

ようこそアトリエ秋葉原へ

ここスペースは写真OKです。

ぜひハッシュタグをつけて呟いてください #pepper_3331

事前確認2点

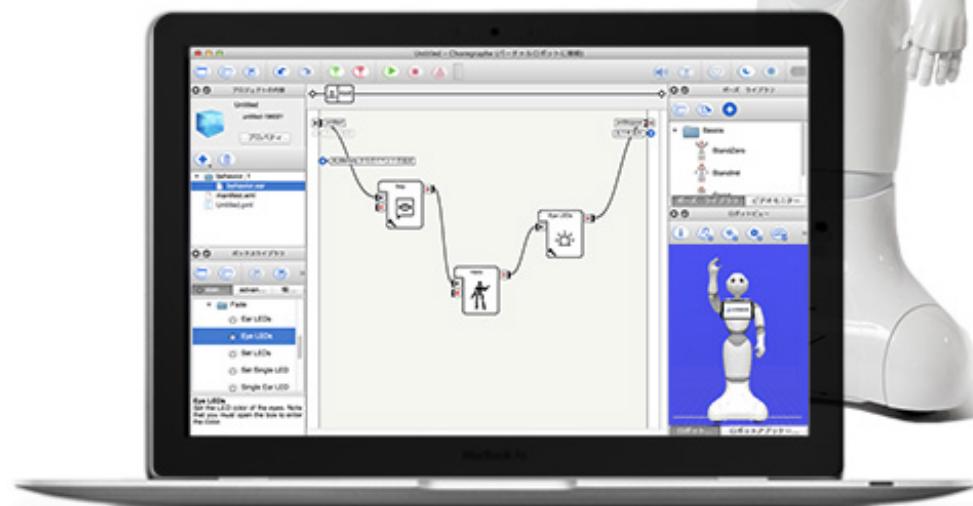
- wifiに接続
- Choregrapheのバージョン(2.5.5)確認

をお願いします。



Atelier Akihabara

ワークショップ Naoqi 2.4.3と2.5.5の違い



免責事項

このワークショップは
アトリエのスタッフが作成したものであり
ソフトバンク公式のものではないことを
ご承知ください。

実体験とコミュニティーで開発を促進する

アトリエ



相互
促進

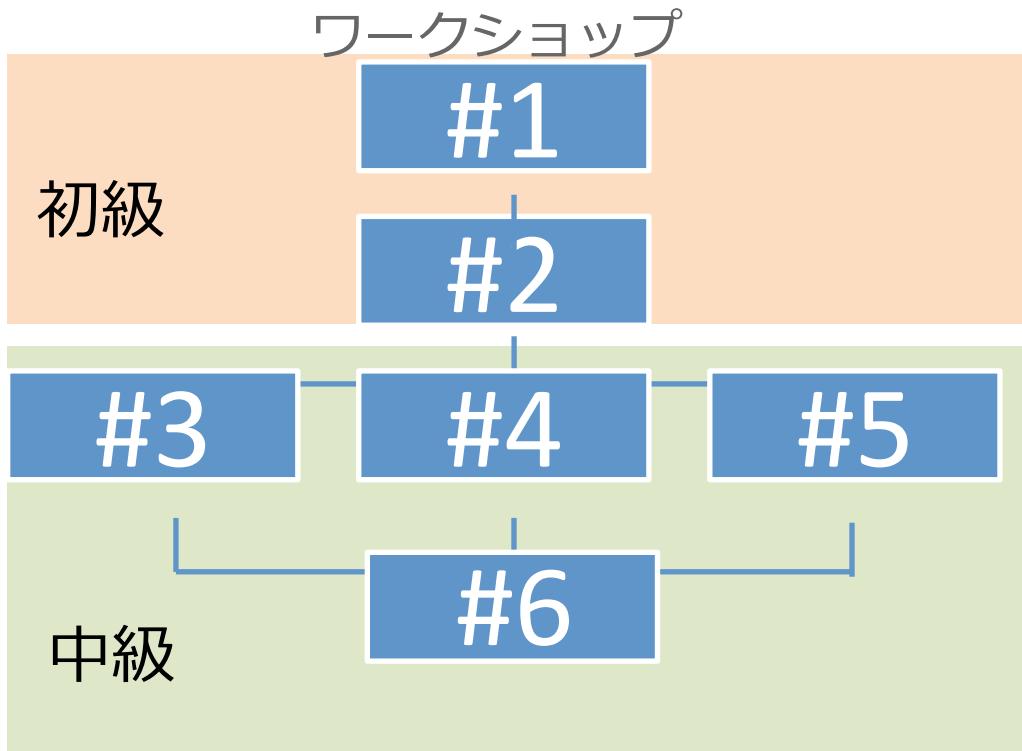
コミュニティー



- ✓ Pepperのアプリ開発を実体験

- ✓ 経験や知見を共有

実体験とコミュニティーで開発を促進する



アトリエについて

実体験とコミュニティーで開発を促進する



アトリエサテライト

有志でPepperと開発スペースを
提供している
企業、大学、コミュニティースペース

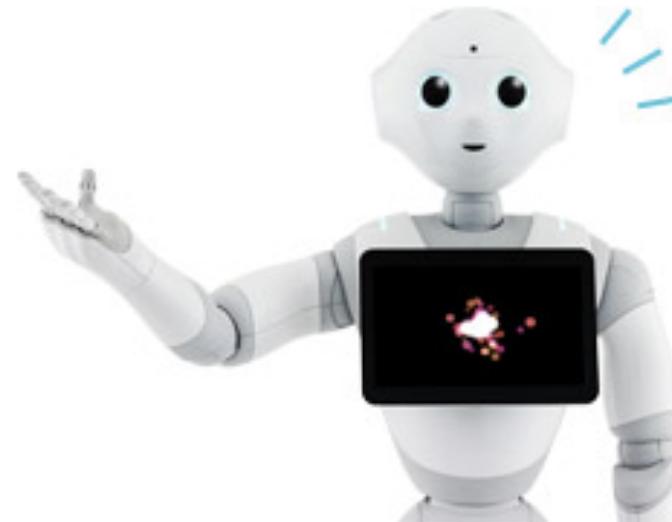
秋葉原で回答できない質問は
各サテライトへ

- お名前
- 所属
- 本日の意気込み

例：

本日の案内を勤めさせていただきます、
飯島 采永 (Iijima Sae)と申します。

Naoqiがバージョンアップしました！



アップデート対象のPepper

モデル	対象
Pepper デベロッパー先行モデル	対象外
Pepper 一般販売モデル	対象外
Pepper for biz	対象

※2017年4月時点

アップデート対象のPepper

モデル	対象
Pepper デベロッパー先行モデル	対象
Pepper 一般販売モデル	対象
Pepper for biz	対象

※2017年7月11日対応開始しました！

1.Naoqiとは

2.Naoqi 2.5.5

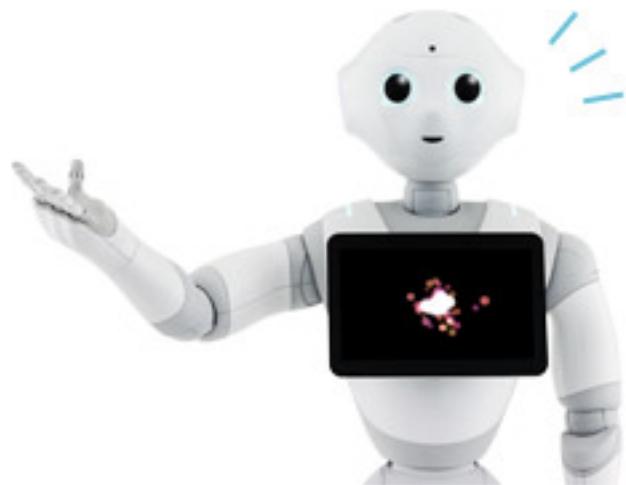
3.タブレットの変更

4.基本BOXの変更点

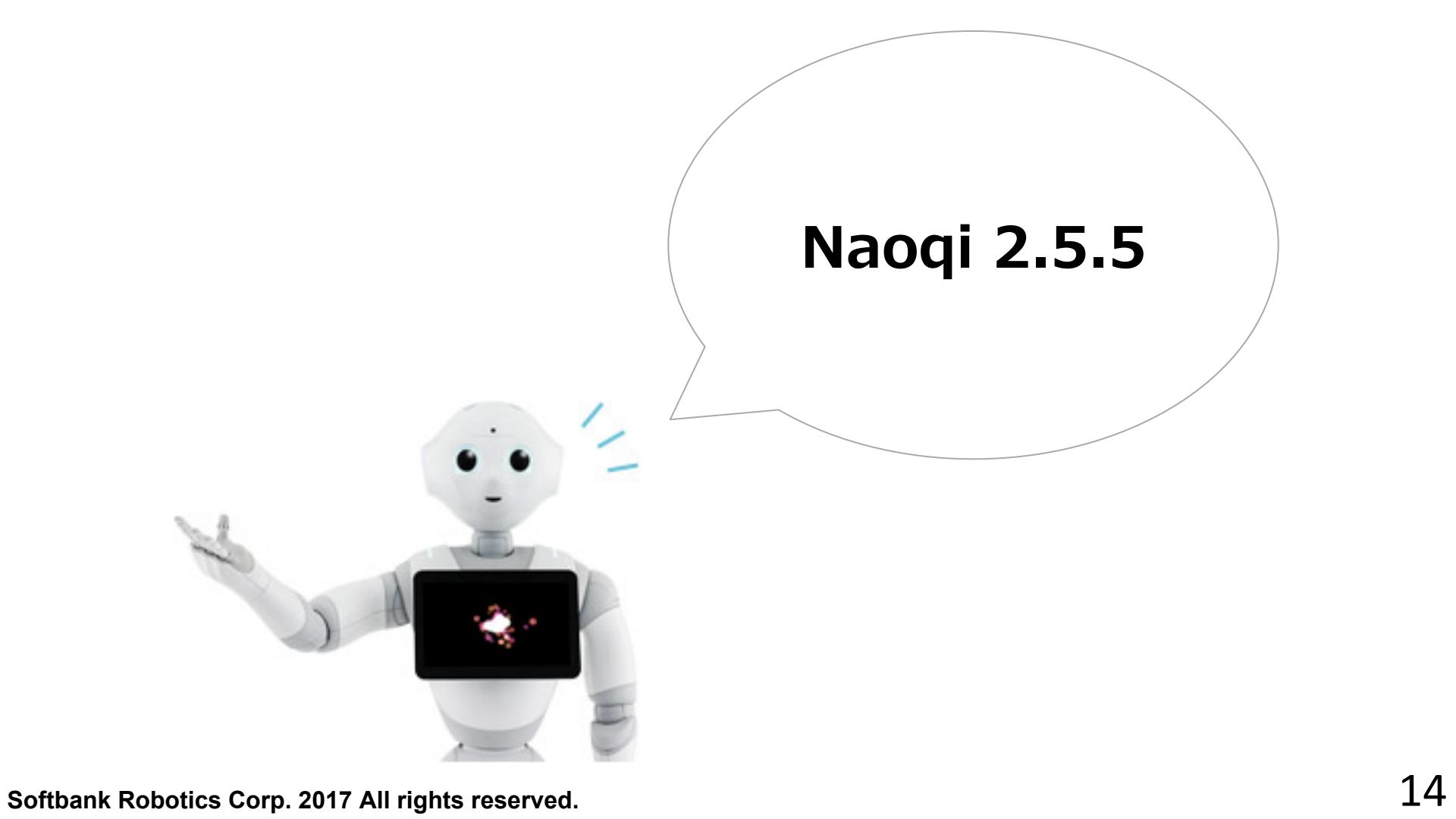
5.翻訳機能

6.SLAM機能(β版)

PepperのOS “Naoqi”



- Naoqiとは?
 - 人型ロボットNao・Pepper・Romeoに搭載されている
 - GentooベースのGNU/Linuxディストリビューション
- 対応するプラットホーム
 - Windows・Mac・Linux
- アプリケーション開発
 - C++・Python・Choregraphe



Naoqi 2.5.5

主な追加点

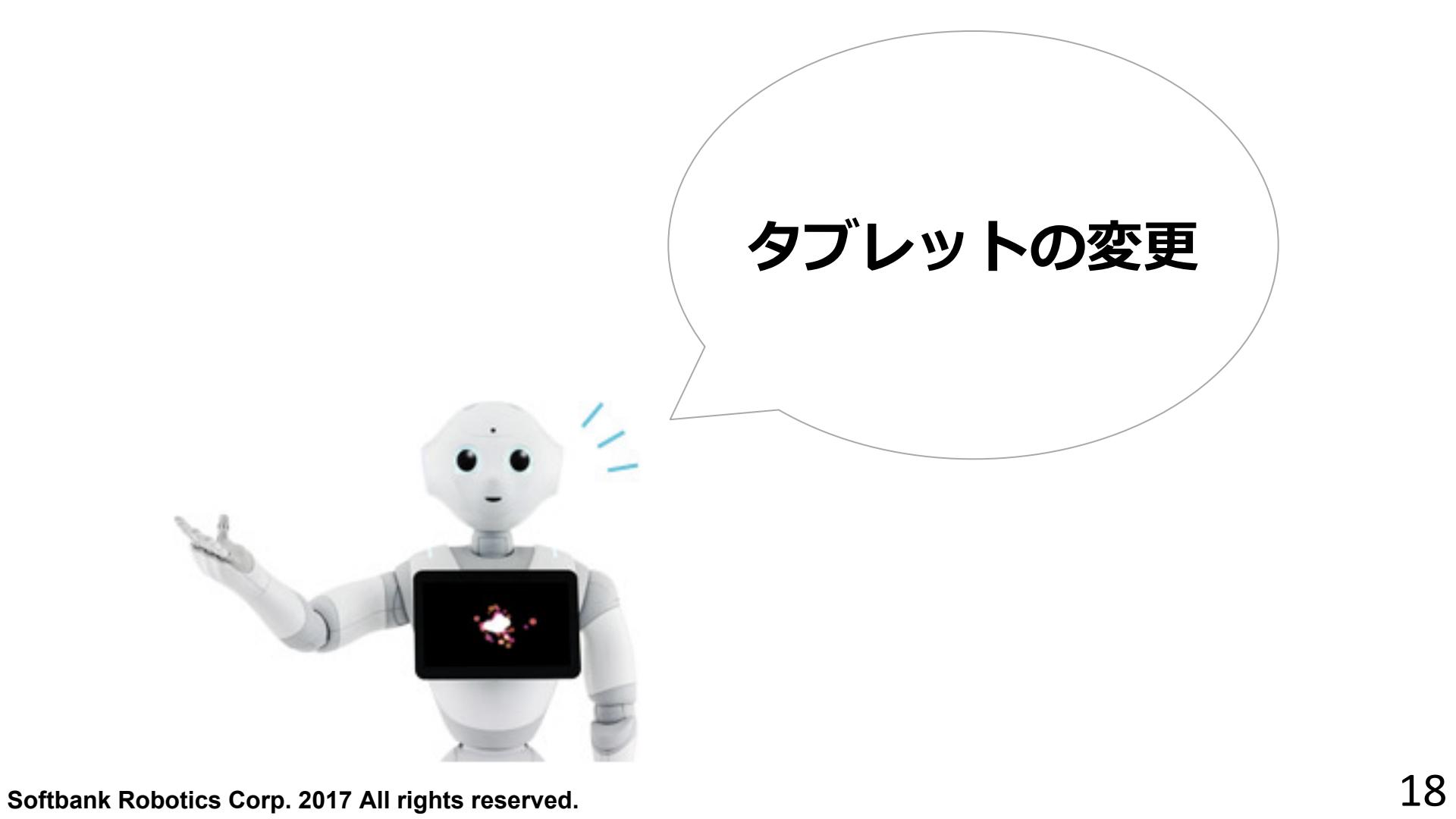
- SLAM機能(自己位置推定と環境地図作成)β版を追加
- 中国語(繁体)に対応 (Pepper for biz)
- クラッシュした際、より詳細なレポートを取得
- ディスプレイアップデートが独立
- 今後発売される新型ディスプレイを搭載したPepperに対応

主な改善点

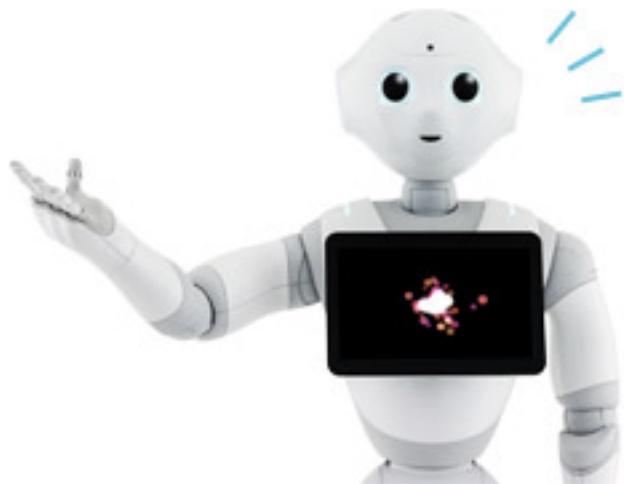
- レーザーセンサーの制御が改善されました。
- 移動距離の自己測定がより正確になりました。
- 特定の状況下でマイクの精度が安定しました。
- Pepperの起動にかかる時間が短縮されました。
- LinuxやPythonなどのライブラリを更新しました。

変更点と不具合

変更点 & 不具合		対応方法
1	今後発売される新型ディスプレイを搭載した Pepper に対応	新型ディスプレイは現行ディスプレイと解像度が異なりますので、対応が必要になります ※詳細は後述します
2	旧バージョンの ChoregrapheのBOXを利用しているアプリが正常に動作しない場合がある	新バージョンの ChoregrapheのBOXと置換してください ※詳細は後述します
3	タイムラインのレイヤーが正常に同期して動かず、タイミングがずれる場合がある	動作を確認し、それが発生している場合は、waitとsay、timeline(アニメーションのみ)を組み合わせる等、timeline外で同期をとる構造に変えてください
4	ASCII以外の文字コードがプロジェクトのファイル名やパスに含まれている場合、ロボットへ転送できない	プロジェクトやプロジェクトに含まれるファイルに、ASCII以外の文字コードを使用しない
5	発話させる文章に"<>"が含まれると正常に発話しない	発話させる文章に"<>"を使用しない



タブレットの変更



新型ディスプレイへの対応

現行のディスプレイ解像度

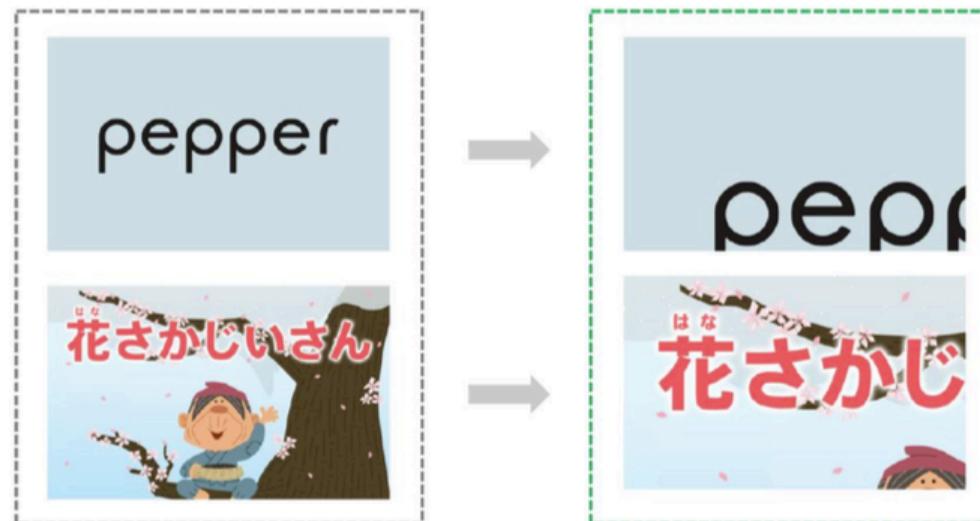
新規格のディスプレイ解像度

横 1707 ピクセル × 縦 1019 ピクセル

横 962 ピクセル × 縦 601 ピクセル

1707 × 1019

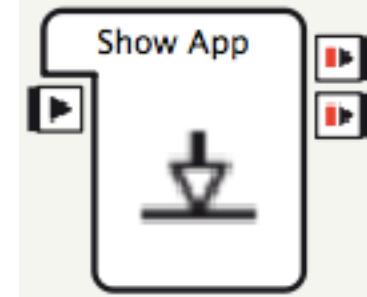
962 × 601



対応時の注意

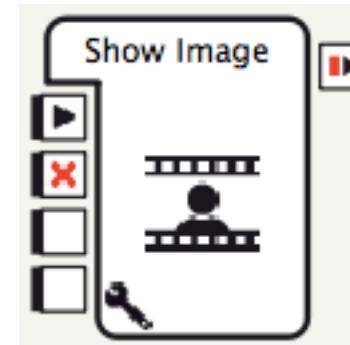
- 対応させる必要があるアプリ

- HTMLを使用してタブレットに表示をさせているアプリケーション



- 対応の必要のないアプリ

- ShowImageを使用して画像を表示させているアプリケーション



新型ディスプレイへの対応方法

- 提供しているスクリプト(adjust.js もしくは adjust23.js)をアプリで表示させるHTML内で読み込み、パッケージ化
1. アプリを確認し、使用するスクリプトを定める
 2. スクリプトを挿入
 3. manifestを修正
 4. パッケージ化、確認

1. アプリの確認

- アプリで読み込まれるHTML内で設定されているパラメータによって使用するスクリプトが異なる
- <head/>エレメント内の<meta/>エレメント表記を確認

例 : <meta name="viewport" content="initial-scale = 1.335,
minimum-scale = 1.335, maximum-scale = 1.335" />

viewportが1.335に設定されている

adjust.js

viewportが1に設定されている
もしくは宣言自体がない

adjust23.js

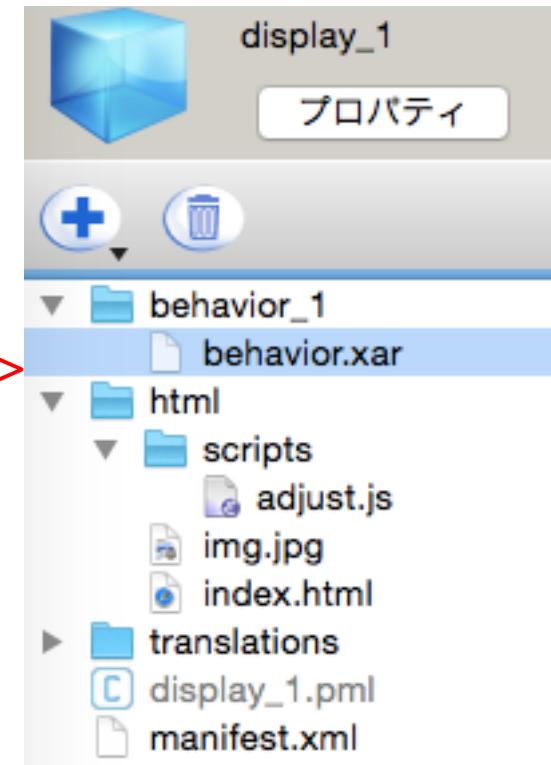
2. スクリプトの挿入

- 対応するスクリプトをhtml/scripts/ディレクトリに追加
- 追加したスクリプトを読み込むようにHTMLを編集
- <head/>エレメント内に以下の記述を追加

```
<script src="scripts/adjust.js"></script>
```

もしくは

```
<script src="scripts/adjust23.js"></script>
```



例外対応が必要なアプリ

- CSS側での制御があるアプリ
 - CSS側でviewport制御を行っている場合には該当行を削除
- jQueryライブラリを使用していないアプリ
 - 以下の記述を追加

```
<script src="/libs/qimessaging/1.0/jquery.min.js"></script>
```
- 複数のHTMLで構成されているアプリ
 - すべてのHTMLに対して同様の処理を行う

3. manifestを修正

- 修正したアプリが新しいバージョンのNaoqiのみで有効になるよう、アプリ内の「manifest.xml」を以下のように修正

修正前：

```
<requirements>
<naoqiRequirement minVersion="2.4"/>
<robotRequirement model="JULIETTE_Y20"/>
</requirements>
```

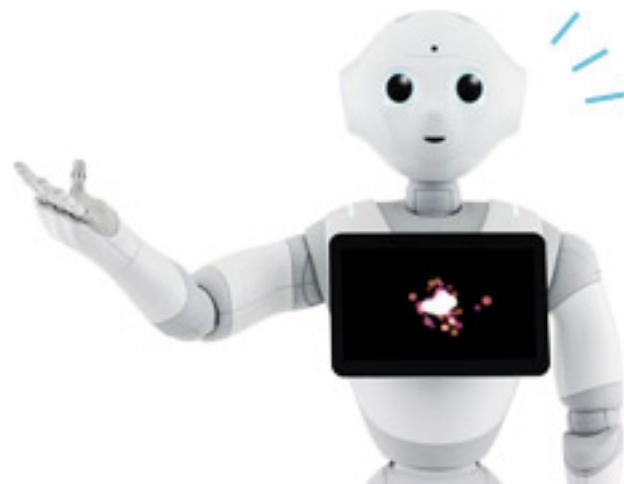
修正後：

```
<requirements>
<naoqiRequirement minVersion="2.5"/>
<robotRequirement model="JULIETTE_Y20"/>
</requirements>
```

4. パッケージ化して確認

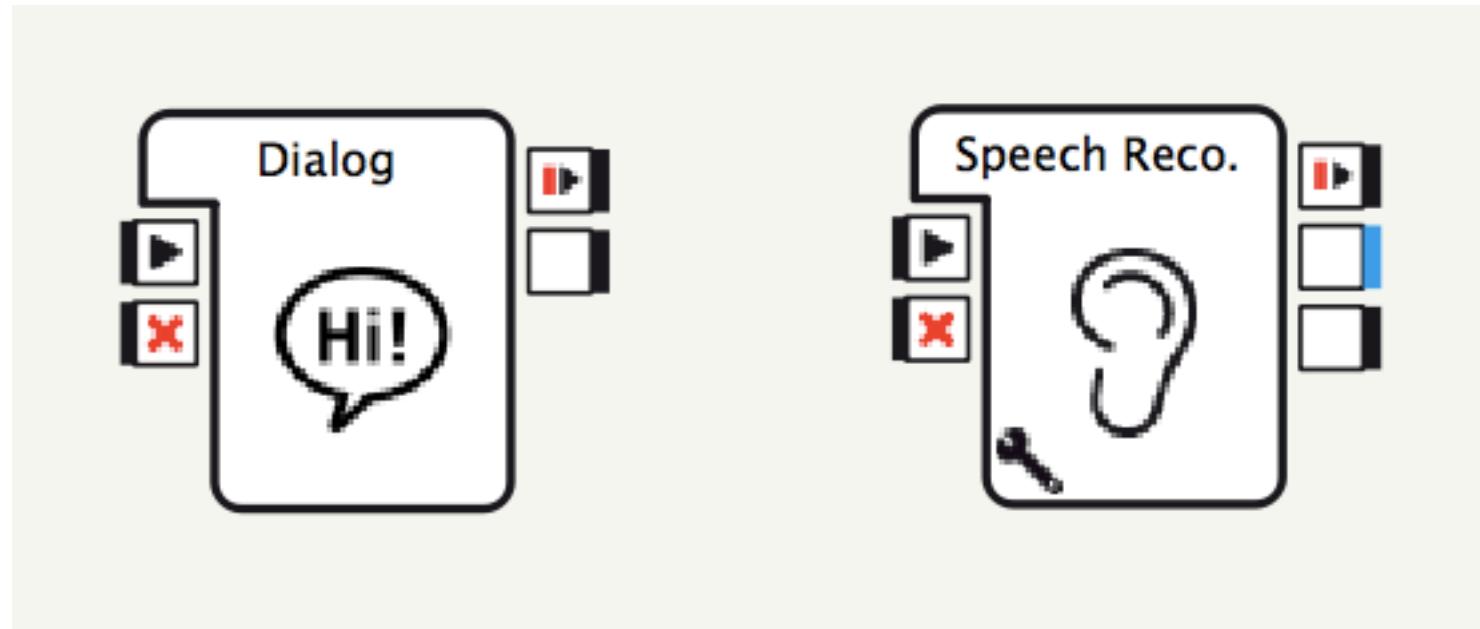
- アプリケーションとしてPepperに送る場合にはパッケージファイルを作成し、送信
- パッケージファイルの作成
 - Choregrapheの[ファイル]>[パッケージをビルド…]
- 実際に実行し、正しく動作するか確認

よく使う BOXの変更点

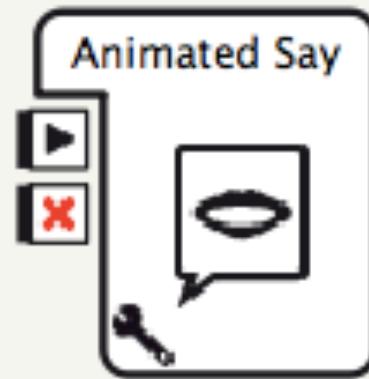
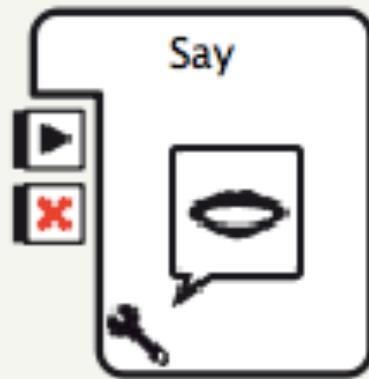


互換性のないBoxの置換

- Choregraphe2.4.3以前のものを使用すると正常に動かない可能性
- Choregraphe2.5.5の同名のBOXと置き換える



変更のある基本BOX



Say Box

Textは設定ボタン➡から変更



今まで

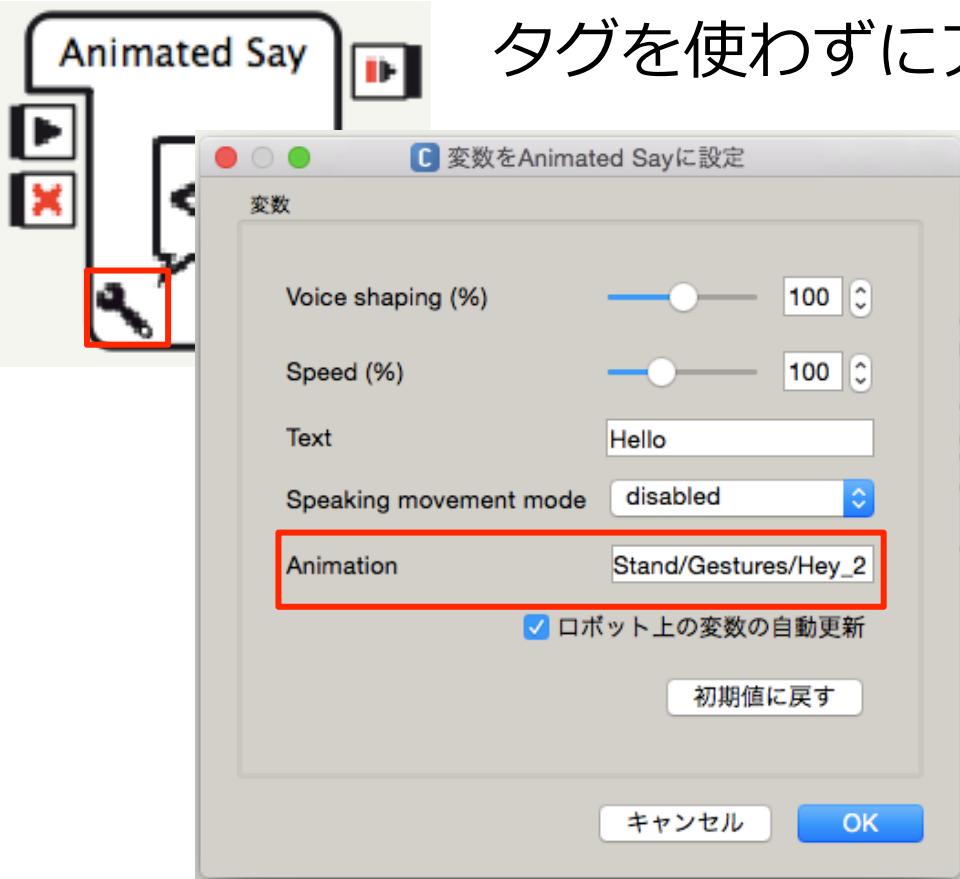
デフォルトのままonStartに繋げると
「Hello」もしくは「こんにちは」と発話

Naoqi2.5.5

デフォルトのままonstartに繋げると
言語設定が日本語の場合
「へろ(Helloのローマ字読み)」と発話
英語の場合「Hello」と発話

※今までのようにダブルクリックすると、スクリプトエディタが開きます

Animated Say Box



タグを使わずにアニメーションが付けられる

変数のAnimationは
Textの読み上げと同時に動く
^start(~/Hey_2)Hello と一緒に

Textにタグを入れても動きます

Dialog

プロジェクトファイル > 新規ダイアログトピック

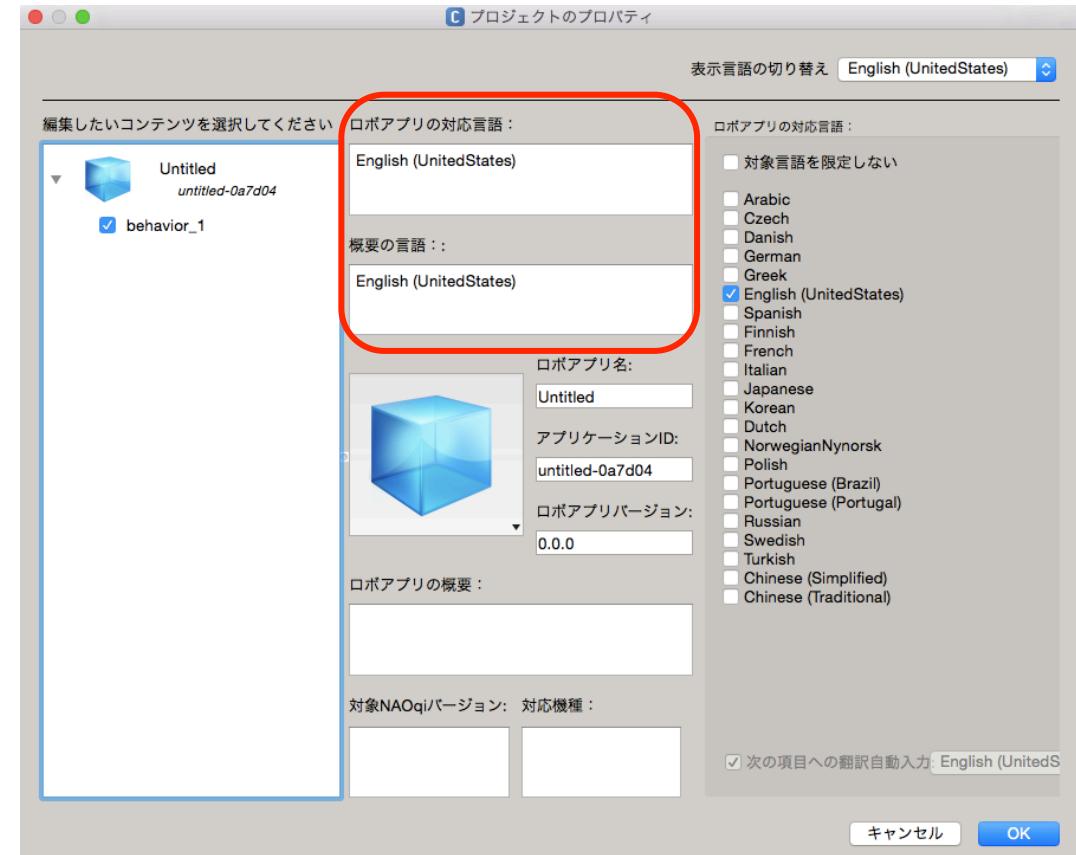
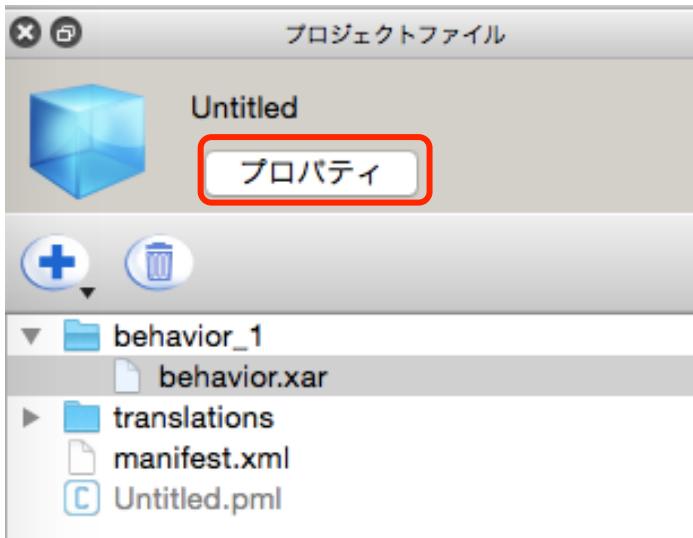


デフォルトでは
言語に表示されるのはEnglishのみ

日本語で話をしたい場合には
ロボアプリの言語設定を
日本語(Japanese)に変える

プロジェクトの言語設定

①プロジェクトのプロパティを開く



プロジェクトの言語設定

②ロボアプリの対応言語
右の表でJapaneseにチェック

ロボアプリの対応言語 :

- Japanese
- 対象言語を限定しない
- Arabic
- Czech
- Danish
- German
- Greek
- English (United States)
- Spanish
- Finnish
- French
- Italian
- Japanese
- Korean
- Dutch
- Norwegian Nynorsk
- Polish
- Portuguese (Brazil)
- Portuguese (Portugal)
- Russian
- Swedish
- Turkish
- Chinese (Simplified)
- Chinese (Traditional)

概要の言語 ::

English (United States) - Japanese

ロボアプリ名:

Untitled

アプリケーションID:

untitled-0a7d04

ロボアプリバージョン:

0.0.0

ロボアプリの概要 :



③概要の言語もクリックすることで編集できる

ロボアプリの対応言語 :

Japanese

概要の言語 ::

Japanese

ロボアプリ名:

ロボアプリ名

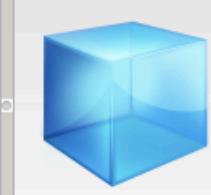
アプリケーションID:

untitled-0a7d04

ロボアプリバージョン:

0.0.0

ロボアプリの概要 :



概要の言語 :

- すべてを選択
- Arabic
- Czech
- Danish
- German
- Greek
- English (United States)
- Spanish
- Finnish
- French
- Italian
- Japanese
- Korean
- Dutch
- Norwegian Nynorsk
- Polish
- Portuguese (Brazil)
- Portuguese (Portugal)
- Russian
- Swedish
- Turkish
- Chinese (Simplified)
- Chinese (Traditional)

音声認識の精度向上

Pepperは人間が喋った任意の言葉を認識できるか？

人間が喋った言葉 (期待する結果)		実験1回目		実験2回目	
		認識された言葉	信頼度	認識された言葉	信頼度
人の名前	高橋	下半身	54.84	電話	54.21
	田中	裸	52.68	ハナカ	56.62
	中村	仲間	52.39	かもね	54.12
	庄司	しょうじき	46.28	しょうじき	49.31
挨拶の言葉	おはよう	おはよう	53.49	おはよう	78.3
	こんにちは	こんにちは	54.93	こんにちは	57.58
	こんばんは	こんばんは	56.71	こんばんは	55.37
	さようなら	さようなら	71.23	さようなら	67.34
食べ物	りんご	貧乏	61.47	流行	58.64
	すいか	した	57.92	した	52.34
	ケーキ	ケーキ	51.33	平気	57.72
	メロン	れおん	46.04	英語	49.46
数値と時間	13時40分	重さんじ四十肩	61.62	13149か	65.56
	午後2時	高校	46.62	こんにちは	49.12
	3331	そうそうじゃない	37.01	3番	40.47
	朝6時30分	さど口三重	61.86	ぼくちん	30.57

2014/12

35

－3文目－ 曖昧な表現に対応する ワイルドカード

u:(私の名前は{*}です) よろしくお願ひします

→何が入ってもいい

スクリプトエディタ

u:(私の名前は{*}です) よろしくお願ひします
u:(私は{*}が好きです) かっこいいですよね

ダイアログ

人間：私の名前は太郎です
Pepper：よろしくお願ひします

人間：私は音楽が好きです
Pepper：かっこいいですよね

—6文目— 変数の利用

input_storing

u:(_[こんにちは おはよう こんばんは])

\$1 今日も頑張ろう！

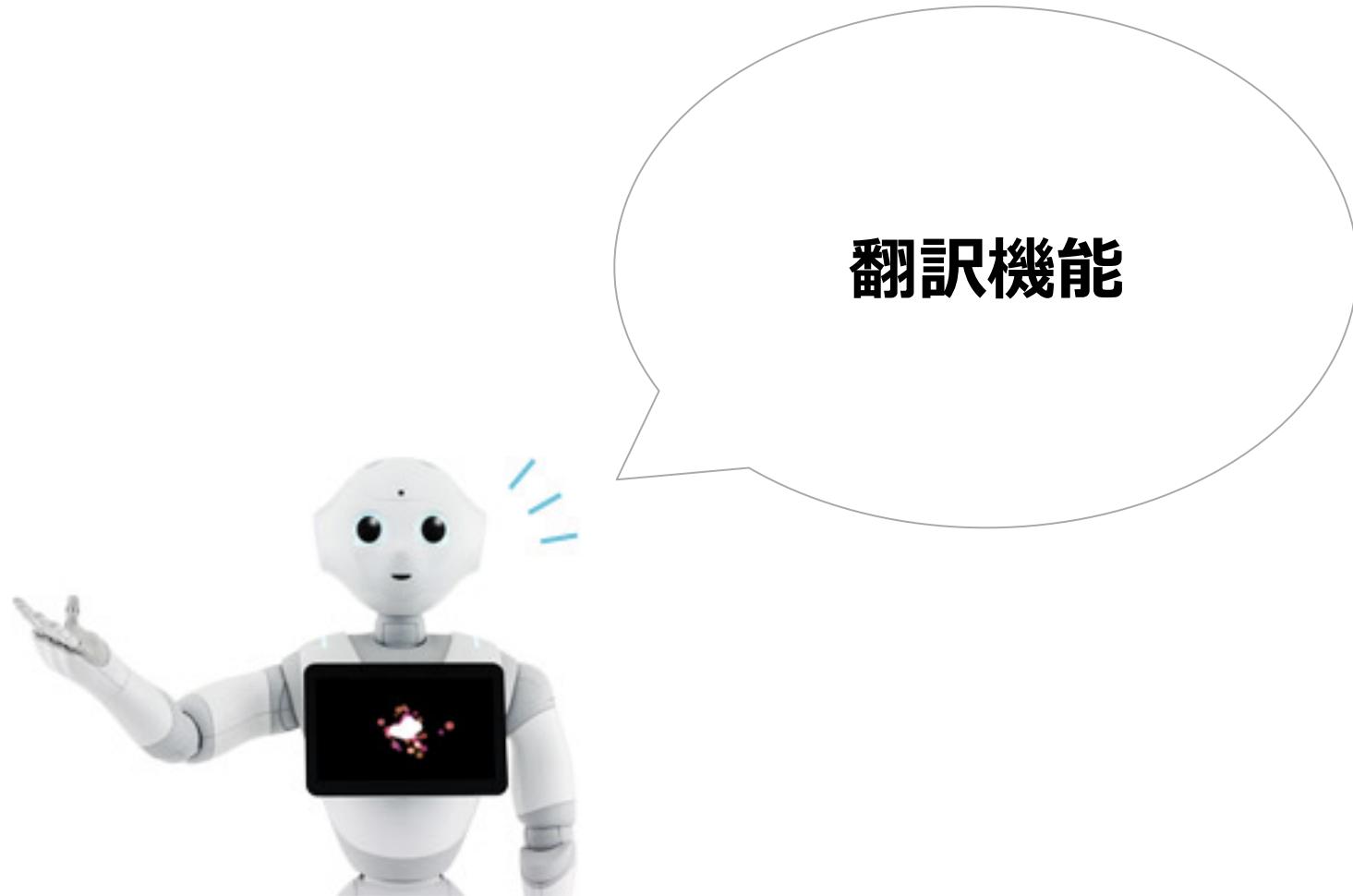
→_の後の入力を変数化して
\$1で呼び出すことができる

音声認識の精度向上

The image shows a software interface for creating dialogue flows and scripts. On the left, there is a visual flowchart editor with a vertical timeline. A blue circle on the timeline has a black line connecting it to a rounded rectangular node labeled "dialog". Inside the "dialog" node is a white speech bubble containing the text "Hi!". To the right of the flowchart is a window titled "スクリプトエディタ" (Script Editor) with the file path "dialog/dialog_jpj.top". The script content is as follows:

```
topic: ~dialog()  
language: jpj  
  
# Defining extra concepts out of words or group of  
words  
#concept:(hello) [hello hi hey "good morning"  
greetings]  
  
# Catching inputs and triggering outputs  
#u:(e:onStart) $onStopped=1  
  
# Replying to speech  
#u:(~hello) ~hello  
  
u:(_{*}) $1
```

At the bottom of the script editor window, there is a status bar with "Ln 1" and a search bar with the placeholder "検索". Below the search bar are several small green icons.



Say Box(再掲)

Textは設定ボタン➡から変更



※今までのようにダブルクリックすると、スクリプトエディタが開きます

今まで

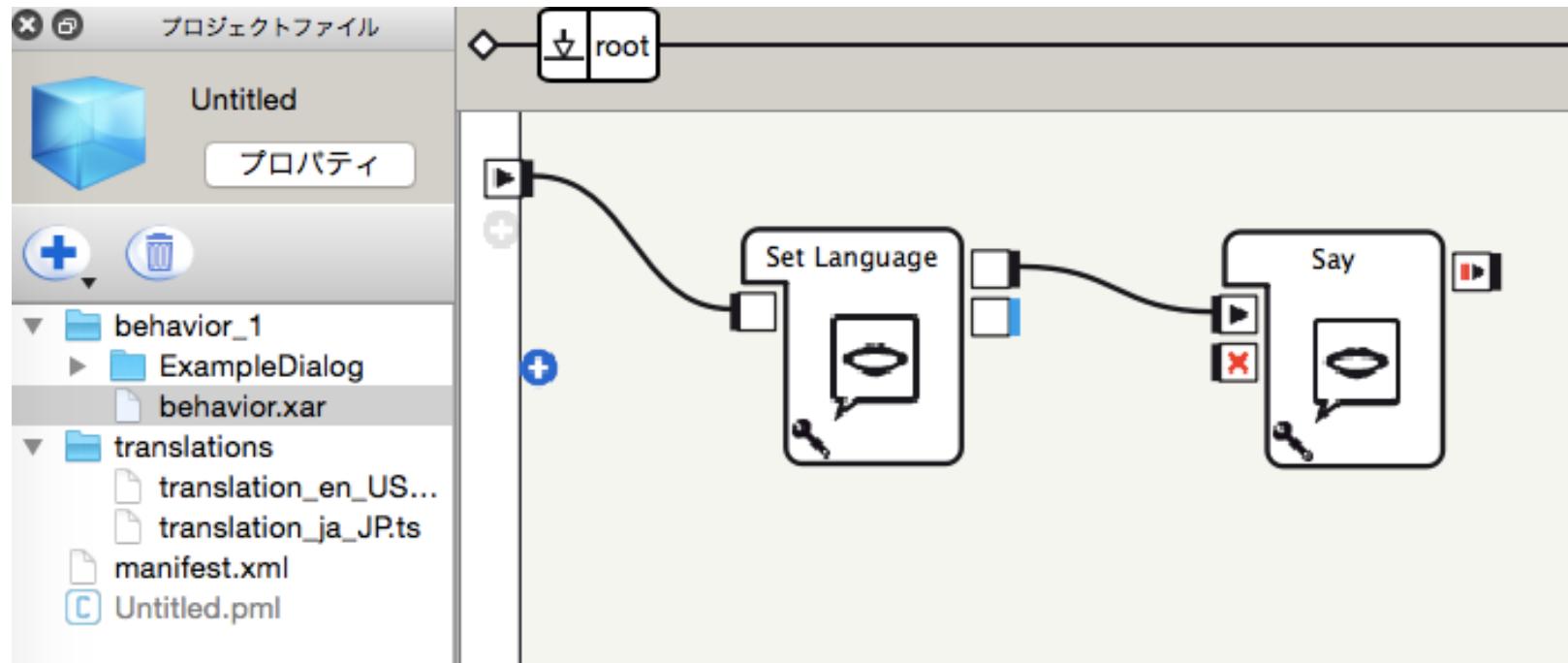
デフォルトのままonStartに繋げると
「Hello」もしくは「こんにちは」と発話

Naoqi2.5.5

デフォルトのままonstartに繋げると
言語設定が日本語の場合
「へろ(Helloのローマ字読み)」と発話
英語の場合「Hello」と発話

翻訳機能

- ロボアプリの言語設定にJapaneseを加えると、設定が日本語なら「こんにちは」、英語なら「Hello」と話す



ローカライズされたストリング



今まで ↑

Say, Animated Sayでは
Localized Textに指定された文字列を発話

Naoqi2.5.5 →

Localized Textがなくなり、新しいタイプ
「ローカライズされたストリング」が追加



翻訳ファイル(.ts)

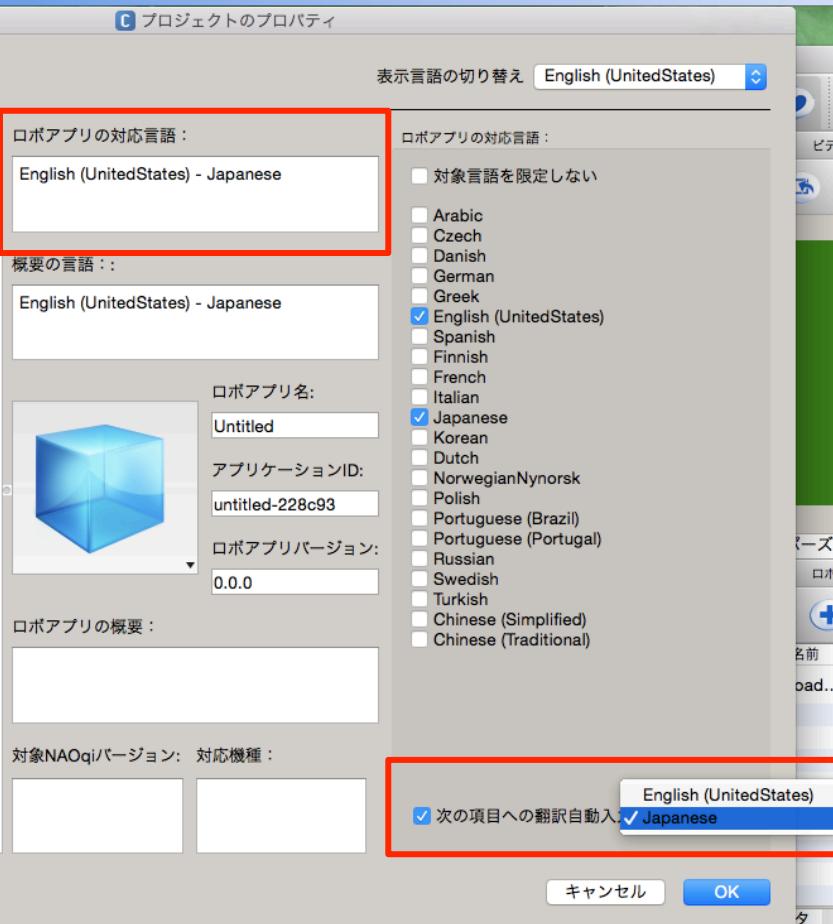
- translationsフォルダと翻訳ファイル(.ts)が追加
- 初期状態ではロボアプリの言語設定が[English(United States)]になっているため、英語用の翻訳ファイルになっている
- 「ローカライズされたストリング」のテキストはこれに記録

注意：

SayボックスのTextに
日本語を入れるときには
ロボアプリの言語設定に
「Japanese」を追加



実際に作ってみよう(1/5)



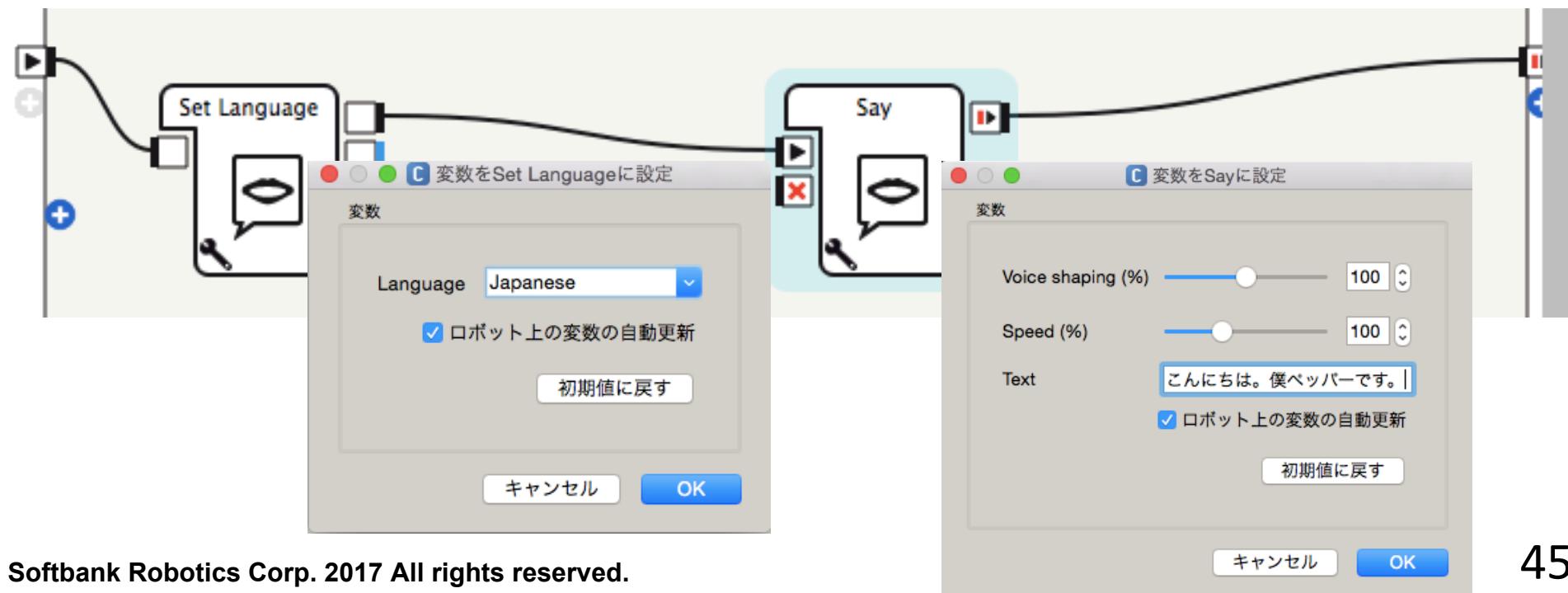
①プロジェクトのプロパティから
ロボアプリの対応言語を
EnglishとJapaneseに設定

②次の項目への翻訳自動入力は
Japaneseに設定

③翻訳ファイル(.ts)が英語と日本語
それぞれ作成されたいたら
プロジェクトを一度保存

実際に作ってみよう(2/5)

- 日本語の翻訳ファイルに自動で記録する
- 以下のようにボックスを配置、変数を設定 → 実行



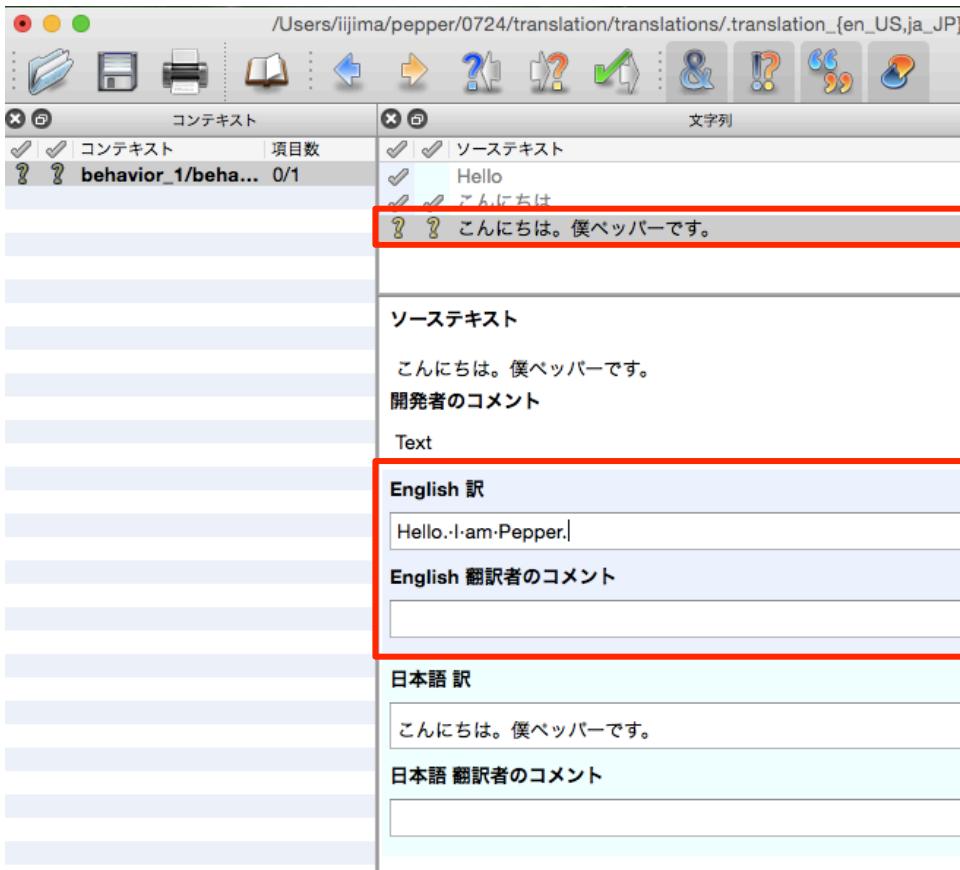
実際に作ってみよう(3/5)

- 英語の翻訳ファイルに記録
- 対応する発話内容を編集

ファイル>アプリをローカライズする
「Qt Linguist」を起動

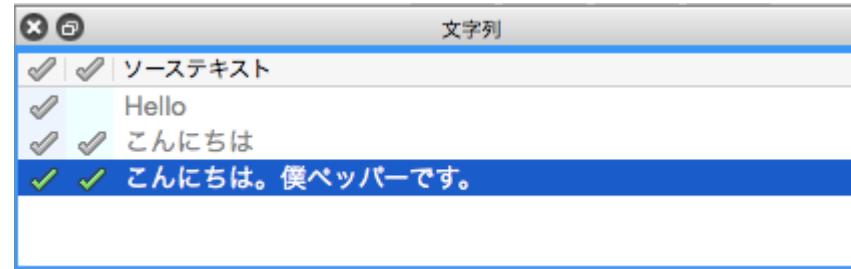


実際に作ってみよう(4/5)



①先ほど入力した文字列で
日本語訳が入っているのを確認
English訳に入力

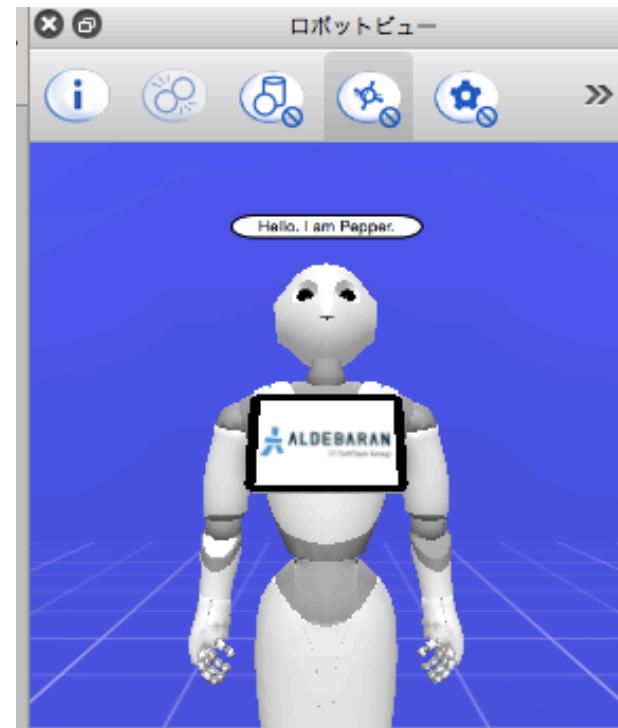
②文字列の横の?マークを
クリックして✓マークに変える

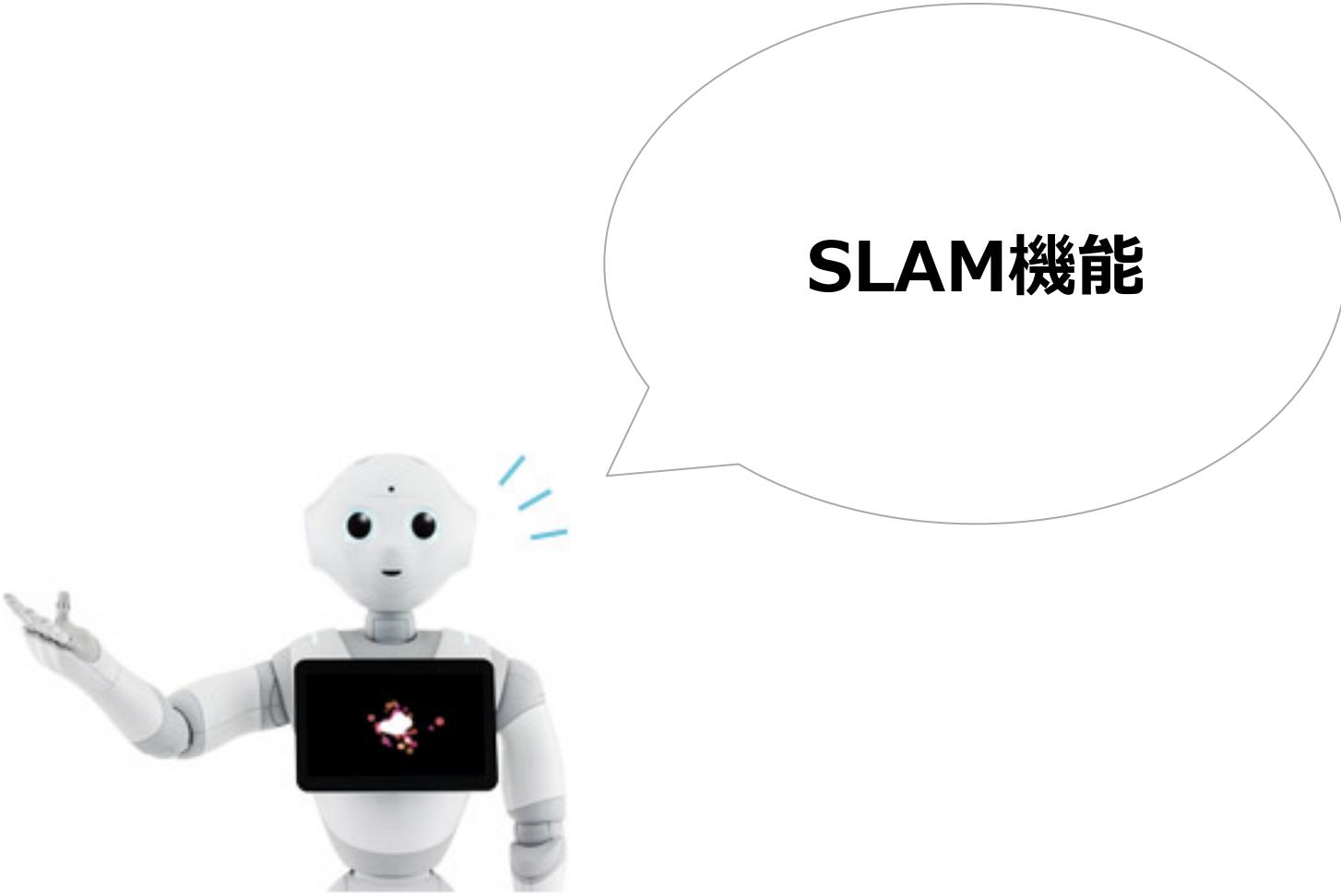


③全て保存□し、閉じる

実際に作ってみよう(5/5)

- Set Languageを[English]に設定
- 実行  → 英語で話す





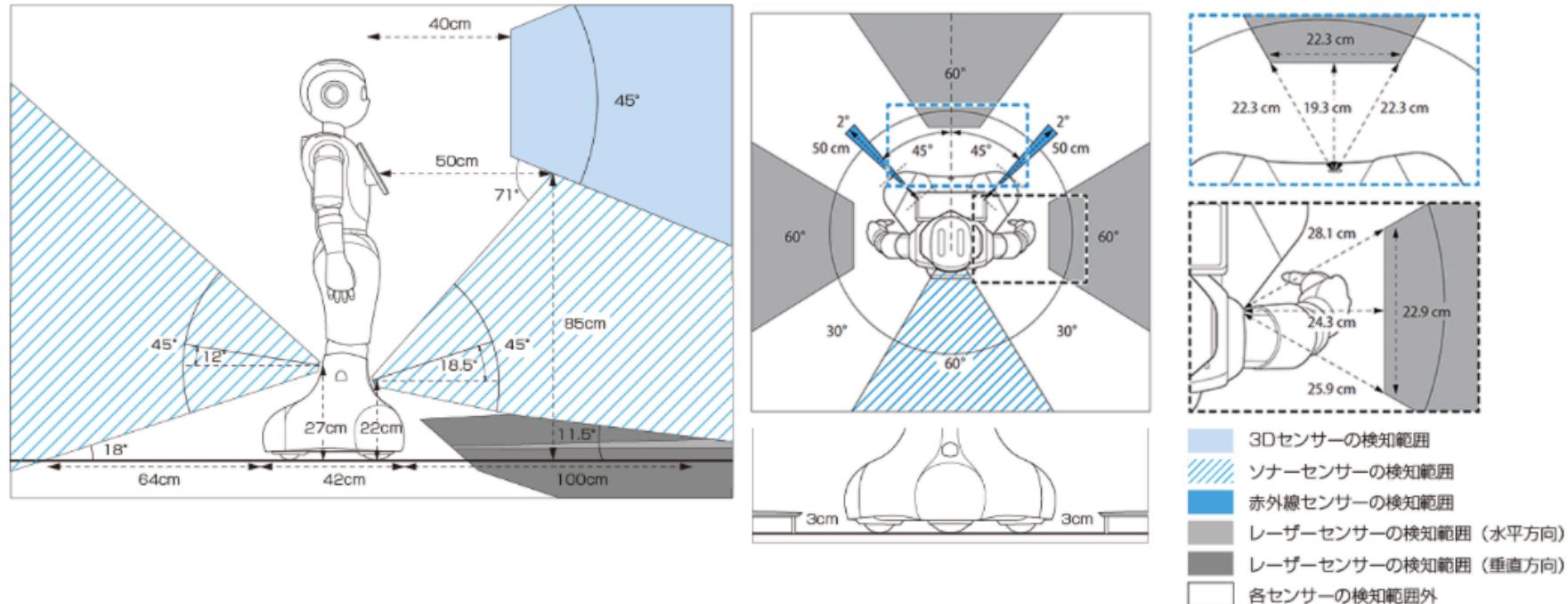
SLAM機能

SLAM機能(β版)

- Simultaneous Localization and Mapping
 - 自己位置推定と環境地図作成を同時に行う
 - PepperのSLAMは同時には行わない
 - ALNavigation API で提供
<http://doc.aldebaran.com/2-5/naoqi/motion/alnavigation.html>
- なぜβ版？
 - 仮リリースであり、サポートデスク正式サポートを受けられないため

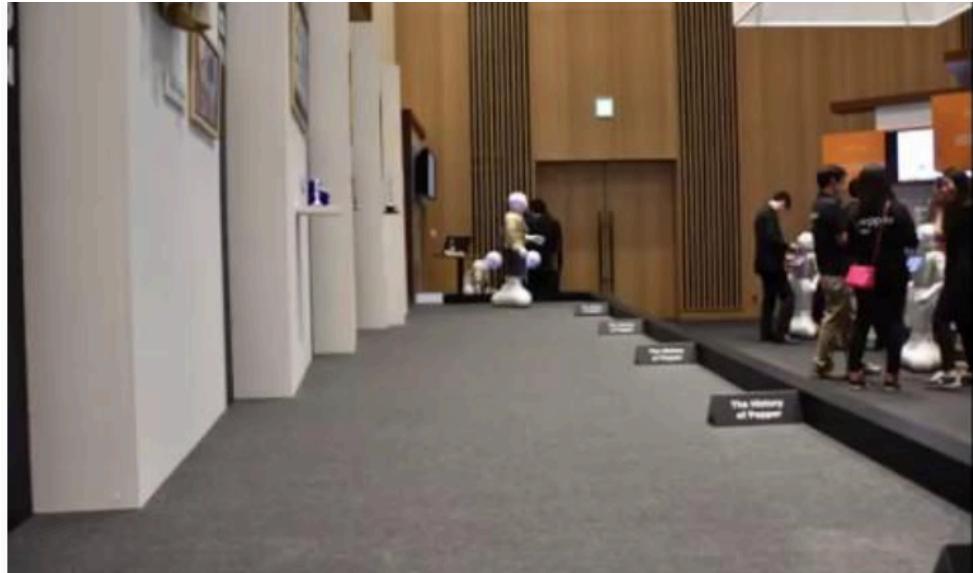
SLAM機能(β版)

- 使っているセンサー：レーザーセンサー・ソナーセンサー
→ 足元にあるセンサーを使っている



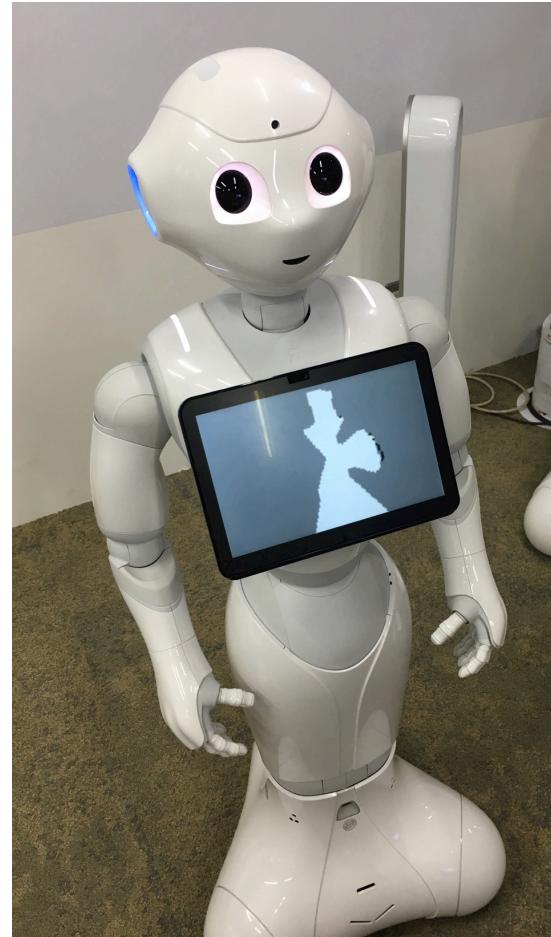
Pepperでの活用事例

- Pepper world



SLAM機能の流れ

1. 地図作成
2. 地図を保存
3. すでに地図が作成されている場合、
その地図の読み込み
4. 自分が地図上のどこにいるのか推定開始
5. 移動
6. 自己位置推定終了



地図作成

- `explore(float)`
 - Pepperが、周辺を適当に動き地図を作成
 - 引数の指定した数値を半径としてPepperが円状に探索開始
- `saveExploration()`
 - `explore`で作成した地図を、Pepperのローカルへと保存
 - 返り値：保存した地図のフルパス
 - `~.explo`という拡張子で保存される

自己位置推定

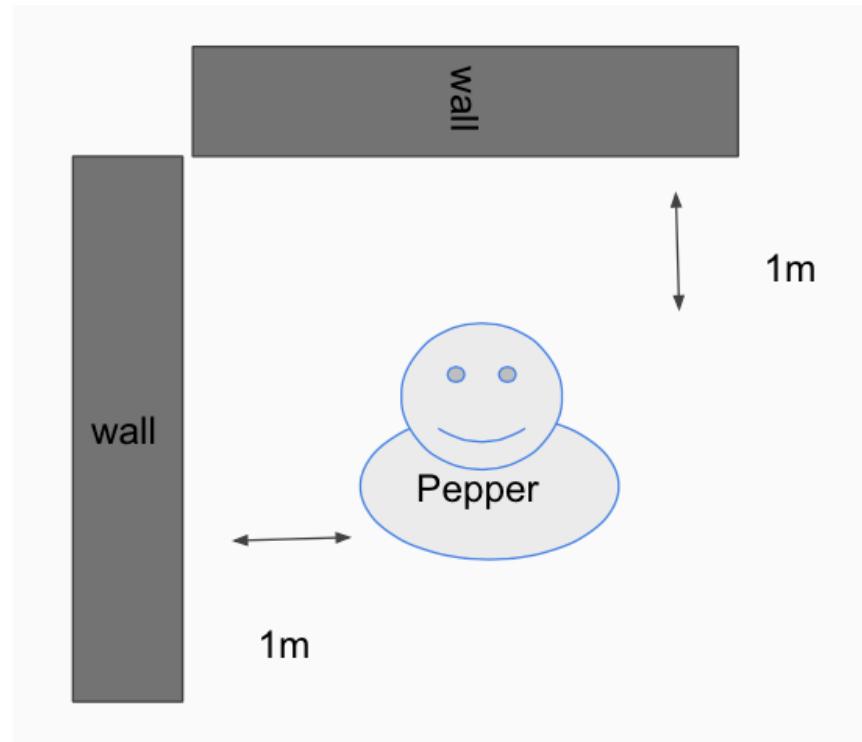
- `startLocalization()`
 - 自己位置推定を開始
 - MAPを読み込み移動する場合は、必ず実行
- `navigateToInMap(x,y,θ)`
 - 指定した位置へと移動 (0,0,0)なら探索開始位置へ
 - Naoqi OS 2.5.5では引数3つ目のθは非対応
- `stopLocalization()`
 - 自己位置推定の終了
 - これをせずに次のstartLocalization()を実行するとエラーとなる

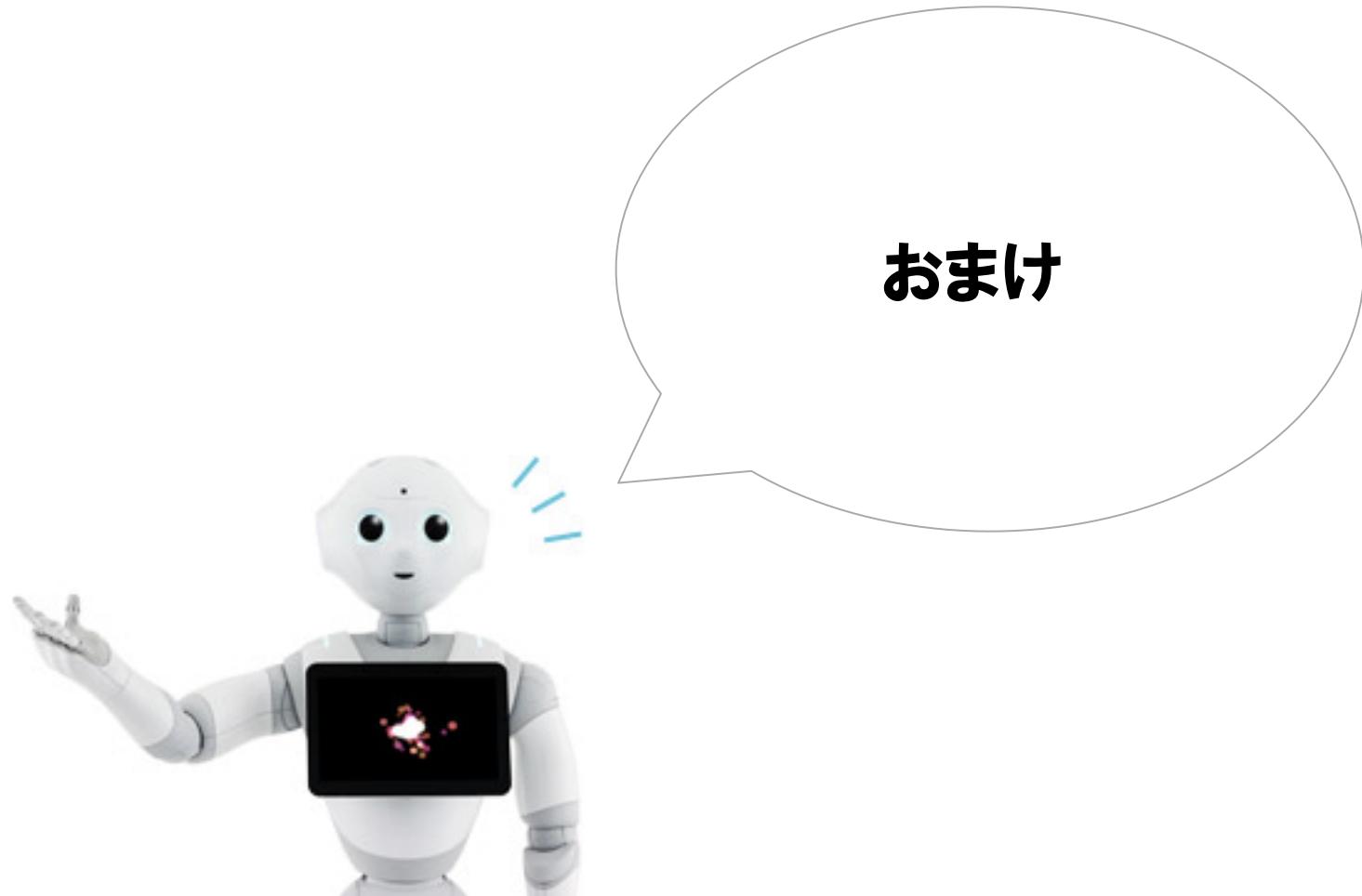
地図読み込み・移動

- `loadExploration(name)`
 - 地図作成で作成した、地図を読み込む
 - `saveExploration()`で取得したフルパスを引数に指定する
- `relocalizeInMap(x,y,θ)`
 - 自己位置推定開始時に、今Pepperがいる地点を教える
 - 必須ではないが、これをやらないとずれる可能性
 - なるべく自己位置推定開始時は、Pepperを地図作成の初期地点へと移動し、`relocalizeInMap(0,0,0)`と指定すると良い。

SLAM利用のTips

- 足元に障害物を設置
- Pepperが通れる幅は1.5m
- 壁を背にして、隅から探索を開始





Pepper デベロッパー ポータル

「Pepper developer」で検索
<https://developer.softbankrobotics.com/jp-ja>

Pepperに関するデベロッパー向けの情報を集約したポータルサイト

- ・技術ドキュメント
- ・事例を共有するショーケース
- ・Pepper SDK for Android Studioのダウンロード
- ・最新ニュースの提供

Pepper アトリエ秋葉原 with SoftBank

「アトリエ秋葉原 ブログ」で検索

- ・ペッパー開発に役立つ記事を見ることができる
- ・イベントの紹介とイベントのレポートが見ることができる
- ・tipsの項目から開発に便利なツールを手に入れることができる

アトリエ秋葉原FBグループ

「アトリエ秋葉原 FB」で検索

- ・アトリエ秋葉原のFacebookグループです
- ・情報共有や質問ができます

Qiita

「Qiita pepper」で検索

- ・エンジニアの情報交換サイト
- ・PepperタグでPepperに関する様々な技術情報がある

おつかれさまでした！
これにてWSは終わりになります。

