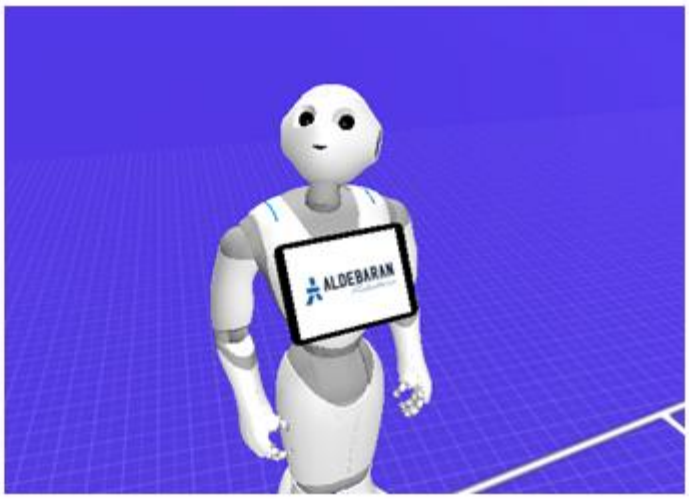
## 頭(首)のポーズづけ

1. アニメーションモード設定後は、目がオレンジ色に点灯します




2. Pepper の頭をさわると、目の一部分が緑色になり、首の関節の固定が解除されます
    - 首の関節を手で自由に動かして、ポーズをつけることができます

3. 再度 Pepper の頭をさわることで、再度固定することができます。目の色はオレンジ色に戻ります
    - 関節の動きはロボットビューにも同期されます


## 腕のポーズづけ

1. 首と同様、アニメーションモード設定後は目はオレンジ色に点灯している状態です
2. Pepper の手の甲をさわると、腕全体の関節の固定が解除されます
    - 手の甲にふれている状態で、腕の関節を動かすことができます


3. 手の甲から手を放すと、腕の関節が固定されます


## モーションダイアログによる関節固定解除

関節の固定の設定・解除は、モーションダイアログからも指定できます。
Pepper が接続された状態でモーションダイアログを開くと、**[チェーンの固定化オン／オフ]ボタン**が有効になります。
このアイコンが緑色の状態が固定オフ、赤色の状態は固定となり、ボタンで切り替えることができます。

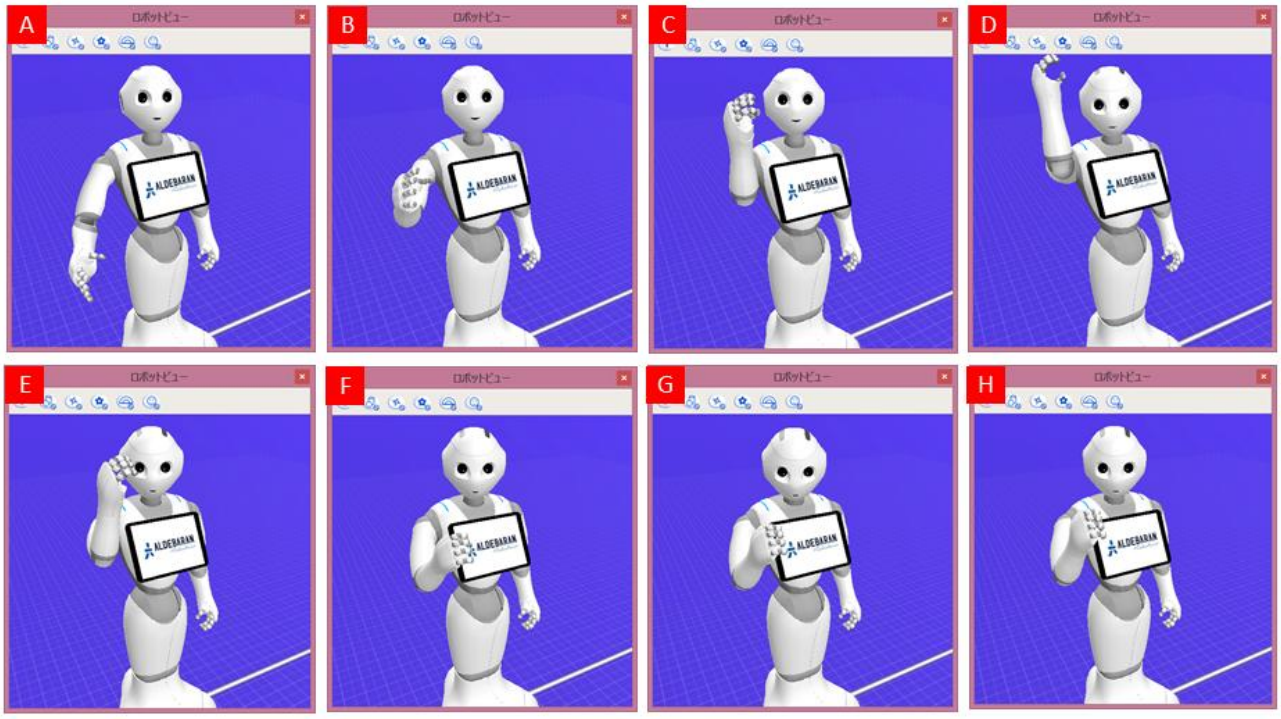
### アニメーションモードの解除

アニメーションモードは、再度[アニメーションモード]ボタンをクリックするか、Pepper の頭を 3 秒以上さわると解除できます。

アニメーションモードでつけたポーズは、Choregraphe によって設定したポーズ同様、ポーズライブラリに登録して好きな時に再生することができます。

## 作成したポーズの実行

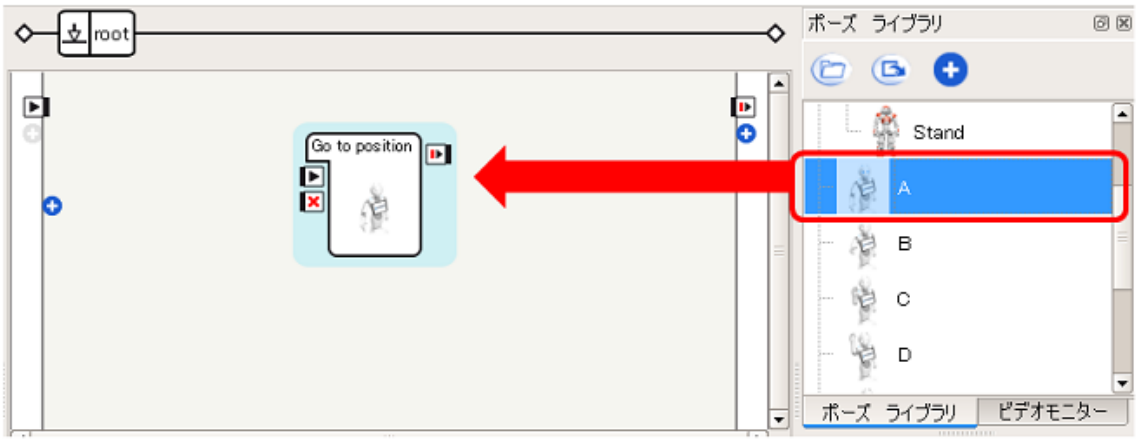
作成したポーズをボックスとして実行する方法について、ガッツポーズを例に説明します。
ここでは、例として、以下のような 8 つのポーズを作ってポーズ ライブラリに登録しているものとします。



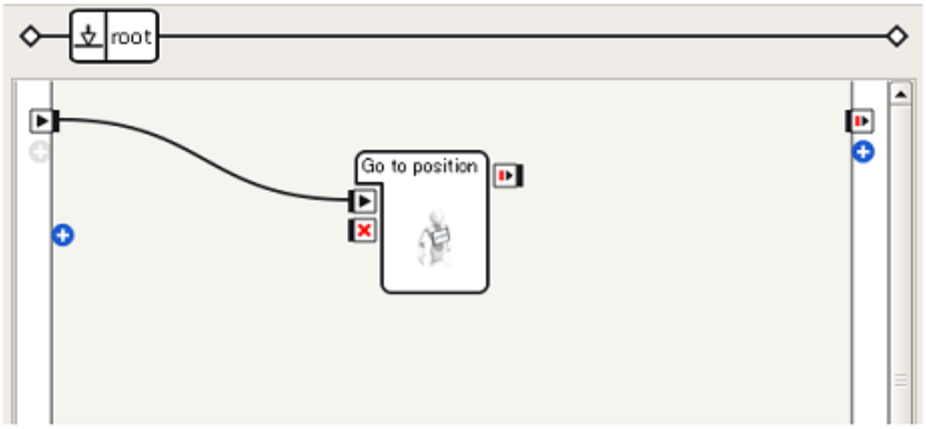

## ポーズの切り替えをおこなうボックス

ポーズをフローダイアグラム中にドラッグ &ドロップすることで、簡単に「ポーズに移行する」ボックスを配置することができます。

1. ポーズ ライブラリからポーズを選択し、フローダイアグラムにドラッグ &ドロップします

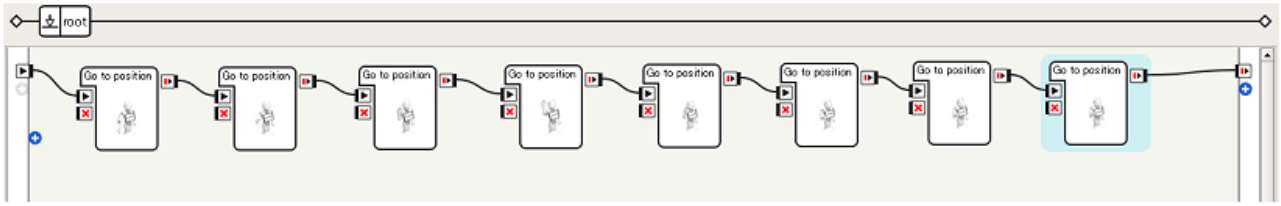



2.
3. 作成されたボックスの onStart 入力にビヘイビアの onStart 入力を接続します




このフローを再生すると、ロボットは現在のポーズから、ポーズ A へとポーズを変更します。ポーズからポーズへの変更にともなう動作は自動的に補間されます。
なお、このポーズとポーズを補間する際に、自分自身に衝突してしまうような(たとえばタブレットの近くを腕が通過するような)動きになってしまう場合に、Anti-self collision 機構によって意図しない動きになってしまう場合があるので注意が必要です。

こんな風に、ポーズを並べることで、順番にポーズを変えていくような動きを簡易的に実現することができます。

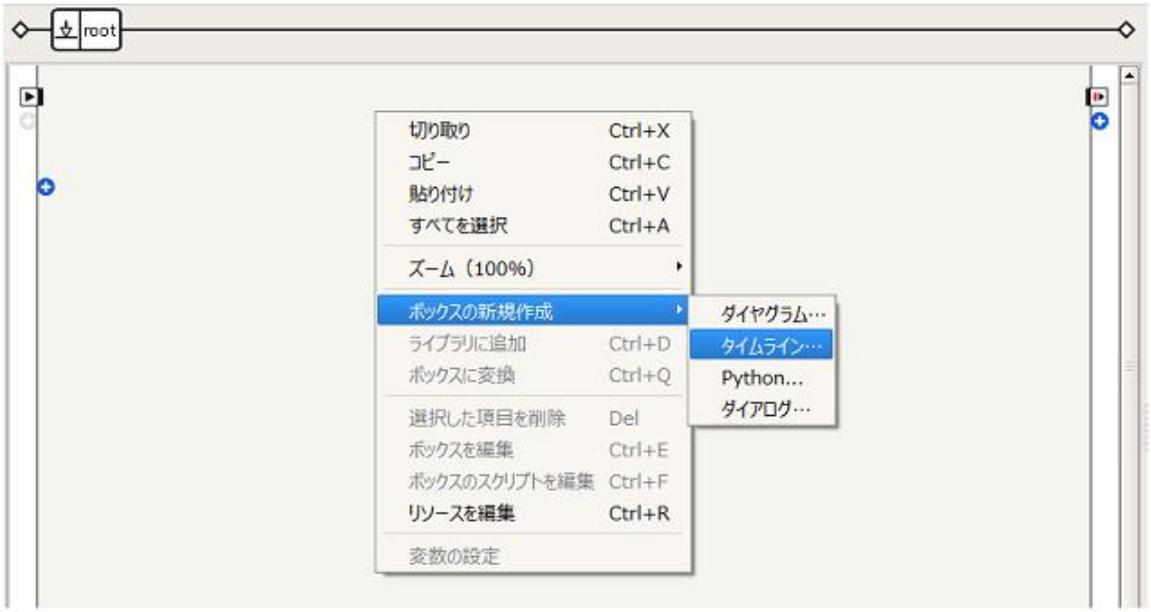



実際に連続した動きのように見せるためには、あるポーズとあるポーズの間はすばやく動かしたり、ゆっくり動かしたりと時系列の制御が必要になります。これには、タイムラインボックスを利用します。

# タイムラインボックスの作成

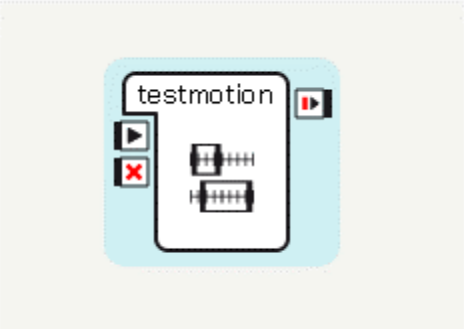
作成したポーズを使って、あらたにモーションをボックスとして定義するには、以下のようにおこないます。

1. フローダイアグラム上で右クリックし、**[ボックスの新規作成]メニュー** の **[タイムライン...]** を選択します



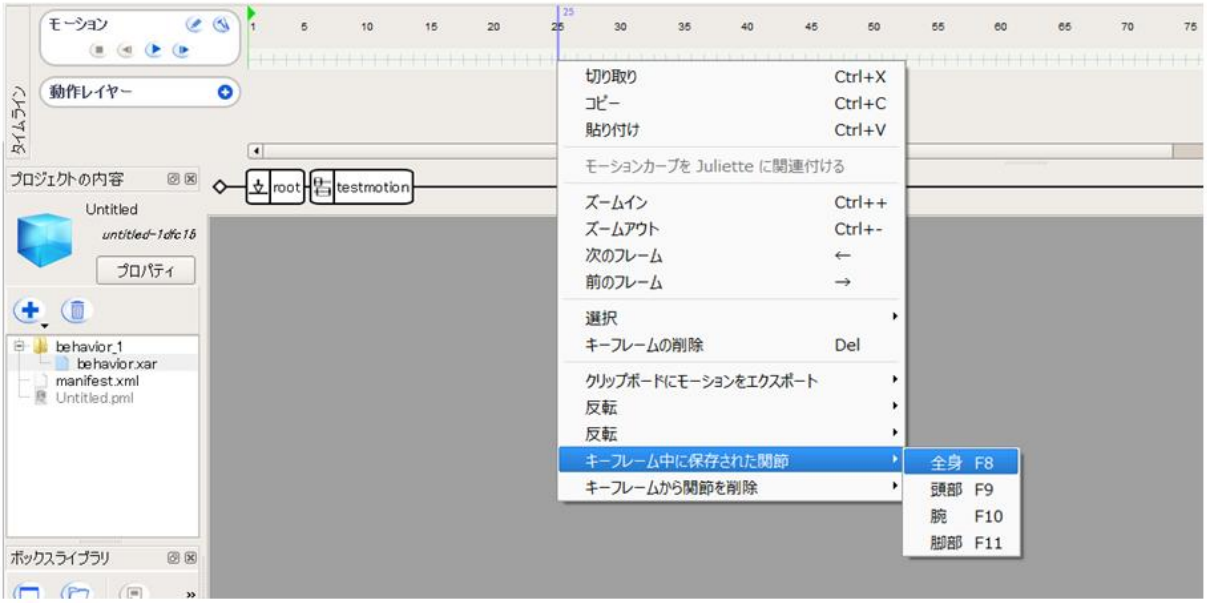

2. ボックスの情報を問い合わせるダイアログが開くので、適当な名前を入力し、[OK]ボタンを押します


3. 新しい、空のタイムラインボックスが作成されます。このボックスをダブルクリックし、タイムラインを開きます



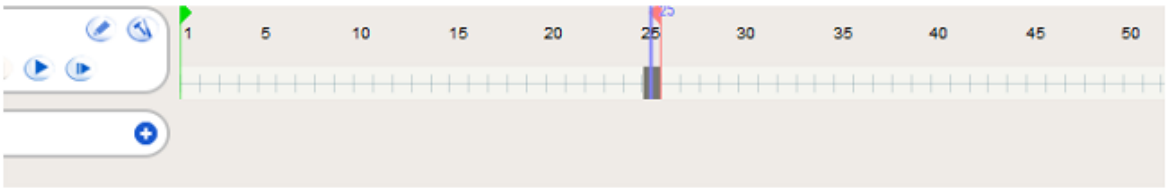

4. 最初のポーズ A を、モーション開始後の 25 フレーム目(デフォルトのフレームレートは 25 フレーム/秒なので、1 秒経過時点)に設定してみます。まず、**ポーズ A をダブルクリック** し、ロボットのポーズをポーズ A に変更します


5. タイムラインの 25 フレーム目で右クリックし、**[キーフレーム中に保存された関節]** の **[全身]** を選択します

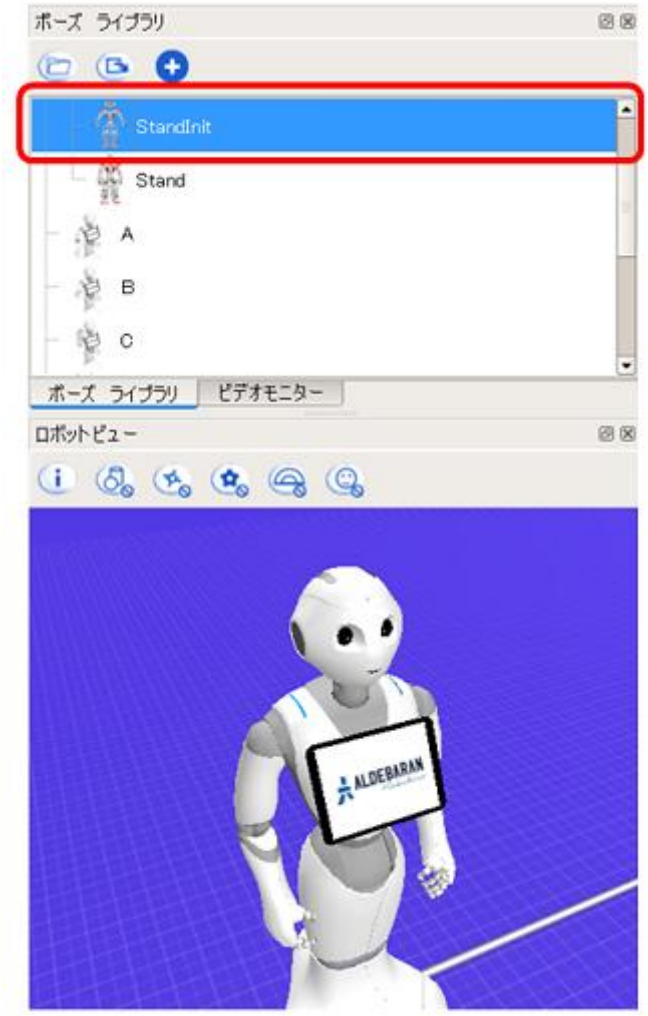



すると、25 フレーム目がモーションキーフレームとなり、グレーで表示されます。これにより、25

フレーム目ではポーズ A になるように関節の各部が調整されることになります。

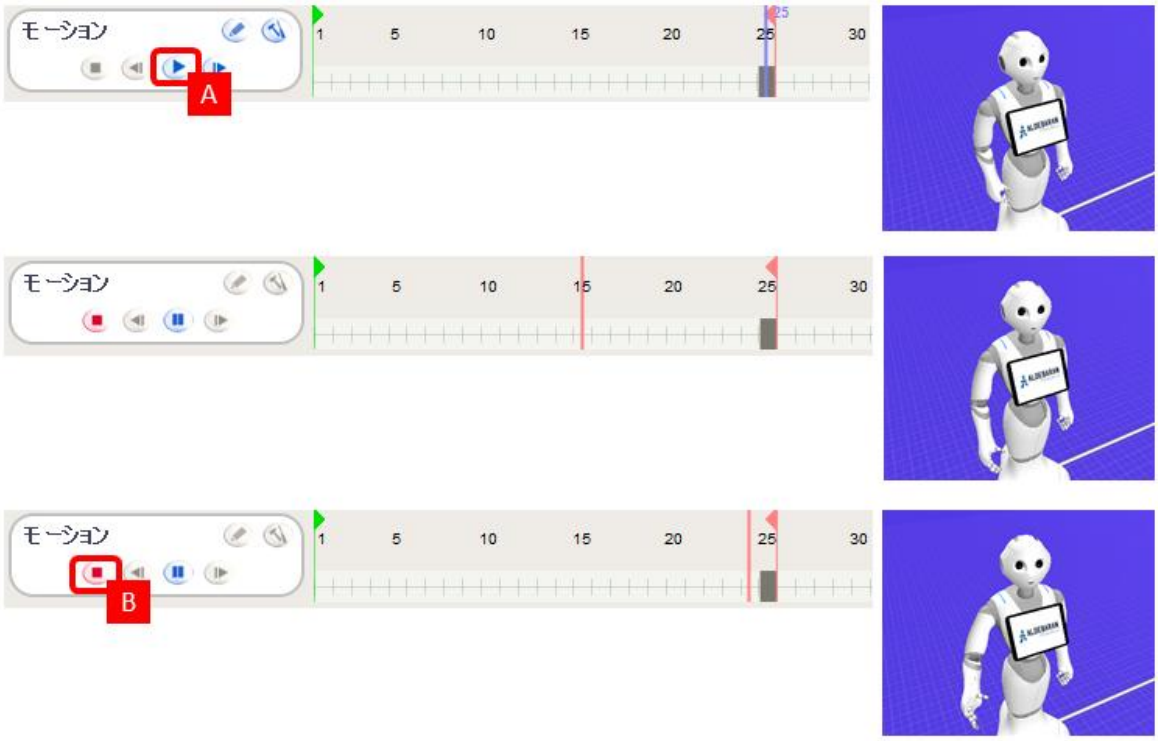



6. ここまで作成した動きを確認してみます。変化がわかりやすいように、ポーズライブラリを使って、ロボットのポーズを標準で用意されている **ポーズ StandInit** にダブルクリックで変更します




7. **再生ボタン[A]** をクリックします
1 秒程度で、ロボットのポーズがポーズ StandInit からポーズ A に移行する様子が確認できます。

停止したい場合は **停止ボタン[B]** をクリックします。

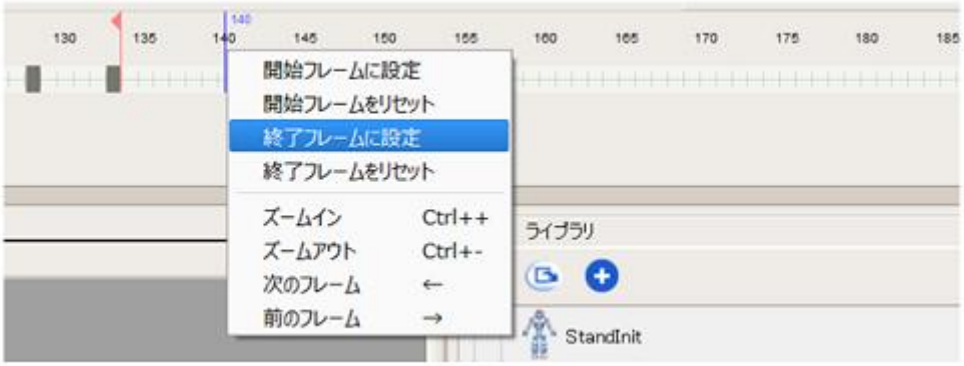



[参考]最初のモーションキーフレームの配置

ここでは最初のモーションキーフレームを 25 フレーム目にしましたが、基本的にタイムラインにおいて、1 フレーム目にモーションキーフレームを配置すべきではありません。
ここまで見てきたとおり、ロボットでは各関節を駆動させてポーズを変化させていきます。そのため、急激な変化を起こさせるようなタイムラインを記述してしまうと、予期しない動作をしてしまうおそれがあります。

8. ここまでの要領で、ポーズ B 以降のモーションキーフレームを作成していきます
   たとえば、以下のようなタイミングで各ポーズを設定してみるとよいかもしれません。
9. 


10. 終了フレームの設定
    このモーションの終了としたいフレーム上で右クリックし、**[終了フレームに設定]** を選択します。



このような手順で、新たなタイムラインボックスを作成し、フレームごとにモーションを作成していくことが可能です。作成したボックスは標準ボックスライブラリのモーションと同様に、他のボックスと並列に実行したり、さまざまな機能と組み合わせることができます。

このようにして、Pepper のポーズを定義し、それらをボックスから必要なタイミングで呼び出すことで、さまざまなジェスチャーをともなう振る舞いを作成することができます。

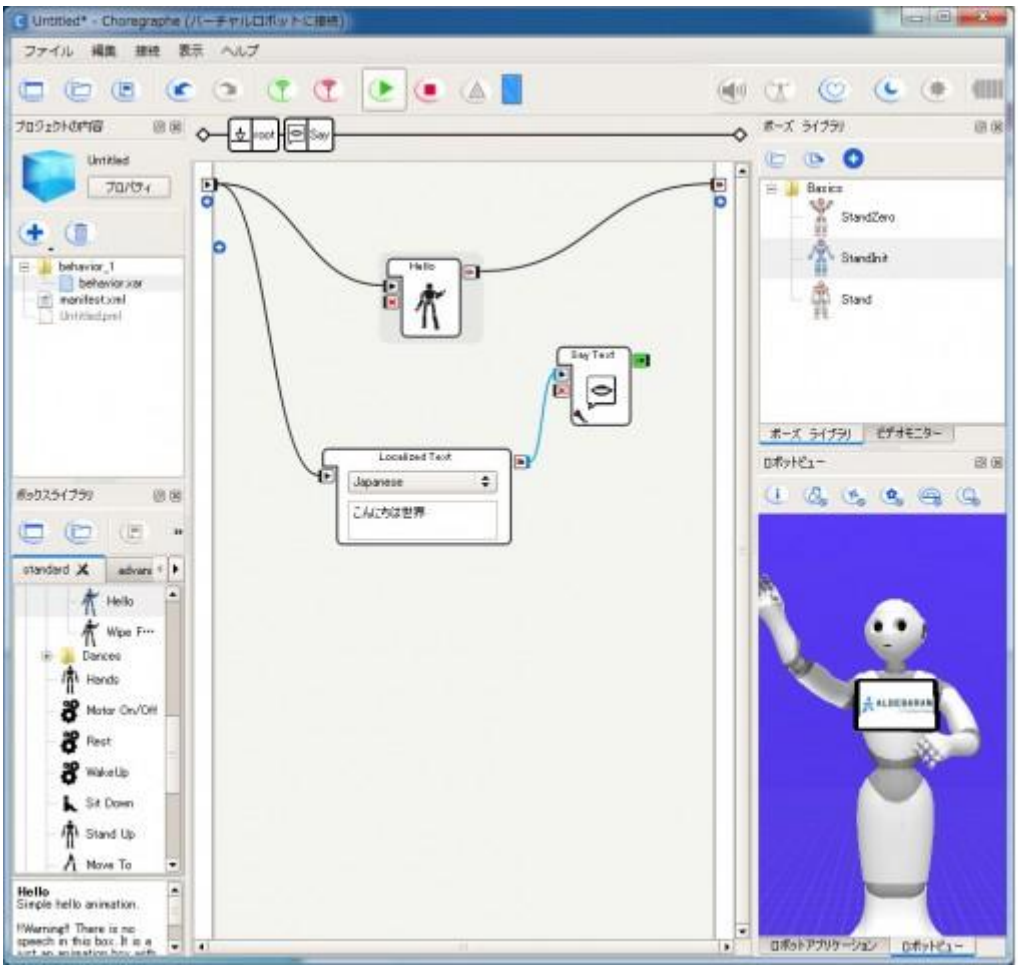
なお、ここまででも見てきたとおり、Pepper は Anti-self collision 機構や、各種センサーなどの、自身や他のものを保護するためのさまざまな機構を備えており、モーションを作る際には、これらの機構と衝突しないように配慮していくことが必要です。これは実体のあるロボットならではの難しさであり、面白さだと思います。

# pepper に手を振らせる(Choregraphe)

公開日: 2014/09/16 : 最終更新日:2014/09/16 pepper アプリ開発

Choregraphe を使って、pepper にあらかじめ用意された動作を簡単に実行させる方法を試してみました。手順は前回の状態からの続きです。

1. 左下のボックスライブラリの「Motions」→「Animations」→「Hello」を選択して中央エリアにドラッグすると、Hello ボックスが生成される
2. Hello ボックスの左側の三角ボタンに中央エリア左端の三角ボタンからドラッグして線をつなげる
3. 同様に Hello ボックスの右側の三角ボタンから中央エリア右端の三角ボタンへドラッグして線をつなげる
4. 画面上部の緑色の再生ボタンを押す




上画像のように線をつなげると、pepper が手を振りはじめると同時に Hello World の吹き出しが出ます。(途中で吹き出しが先に消えてしまうので、上画像では吹き出しが消えています)

手を振るボックスの後に Say text のボックスをつなげると、手を振り終わってから吹き出しが出ます。もっと細かく動作タイミングを制御するにはプログラムソースを書かないとだめなのかな？ボックスの配置位置で制御できるのかな？