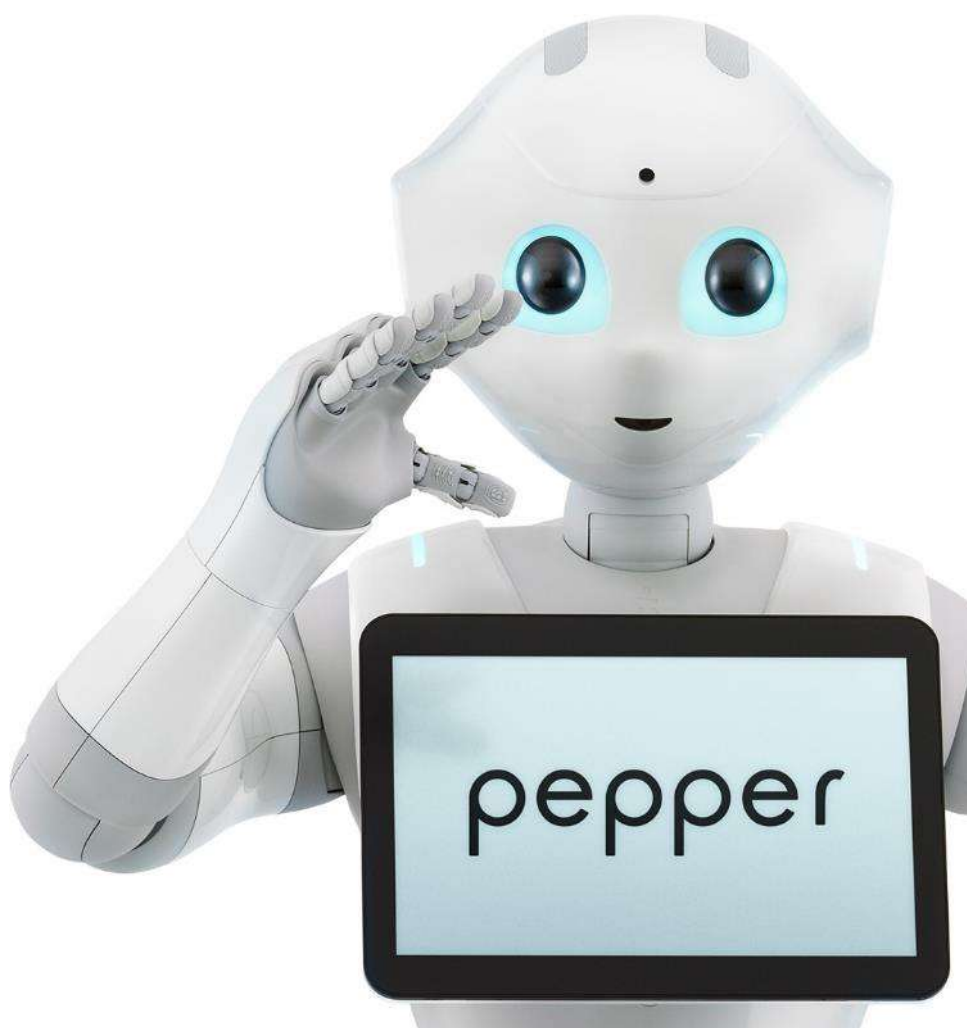


pepper

Pepper パートナープログラム

ロボアプリパートナー (Basic)

学習用ワークブック



pepper

Pepper パートナープログラム

ロボアプリパートナー (Basic)

学習用ワークブック

内容

1.	はじめに	11
1.1.	ワークブックについて	11
1.1.1.	用語の説明	11
1.1.2.	動作環境	11
1.2.	前提知識	11
1.3.	Pepper の種類	12
2.	Pepper のハードウェア仕様	12
2.1.	マイク・スピーカー・カメラ・センサーの位置と種類	12
2.2.	関節制御	18
2.3.	ディスプレイ	18
2.3.1.	新型と旧型	18
2.3.2.	解像度	19
2.3.3.	新型、旧型の見分け方	19
3.	Pepper の取り扱い方法	20
3.1.	箱からの取り出し方	20
3.2.	梱包	20
3.3.	移動	20
3.3.1.	セーフレスト状態	20
3.3.2.	手押し移動の場合	21
3.3.3.	本機を持ち上げて移動する場合	21
3.4.	起動と停止	22
3.4.1.	起動と停止のフロー	22
3.4.2.	強制終了	22
3.5.	Pepper の設定	22
3.5.1.	設定アプリの起動方法(一般販売モデル)	22
3.5.2.	設定アプリの起動方法(Pepper for Biz モデル)	23
3.5.3.	基本設定	23
3.5.4.	ネットワークの設定	24
3.5.5.	ソフトウェアアップデート	24
3.5.6.	ソフトウェアアップデート	25
3.5.7.	ロボットウェブページ	26
3.5.8.	設定アプリが起動できない場合(有線接続)	26
3.6.	Pepper for Biz との連携	27
3.6.1.1.	お仕事かんたん生成	27
3.6.1.2.	ロボアプリ配信管理	27
3.6.1.3.	インタラクション分析	28
3.7.	ロボアプリの起動方法	28
3.7.1.	ロボアプリランチャー	28
3.7.2.	お仕事かんたん生成	28
3.7.3.	その他の起動方法	28
3.7.3.1.	Pepper にアプリをインストールしていない場合	28
3.7.3.2.	Pepper にアプリをインストールしている場合	29
3.8.	肩の LED ランプ	29
4.	Choregraphe の使い方	30
4.1.	Choregraphe のインストール	30

4.1.1.	動作環境	30
4.1.2.	インストール	30
4.2.	Choregraphe の初期設定	30
4.2.1.	日本語化	30
4.2.1.1.	【Mac】	30
4.2.1.2.	【Windows】	31
4.2.2.	バーチャルロボットの変更	33
4.3.	Choregraphe の画面構成	35
4.3.1.	パネル	35
4.3.2.	ツールバー	36
4.4.	プロジェクトファイルパネル	37
4.4.1.	ボックスライブラリパネル	39
4.4.2.	フローダイアグラムパネル	39
4.4.3.	ポーズライブラリパネル	40
4.4.4.	ビデオモニターパネル	40
4.4.5.	ロボアプリ一覧パネル	41
4.4.6.	メモリウォッチャーパネル	41
4.4.7.	ダイアログパネル	42
4.4.8.	ログビューアパネル	42
4.5.	Pepper との接続	42
4.5.1.	Choregraphe の設定	42
4.5.1.1.	バーチャルロボットとの接続	43
4.5.1.2.	実機との接続	43
4.6.	基本的な開発方法	43
4.6.1.	Choregraphe のプログラミングスタイル	43
4.6.2.	プロジェクトの作成と保存	44
4.6.3.	インストール/アンインストール	45
4.6.4.	パッケージファイル	46
4.7.	はじめてのロボアプリ	48
5.	ボックス	51
5.1.	ボックスの構成要素	51
5.2.	入力・出力の性質	52
5.3.	入力・出力のデータ型	53
5.4.	変数(パラメータ)	53
5.5.	ボックスの編集	54
5.5.1.	ボックスの編集方法	54
5.5.2.	追加用のウィンドウ	56
5.5.3.	編集用のウィンドウ	58
5.6.	ボックスの種類	58
5.6.1.	Python ボックス	59
5.6.1.1.	Pytho ボックスの作成	59
5.6.1.2.	Python ボックスのスキプトの編集	59
5.6.2.	ダイアグラムボックス	60
5.6.2.1.	ダイアグラムボックスの作成	60
5.6.2.2.	ダイアグラムボックスの編集	61
5.6.3.	ダイアログボックス(Dialog ボックス)	62
5.6.4.	タイムラインボックス	62

5.7.	ボックスライブラリ	62
5.7.1.	ライブラリの追加	62
5.7.2.	ライブラリの作成	63
6.	会話	65
6.1.	関連ボックス	65
6.2.	Set Language ボックス	66
6.3.	Say ボックス	67
6.4.	イントネーションと抑揚の調整	69
6.4.1.	調整できる項目	69
6.4.2.	その他の調整方法	69
6.4.3.	セリフの確認方法	70
6.4.3.1.	ロボットウェブページで確認	70
6.4.3.2.	Choregraphe で確認	70
6.4.4.	調整済みテキストのサンプル集	71
6.5.	Say Text ボックス	71
6.6.	Animated Say ボックス	72
6.7.	Speech Reco. ボックス	74
6.7.1.	聞き取りたい言葉の登録	74
6.7.2.	認識率	75
6.7.3.	聞き取った後の処理	75
6.7.4.	聞くと話すは完全分離	77
6.8.	Dialog ボックス	78
6.8.1.	作成手順	78
6.8.2.	トピックファイル	80
6.8.3.	ユーザルール	80
6.8.4.	ユーザサブルール	81
6.8.5.	複数の単語登録	82
6.8.6.	オプションを用いた単語登録	83
6.8.7.	ボックスからの出力	83
6.8.8.	イベント	84
6.8.8.1.	イベントとは?	85
6.8.8.2.	ボックス開始のイベント	85
6.8.8.3.	聞き取れなかった時のイベント	85
6.8.8.4.	放置された時のイベント	85
6.8.9.	ビヘイビア、サウンドの実行	87
6.8.10.	proposal	87
6.8.11.	コラボラティブとノンコラボラティブ	89
6.9.	会話作成時の注意点	90
6.9.1.	聞き返しとタイムアウト	90
6.9.2.	聞き取り精度向上	90
7.	モーション	91
7.1.	ポーズ関連のパネル	91
7.1.1.	ロボットビュー	91
7.1.2.	ポーズライブラリ	91
7.2.	ポーズの作成	92
7.2.1.	バーチャルロボットによるポーズ作成	92
7.2.2.	実機によるポーズ作成	93

7.2.2.1.	アニメーションモード	93
7.2.2.2.	頭の場合	94
7.2.2.3.	腕の場合	94
7.3.	Timeline ボックス	95
7.3.1.	タイムラインパネル	95
7.3.2.	モーションの作成	96
7.3.3.	モーションとセリフの同期	98
7.3.3.1.	Behavior レイヤーの名前変更	99
7.3.3.2.	セリフの追加	99
7.3.3.3.	キーフレームの分割	101
7.3.4.	モーションとその他の処理の同期	102
7.3.5.	ミラーリング/反転	103
7.3.5.1.	インスペクタパネルを使用したミラーリング	103
7.3.5.2.	Timeline ボックスを使用したミラーリング	103
7.3.5.3.	反転	103
7.3.6.	モーションのサンプル集	104
7.4.	移動 (平面動作)	104
7.5.	会話系ボックスとの連携	106
7.5.1.	新規 Behavior の作成	106
7.5.2.	セリフのモーションを組み込む	107
7.6.	モーション作成時の注意点	107
7.6.1.	ポーズを 1 フレーム目に登録しない	108
7.6.2.	ポーズを連続で登録しない	108
7.6.3.	中途半端な姿勢を維持しない	109
7.6.4.	腰の角度を深くしすぎない	109
7.6.5.	タイムラインは 500 フレーム以内	109
7.6.6.	インスペクタパネルを確認する	109
7.6.7.	センサーの死角や予測できない利用者の動き	110
7.6.8.	人と目を合わせる	111
7.6.9.	開発時	111
7.6.10.	独自モーションとの競合回避	111
8.	イベント	111
8.1.	イベントを検知する方法	111
8.2.	タッチ系イベント	112
8.3.	対人系イベント	115
9.	ディスプレイ	117
9.1.	ディスプレイ表示の基本的な仕組み	117
9.2.	画像表示	117
9.2.1.	画像の指定	117
9.2.2.	入出力	117
9.2.3.	画像表示時の注意点	119
9.3.	タッチ位置の座標取得	119
9.4.	HTML&CSS&JavaScript の使用	120
9.4.1.	プロジェクトの構成	120
9.4.2.	HTML ファイルの表示/非表示	121
9.5.	Pepper とディスプレイの連携	122
9.5.1.	ディスプレイ → Pepper	122

9.5.1.1.	ライブラリの読み込み	122
9.5.1.2.	独自イベントの作成	122
9.5.1.3.	イベントの発生とデータ転送	123
9.5.2.	Pepper → ディスプレイ	126
9.5.2.1.	Raise Event ボックスによるイベントの発生とデータ転送	126
9.5.2.2.	ディスプレイ側のイベント監視とデータ受信	126
9.6.	異なるディスプレイの対応	128
9.6.1.	ロボアプリの実装方法ごとの対応方法	128
9.6.1.1.	viewport が 1.335 に設定されている	128
9.6.1.2.	viewport の指定がない、もしくは 1 が設定されている	129
9.6.1.3.	画面表示使用する HTML がレスポンシブルデザインに対応している	129
9.6.2.	新型ディスプレイへの対応手順	129
9.6.3.	例外ケース	130
9.6.3.1.	CSS 側での制御があるロボアプリ	130
9.6.3.2.	jQuery ライブラリを利用していないロボアプリ	130
9.6.3.3.	複数の HTML より構成されているロボアプリ	130
9.6.4.	スクリプトの内容	130
9.6.4.1.	adjust.js	130
9.6.4.2.	adjust23.js	130
9.7.	ディスプレイ表示の注意点	131
9.7.1.	字はできるだけ大きくする	131
9.7.2.	UI 要素の配置が上下左右に偏らないようにする	131
9.7.3.	文字と背景のコントラスト比はできるだけ高くする	131
9.7.4.	ボタン連打に対応する	131
9.7.5.	タッチした時に表示が上下左右にブレないようにする	131
9.7.6.	ピンチイン/ピンチアウトで拡大/縮小しないようにする	131
9.7.7.	アプリ起動中はバブル状態に戻らないようにする	133
9.7.8.	アプリが終了したら表示した内容を消す	133
9.7.9.	操作は音声とディスプレイ両方でできるようにする	133
9.7.10.	画像作成時の注意点	133
9.7.11.	ディスプレイのタイムアウト処理	133
10.	オートノマスライフ	135
10.1.	オートノマスライフとは	135
10.1.1.	オートノマスライフの主な機能	135
10.1.1.1.	インストールされているアクティビティの自律的な起動	135
10.1.1.2.	Basic Awareness と Breathing Animation の自動実行	136
10.1.1.3.	ライフサイクル管理	137
10.1.1.4.	安全確保のための反応	138
10.1.2.	アクティビティ	139
10.1.3.	オートノマスライフの停止と再開	139
10.2.	Interactive アクティビティ	140
10.2.1.	性質の設定	140
10.2.2.	トリガーセンテンス	141
10.2.3.	アプリアイコン	142
10.2.3.1.	アイコンの作成要領	142
10.2.3.2.	アイコンのデザイン可能範囲	142
10.2.3.3.	アイコンテンプレートの入手	143

10.2.3.4.	アイコンの設定	143
10.3.	Solitary アクティビティ	143
10.3.1.	性質の設定	144
10.3.2.	起動トリガー条件	144
10.3.2.1.	起動トリガー条件の設定方法	144
10.3.2.2.	Launchpad イベント	146
10.3.2.3.	ビヘイビアパス	147
10.4.	アクティビティの連携	149
11.	標準ボックス	150
11.1.	Take Picture ボックス	150
11.2.	Record Sound ボックス	151
11.3.	Play Sound ボックス	151
11.4.	Record Video ボックス	151
11.5.	Play Video ボックス	152
11.6.	Basic Awareness ボックス	154
11.6.1.	Breathing	155
11.7.	Comment ボックス	155
11.8.	Log ボックス	156
12.	NAOqi OS(API)	156
12.1.	API ドキュメントの見方	157
12.2.	NAOqi Vision API	159
12.2.1.	ALPhotoCapture	159
12.2.2.	ALPeoplePerception	160
12.2.3.	ALFaceCharacteristics	161
12.3.	NAOqi Audio API	162
12.3.1.	ALVoiceEmotionAnalysis	162
12.4.	NAOqi Motion API	163
12.4.1.	ALNavigation	163
12.5.	NAOqi Sensors & LEDs	163
12.5.1.	ALLeds	163
13.	お仕事かんたん生成 2.0	165
13.1.	お仕事の概要	165
13.1.1.	お仕事について	165
13.1.2.	お仕事かんたん生成 2.0 について	165
13.2.	お仕事かんたん生成 2.0 の基本設定	165
13.2.1.	Pepper の機体リストを管理する	166
13.3.	お仕事の新規作成	166
13.4.	キーワード	166
13.4.1.	語尾を高くする	167
13.4.2.	キーワードに間を入れる	167
13.4.3.	キーワードを登録する	167
13.5.	メディアライブラリ	169
13.6.	お仕事の編集	169
13.7.	ボックスの操作	173
13.7.1.	各ボックスでの編集画面の共通操作について	174
13.7.2.	ディスプレイ設定	174
13.7.2.1.	セリフ設定	175
13.7.3.	呼び込みボックス	177
13.7.4.	トークボックス	178

13.7.5.	メニューボックス	179
13.7.6.	質問ボックス	179
13.7.7.	公式アプリボックス	180
13.7.8.	印刷ボックス	180
13.7.9.	ジャンプボックス	182
13.7.10.	ベンダーアプリボックス	182
13.7.11.	属性分岐ボックス	183
13.7.12.	ランダム分岐ボックス	184
13.8.	お仕事の管理	184
13.8.1.	お仕事を管理する	185
13.8.2.	お仕事一覧画面の見かた	185
13.8.3.	お仕事を複製する	186
13.8.4.	お仕事の閲覧に制限をかける	186
13.8.5.	お仕事の配信期間を設定する	187
13.8.6.	お仕事データをエクスポート／インポートする	187
13.9.	ロボアプリの配信	190
13.9.1.	アプリリストから配信する	190
13.9.2.	パートナーアプリを配信する	190
13.9.3.	マイアプリを配信する	190
13.10.	Pepper 側でのお仕事の更新	191
13.10.1.	Pepper 側でお仕事を更新されるタイミング	191
14.	インタラクション分析	191
14.1.	インタラクション分析	191
14.1.1.	取得できるデータについて	191
14.1.2.	インタラクション分析で表示される情報	192
15.	Pepper for Biz	192
15.1.	一般販売モデルとの比較 (仕様編)	192
15.2.	一般販売モデルとの比較 (開発条件編)	193
15.3.	ロボアプリ品質チェックリスト	194
15.4.	ロボアプリ配信管理	194
15.5.	ビヘイビアパス	195
15.6.	ソフトウェアの違い	196
15.7.	ロボアプリ公開申請	196
15.7.1.	manifest.xml の修正	196
15.7.2.	ロボアプリバージョンの説明	197
15.7.3.	manifest.xml のエレメント	198
16.	付録	200
16.1.	演出	200
16.1.1.	人を惹きつける方法	200
16.1.2.	安全性の確保	200
16.2.	再起動方法	200
16.2.1.	通常再起動	200
16.2.2.	NAOqi のみ再起動	201
16.2.3.	強制再起動	201
16.2.4.	緊急停止	201
16.2.5.	自己修復モードで起動	201
16.3.	qicli コマンド	201

16.3.1.	SSH 接続の方法	201
16.3.2.	セーフティー機能解除	202
16.3.3.	イベント発生	202
16.3.4.	ログを確認する	203
16.4.	ロボットウェブページ (advanced)	203
16.5.	運用時の注意点	204
16.5.1.	ネットワーク環境	204
16.5.2.	設置場所	204
16.5.3.	複数台体制の検討	204
16.6.	Pepper の商標・キャラクターについての制限	204
16.7.	メディア掲載・公共環境での稼働について	205
16.7.1.	禁止事項	205
16.7.2.	事前申請が必要	205
16.7.3.	キャラクターに関する制限	205
16.7.4.	商標について	205
16.7.5.	名称 (Pepper) に関する制限	205
16.7.6.	Pepper の画像について	206

1.はじめに

1.1. ワークブックについて

本書はソフトバンクロボティクス株式会社 Pepper パートナープログラム事務局が、技術者を対象として「ロボアプリパートナー（Basic）」用に作成した学習用ワークブックとなります。内容等確認して頂く場合、Pepper 本体が必要になる場合もありますので、ご了承ください。

1.1.1. 用語の説明

本書で使用する用語の一部を、以下で説明します。

表 1.1-1 用語の説明

用語	説明
ロボアプリ	Pepper 向けに開発されたアプリ。
ボックス	処理の単位。ロボアプリはボックスと言う小さい処理の単位を組み合わせてアプリの開発を行います。
ビヘイビア(behavior)	ボックスを組み合わせて作成した一連の処理の単位。1つのロボアプリが複数のビヘイビアから構成される場合もあります。

1.1.2. 動作環境

本書に記載されているサンプルコードは以下の環境にて動作させることを前提に作成されています。

表 1.1-2 動作環境

環境	バージョン
Choregraphe	2.5.5
NAOqi	2.5.5

1.2. 前提知識

本書は以下の前提知識を習得されている方向けに作成されています。前提知識の習得に関しては、事前に各自で実施しておく必要があります。

表 1.2-1 前提知識

前提知識	主な用途
Python 2.7.x	Choregraphe 上で NAOqi の API を利用する際に使用します。
HTML、CSS	Pepper の胸のディスプレイに画面表示を行う際は、HTML、CSSにて画面のレイアウトを行います。
javascript、jQuery	画面表示に使用する HTML、CSS を動的に更新したり、Pepper 本体とディスプレイを連携させる際に使用します。

1.3. Pepper の種類

Pepper には、大きく以下の 3 種類が存在します。

表 1.3-1 Pepper の種類

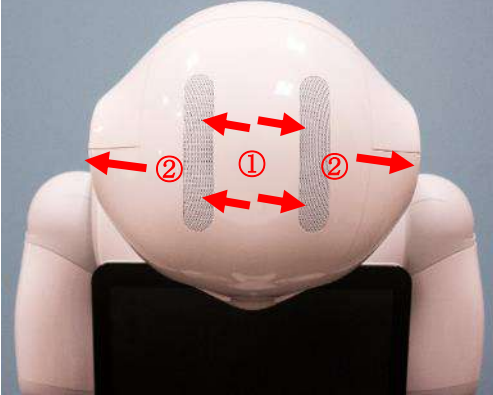
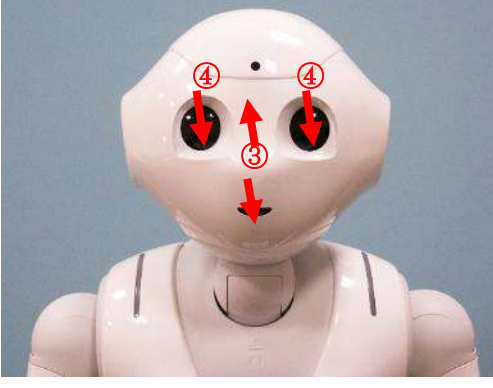
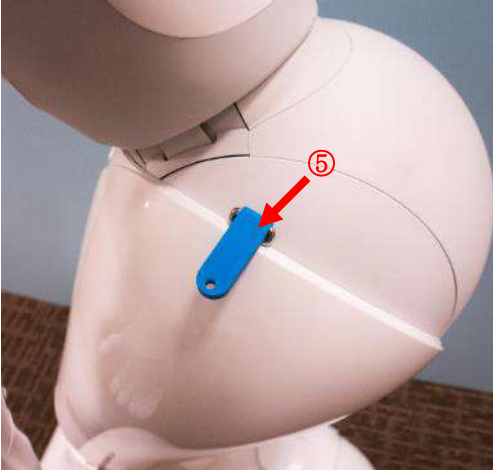
種別	説明
デベロッパー先行モデル	Pepper リリース当初に、開発者向けに販売された先行モデルで、現在は販売を行っていません。一部、CPU が古い場合などがあるため、開発や運用にデベロッパー先行モデルを利用する場合は注意が必要です。デベロッパー先行モデルは本書の対象範囲外です。
一般販売モデル	家庭向けに販売されているモデルです。会話アプリなど家庭向けのアプリが標準搭載されています。
Pepper for Biz モデル	ビジネス向けに販売されているモデルです。 お仕事かんたん生成と呼ばれる CMS にて、受付機能や接客機能のカスタマイズやロボアプリのリモートインストールなどを行うことができます。また、一般販売モデルで利用できる機能の一部が Pepper for Biz モデルでは利用できません。 2017 年 4 月 4 日以降の Pepper for Biz モデルは胸のディスプレイの仕様変更されていることにも注意が必要です。


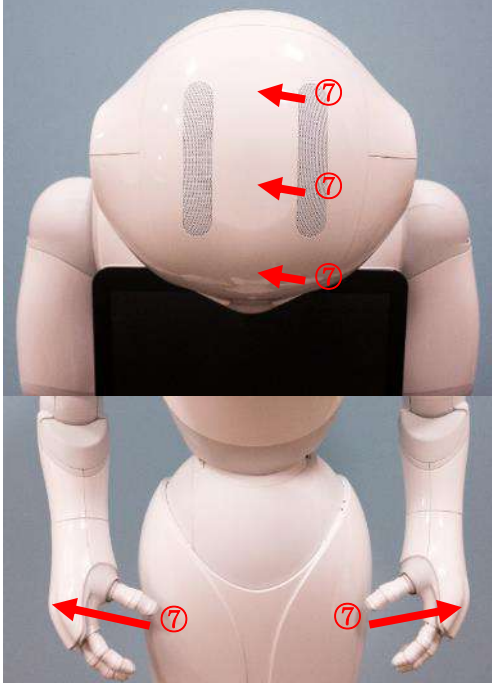

2. Pepper のハードウェア仕様

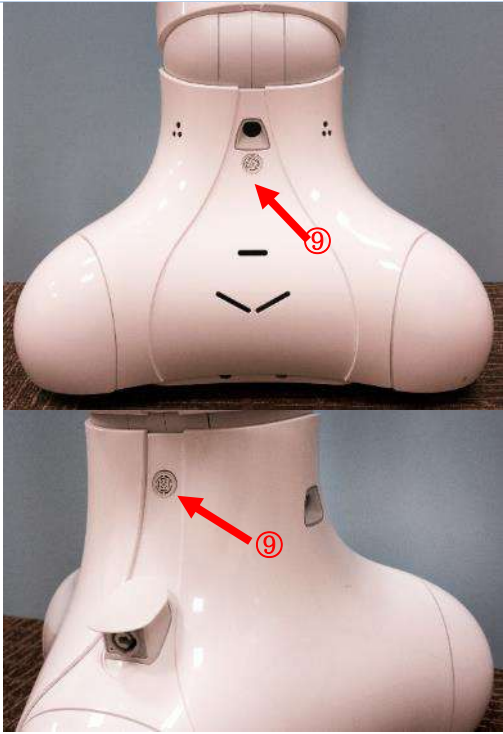

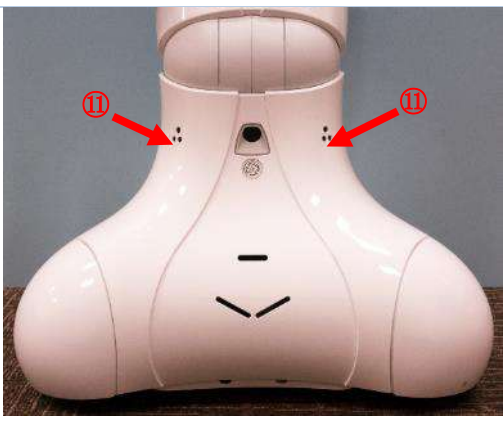
本項では、Pepper に搭載されているセンサーやデバイスのなどのハードウェアについて説明します。

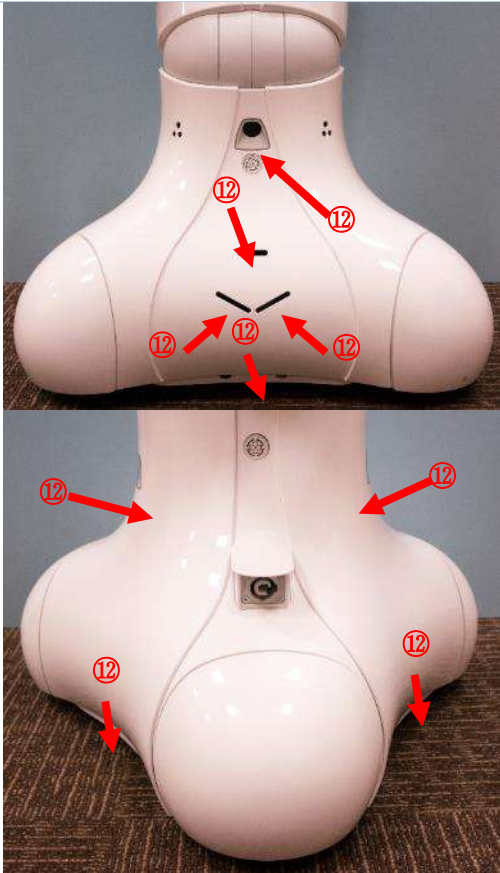
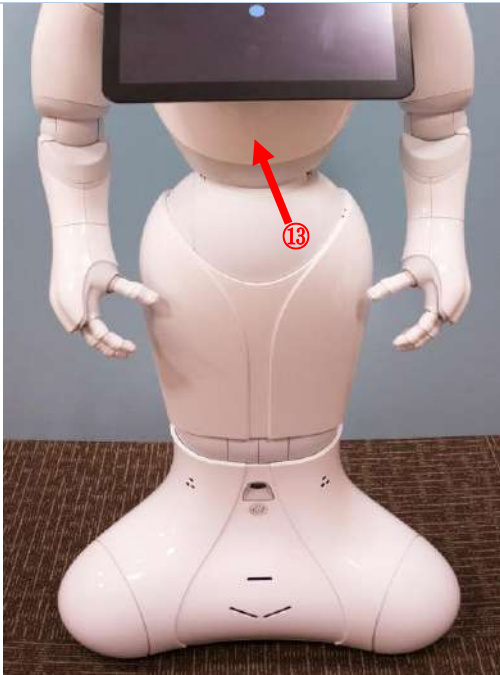
2.1. マイク・スピーカー・カメラ・センサーの位置と種類

表 2.1-1 マイク・スピーカー・カメラ・センサーの位置と種類

	<p>①マイク</p>	<p>頭部の左の前後、右の前後</p>
	<p>②スピーカー</p>	<p>頭部の左右の耳の部分 (ディスプレイ)</p>
	<p>③2D カメラ</p>	<p>額と口 【出力 2560×1920, 1fps】 【出力 640×480, 30fps】</p>
	<p>④3D カメラ</p>	<p>右目 (赤外線照射) 左目 (赤外線検出) 【ASUS Xtion 3D センサー】 出力 320×240, 20fps】</p>
	<p>⑤解除キー 腰</p>	<p>腰のブレーキ解除キー</p>

	<p>⑥解除キー 脚</p>	<p>脚のひざ部のブレーキ解除キー</p>
	<p>⑦タッチセンサー</p>	<p>頭部中央の前方、中央、後方 左右の手の甲</p>
	<p>⑧バンパーセンサー</p>	<p>脚部の前方の左右と後方</p>

	<p>⑨ソナー</p>	<p>脚部の正面と後方</p>
	<p>⑩充電フラップ</p>	<p>脚部後部の AC アダプタの 充電プラグを接続する</p>
	<p>⑪赤外線センサー</p>	<p>脚部 2 か所</p>

	<p>⑫レーザーセンサー</p>	<p>脚部の正面と左右前方の3か所穴が開いているところ 足元の前方、左後方、右後方</p>
	<p>⑬ジャイロセンサー</p>	<p>胸部と脚部</p>

 A close-up photograph of a white robot's chest area. A red arrow points to a small, circular button on the robot's chest, which is labeled with the number 14 in a red circle.	<p>⑭胸部ボタン</p>	<p>ディスプレイの裏側</p>
--	---------------	------------------

2.2. 関節制御

ロボアプリから制御できる Pepper の関節について把握しておきましょう。

表 2.2-1 Pepper の関節

部位		動き
首		<ul style="list-style-type: none"> ・首を支点にして頭を左右に振る動き(ヨー) ・首を支点にして頭を上下に振る動き(ピッチ)
腕	肩	<ul style="list-style-type: none"> ・肩を支点にして腕全体を前後に回す動き(ピッチ) ・肩を支点にして腕全体を左右に開く動き(ロール)
	肘	<ul style="list-style-type: none"> ・肘を支点にして腕を上下にひく動き(ロール) ・肘を支点にして腕を左右に振る動き(ヨー)
	腕	<ul style="list-style-type: none"> ・腕を支点にして手を裏表に回す動き(ヨー)
	手	<ul style="list-style-type: none"> ・手のひらを開閉する動き
下半身	腰	<ul style="list-style-type: none"> ・腰を支点にして上半身を前後に傾ける動き(ピッチ) ・腰を支点にして上半身を左右に傾ける動き(ロール)
	膝	<ul style="list-style-type: none"> ・膝を支点にして上半身を前後に傾ける動き(ピッチ)

2.3. ディスプレイ

Pepper の胸部にはタッチセンサー付きディスプレイ(タブレット: Android 4.0.4 ベース)が搭載されています。

2.3.1. 新型と旧型

2017年4月4日以降に出荷される Pepper for Biz モデルは、新しい NAOqi 2.5.5 が搭載されディスプレイのアップグレードも行なわれました。新型ディスプレイは、旧型と比較し、画面の解像度は下がりますが処理速度が早くなります。ハードウェア仕様が異なるため、新旧のディスプレイに対応するためにはロボアプリ側での対応が必要になります。

2.3.2. 解像度

Pepper のディスプレイの解像度は以下の通りです。

表 2.3-1 ディスプレイの解像度

タイプ (ハードウェアバージョン)	解像度
旧型 (1.6、1.7)	1707px(W) × 1067px (H)
新型 (1.8a)	962px(W) × 601px(H)

2.3.3. 新型、旧型の見分け方

新型ディスプレイを搭載した Pepper for Biz モデルは、Robot ID (Body ID) が AP990483 以降となり、製品型番は TR18AA1 (請求書表記 18a) となります。Robot ID (Body ID) は首の後ろのカバーを外したところにある緊急停止ボタン付近に表記されています。



図 2.3-1 Robot ID の位置

3. Pepper の取り扱い方法

Pepper が届いたら梱包箱から安全に取り出す必要があります。取り出した後は Pepper を適切な場所へ移動してください。間違った手順で Pepper を扱うと故障や転倒の原因になりますので、本項で基本的な扱い方を確認しましょう。

3.1. 箱からの取り出し方

Pepper を箱から取り出す際の手順は、下記のとおりです。

1. 梱包箱を立てた状態で蓋を開けてください（梱包箱の上下を間違わないように注意）。
2. 梱包箱の上部のミミ（フラップ）を固定用の穴に差し込み固定します。
3. Pepper を押さえながら緩衝材を取り出します（関節が固定されていないため、Pepper が前のめりになり、倒れやすいので注意）。
4. 首を前に倒して、脇を抱えて持ち上げ、外に出してください。
5. アクセサリーボックス（充電器一式）を取り出します。

3.2. 梱包

Pepper を梱包する際の手順は、下記のとおりです。

1. 蓋を開けます。
2. 梱包箱の上部のミミ（フラップ）を固定用の穴に差し込み固定します。
3. 梱包箱から緩衝材を取り出します。
4. Pepper をセーフレストの姿勢にし、シャットダウンします。
5. 電源 OFF の状態で、膝と腰の部分に付属の解除キーを差し、関節の固定を解除します。
6. 緊急停止ボタンを押し、意図しない動作を防止してください。
7. 充電フラップを閉じます。（閉じない状態で梱包すると封緘時に緩衝材が浮き、故障の原因になりますので必ず閉じてください。）
8. Pepper の前面から脇を抱えて持ち上げ、足の方から押し込みます。
9. 梱包箱をゆっくり倒してください。
10. 頭、肩、手、腕を押し込みます。
11. アクセサリーボックス（充電器一式）を入れ、緩衝材（胸部のディスプレイ用および蓋）をはめます。
12. 梱包箱を閉じます。

3.3. 移動

Pepper を移動させる際の注意点は以下の通りです。

3.3.1. セーフレスト状態

セーフレスト状態は、Pepper の重心を低く保つことができ、倒れにくい姿勢です。

Pepper はスリープモード、レストモード、もしくはシャットダウンの際にセーフレスト状態になります。

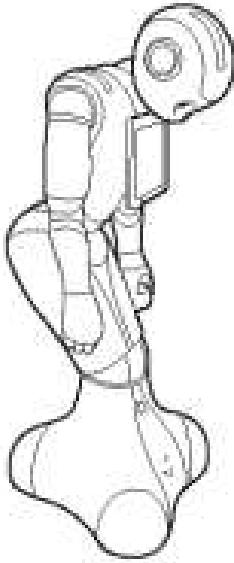


図 3.3-1 セーフレスト状態の Pepper

3.3.2. 手押し移動の場合

1. 姿勢を安定させるため、セーフレストの姿勢にします。
 2. 腰／ひざの解除キーがついている場合は取り外します。
 3. 充電フラップを開きます。
 4. 重心が足元にあるため、肩に手を置き、もう一方の手をおしりにあてて静かに前に押し移動させます。
 5. 周囲（特に進行方向の足元）に障害物がないか確認しながら移動させます。
- ※ 移動をしている最中に緊急停止ボタンや胸部ボタンを押さないよう注意してください。
 - ※ 電源 ON 時は、Pepper のホイール以外は動作が継続しているので注意してください。
 - ※ 移動の際は、電源 OFF、もしくはレストモードで行ってください。スリープモードでもセーフレスト状態になりますが、頭部のタッチセンサーを触るだけで動き出し、ため危険です。

充電フラップを開けることでオムニホイールを停止することができます。手押しで移動する際に、オムニホイールが急に動作すると、転倒などに繋がりがやすく非常に危険です。手押しで移動する際は充電フラップを開けるのを忘れないようにしましょう。安全上、オムニホイールの動作を制限したい場合にも、充電フラップの開閉は利用することができます。

3.3.3. 本機を持ち上げて移動する場合

1. シャットダウンし、セーフレスト姿勢にする
2. 緊急停止ボタンを押します。
3. 背後から本機を支えながら腰とひざの解除キーを取り付けます。
4. 背後から胸の下に手を入れ、抱きかかえるようにして持ち上げて移動させてください。

3.4. 起動と停止

3.4.1. 起動と停止のフロー

- 電源 OFF のときに胸部ボタンを 1 回押すと起動します。
- 起動時（ウェイクアップモード時）に胸部ボタンを 2 回押すとレストモードになります。
- レストモード時に胸部ボタンを 2 回押すとウェイクアップモードになります。
- ウェイクアップモード時に額のカメラを隠しながら、前頭部の一番手前の頭部タッチセンサーを 3 秒タッチすると、スリープモードになります。
- スリープモード時に頭部のタッチセンサーを触ると、ウェイクアップモードになります。

※ 起動時、停止時に聞こえる、「OGNAK GNUK」と「GNUK GNUK」は、Bidi 語というロボット用の言葉で、それぞれ「こんにちは」と「さようなら」という意味です（ソフトバンクロボティクスのコミュニティサイトより）。

3.4.2. 強制終了

- 電源 ON 時は胸部ボタンを 3 秒間押すと通常のシャットダウンを行います。
- 胸部ボタンを 8 秒以上押すと強制シャットダウンを行います。強制シャットダウンはアプリの終了処理などが呼ばれずデータが保存されないことがありますので注意してください。
- 緊急時は首の後ろのカバーの中にある緊急停止ボタンを押すと、頭および体への電気供給がすべて停止し、即座に動作を終了することができます。
- 緊急停止ボタンは緊急時以外に使用しないでください。
- 緊急停止ボタンはカバーの上からたたき押すことも可能です。
- 緊急停止ボタンを押すと、起動にロックがかかり、ロックを解除するまで電源は入りません。
- 緊急停止ボタンをつまんで時計回りに回すことでロックを解除することができます。
- 移送時には通常のシャットダウンを行った後に、緊急停止ボタンを押して起動のロックをかけてください。移送時に箱の中で電源が入ると故障の原因となります。

3.5. Pepper の設定

Pepper の声の大きさ（スピーカーのボリューム）、ディスプレイの輝度、ネットワーク環境の設定などは、標準でインストールされている設定アプリで行います。

3.5.1. 設定アプリの起動方法(一般販売モデル)

一般販売モデルでの設定アプリの起動方法は以下の通りです。

1. Pepper 起動後、バブル画面が表示されます。
2. バブル画面をタッチするとロボアプリランチャーが起動します。
3. ロボアプリランチャー内の設定アプリのアイコンをタップします。
4. 設定アプリが起動し、設定画面が表示されます。

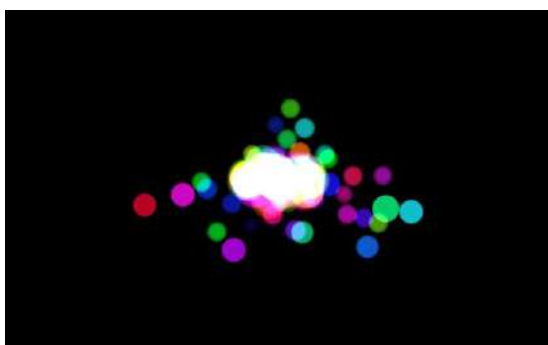


図 3.5-1 バブル画面



図 3.5-2 ロボアプリランチャー

3.5.2. 設定アプリの起動方法(Pepper for Biz モデル)

一般販売モデルと Pepper for Biz モデルでは、設定画面の表示方法が異なります。Pepper for Biz モデルでの設定アプリの起動方法は以下の通りです。

1. Pepper 起動後、お客様の呼び込み動作が始まります。
2. 1m 以内で人を認識したら呼び込み動作が終了します。
3. 呼び込み動作が終了したら、Pepper のディスプレイの左上 (赤枠内) を長押しします。
4. 管理メニューが表示されパスワードを入力し [OK] をクリックします。
5. [設定] ボタンをクリックします。
6. [基本設定] ボタンをクリックします。
7. 設定アプリが起動し、設定画面が表示されます。

※ 管理メニューのパスワードの初期値は"9999"です。



図 3.5-3 長押しするエリア

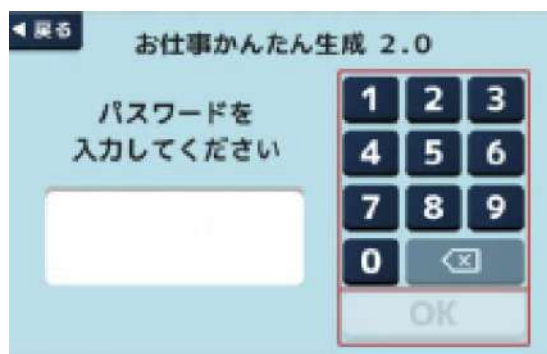


図 3.5-4 管理メニュー

3.5.3. 基本設定

顔のアイコンのボタンをタッチすると、声の大きさやディスプレイの輝度を調整できます。また、NAOqi のバージョンやバッテリー充電率を確認することができます。



図 3.5-5 設定アプリ (基本情報)

3.5.4. ネットワークの設定

地球のアイコンのボタンをタッチすると、ネットワーク環境を設定することができます。インターネットに接続されていると、ボタンの下にある小さい丸が緑に、接続されていないとオレンジ色になります。



図 3.5-6 設定アプリ (ネットワーク設定)

現在の IP アドレスは設定画面を表示して確認するか、Pepper 起動中に胸部ボタンを 1 回押して、機体名と IP を発話させることで確認することができます。

3.5.5. ソフトウェアアップデート

下向き矢印 (ダウンロード) のアイコンをタッチすると、アプリのアップデート状況を確認

認することができます。アップデートがある場合、ボタンの下にある小さい丸がオレンジ色になります。アップデートがあるときは、必ずアップデートしてから設定アプリを終了（画面右上の[X]ボタンをタッチ）させてください。

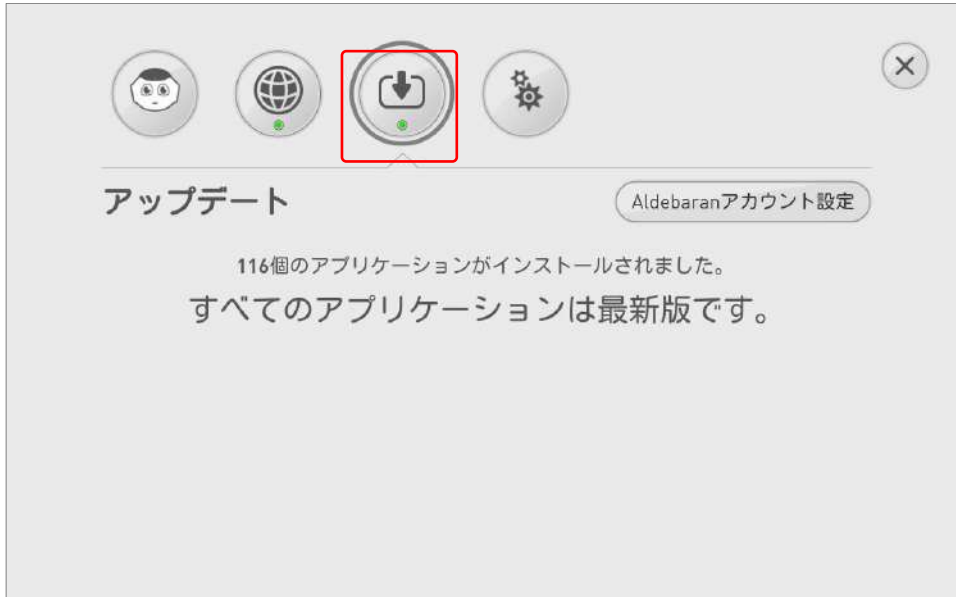


図 3.5-7 設定アプリ (アップデート)

3.5.6. ソフトウェアアップデート

歯車のアイコンをタッチすると、言語やタイムゾーンの設定ができます。



図 3.5-8 設定アプリ (詳細設定)

3.5.7. ロボットウェブページ

設定アプリと同等の画面をパソコンの Web ブラウザで表示することができます。これをロボットウェブページと言います。ロボットウェブページは、Pepper と同一ネットワーク内にあるパソコンから以下の書式の URL を開くことで表示することができます。

http://(Pepper の IP)

図 3.5-9 ロボットウェブページの URL

ロボットウェブページ表示時に Basic 認証用のダイアログが表示された場合は、以下の情報を入力することで認証を行うことができます。

username: nao

password: nao

※パスワードを変更している場合は変更したパスワードを入力してください。

ロボットウェブページには「ここだよ」と設定された吹き出しがあり、この吹き出しに任意の文字列を入力し、Enter (return) キーを押すと、入力した文字列を Pepper が発話します。簡単なセリフのイントネーションチェックなどに利用できます。



図 3.5-10 ロボットウェブページ

3.5.8. 設定アプリが起動できない場合(有線接続)

ロボアプリランチャーが起動しない、管理メニューのパスワードがわからないなどで設定アプリを起動できない場合は、Pepper とパソコンを優先接続した後にロボットウェブページを開くことで設定を行うことができます。

有線接続の手順は以下の通りです。

1. Pepper の腰用の解除キーを準備します。
2. Pepper の後頭部にある二つ穴の部分に解除キーを差し込みます。
3. Pepper 後頭部のパネルをゆっくり下方向に引っ張りながら外します。
4. 開口部の左下に LAN ケーブルの差込口があるので、Pepper とパソコンを LAN ケ

ケーブルで接続します。



図 3.5-11 Pepper と有線接続

5. 胸部ボタンを1回押し、PepperがIPを発話する様になるのを待ちます(認識に時間がかかる場合はLANケーブルを刺した状態でPepperを再起動すると認識する場合があります)。
 6. パソコンのブラウザで「[http://\(PepperのIP\)](http://(PepperのIP))」にアクセスし、ロボットウェブページを開きます。
- ※ 有線接続するにあたり、インターネット共有などパソコン側の設定が必要な場合があります。

開発中は何度もChoregrapheからアプリを転送して実行します。開発用PCとPepperを有線で接続することで、安定した転送を行くことができるようになり作業効率が向上します。Pepperの移動やモーションなどで、ケーブルやパソコンが引っ張られることが考えられますので、LANケーブルの長さにはゆとりを持たせるなどの注意が必要です。

3.6. Pepper for Biz との連携

Pepper for Biz モデルでは、「お仕事かんたん生成」、「ロボアプリ配信管理」および「インタラクション分析」などのWebサービスと連携したサービスを利用することができます。

3.6.1.1. お仕事かんたん生成

業務に最適なテンプレートを選択するだけで簡単にPepperのお仕事がカスタマイズすることができます。お仕事を配信するにはPepper側にSBRアカウントを登録し、お仕事かんたん生成2.0の管理画面でお仕事を反映することで配信がなされます。

※ どのロボットが紐付いているか管理画面上で確認することはできません。

3.6.1.2. ロボアプリ配信管理

管理者がPepperに対して配信するロボアプリを管理できる機能を提供します。ロボアプリ配信管理とPepperとの関連付けの確認方法は以下の通りです。

1. ロボアプリ配信ページ (<https://adm.aldebaran.com/>) にアクセスします。
2. [ログイン]をクリックします。
3. 登録されたSBRアカウントと対応するパスワードで[サインイン]をクリックしログインします。
4. 正常にログインできることを確認します。

3.6.1.3. インタラクション分析

Pepper で取得したデータをクラウドに蓄積し、管理画面で「見える化」し、集客施策の効果測定など、データに基づいた戦略的マーケティングが実現できます。インタラクション分析と Pepper との関連付けの確認方法は以下の通りです。

1. インタラクション分析ページ (<https://softbankrobotics.com/portal/interaction-analytics/auth/login/>) にアクセスします。
2. [ログイン]をクリックします。
3. 登録された SBR アカウントと対応するパスワードで [サインイン] をクリックしログインします。
4. [Allow]をクリックします。
5. 正常にログインできることを確認します。

3.7. ロボアプリの起動方法

Pepper にインストールされているロボアプリを起動する方法について説明します。

3.7.1. ロボアプリランチャー

一般販売モデルは Pepper がソリタリー状態の時にディスプレイをタッチすることでロボアプリランチャーを起動できます。ロボアプリランチャーを起動することで、インストールされているロボアプリの一覧を見ることができます。また、一覧からアイコンをタップすることでロボアプリを起動することができます。



図 3.7-1 ロボアプリランチャー

3.7.2. お仕事かんたん生成

Pepper for Biz モデルは「お仕事かんたん生成」と「ロボアプリ配信管理」で設定した内容に合わせてロボアプリを起動することができます。

3.7.3. その他の起動方法

3.7.3.1. Pepper にアプリをインストールしていない場合

起動したいロボアプリのプロジェクトファイルを持っている場合は、Choregraphe でプロジェクトを開き、Pepper 本体と接続し、ツールバーの再生ボタンをクリックすることで起動できます。

3.7.3.2. Pepper にアプリをインストールしている場合

一般販売モデルと Pepper for Biz モデルの両方で使えるロボアプリの起動方法は以下の通りです。

- Choregraphe のロボアプリ一覧パネルから起動する。
- qicli コマンドを実行して起動する (直接アプリを指定して起動するコマンド)。

以下の方法は一般販売モデルのみ使用できるロボアプリの起動方法です。

- トリガーセンテンスで起動する (オートノマスライフがオンの状態)。
- トリガー条件で起動する (オートノマスライフがオンの状態)。
- Qicli コマンドでトリガー条件を満たすためのイベントを発行し起動する。

3.8. 肩の LED ランプ

肩の LED ランプは NAOqi の状態に合わせて点灯します。肩のランプを制御すると NAOqi からの通知を正しく受けることができなくなる可能性があります。

表 3.8-1 肩の LED ランプ

色 (光り方)	状態
白 (点灯)	正常
緑 (点滅)	通知情報あり
黄 (早く 2 回点滅)	警告
赤 (早く 2 回点滅)	エラー
赤 (遅く点滅)	使用不可

複数の通知がある場合は緊急性の高いものから順に表示されます。胸部ボタンを 1 回押すことで、Pepper が機体名、IP アドレスを発話した後、通知内容を発話します。エラーの詳細や対処方法は公開されているマニュアル等を参照してください。LED 系のボックスを使用すると、Pepper の目、耳、肩の LED の制御が可能で、色の指定や点滅のタイミングの制御を設定できます。

4.Choregraphe の使い方

Choregraphe はロボアプリを作成するための統合開発環境です。ドラッグ&ドロップを中心とした操作で、プログラミング経験の少ない方でも導入しやすいように設計されています。この章では、初期設定、画面構成、基本的な操作方法などについて解説します。

4.1.Choregraphe のインストール

4.1.1. 動作環境

Choregraphe のインストールに必要な動作環境は、下記のとおりです。

表 4.1-1 Choregraphe の動作環境

項目名	動作環境
ハードウェア	1.5GHz CPU / 512MB RAM / OpenGL 対応グラフィックカード
OS	Ubuntu 14.04 (Trusty Tahr) and later Microsoft Windows 7 および 8.1 (Windows10 は正式対応していません) Mac OS X 10.10.2 Yosemite 以上

4.1.2. インストール

Choregraphe は、ソフトバンクロボティクス株式会社のデベロッパー向け WEB サイトからダウンロードできます。ご使用の環境に対応したインストーラをダウンロードし、インストールを行ってください。

[デベロッパー向け Web サイト]

<https://www.softbank.jp/robot/developer/dev-support/documents/>

※ ライセンスキーはデベロッパー向け Web サイトに掲載されているライセンスキーをご利用ください

4.2.Choregraphe の初期設定

Choregraphe をインストールして最初に起動すると、メニューなどの表記が英語です。また、ウィンドウ内に表示されるロボットが Pepper ではありません。この節では、ロボアプリを開発しやすいように初期設定を行います。

4.2.1. 日本語化

メニューなどの表記を日本語にするには、以下の手順で設定します。

4.2.1.1. 【Mac】

1. [Choregraphe]メニューの[Preferences...]を選択する
2. [Language]の値を"空白 (System default) "から"日本語"に変更する
3. [OK]ボタンをクリックする

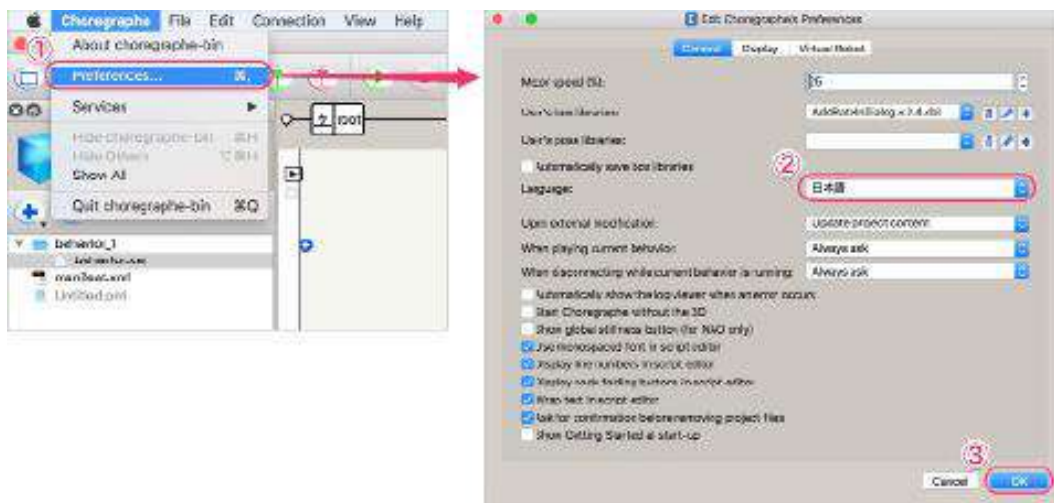


図 4.2-1 Choregraphe の日本語化（Mac の場合）

4.2.1.2. 【Windows】

1. [Edit]メニューの[Preferences]を選択する
2. [Language]の値を"空白（System default）"から"日本語"に変更する
3. [OK]ボタンをクリックする

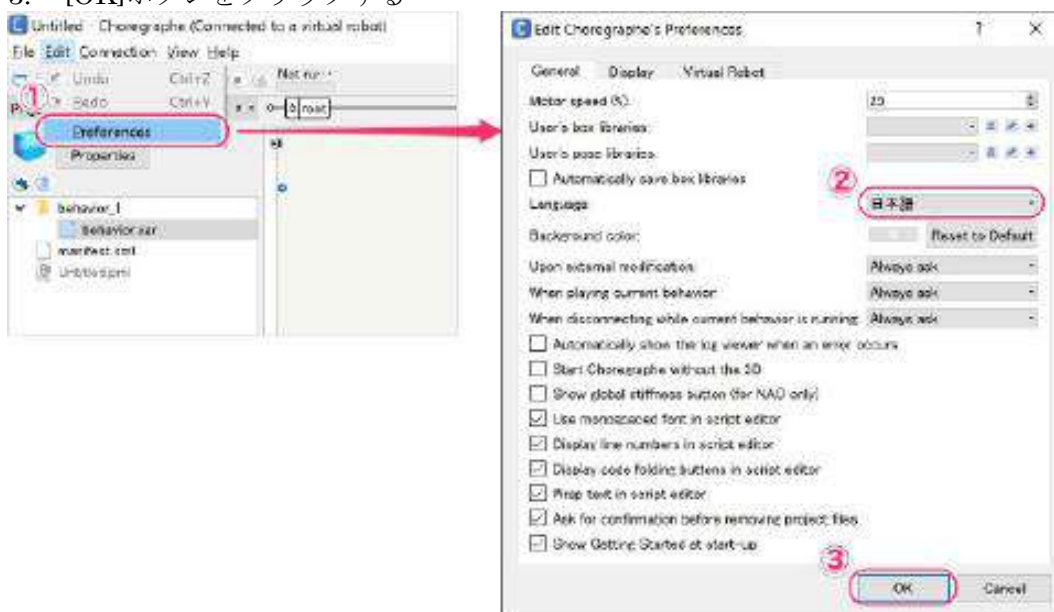


図 4.2-2 Choregraphe の日本語化（Windows の場合）

[OK]ボタンクリック後に、Choregraphe の再起動を促す以下のウィンドウが表示されます。
[OK]ボタンをクリックして、Choregraphe を終了し、もう一度起動してください。



図 4.2-3 Choregraphe の日本語化 (再起動を促すウィンドウ)

4.2.2. バーチャルロボットの変更

Choregraphe のウィンドウ右上にロボットが表示されます。これをバーチャルロボットと言います。初期設定のバーチャルロボットはソフトバンクロボティクスの NAO（<https://www.ald.softbankrobotics.com/ja/クールなロボット/nao>）になっています。ロボアプリの開発に向けて、バーチャルロボットを Pepper に変更することをお勧めします。

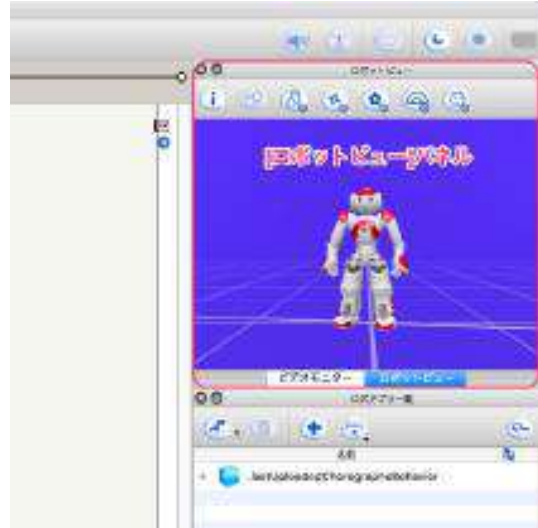


図 4.2-4 バーチャルロボット

変更手順は以下の通りです。（画像は日本語化を終えた状態です）

1. [Choregraphe]メニューの[Preferences...]を選択（Windows は[編集]メニューの[設定]）
2. [バーチャルロボット]タブを選択する
3. [ロボットモデル]を"PepperY20(V16)"に変更する
4. [OK]ボタンをクリックする



図 4.2-5 バーチャルロボットの変更

再起動を促すウィンドウが表示されますが、[Yes]ボタンをクリックしてしばらくすると、（Choregraphe を再起動しなくても）自動的にバーチャルロボットが変更されます。

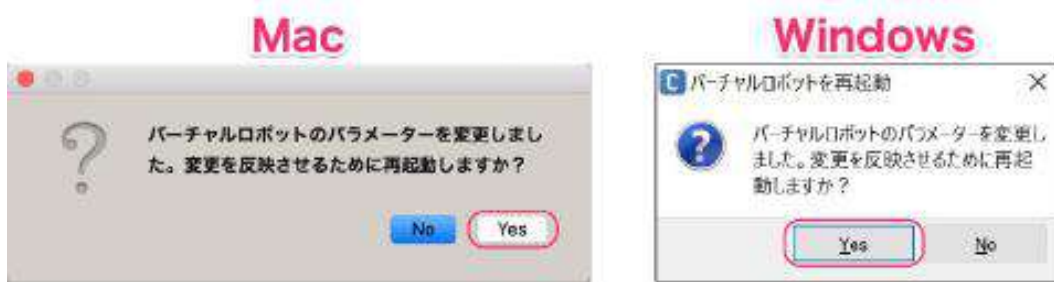


図 4.2-6 仮想ロボットの変更（再起動を促すウィンドウ）

GPU との相性問題

開発 PC によっては仮想ロボットが正常に動作しないことがあります。特に Windows 系 PC の場合、搭載 GPU（Graphics Processing Unit）との相性問題が発生することが多いです。仮想ロボットが動作せず、他に代替開発 PC がない場合、Choregraphe を 3D（仮想ロボット）なしで利用することができます。



図 4.2-7 3D なしで起動（Windows 版の設定画面）

Windows PC で初回起動時にクラッシュしてしまう場合、コマンドプロンプトから以下のように「--no-ogre」オプションを付けて起動してください。

```
choregraphe-bin.exe --no-ogre
```

図 4.2-8 クラッシュ回避のコマンド

4.3. Choregraphe の画面構成

Choregraphe はプログラミングを支援するための様々な機能が揃っています。この節では、Choregraphe の画面構成を見ながら、主な機能を解説します。

4.3.1. パネル

Choregraphe のウィンドウは、複数の矩形で分割されています。1つの矩形をパネルと言います。

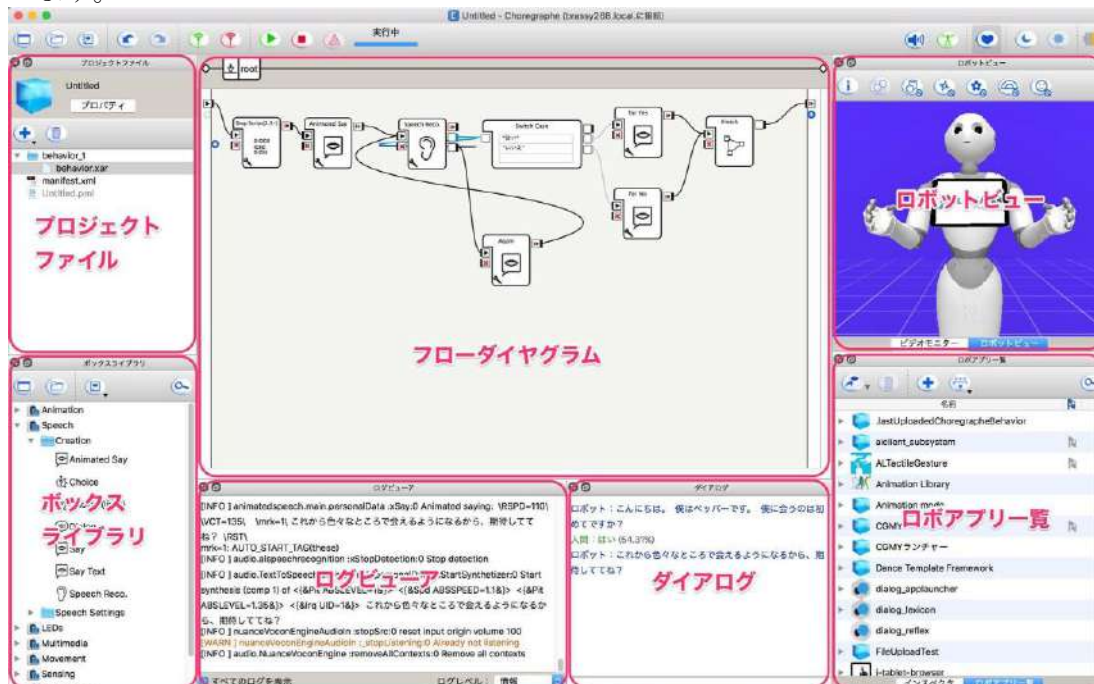


図 4.3-1 メニューバーの表示メニュー

Choregraphe のメニューバーから、[表示] を選び、一覧から画面に表示させたい内容にチェックを入れるとパネルが表示されます。主なパネルには以下の様なものがあります。

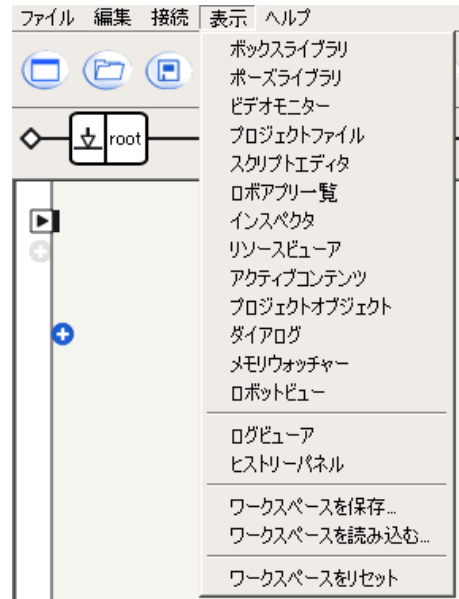


図 4.3-2 メニューバーの表示メニュー

表 4.3-1 Choregraphe の主なパネル

パネル名	説明
プロジェクトファイル	アプリに関連するファイルを管理する。
ボックスライブラリ	プログラムの部品となるボックスがカテゴリに分類されて格納されている。
フローダイアグラム	プログラムの内容を表示する。ボックスライブラリからボックスをドラッグ&ドロップし、ボックス同士を線でつないでロボアプリの開発を行う。
ロボットビュー	バーチャルロボットまたは実機の Pepper の動作を表示する。
ロボアプリ一覧	実機の Pepper のストレージにインストールされているアプリを一覧で表示する。インストールやアンインストールも行える。
ログビューア	Pepper のシステムログを表示する。ログの種類でフィルタリングできる。
ダイアログ	Pepper とユーザの会話内容を表示する。ユーザの言葉を認識させた場合は、音声の認識率も表示される。

4.3.2. ツールバー

Choregraphe のウィンドウ上部に、開発中よく使用する機能をボタンで呼び出せるようにツールバーが配置されています。



図 4.3-3 Choregraphe のツールバー

ツールバーのボタンの機能は以下の通りです。

表 4.3-2 ツールバーのボタン

ボタン名	説明
プロジェクトを新規作成	新しいプロジェクトを作成する。
プロジェクトを開く	既存のプロジェクトを開く。
プロジェクトを保存	開いているプロジェクトの変更を保存する。
元に戻す	編集内容を 1 つ前に戻す。
やり直し	同じ編集内容をもう一度行う。
接続	Pepper と接続する。
切断	接続している Pepper とのリンクを切断する。
アップロードして再生	フローダイアグラムに表示されているプログラムを Pepper に送って実行する。
停止	実行中のアプリを停止する。
デバッグ/エラー出力	現在使用不可
ボリューム	Pepper のスピーカーの音量を調整する。
アニメーションモード	Pepper に任意のポーズを取らせるときに使用する。（詳細は第 4 章）
オートノマスライフ	Pepper を人らしく振る舞わせたり、アプリ起動条件の監視、関節モーターの温度監視などを行うミドルウェアのオン/オフを切り替える。（詳細は第 7 章）
Rest	Pepper をセーフレスト状態にして休ませる。
Wake Up	休んでいる Pepper を起こす。
バッテリーレベル	カーソルを重ねるとバッテリーの充電率を表示する。

4.4. プロジェクトファイルパネル

プロジェクトおよび、ビヘイビアのプロパティ、およびプロジェクト内のファイルの確認ができます。初期状態では、以下のファイルが作成されます。



図 4.4-1 プロジェクトファイルパネル

表 4.4-1 初期状態のファイル一覧

ファイルパス	概要
/Untitled.pml	Choregraphe がプロジェクトのファイル構成などを管理するためのファイル。
/manifest.xml	ロボアプリの設定ファイル。
/behavior_1/behavior.xar	Pepper の動作(ビヘイビア)を記述するためのファイル。
/translations/ translation_en_US.ts	言語ごとに Pepper のフレーズを管理するためのファイル。

※ Untitled と behavior_1 は保存の際に名称を変更できますが、他のファイル名は変更できません。

4.4.1. ボックスライブラリパネル

プログラムの部品となるボックスがカテゴリに分類されて格納されています。ボックスライブラリには、新規のボックス・既存のボックスが追加できます。ツリー表示を展開することで、ボックスの検索や選択を行うことができます。



図 4.4-2 ボックスライブラリパネル

4.4.2. フローダイアグラムパネル

Choregraphe の画面中央にある[フローダイアグラム]パネルにボックスを配置し、結線することで Pepper の動作をプログラムできます。[ボックスライブラリ]パネルから配置したいボックスを選択し、ドラッグ&ドロップでフローダイアグラムに配置します。さらに、ボックスの入力・出力コネクタをドラッグし、他の入力・出力コネクタでドロップすることで結線することができます。

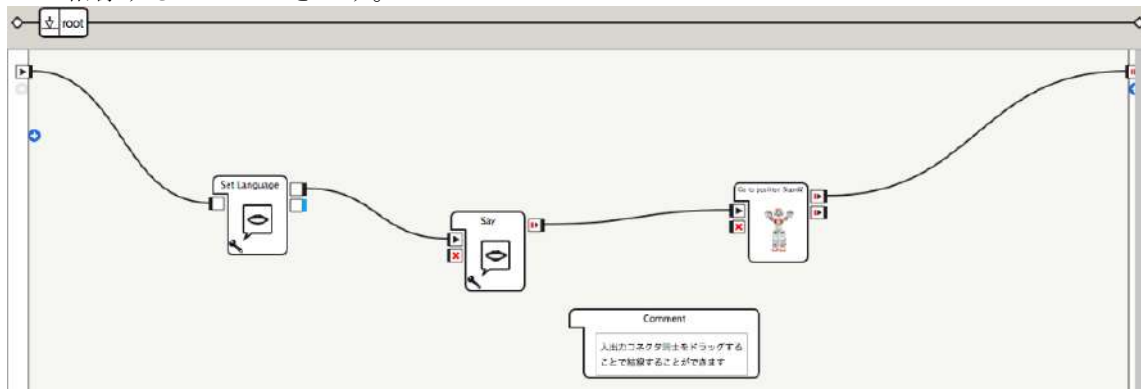


図 4.4-3 フローダイアグラムパネル

ダイアログボックスは階層構造になっているため、ダブルクリックで下の階層を開くことができます。下層のボックスを開いた後は、[フローダイアグラム]パネル左上の「root」アイコンをクリックすることで戻ることが可能です。

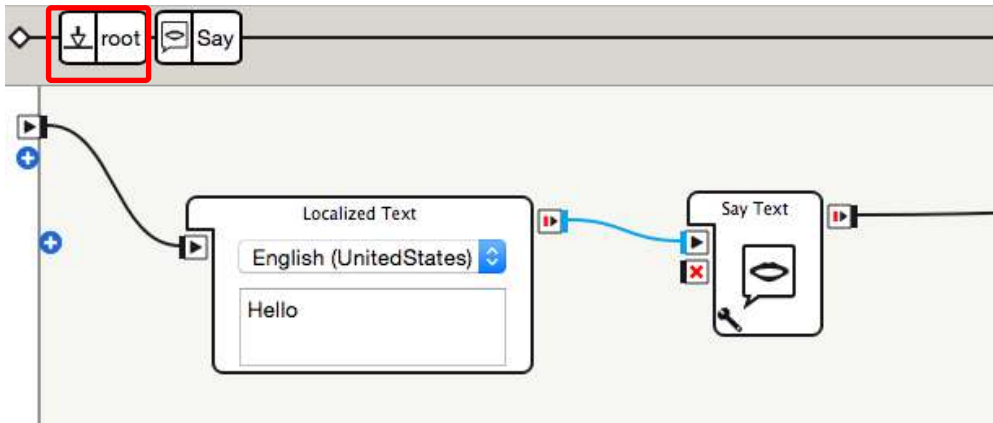


図 4.4-4 root の位置

4.4.3. ポーズライブラリパネル

[ポーズライブラリ]パネルに登録されているポーズは、ボックスと同様フローダイアグラムパネルにドラッグ&ドロップすることで配置することができます。Pepperに登録されているポーズをとらせたい場合に使用します。また、新規のポーズをとらせて保存でき、そのポーズライブラリをファイルとしてエクスポート・インポートできます。

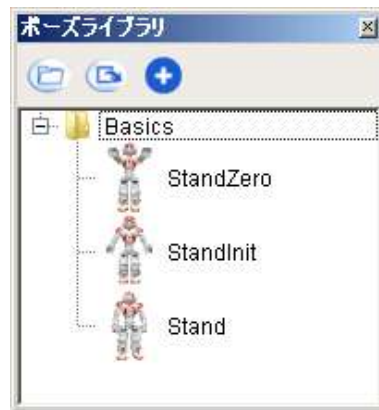


図 4.4-5 ポーズライブラリパネル

4.4.4. ビデオモニターパネル

Pepper 本体が認識している視界を確認できます。Pepper 本体に接続している時のみ映像が表示されます。バーチャルロボット接続時には何も表示されません。



図 4.4-6 ビデオモニターパネル

4.4.5. ロボアプリ一覧パネル

[ロボアプリ一覧]パネルには、Pepper 本体にインストールされているロボアプリの一覧が表示されます。ビヘイビア名の右側に表示される実行ボタンをクリックするとそのロボアプリが実行されます。パネル右側の検索ボタンからはインストール済みロボアプリを検索できます。ロボアプリのインストール・アンインストールもこのパネルから行います。

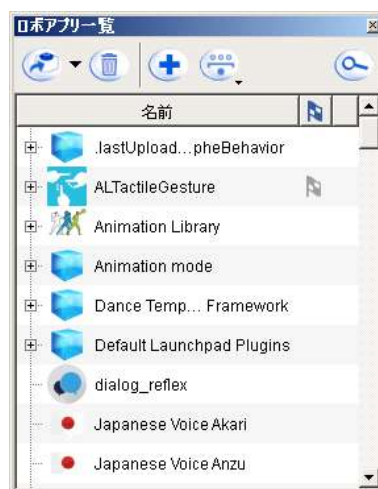


図 4.4-7 ロボアプリ一覧パネル

4.4.6. メモリウォッチャーパネル

[メモリウォッチャー]パネルを使用することで Pepper の制御を管理する NAOqi フレームワークのメモリの内容(ALMemory の値)を確認することができます。表示されていない場合は表示メニューから[メモリウォッチャー]にチェックを入れると Choregraphe の中央下部に表示されます。

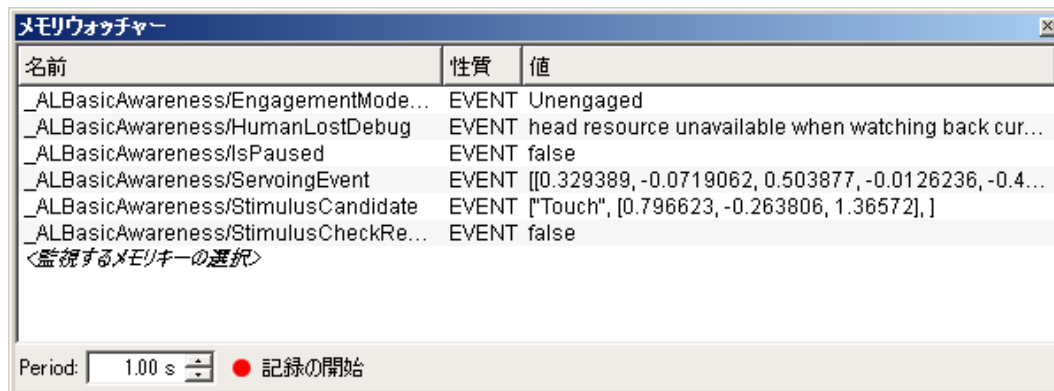


図 4.4-8 メモリウォッチャーパネル

[メモリウォッチャー]パネルではメモリの取得間隔がデフォルトで 1 秒に設定されています。テキストフィールドへ入力するか右側のトグルをクリックすると間隔を変更できます。また、画面内の最後の行の「監視するメモリアドレスの選択」をダブルクリックするか、パネル内で右クリックメニューにて監視対象を追加できます。[メモリウォッチャー]パネルで監視するメモリアドレスを増やしすぎると Choregraphe が重くなってしまふことに注意してください。

4.4.7. ダイアログパネル

[ダイアログ]パネルは表示メニューの[ダイアログ]をクリックすると表示できます。Pepper の発話内容と Pepper が音声認識した内容、音声認識時の認識率が表示されます。

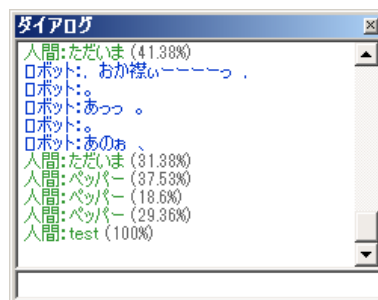


図 4.4-9 ダイアログパネル

また、バーチャルロボット接続中の Pepper の発話内容は[ダイアログ]パネルにて確認できます。音声は Pepper 本体に接続している場合のみ再生され、バーチャルロボット接続中は再生されません。[ダイアログ]パネルの下部にあるテキストフィールドに文字を入力することで、バーチャルロボットに接続している状態でも対話することができます。テキストが日本語であっても問題ありません。

4.4.8. ログビューアパネル

内部のログを確認できます。ログレベルを操作することで、Fatal、Error、Warning、Info、Verbose、Debug の 6 種類から絞り込みできます。ログビューア下部のプルダウンから選択してください。すべてのログを表示したい場合は、「すべてのログを表示」をチェックしてください。

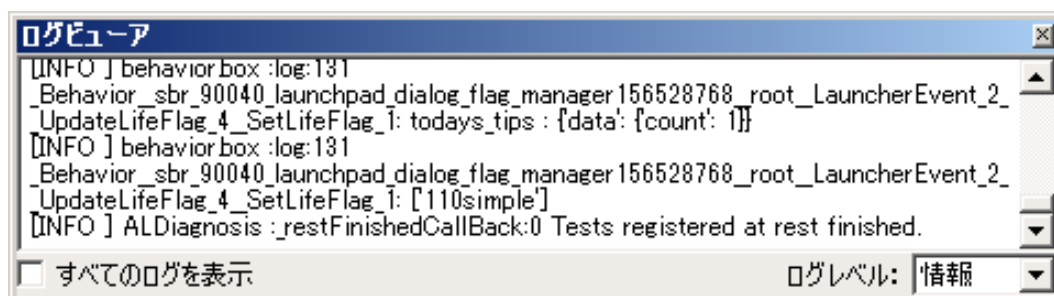


図 4.4-10 ログビューアパネル

4.5. Pepper との接続

ロボアプリを作成して実行/テストするには、開発用 PC と Pepper を接続する必要があります。簡単な会話や動きはバーチャルロボットでも確認できますが、ディスプレイやセンサーを使用するアプリは実機の Pepper に接続する必要があります。この節では、Pepper と開発用 PC を接続するための設定方法を解説します。

4.5.1. Choregraphe の設定

Pepper の設定が終了したら、Choregraphe から Pepper へ接続する設定を行います。新たに Pepper との接続を行う場合、ツールバーの[切断]ボタンをクリックして、ロボットビューからバーチャルロボットがいなくなった状態から始めることをお勧めします。

4.5.1.1. バーチャルロボットとの接続

バーチャルロボットとはロボアプリ開発のエミュレータ機能の一部で、バーチャルロボット上で開発したロボアプリを動作させることができます。バーチャルロボットと接続するには、[接続]メニュー→[バーチャルロボットに接続]を選択します。



図 4.5-1 バーチャルロボットに接続

バーチャルロボットには、ディスプレイの画面表示や写真撮影など対応していない機能が多数あることに注意が必要です。

4.5.1.2. 実機との接続

実機の Pepper と接続する手順は以下の通りです。

1. ツールバーの[接続]ボタンをクリックする
2. Pepper の一覧から接続したい機体を選択する
3. [選択]ボタンをクリックする



図 4.5-2 実機の Pepper に接続

Pepper の一覧から選択せずに、右側の[固定 IP またはホスト名を使ってください]に IP アドレスを入力しても接続できます。1 台の Pepper に同時に接続できる開発用 PC は 1 台だけです。

4.6. 基本的な開発方法

ロボアプリ開発の基本的な流れについて説明します。

4.6.1. Choregraphe のプログラミングスタイル

Choregraphe のプログラミングスタイルは、マウスによるドラッグ&ドロップが主な操作

になります。基本的な手順は以下の通りです。

1. [ボックスライブラリ]パネルから必要なボックスを[フローダイアグラム]パネルにドラッグ&ドロップする
2. [フローダイアグラム]パネル左端の[onStart]アイコンから最初のボックスに向かってドラッグして結線する
3. 最後のボックスから[フローダイアグラム]パネル右端の[onStopped]アイコンに向かってドラッグして結線する
4. ツールバーの[アップロードして再生]ボタンをクリックしてアプリを実行する

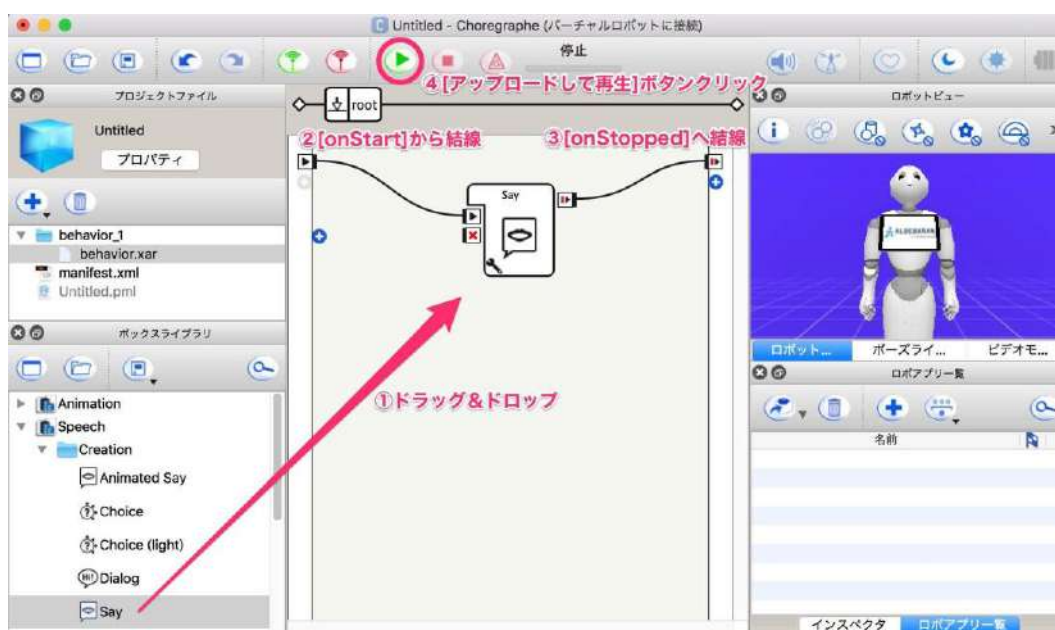


図 4.6-1 Choregraphe のプログラミングスタイル

注意点

インストールボタンでインストールしたロボアプリは、プロパティで設定した ApplicationID が使用されます。ツールバーの再生ボタンでインストールしたロボアプリは、プロパティの設定によらず「.lastUploadedChoregrapheBehavior」が ApplicationID となります。ApplicationID を指定して起動する場合などは、「.lastUploadedChoregrapheBehavior」を指定する必要があります。また、ツールバーの再生ボタンで、複数のロボアプリをインストールしようとしても、ApplicationID が重複するため、すべて上書きインストールとなります。

4.6.2. プロジェクトの作成と保存

多くの統合開発環境は、1つのアプリに関わるファイル群を1つのプロジェクトという単位で管理します。Choregraphe も同様で、作成したアプリを保存するときにプロジェクトを作成します。プロジェクトを作成して保存する手順は以下の通りです。

1. [ファイル]メニューの[プロジェクトを保存]を選択する
2. [プロジェクトに名前をつけて保存]ウィンドウの[プロジェクト名]に任意の名前を設定する
3. [保存]ボタンをクリックする

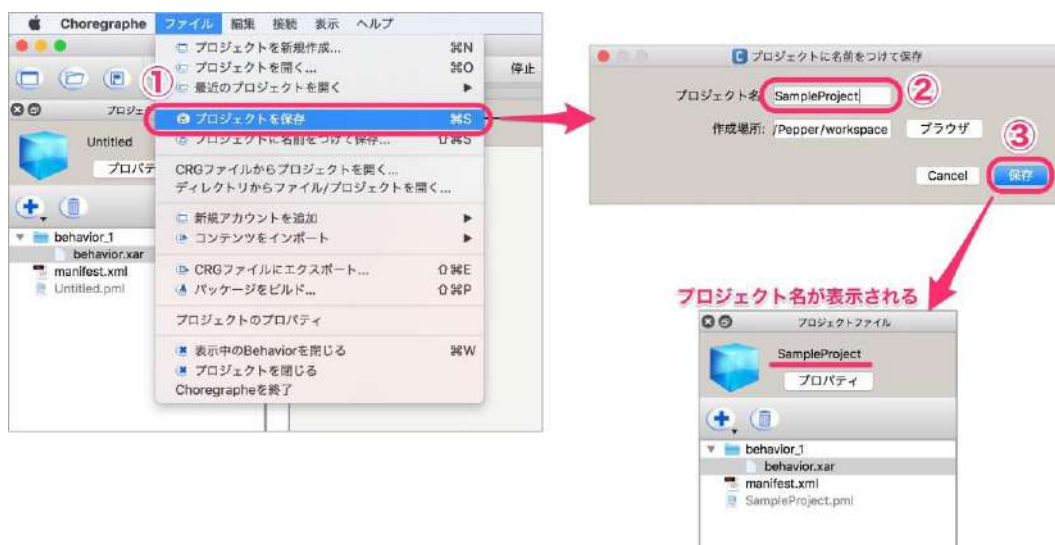


図 4.6-2 プロジェクトの作成と保存

プロジェクト名は半角英数字をお勧めします。日本語でも保存できますが、稀に不具合の原因になります。[プロジェクトに名前をつけて保存]ウィンドウの[作成場所]はプロジェクトを保存したいフォルダを指定してください。

4.6.3. インストール/アンインストール

Pepper にロボアプリをインストールするには、Choregraphe の[ロボアプリ一覧]パネルを使用します。

インストール手順は以下の通りです。

1. [ロボアプリ一覧]パネルを表示する。
2. [ロボアプリ一覧]パネル左上の[現在のプロジェクトをロボットにインストール]ボタンをクリックする。



図 4.6-3 Pepper へのインストール

Pepper にインストールされているロボアプリをアンインストールする手順は以下の通りです。

1. [ロボアプリ一覧]パネルを表示する。
2. [ロボアプリ一覧]パネルからアンインストールしたいロボアプリを選択する。
3. [ロボアプリ一覧]パネル上部の[選択したロボアプリをロボットから削除]ボタンをクリックする。



図 4.6-4 アプリのアンインストール

4.6.4. パッケージファイル

プロジェクトを圧縮して 1 つのファイルにまとめたものをパッケージファイルと言います。パッケージファイルを用いてインストールすることもできます。パッケージファイルの作成とインストール手順は以下の通りです。

1. [ファイル]メニューの[パッケージをビルド]を選択する
2. [パッケージのエクスポート先を指定してください]ウィンドウの[Save As]に任意のプロジェクト名を入力して、[Save]ボタンをクリックする
3. [ロボットアプリ一覧]パネルの[現在のプロジェクトをロボットにインストール]ボタン右側にある下向き三角形をクリックする
4. [パッケージファイルを選択してインストール]を選択する
5. インストールしたいパッケージファイルを選択して[オープン]ボタンをクリックする

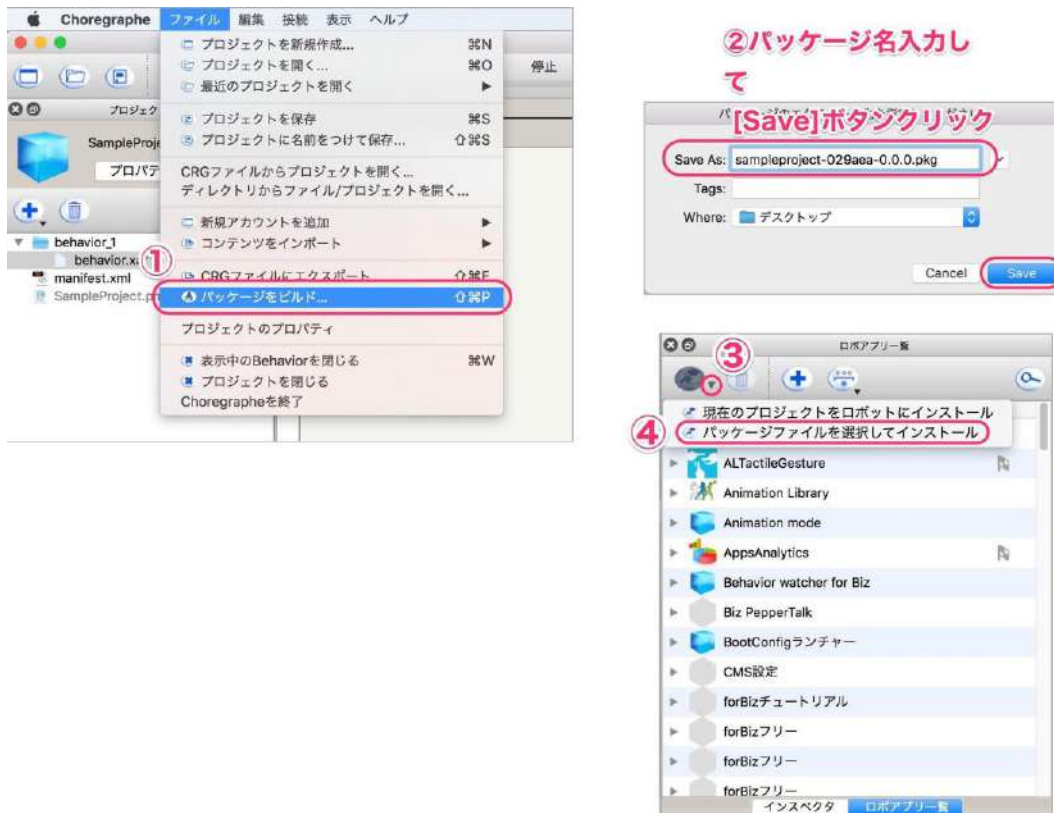


図 4.6-5 パッケージファイルの作成とインストール

4.7. はじめてのロボアプリ

「こんにちは」と発話するロボアプリを開発し、起動してみましょう。

1. ボックスライブラリの中から[Set Language]ボックス、[Say]ボックスを探して[フローダイアグラム]パネルに配置します。[フィルターを表示]ボタンを用いれば、簡単に検索できます。

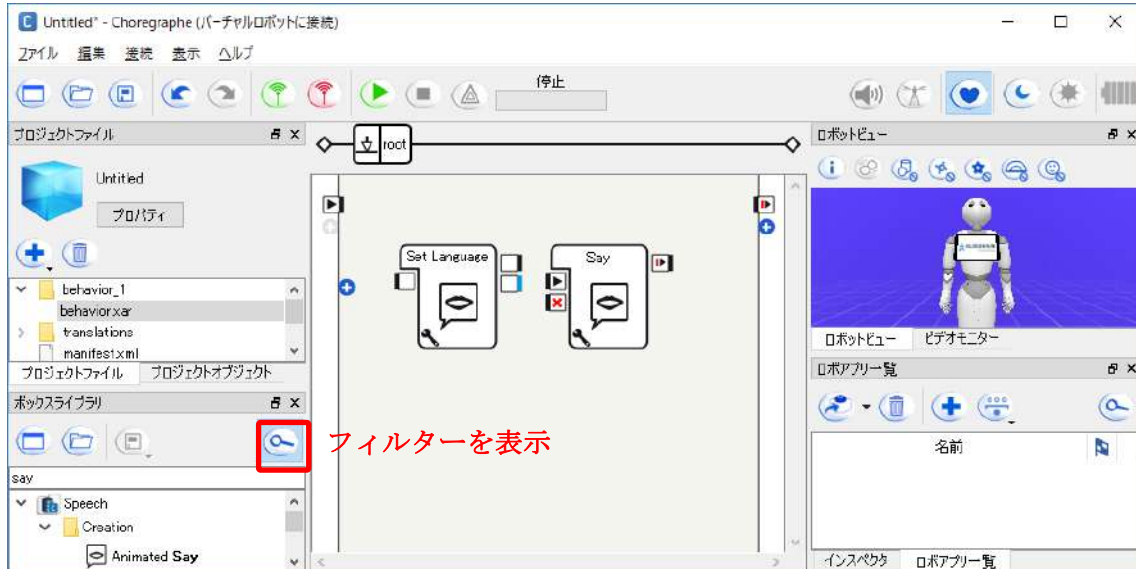


図 4.7-1 Set Language ボックスと Say ボックスの配置

2. [フローダイアグラム]パネルの左辺にある onStart をドラッグして、[Set Language]の onset にドロップし接続します。

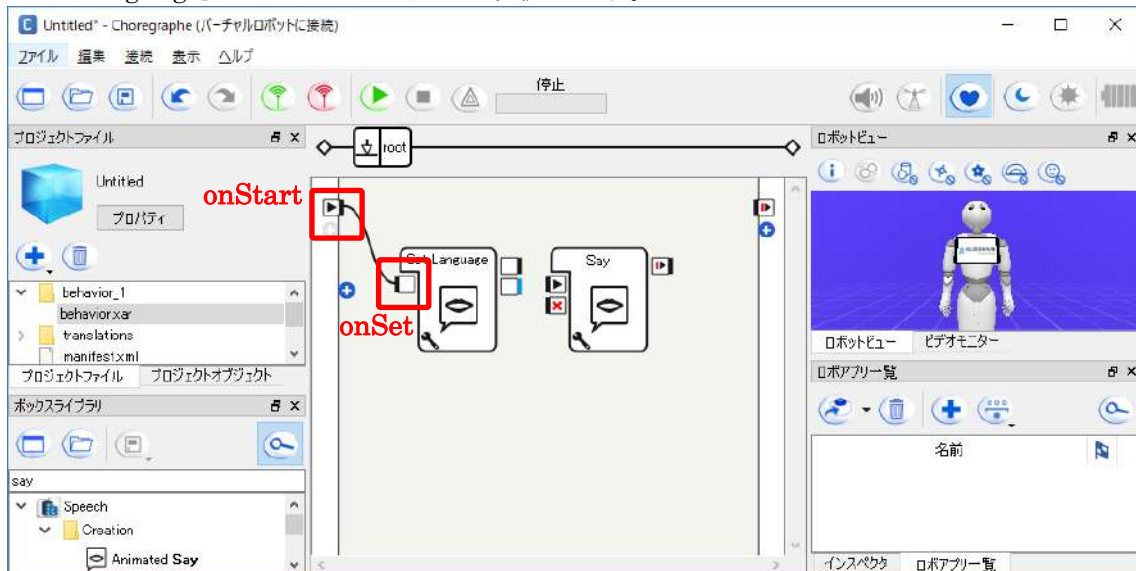


図 4.7-2 root の onStart と Set Language ボックスの onSet の接続

3. 同様に[Set Language]ボックスの onReady と[Say]ボックスの onStart を接続します。
4. [Say]ボックスの onStopped と[フローダイアグラム]パネルの onStopped を接続します。

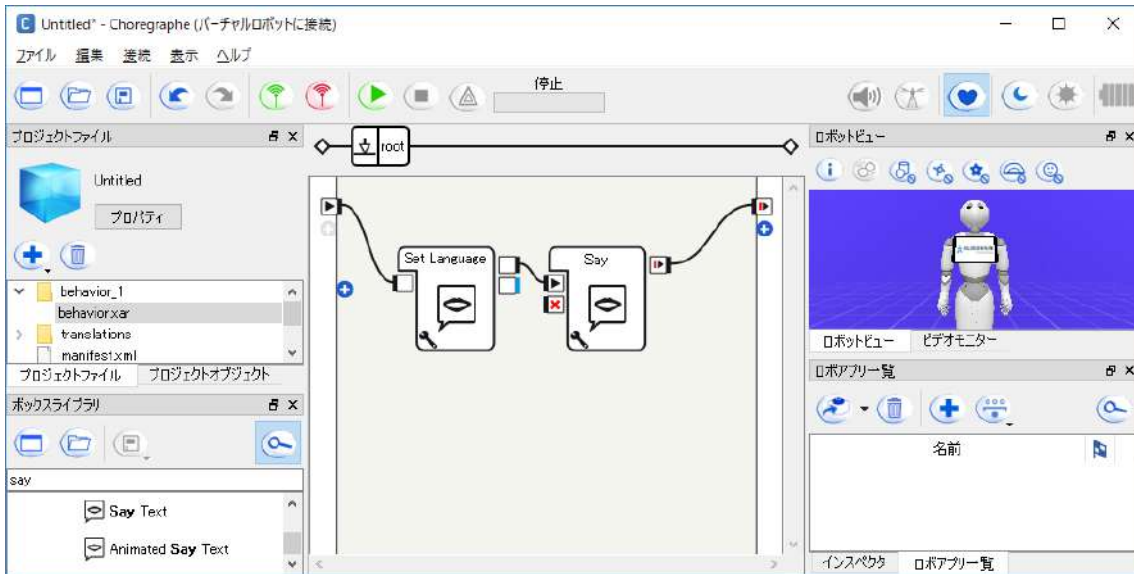


図 4.7-3 各ボックスの入力と出力を接続

5. [Set Language]ボックスのパラメータ設定ボタンをクリックし、設定ウィンドウを表示します。

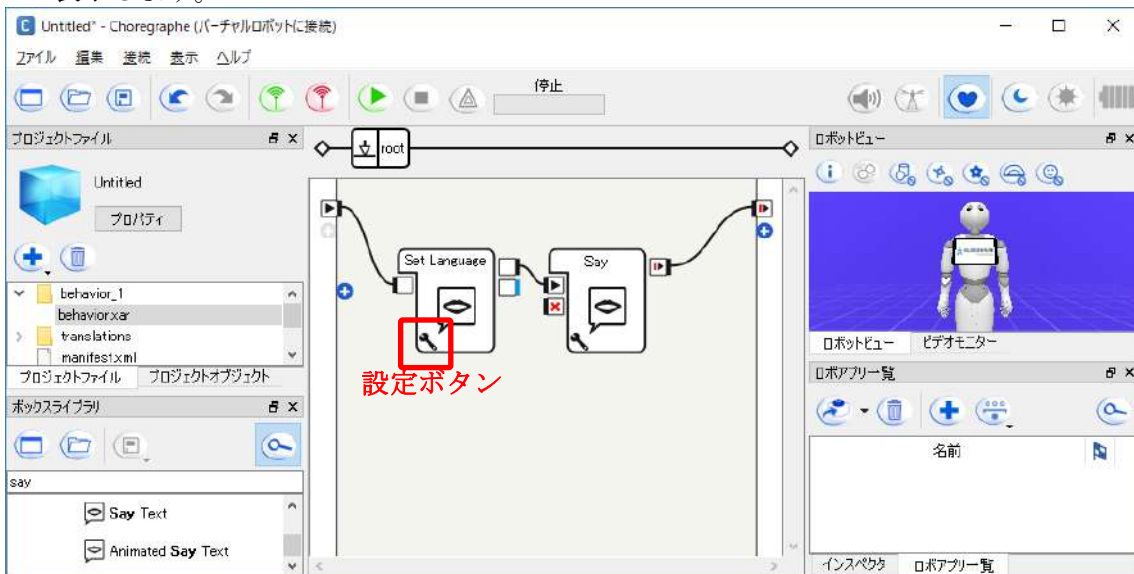


図 4.7-4 パラメータボタン

6. 設定ウィンドウの「Language」で「Japanese」を選択し、[OK]ボタンをクリックします。

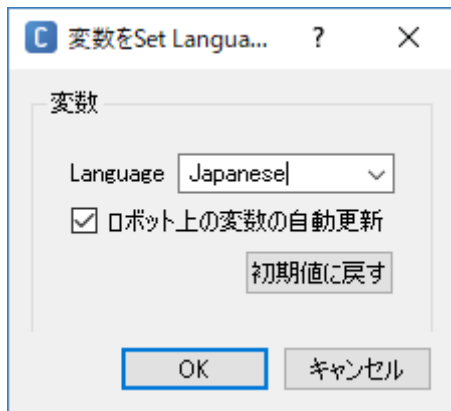


図 4.7-5 Set Language の設定ウィンドウ

7. 同様に[Say]ボックスの設定ウィンドウを開き、「Text」を「こんにちは」に書き換えて[OK]ボタンをクリックします
8. ツールバーの[再生]ボタンをクリックして、Pepper が「こんにちは」と発話するのを確認してください

5. ボックス

Choregraphe でロボアプリを作成する上で、ボックスという概念は非常に重要です。「ボックスはプログラムの部品」と考えることができます。ボックスを複数つなげることで複雑なプログラムを比較的簡単に作成することができます。ボックスは入力、出力、変数（パラメータ）などで構成されています。この章では、ボックスの構造について解説します。

5.1. ボックスの構成要素

ボックスはプログラムの部品です。ボックスをつなぎ合わせることで、1つのロボアプリを構築することができます。ボックスを構成する要素は以下の通りです。

表 5.1-1 ボックスの構成要素

要素名	説明
ボックス名	ボックスの名前になります。プログラム上で特別な意味を持ちませんが、そのボックスが何をしているかがわかるように命名してください。ボックス名の開始文字は、半角英文字を利用してください。記号 "/" を用いると Choregraphe でボックスを開けなくなることがあります。ご注意ください。
イメージアイコン	ボックスアイコンはプログラム上特別な意味を持ちません。ボックスの働きに相応しいイメージを指定してください。
入力	前の処理からの流れを受け付ける入り口。何らかのデータ、シグナル等がボックスに対して入力されます。
出力	後の処理へ流れを送り出す出口。ボックスの中である処理がされ、結果をシグナルとして他のボックスに伝えます。
パラメータ設定ボタン	ボックス内のプログラムで使用する変数の値を設定するウィンドウを表示します。この変数は、ボックスと関連づけられ、専用の関数によって設定・取得(<code>setParameter / getParameter</code>) することができます。Python における変数とは違うので注意してください。

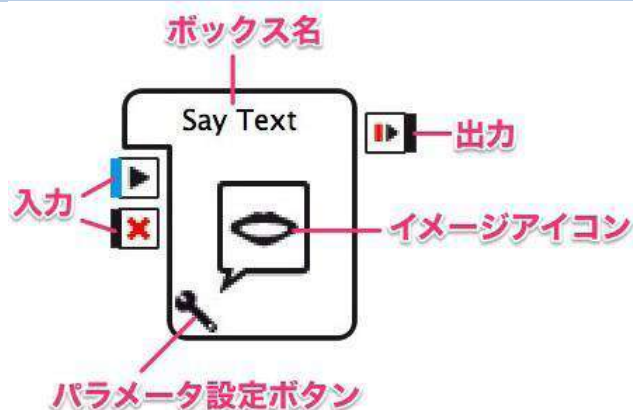


図 5.1-1 ボックスの構成要素 (例: [Say Text]ボックス)

5.2. 入力・出力の性質

ボックスを[フローダイアグラム]パネルに追加すると、入力・出力に関するアイコンが表示されているのがわかります。各部の説明は以下の通りです。

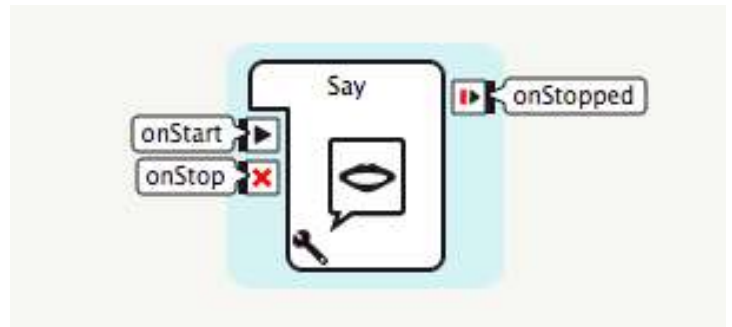







図 5.2-1 入力・出力の性質

表 5.2-1 入力・出力の性質の一覧

種別	項目名	説明
入 力	onStart 	このボックスの処理を開始させる場合に、結線します。
	onStop 	このボックスの処理を終了させる場合に、結線します。
	onEvent 	開始または終了以外の処理をさせる場合に使用します。
出 力	onStopped 	このボックス内で停止処理が終了した時に、ここからシグナルが送られます。
	即時(punctual) 	終了以外で出力が必要な場合に使用します。

5.3. 入力・出力のデータ型

ボックスの入力・出力はデータの型を持っています。入力・出力するデータの型にあったデータ型を選択してください。対応しないデータ型で入力・出力を行った場合、データの受け渡しは保証されません。

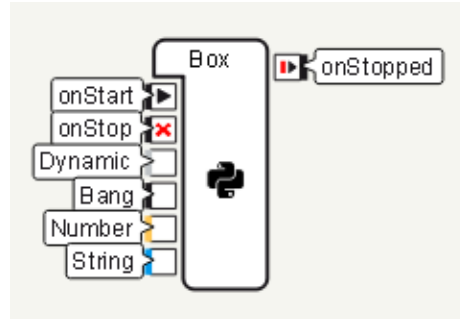


図 5.3-1 入力・出力のデータ型

表 5.3-1 入力・出力のデータ型の一覧

項目	色	説明
Dynamic	灰 □ □	入力・出力されるデータの型は数値、文字列、数値の配列、文字列の配列になります。無しでも構いません。
Bang	黒 ■ ■	入力・出力されるデータはありません。Bang が指定された入力・出力にデータが送られてきてもエラーにはなりません、データは受け付けられません。
Number	黄 □ □	入力・出力されるデータは数値です。
String	青 □ □	入力・出力されるデータは文字列です。

5.4. 変数(パラメータ)

ボックスの種類によっては、左下にスパナのアイコンがあります。これを[パラメータ設定]ボタンと言います。[パラメータ設定]ボタンをクリックすると、ボックスの種類に応じた変数 (パラメータ) 値を設定することができます。[Say Text]ボックスには、「Voice shaping(%)」と「Speed(%)」という 2 つの変数があります。



図 5.4-1 ボックスの変数の例 ([Say Text]ボックス)

5.5. ボックスの編集

ボックスの編集について説明します。

5.5.1. ボックスの編集方法

ボックスの入力・出力・変数の情報を変更する場合は、ボックスを右クリックし、メニューから[ボックスを編集]を選択し、ボックスの編集ウィンドウを表示することで行います。



図 5.5-1 ボックスのメニュー

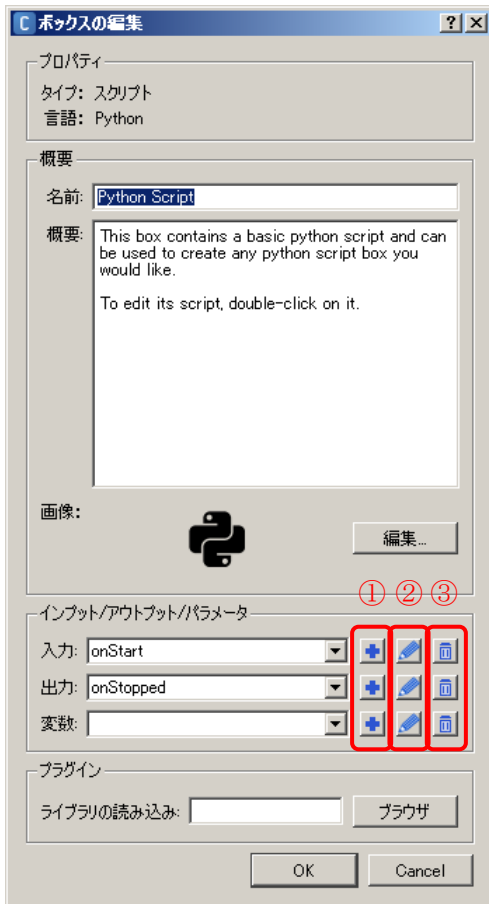


図 5.5-2 ボックスの編集ウィンドウ

- ①入力、出力、変数の追加用のウィンドウを表示します。
- ②入力、出力、変数の編集用のウィンドウを表示します。
- ③入力、出力、変数を削除します。

また、入力部分のアイコンを右クリックすると入力に関するメニューが表示され、出力部分のアイコンを右クリックすると出力に関するメニューが表示されます。それぞれ、前述の①②③に対応しています



図 5.5-3 入力・出力のメニュー

5.5.2. 追加用のウィンドウ

入力・出力の追加用のウィンドウは名前、ツールチップ、型、性質を設定できます。

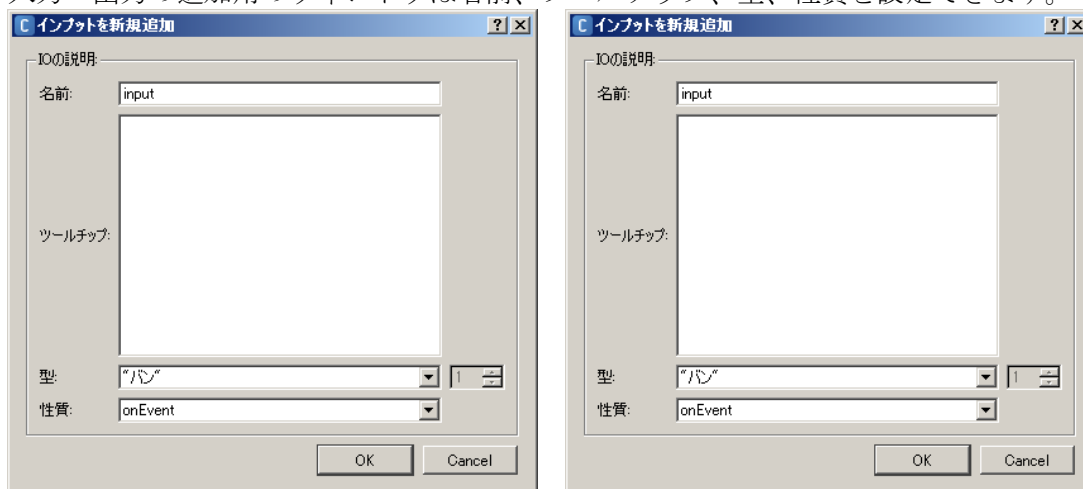


図 5.5-4 入力・出力の追加用のウィンドウ

変数の追加用のウィンドウは追加する変数のデータの型によって設定内容が異なります。

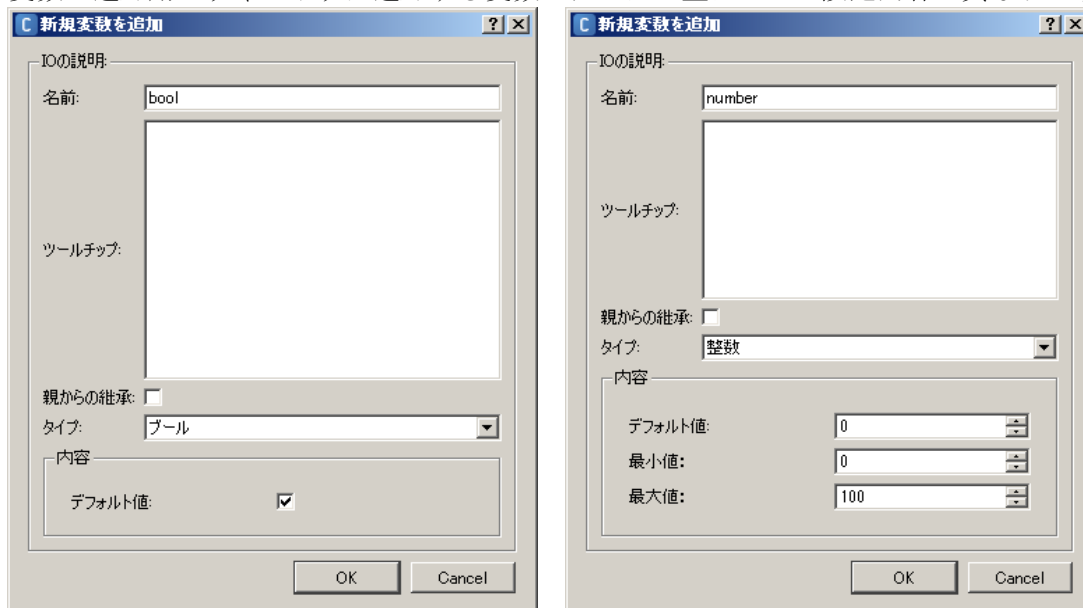




図 5.5-5 変数の追加用のウィンドウ

変数を追加した場合、ボックスの左下にパラメータ設定ボタンが追加され、クリックすることでパラメータ設定ウィンドウが表示されます。

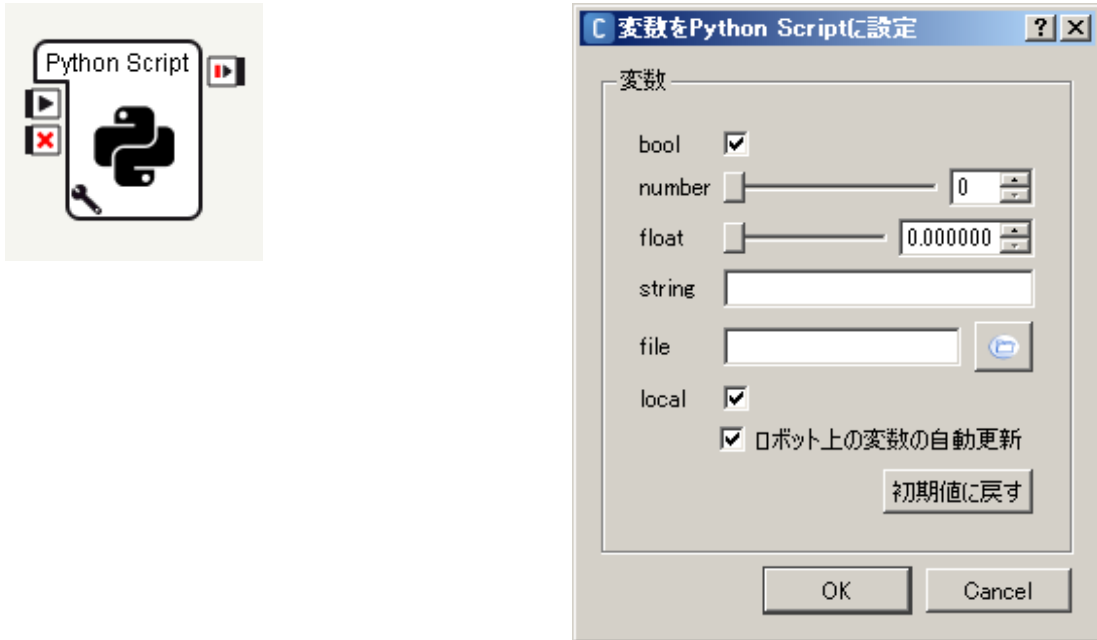


図 5.5-6 パラメータ設定ボタンとパラメータ設定ウィンドウ

5.5.3. 編集用のウィンドウ

入力、出力、変数の各ウィンドウの内容は追加をするときと同じです。既に複数のパラメータが設定されている場合は、対象とするパラメータを左のプルダウンから選択した後に編集します。

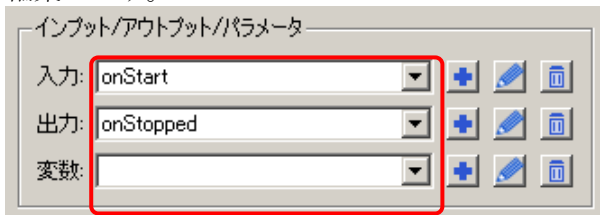


図 5.5-7 編集する項目を選択するプルダウン

5.6. ボックスの種類

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。

表 5.6-1 ボックスの種類一覧

ボックスの種類	説明
Python ボックス	Python でスクリプトを記述し、Pepper を制御することができるボックスです。
ダイアグラムボックス	ボックスやボックスとボックスの結線などをグループ化して一つのボックスとすることができるボックスです。
ダイアログボックス	Pepper とユーザとの会話を、QiChat というスクリプト言語を

(Dialog ボックス)	使って定義するためのボックスです。QiChat を使うことで、効率的に会話の内容を定義することができます。QiChat によって記述されたダイアログボックス用のファイルをトピックファイル (.top) と呼びます。
タイムラインボックス	Pepper のポーズや動きなどを定義するためのボックスです。

5.6.1. Python ボックス

Python ボックスの作成方法、および基本的な事項について説明します。

5.6.1.1. Python ボックスの作成

[フローダイアグラム]パネルの何も無いところで右クリックし、[ボックスの新規作成]、[Python]を選択することで Python ボックスを作成することができます。

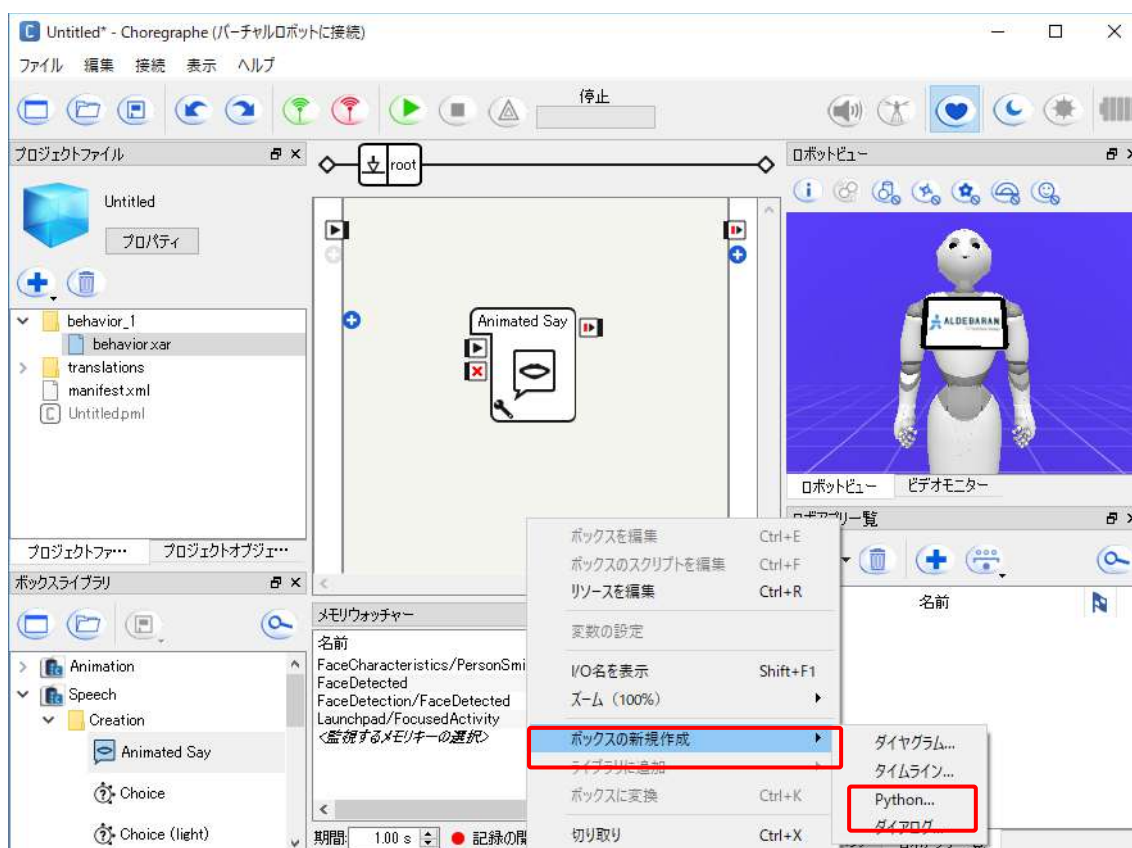


図 5.6-1 ボックスライブラリの作成

5.6.1.2. Python ボックスのスキプトの編集

Python ボックスをダブルクリックすると[スクリプトエディタ]パネルが開き、スクリプトの閲覧、編集が可能です。

Python ボックスは、入力に対応するハンドラメソッドを作成する必要があります。ハンドラメソッドのメソッド名はボックスの入力名の前に接頭辞"onInput_"をつけたものになります。入力[onStart]に、シグナルが送られた場合は、Python スクリプトの onInput_onStart メソッドがコールされます。また、Python スクリプト内では出力名と

同名のメソッドをコールすることが可能です。コールした場合は、ボックスの出力からシグナルが送出されることになります。

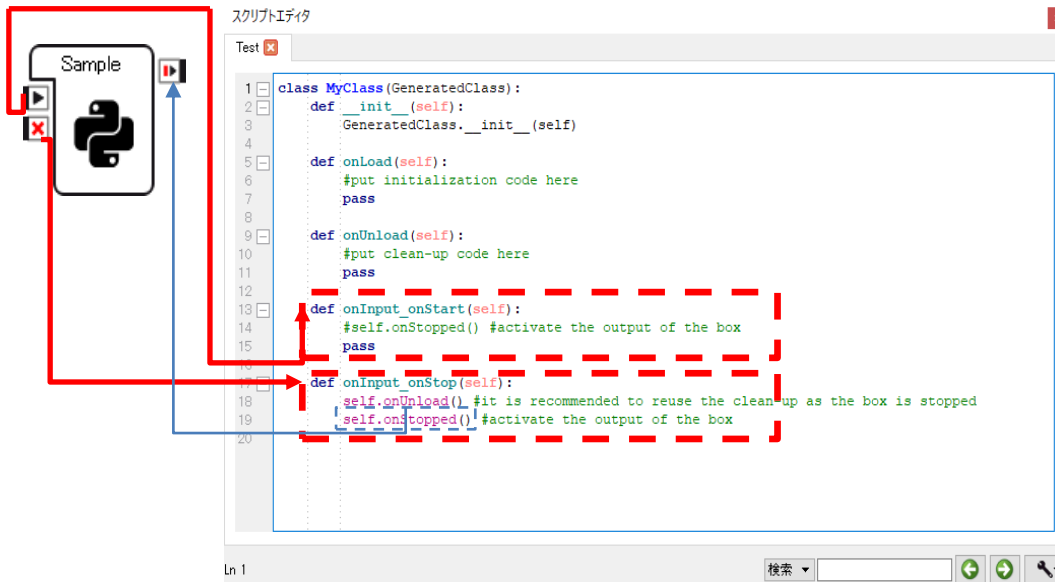


図 5.6-2 Python ボックスのスク립トエディタ

5.6.2. ダイアグラムボックス

ダイアグラムボックスの作成方法、および基本的な事項について説明します。

5.6.2.1. ダイアグラムボックスの作成

Python ボックスと同様に、[フローダイアグラム]パネルの何もないところで右クリックし、[ボックスの新規作成]、[ダイアグラム]を選択することでダイアグラムボックスを作成することができます。また、グループ化したいボックスを選んだ状態で、右クリックメニューから[ボックスに変換]を選択することでもダイアグラムボックスを作成することができます。

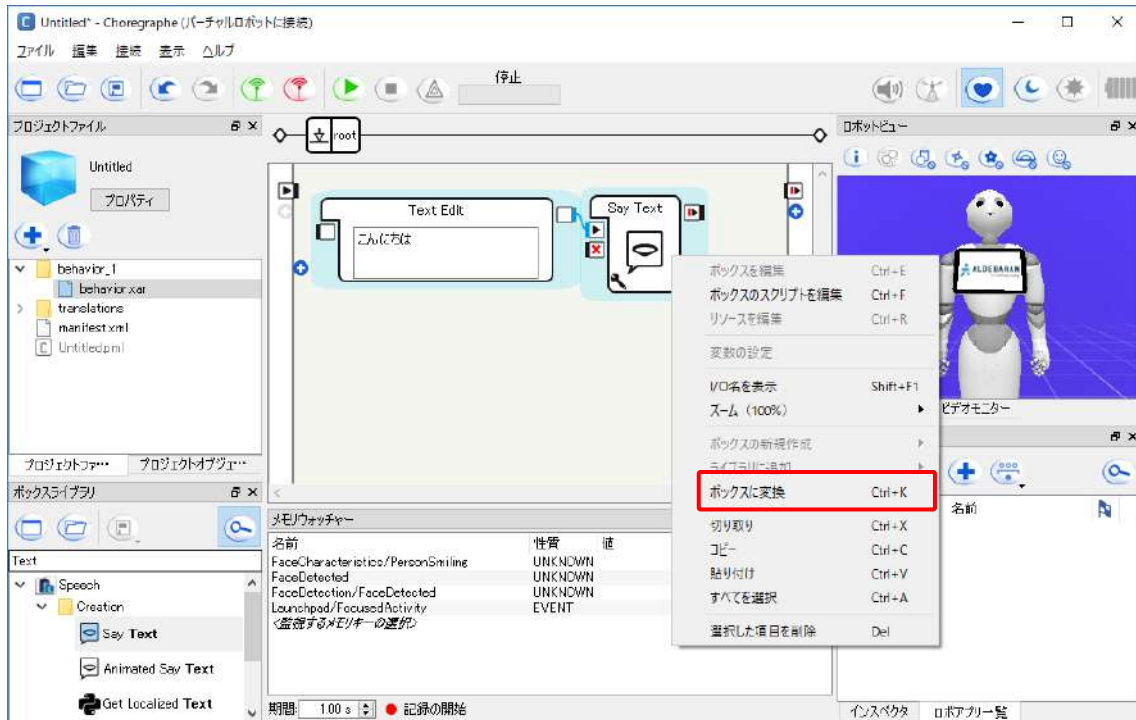


図 5.6-3 ダイアグラムボックスの作成

5.6.2.2. ダイアグラムボックスの編集

ダイアグラムボックスをダブルクリックすると[フローダイアグラム]パネルに、グループ化されているボックスが表示されます。ダイアグラムボックスの入力にシグナルが送られると、[フローダイアグラム]パネルの左端にある入力からシグナルが流れてきます。同様に[フローダイアグラム]パネルの右端にある出力にシグナルを送ると、ダイアグラムボックスの出力からシグナルが送出されます。また、ダイアグラムボックスの中にダイアグラムボックスを作成するなど階層構造を持つことができますが、[フローダイアグラム]パネルの上部には現在表示しているダイアグラムボックスの位置が表示されており、クリックすることで上位の階層に戻ることができます。

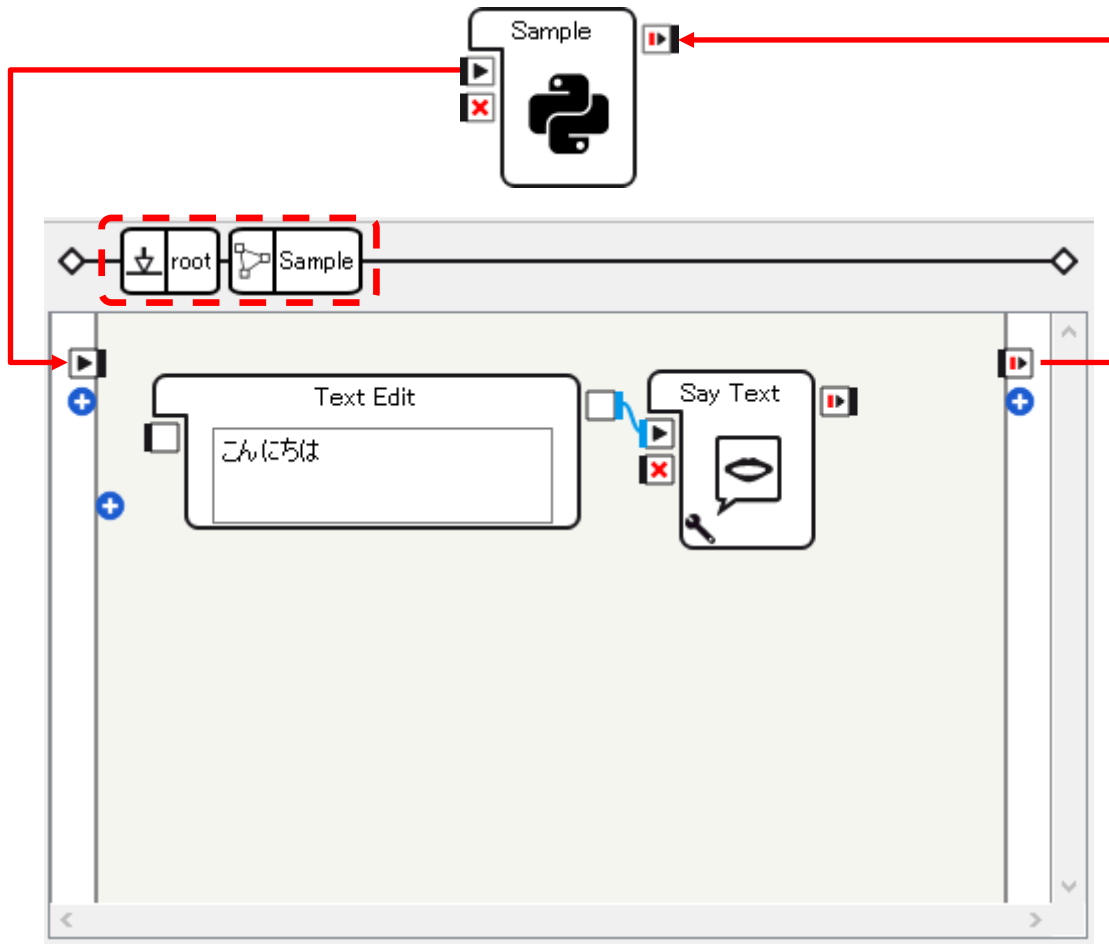


図 5.6-4 ダイアグラムボックスの編集

5.6.3. ダイアログボックス(Dialog ボックス)

ダイアログボックスについては、「Dialog ボックス」にて詳しく説明します。

5.6.4. タイムラインボックス

タイムラインボックスについては、「Timeline ボックス」にて詳しく説明します。

5.7. ボックスライブラリ

[ボックスライブラリ]パネルには多くの標準ボックスが登録されています。ボックスライブラリに標準以外のボックスを追加したり、独自に作成したボックスをライブラリとして他の開発者に配布することができます。

5.7.1. ライブラリの追加

ボックスライブラリは「.cbl」という拡張子のファイルです。[ボックスライブラリ]パネルにボックスライブラリを追加する手順は以下の通りです。

1. [ボックスライブラリ]パネルの[ボックスライブラリを開く]ボタンをクリックする。
2. ボックスライブラリのファイル (.cbl) ファイルを選択する。

3. [オープン]ボタンをクリックする。

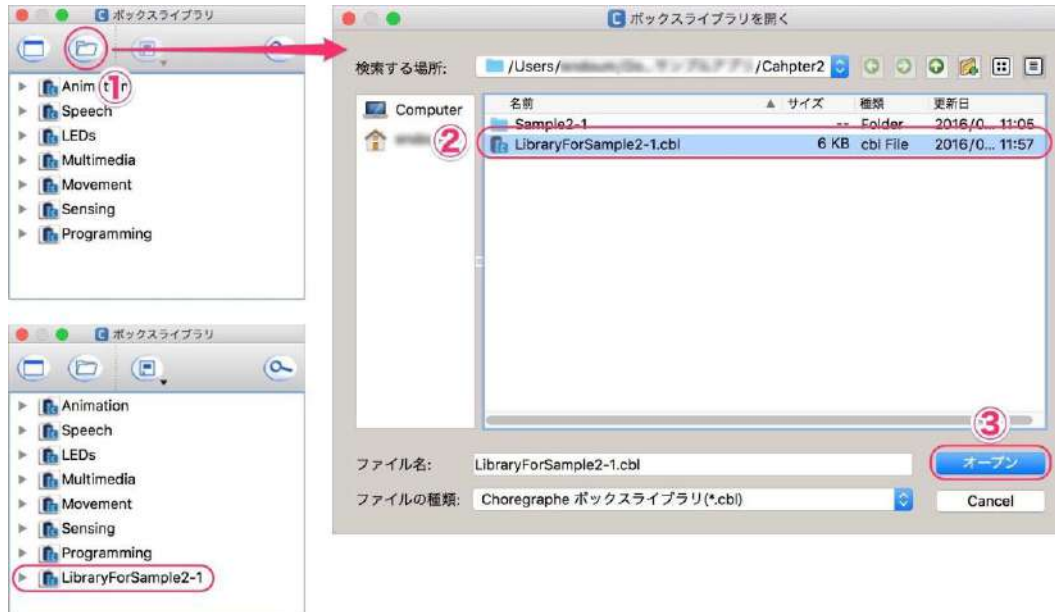


図 5.7-1 ボックスライブラリの追加

5.7.2. ライブラリの作成

独自に作成したボックスをライブラリとして他の開発者に提供する手順は以下の通りです。

1. [ボックスライブラリ]パネルの[新規ボックスライブラリ]ボタンをクリックする。
2. [ファイル名]に任意のライブラリ名を入力する。
3. [保存]ボタンをクリックすると、[ボックスライブラリ]パネルにライブラリが追加される。
4. 3で追加されたライブラリに独自に作成したボックスをフローダイアグラムからドラッグ&ドロップするとボックスが追加され、ライブラリ名に「*（アスタリスク）」が付きます。
5. ライブラリを右クリックして[ライブラリを保存]を選択する。
6. 3で作成されたボックスライブラリ（.cbl）ファイルを他の開発者に配布する。

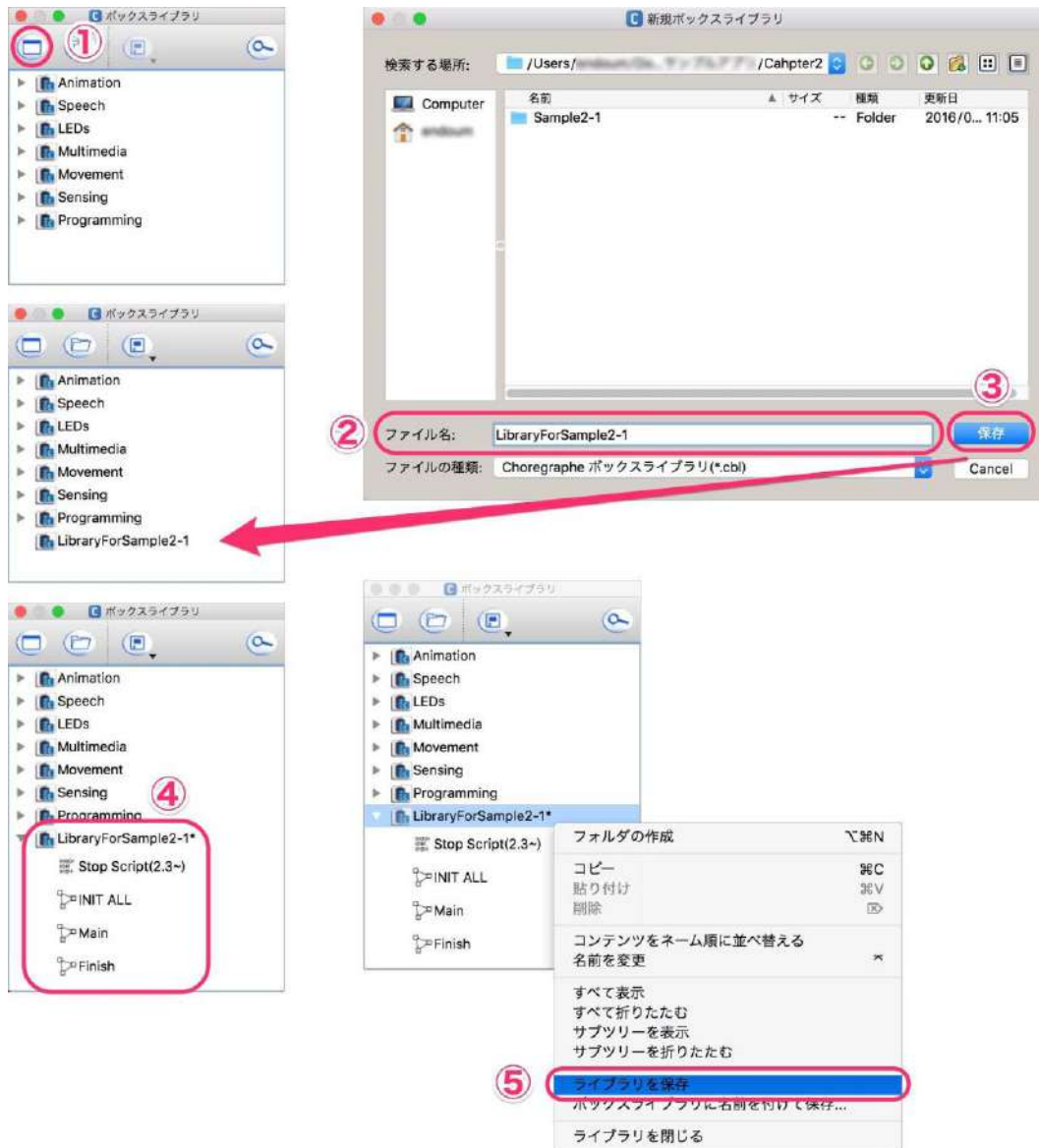


図 5.7-2 ブロックライブラリの作成

Sample 5-1

表 5.7-1 sample5-1

項目	説明
プロジェクト名	sample5-1
ファイルパス	chapter5/sample5-1/sample5-1.pml
概要	新規にプロジェクトを作成し、提供されたボックスライブラリ（SampleLibrary.cbl）を用いて、以下のようにフローダイアグラムを構成して動作を確認してください。

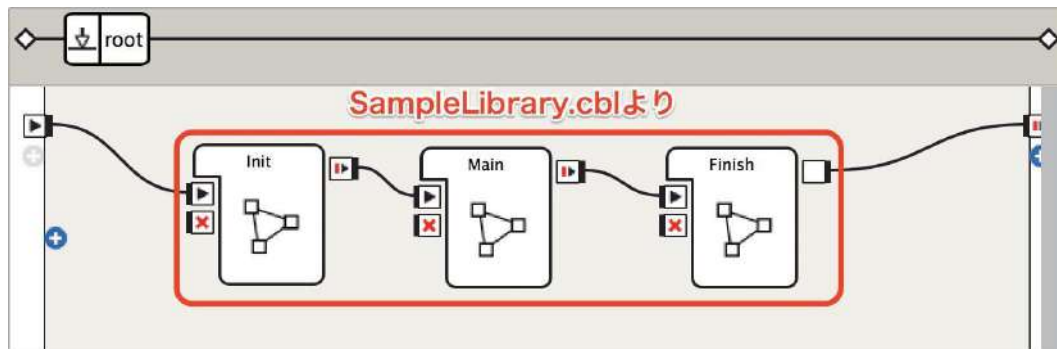


図 5.7-3 sample2-1 のフローダイアグラム

動作確認時は、Pepper の周囲半径 90cm 以上の空間を確保してください。人や物が近くに存在すると、セーフティー機能が働いて想定通りに動かない場合があります。このサンプルアプリは実機に接続して、オートノマスライフをオンの状態で実行してください。

ツールバーの [オートノマスライフ] ボタンをクリックして、ハートマークの内側が青い状態にしてください。



図 5.7-4 オートノマスライフがオンの状態

6. 会話

Pepper の一番重要なユーザインターフェイスは会話です。ユーザとのやり取りをすべてディスプレイで行ってしまったら、そのアプリは Pepper をプラットフォームとする必要はありません。この章では、Pepper との会話の仕方、声質やイントネーションの調整方法、会話に関する開発上の注意点などを解説します。

6.1. 関連ボックス

Pepper との会話を実現するために使用する主なボックスは以下の通りです。

表 6.1-1 会話関連の主なボックス

アイコン	ボックス名	説明
	Set Language	言語の設定を変更する。
	Say	特定の文字列を読み上げる。
	Say Text	入力[onStart]に渡された文字列を読み上げる。
	Animated Say	特定の文字列を読み上げながら、自動的に動作が付く。
	Speech Reco.	ユーザの言葉を聞き取り、判定する。
	Dialog	複数往復する複雑な会話を組み立てる。

次の節から、これら 6 つのボックスの使い方を紹介します。

6.2. Set Language ボックス

[Set Language]ボックスは、Pepper の言語を変更するボックスです。[Set Language]ボックスの変数は以下の通りです。

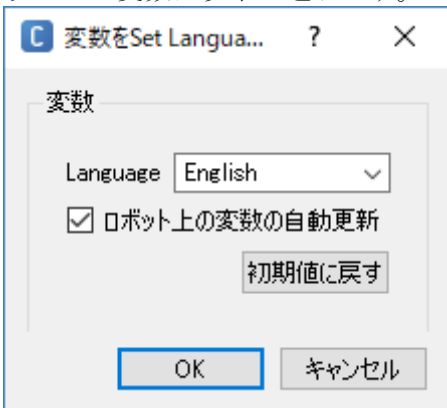


図 6.2-1 Set Language ボックスの変数

表 6.2-1 Set Language ボックスの変数

項目	説明
Language	変更したい言語。

Pepper の設定アプリで設定されている言語から変更する必要がなければ設定する必要はありません。バーチャルロボットで会話のテストをしたい場合は、会話が始まる前に[Set Language]ボックスを配置して、言語設定をする必要があります。

6.3. Say ボックス

[Say]ボックスは引数で指定された文字列を読み上げるボックスです。[Say]ボックスの変数は以下の通りです。

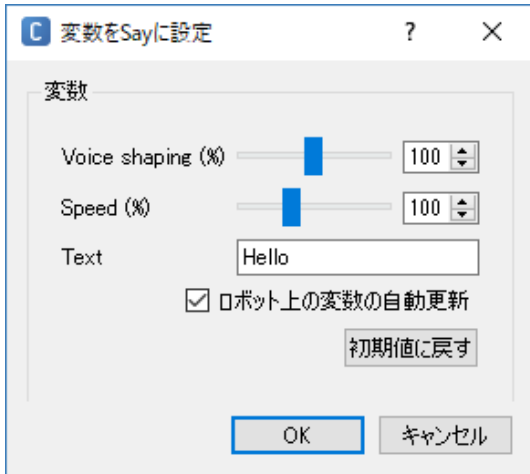
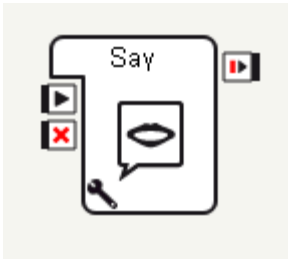


表 6.3-1 Say ボックスの変数

項目	説明
Voice shaping(%)	読み上げ時の声の高さ。
Speed(%)	読み上げ時の声の早さ。
Text	読み上げる文字列。

図 6.3-1 Say ボックスの変数

「Voice shaping(%)」と「Speed(%)」は両方とも値が "100" ですが、Pepper の推奨値は前者が "135" で、後者が "110" です。喜怒哀楽を表現するためにセリフの一部の声質を変更することができます。

Sample 6-1

表 6.3-2 Sample6-1

項目	説明
プロジェクト名	sample6-1
ファイルパス	chapter6/sample6-1/sample6-1.pml
概要	[Say]ボックスを使用したサンプルアプリです。声質のパラメータ値が推奨値と初期値で雰囲気が違うことを確認してください。

6.4. イントネーションと抑揚の調整

Pepper はバージョンアップを重ねる度にイントネーションの精度は上がっていますが、まだ違和感が残る場合があります。以下のように調整すると自然な発声にできます。

```
\rspd=110\ \vct=160\ええっ、 \pau=700\ \vct=145\ それ、
\pau=200\ \vct=155\ 本当なの？
```

図 6.4-1 「ええっ、それ本当なの？」の調整済みのフレーズ

6.4.1. 調整できる項目

イントネーションや抑揚の調整のために使用できる項目は以下の通りです。

表 6.4-1 調整項目

キー名	値の範囲	用途
vct	50～200 (デフォルト: 100)	声の高さを設定する。
rspd	50～400 (デフォルト: 100)	声の早さを設定する。
pau	ミリ秒	一時停止時間を設定する。単位はミリ秒。
vol	0～100%	音量を設定する。

Pepper の標準の声は vct=135, rspd=110 です。Say ボックスなどの発話をさせるボックス内の初期値は vct、rspd ともに 100 なので、標準値に調整するか、意図的に高い声や低い声に調整して発話させましょう。

6.4.2. その他の調整方法

イントネーションを調整するには、以下の様な方法もあります。

- 漢字をひらがなにする。
- ひらがなの一部をカタカナにする。
- ひらがなの一部を別の漢字にする。
- 漢字を一音ずつ別の漢字にする。
- 「、」やスペースで間を空ける。
- 語尾を上げるには「？」を付ける。
- 語尾に勢いを付けるには「っ」や「ッ」を付けたり、「！」を1つ以上付ける。
- 語尾を伸ばすには「ー」を1つ以上付けたり、「あ」「い」「う」「え」「お」を複数並べる。
- 「ゃ」「ゅ」「ょ」を試してみる。
- 「、」「。」「(スペース)」「|」「^」を試してみる。

声の高さなどを調整した後は、アプリが終了しても、その状態を維持しますので、自分が作成したロボアプリの前に動作していたロボアプリの設定を引き継いでしまいます。回避するためには、発話テキストの先頭には声の高さや早さを設定するようにしましょう。

表 6.4-2 調整例の一覧

セリフ	調整後のセリフ
やったー！	\rspd=102\\vct=155\ヤッ田ーーーーーッ
いってらっしゃい	\rspd=115\\vct=140\イッテラッしやーあいつ
違うでしょ	\rspd=115\\vct=130\ちが\ vct=155\ 鶯でしょー？
それじゃあ、いきますよ！	\rspd=115\\vct=135\それじゃあ、\ vct=140\いきますよー？？
3、2、1	\rspd=120\\vct=140\さーん、、、 \pau=700\\rspd=115\\vct=140\にー、、、 \pau=700\\rspd=115\\vct=140\いーち、、、

6.4.3. セリフの確認方法

イントネーションと抑揚の調整時にセリフを確認する簡単な方法について説明します。

6.4.3.1. ロボットウェブページで確認

ロボットウェブページを表示して、セリフを入力し、Enter キーを押すことで確認できます。ロボットウェブページは"http://(Pepper の IP アドレス)"で開くことができます。

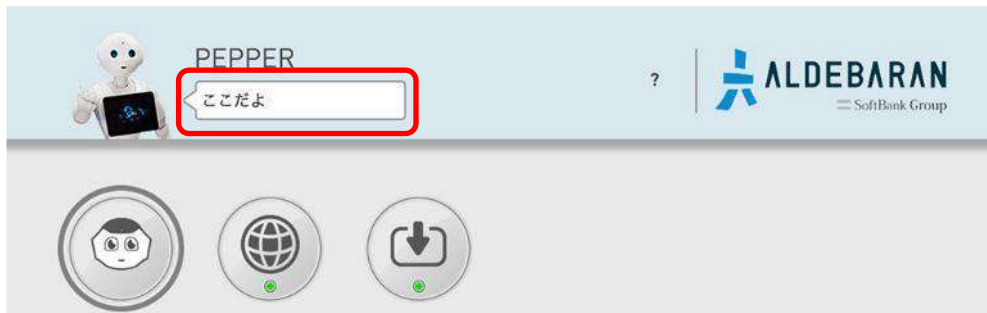


図 6.4-2 ロボットウェブページで確認

6.4.3.2. Choregraphe で確認

空のプロジェクトを作成し、結線していない[Say Text]ボックスを配置します。Pepper 本体と接続し、ツールバーの再生ボタンからプロジェクトを実行します。実行後に[Say Text]ボックスの入力をダブルクリックし、確認したいセリフを入力し OK を押します。

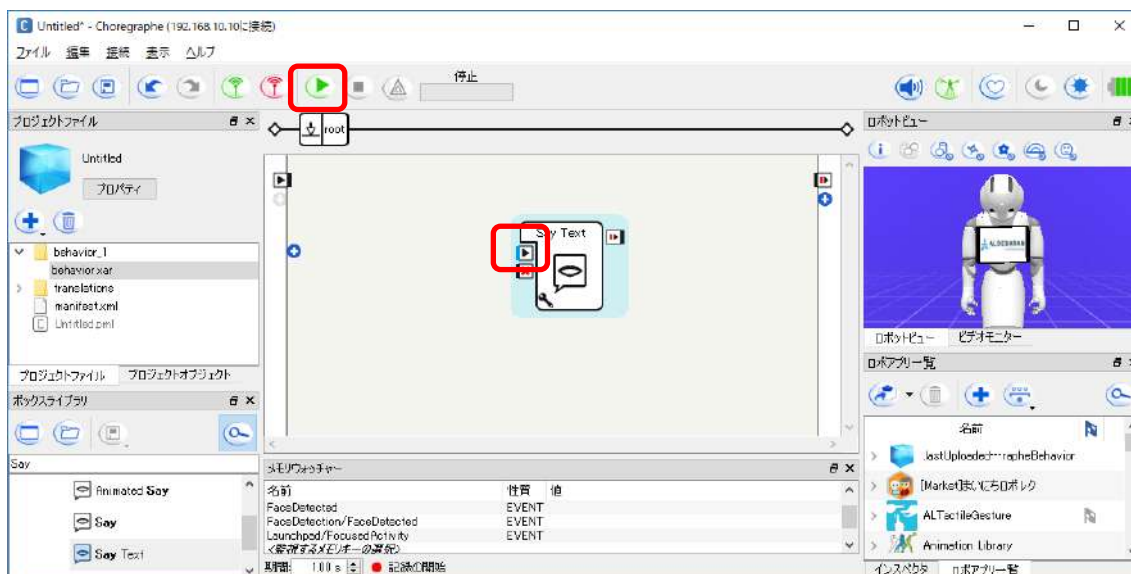


図 6.4-3 Choregraphe で確認

※ただし、この方法は Pepper と Choregraphe のバージョンが揃っている必要があります。

6.4.4. 調整済みテキストのサンプル集

調整済みテキストについて、いくつかサンプルがありますので参考にしてください。

- Pepper セリフ集_Ver1.1.xlsx (本ドキュメントとともに配布しております)
- Pepper for Biz (法人向けモデル) サポートの Pepper セリフ集 (<https://www.softbank.jp/robot/biz/support/tool/intonation/>)

6.5. Say Text ボックス

[Say Text]ボックスは入力で受け取った文字列を読み上げるボックスです。[Say Text]ボックス自体にテキストを用意する機能はありませんので、[Text Edit]ボックスなどと組み合わせ使用します。[Say Text]ボックスの変数は以下の通りです。



図 6.5-1 Say Text ボックスの変数

表 6.5-1 Say Text ボックスの変数

項目	説明
Voice shaping(%)	読み上げ時の声の高さ。
Speed(%)	読み上げ時の声の早さ。

Sample 6-2

表 6.5-2 Sample6-2

項目	説明
プロジェクト名	sample6-2
ファイルパス	chapter6/sample6-2/sample6-2.pml
概要	[Say Text]ボックスを使用したサンプルアプリです。[Say]ボックスとの使い方の違いを確認してください。

6.6. Animated Say ボックス

[Animated Say]ボックスは、引数で指定された文字列の読み上げとモーションを行うボックスです。内部では ALAnimatedSpeech を使用しています。実際に試すと分かりますが、Pepper が直立不動でベラベラと喋り続けると、ユーザに少なからず恐怖を与えます。基本的に Pepper が喋っている間は Pepper の体を動かす必要がありますが、一語一句に独自の動作を付けていくのはとても時間がかかります。[Animated Say]ボックスを使用すると自動的にモーションが付き、最低限の動作は行ってくれます。

[Animated Say]ボックスで設定できる変数は以下の通りです。

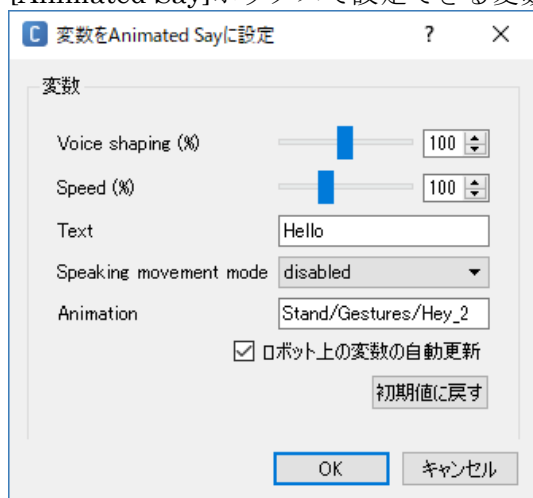


図 6.6-1 Animated Say ボックスの変数

表 6.6-1 Animated Say ボックスの変数

項目	説明
Voice shaping(%)	読み上げ時の声の高さ。
Speed(%)	読み上げ時の声の早さ。
Text	読み上げる文字列。
Speaking movement mode	モーションの自動実行に関する設定。
Animation	モーションの自動実行の設定で disabled を選択した場合は、本パラメータにて実行するモーションを直接指定することができます。自動実行の設定が、random か contextual の場合、本設定は無効になります。

[Animated Say]ボックスのパラメータ設定ボタンをクリックすると、[Say]ボックスと同じパラメータの他に、[Speaking movement mode]というパラメータがあります。

[Speaking movement mode]パラメータの値は"contextual（初期値）"、"random"、"disabled"から選択できます。それぞれの意味は以下の通りです。

表 6.6-2 Animated Say ボックスの Speaking movement mode パラメータの値

値	説明
disabled	モーションの自動実行を行わない。
random	モーションをランダムで実行する。
contextual	モーションを文脈から判断して実行する。

Pepper にインストールされている動作の一覧を参照する手順は以下の通りです。

1. Choregraphe の[ヘルプ]メニューから[参照 API]を選択して NAOqi の API ドキュメントを表示する。
2. 左側のメニュー（青い背景のエリア）から[NAOqi Motion]→[ALAnimationPlayer]→[ALAnimationPlayer - Advanced]を選択する。



図 6.6-2 Pepper にインストールされている動作

Sample 6-3

表 6.6-3 Sample6-3

項目	説明
プロジェクト名	sample6-3
ファイルパス	chapter6/sample6-3/sample6-3.pml
概要	[Animated Say]ボックスを使用したサンプルアプリです。自動で動作が付くことを確認してください。

6.7. Speech Reco.ボックス

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ふさわしいものを選択し、文字列として出力します。

6.7.1. 聞き取りたい言葉の登録

[Speech Reco.]ボックスでユーザの言葉を聞き取るには、変数（パラメータ）で聞き取りたい言葉のリストを登録します。[Speech Reco.]ボックスのパラメータ設定内容は以下の通りです。



図 6.7-1 Speech Reco.ボックスの変数

表 6.7-1 Speech Reco.ボックスの変数

変数名	説明
Word list	聞き取りたい言葉を「; (セミコロン)」で区切ったリスト。
Confidence threshold (%)	言葉の認識率の閾 (しきい) 値。
Visual Expression	認識エンジンの状態を示す LED の動作を有効または無効にする。
Enable word spotting	オフなら完全一致、オンなら部分一致。

6.7.2. 認識率

[Speech Reco.]ボックス動作中、Pepper はユーザが話した言葉が[Word list]パラメータに登録されている言葉にどれだけ似ているかを計算した結果を認識率(%)として出力します。認識率は[ダイアログ]パネル ([表示]メニュー → [ダイアログ]) を表示すると確認できます。

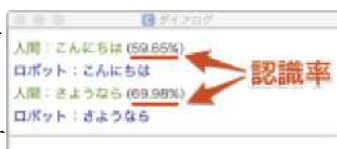


図 6.7-2 認識率

ユーザが発した言葉を[Word list]パラメータに登録されたすべての言葉に照らし合わせて計算された認識率が、どれも[Confidence threshold]パラメータの値を超えない場合は、聞き取れなかったと判断されます。

Pepper の設置環境に依存しますが、[Confidence threshold]パラメータの値は"40"前後で調整するのが現実的です。閾値を高くすると、聞き取っていても、正確さに欠ける場合には反応しなくなります。反対に閾値を低くすると、間違った言葉にも反応してしまうことが多くなります。

6.7.3. 聞き取った後の処理

[Speech Reco.]ボックスには出力が3つあり、それぞれの違いは以下の通りです。

表 6.7-2 Speech Reco.ボックスの出力

出力名	説明
onStopped	ボックスが終了した時に出力される。
wordRecognized	言葉を認識した (聞き取った言葉を出力) 時に出力される。
onNothing	聞き取れなかった時に出力される。

ユーザの発した言葉が[Word list]パラメータに登録されている言葉のいずれかだと判断した場合、出力[wordRecognized]から後続へ処理が継続します。聞き取れなかった場合、出力[onNothing]から後続へ処理が継続します。通常、[Switch Case]ボックスと結線して使

用します。
 ロボアプリでは必ず聞き取れなかった場合の処理を実装してください。

Sample 6-4

表 6.7-3 Sample6-4

項目	説明
プロジェクト名	sample6-4
ファイルパス	chapter6/sample6-4/sample6-4.pml
概要	[Speech Reco.]ボックスを使用したサンプルアプリです。 変数の設定の仕方、聞き取れた場合の処理と聞き取れなかった場合の処理の実装の仕方を確認してください。

6.7.4. 聞くと話すは完全分離

[Speech Reco.]ボックスは、
[wordRecognized]または[onNothing]から
出力した場合、ボックス自体は終了せず、
Pepper は聞き取りモード（目と耳が青く
クルクル回っている状態）のままになりま
す。この状態のまま Pepper が喋ると、自
身の声に反応して想定外の動作をする可
能性があります。

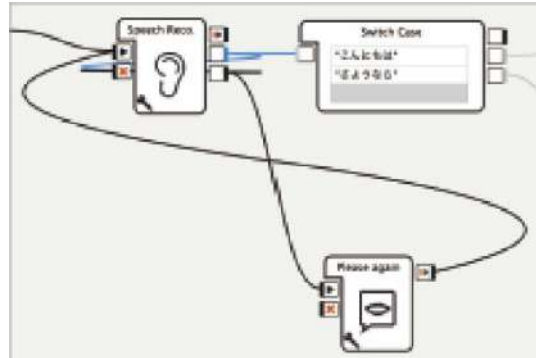


図 6.7-3 聞くと話すは完全分離

この問題を回避するために、「聞く」と「話す」は完全に分離する必要があります。そのためには、Pepper が言葉を聞き取った直後に、[Speech Reco.]ボックスを停止させます。動作中の[Speech Reco.]ボックスを停止させるには、入力[onStop]への結線が必要です。

Sample 6-5

表 6.7-4 Sample6-5

項目	説明
プロジェクト名	sample6-5
ファイルパス	chapter6/sample6-5/sample6-5.pml
概要	sample6-4 を改良して、聞くと話すを完全に分離したサンプルアプリです。Pepper が喋っている間は聞き取りモードが止まっていることを確認してください。

6.8. Dialog ボックス

[Dialog]ボックスは Pepper とユーザの間で、数往復のやり取りをさせたい場合に使用するボックスです。言葉のやり取りは[Speech Reco.]ボックスと[Say]ボックスの組み合わせでもできますが、複雑なやり取りや、長いやり取りを実現しようとする、多くのボックスを配置する必要がある等の難点があります。Dialog ボックスでは会話のルールを QiChat というスクリプト言語を使って定義するため、複雑なやり取りや長いやり取りを効率的に実現することができます。また、QiChat によって記述されたファイルをトピックファイル (.top) と呼びます。[Dialog]ボックスは、それ単体では機能せず、トピックファイルと関連付けて利用します。トピックファイルの作成方法は[プロジェクトファイル]パネルから新規ダイアログトピックを追加するか、[フローダイアグラム]パネルで右クリックし、[ボックスの新規作成]から[ダイアログ]を選択することで作成できます。

6.8.1. 作成手順

これまで紹介したボックスは、[ボックスライブラリ]パネルから[フローダイアグラム]パネルへドラッグ&ドロップして作成しましたが、[Dialog]ボックスは以下の手順で作成します。

1. [プロジェクトファイル]パネルの[+]ボタンをクリック
2. プルダウンメニューから[新規ダイアログトピック...]を選択
3. [ダイアログトピックを新規追加]ウィンドウの[名前]に任意の名前を入力
4. [ロボアプリの対応言語]から"Japanese(jpj)"だけにチェックを入れる
5. [追加]ボタンをクリック
6. 3で指定した名前のフォルダが[プロジェクトファイル]パネルに追加されていることを確認
7. 追加されたフォルダ内の○○dlg ファイルを[フローダイアグラム]パネルにドラッグ & ドロップ
8. [フローダイアグラム]パネルを右クリックし、ボックスの新規作成からも作成することができます。



図 6.8-1 Dialog ボックスの作り方

6.8.2. トピックファイル

[プロジェクトファイル]パネルからトピックファイルをダブルクリックすると、[スクリプトエディタ]ウィンドウが開いて、会話のルールを記述できます。トピックファイルの1から2行目は変更しないでください。"#"以降は行末までコメントになります。

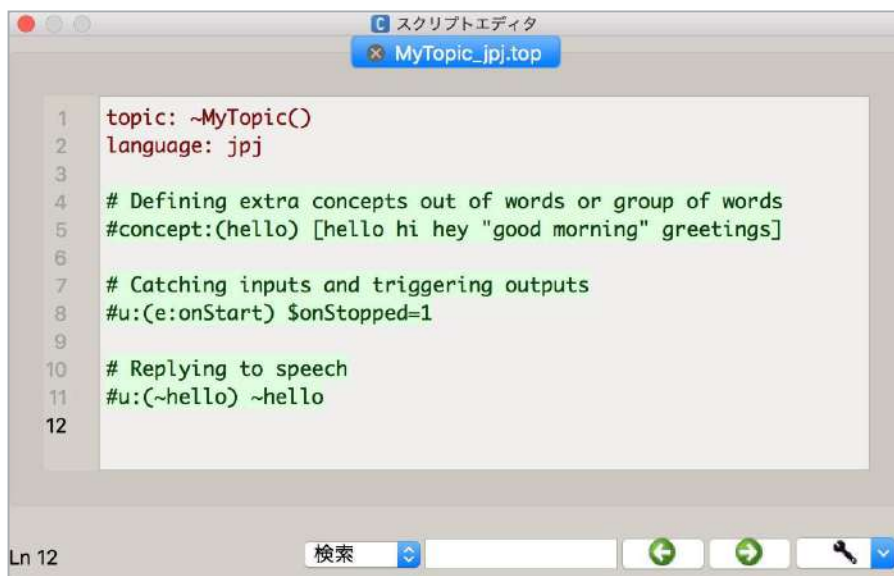


図 6.8-2 トピックファイル

6.8.3. ユーザルール

トピックファイルの最も基本的な構文としてユーザルールがあります。書式は以下の通りです。*input*の部分に、ユーザの入力(想定されるユーザの発話内容)を設定し、*answer*の部分にそれに対する Pepper の発話内容を記述します。

```
u: (input) answer
```

図 6.8-3 ユーザ入力の書式

ユーザが「こんにちは」と言ったら、Pepper が「こんにちはです」と返すスクリプトは以下のようになります。

topic: ~sample()

language: jpj

u:(こんにちは)こんにちはです

図 6.8-4 ユーザ入力サンプル

1 行目、2 行目はトピック名と言語設定で、トピックファイル作成時に自動で生成されます。トピック名については、重複するとエラーになることがありますので、他のロボアプリと被らないような名前を設定しておくといよいでしょう。

Sample 6-6

表 6.8-1 Sample6-6

項目	説明
プロジェクト名	sample6-6
ファイルパス	chapter6/sample6-6/sample6-6.pml
概要	[Dialog]ボックスを使用して 1 往復の会話を行うサンプルアプリです。会話の基本文法、コメントの入れ方、音声調整用コマンドの使い方を確認してください。アプリは自動で終了しないので、Choregraphe のツールバーの[停止]ボタンを押してください。

6.8.4. ユーザサブルール

ユーザ入力はサブルールを使って階層化することができます。書式は以下の通りですが、ちょっと分かりにくいのでサンプルで説明します。

```
u:(input1) answer
  u1:(input2) answer
  u1:(input3) answer
    u2:(input4) answer
      u3:(input5) answer
        u2:(input6) answer
```

図 6.8-5 ユーザ入力のサブルールの書式

以下のサンプルでは、ユーザの「動物」という発話を認識すると、Pepper は「突然ですが、動物は好きですか?」と返します。続けて、ユーザが「はい」と回答すれば、Pepper は「犬と猫では、どちらが好きですか?」と返します。このように u、u1、u2 など "u+(数字)" を用いることで会話の階層化を行うことができます。u:(動物)の部分のユーザ入力があるまで、u1:(はい)や u1:(いいえ)の部分が反応することはありません。同様に u1:(はい)の部分のユーザ入力があるまで、u2:(犬)、u2:(猫)の部分が反応することはありません。また、サブルールは 7 階層までしか使用できず、u8 を使用するとエラーが発生します。

```
topic: ~sample()
language: jpj
```

```
u:(動物) 突然ですが、動物は好きですか？
u1:(はい) 犬と猫では、どちらが好きですか？
u2:(犬) 私はパグが好きです。
u2:(猫) 私はシャムが好きです。
u1:(いいえ) そうですね。残念です。
```

図 6.8-6 ユーザ入力のサブルールサンプル

Sample 6-7

表 6.8-2 Sample6-7

項目	説明
プロジェクト名	sample6-7
ファイルパス	chapter6/sample6-7/sample6-7.pml
概要	複数回往復する会話を実装しているサンプルアプリです。サブルールの記述方法を確認してください。アプリは自動で終了しないので、Choregraphe のツールバーの[停止]ボタンを押してください。

6.8.5. 複数の単語登録

Pepper からユーザへ「○○は好きですか？」という質問をしたとき、以下の様なサブルールの記述方法だと「はい」または「いいえ」に限定されてしまいます。「好き」や「嫌い」と答えられてしまうと、Pepper が無反応になります。

```
u:(ユーザのセリフ)○○は好きですか？
u1:(はい) 「はい」に対する Pepper の返答
u1:(いいえ) 「いいえ」に対する Pepper の返答
```

図 6.8-7 ユーザ入力のサブルールサンプル

肯定または否定の複数の単語に対して同じ反応をさせたい場合は、[]を使用して、その中に半角スペース区切りで単語を並べます。

```
u:(ユーザのセリフ)○○は好きですか？
u1:([はい 好き]) 「はい」に対する Pepper の返答
u1:([いいえ 嫌い]) 「いいえ」に対する Pepper の返答
```

図 6.8-8 ユーザ入力のサブルールサンプル

[]の中の単語数が増えすぎると、トピックファイル内の見通しが悪くなり、後の保守性にも悪影響を及ぼします。[]をユーザのセリフの()内に直接記述せずに、**concept** 文を用い

るとトピックファイルの上部にまとめることができます。

```
concept:(キー)[単語 1 単語 2 単語 3 . . .]
```

図 6.8-9 concept 文

concept 文に登録した複数の単語をユーザのセリフに当てはめるには以下のように記述します。

```
u:(~キー)Pepper の返答
```

図 6.8-10 concept 文の使い方

例えば以下の様な使い方をします。

```
concept:(yes)[はい 好き]
```

```
concept:(no)[いいえ 嫌い]
```

```
u:(ユーザのセリフ)○○は好きですか？
```

```
u1:(~yes) 「はい」 に対する Pepper の返答
```

```
u1:(~no) 「いいえ」 に対する Pepper の返答
```

図 6.8-11 concept 文のサンプル

Sample 6-8

表 6.8-3 Sample6-8

項目	説明
プロジェクト名	sample6-8
ファイルパス	chapter6/sample6-8/sample6-8.pml
概要	ユーザの多様な言葉に対応できるように、複数単語登録を行ったサンプルアプリです。アプリは自動で終了しないので、Choregraphe のツールバーの[停止]ボタンを押してください。

6.8.6. オプションを用いた単語登録

ユーザの回答内容として、「好き」が想定される場合、ユーザは「大好き」と回答するかもしれませんが。複数の単語として「好き」と「大好き」の両方を記述することもできますが、オプションを用いたほうが適切に記述することができます。

オプションとなりうるワードを{}で括ることでオプションを使用することができます。

```
u:(ユーザのセリフ)○○は好きですか？
```

```
u1:({はい {大}好き}) 「はい」 に対する Pepper の返答
```

```
u1:({いいえ {大}嫌い}) 「いいえ」 に対する Pepper の返答
```

図 6.8-12 ユーザ入力の子ルールのサンプル

6.8.7. ボックスからの出力

[Dialog]ボックスは[Speech Reco.]ボックスと同様に、明示的に終了しないと次のボックスへ処理を流したり、アプリ終了などをすることができません。

トピックファイルの中で[Dialog]ボックスを終了するには、以下の様な記述が必要です。

```
$出力名 = 値
```

図 6.8-13 ボックスからの出力の書式

例えば次の様に記述します。

```
u:(ユーザのセリフ)○○は好きですか？
```

```
u1:(はい)「はい」に対する Pepper の返答 $onStopped = 1
```

```
u1:(いいえ)「いいえ」に対する Pepper の返答 $onStopped = 2
```

図 6.8-14 ボックスからの出力のサンプル

\$出力名に設定する値は数字でも文字列でも構いません。文字列の場合は値の前後に"(ダブルコーテーション)を付けます。[Dialog]ボックス終了後、会話の終わり方によって次の処理に変化を与えたいなら、上記の例のように会話の終了毎に違う値を\$出力名に設定すべきです。[Dialog]ボックス終了後、会話の終わり方によって次の処理に変化を与える必要が無いなら、\$出力名に設定する値は同じもので構いません。

Sample 6-9

表 6.8-4 Sample6-9

項目	説明
プロジェクト名	sample6-9
ファイルパス	chapter6/sample6-9/sample6-9.pml
概要	[Dialog]ボックスを終了するサンプルアプリです。[Dialog]ボックスの出力[onStopped]のデータ型を数 (Number) に変更してあります。[Dialog]ボックス終了時に 1~3 の値が[onStopped]から出力されます。標準のボックスである[Switch Case]で[Dialog]ボックスから出力された値を判定し、最後に異なる一言を追加で喋るようになっています。

6.8.8. イベント

これまで紹介してきた[Dialog]ボックスの使い方 (トピックファイル内の記述方法) の最大の欠点は、ユーザがある特定の言葉を話しかけないと会話が始まらないということです。開発者は Choregraphe を用いてアプリの内容を見れば、どんな言葉をかければ会話が始まるか分かります。しかし、ユーザはそのようなことを知る由もなく、Pepper の前に立っても何も始まらないので「？」となって立ち去ってしまうでしょう。ロボアプリは、Pepper 側からユーザにアプローチして、どんな言葉をかけて欲しいか誘導しなければなりません。それを実現する方法としては、[Dialog]ボックスの前に[Say]や[Animated Say]ボックスを配置して、Pepper から話しかけるようにする方法があります。また、[Dialog]ボックスが始まったことを検知して、それをトリガーとして Pepper から会話を始める方法も考えられます。Pepper のセンサーから周囲の環境変化を検知したり、アプリ内の状態変化を検知するには、イベントを使用します。ここでは、[Dialog]ボックスの中でイベントを使用する方法を解説します。

6.8.8.1. イベントとは？

イベントとは、Pepper がセンサーで捉えた環境の変化やアプリ内の状態変化を監視して、アプリに通知してくれる仕組みです。トピックファイル内でイベントを記述する書式は以下の通りです。

e:イベント名

図 6.8-15 イベントの書式

6.8.8.2. ボックス開始のイベント

[Dialog]ボックスが開始されたことを示すイベントは以下のように記述します。

u:(e:onStart)Pepper のセリフ

図 6.8-16 イベントの使い方

このようにトピックファイルに記述しておけば、それに関連付けられている [Dialog]ボックスが開始された場合に Pepper が話し始めます。

6.8.8.3. 聞き取れなかった時のイベント

Pepper が質問して、ユーザの答えが用意された回答のどれにも当てはまらず、聞き取ることができなかった場合は、**Dialog/NotUnderstood** というイベントが発生します。以下のように利用します。

u:(e:onStart)突然ですが、犬と猫ではどちらが好きですか？
 u1:(犬)僕もワンちゃん好きですよ
 u1:(猫)最近、日本は空前の猫ブームだそうです
 u1:(e:Dialog/NotUnderstood)すみません、聞き取れませんでした。
 犬か猫で教えてくださいね。^stayInScope

図 6.8-17 聞き取れなかった時のイベント

ここで登場した^stayInScope というコマンドは、「u1 の範囲にとどまってください」という意味です。^stayInScope を記述しないと、もう一度ユーザが「犬」または「猫」と答えても Pepper は反応できなくなってしまいます。

6.8.8.4. 放置された時のイベント

PC、Web、スマホのアプリでは必要はありませんが、ロボアプリではユーザがいなくなっ
て放置されてしまう可能性があることを考慮しなければなりません。アプリ実行中にユー
ザに放置されたら、タイムアウト処理を行ってアプリを終了する必要があります。タイム
アウトの目安は 15 から 20 秒です。[Dialog]ボックスで会話が途切れた場合、
Dialog/NotSpeaking というイベントが発生します。

以下のように利用します。

u:(e:Dialog/NotSpeaking15)あれ？僕、放置されてますか？終了しちゃいますよお？

図 6.8-18 放置されたときのイベント

Dialog/NotSpeaking の末尾には 5/10/15/20 のいずれかが付き、それぞれタイムアウトの秒数を意味します。

Sample 6-10

表 6.8-5 Sample6-10

項目	説明
プロジェクト名	sample6-10
ファイルパス	chapter6/sample6-10/sample6-10.pml
概要	[Dialog]ボックスでイベントを使用したサンプルアプリです。すべてのイベントの動作を確認してください。

6.8.9. ビヘイビア、サウンドの実行

QiChat で他のビヘイビアの実行やサウンド再生などをコントロールすることができます。

表 6.8-6 アニメーション、サウンドの実行方法一覧

種別	書き方	説明
ビヘイビア	<code>^start</code> (アプリケーション ID/ビヘイビア名)	開始と終了を明示的に指定することでビヘイビアの実行終了を待つことができます。
	<code>^wait</code> (アプリケーション ID/ビヘイビア名)	
	<code>^run</code> (アプリケーション ID/ビヘイビア名)	途中で中断されることがあります。
サウンド	<code>^startSound</code> (サウンドファイル名)	開始と終了を明示的に指定することでサウンドの再生終了を待つことができます。
	<code>^waitSound</code> (サウンドファイル名)	
	<code>^runSound</code> (サウンドファイル名)	途中で中断されることがあります。

6.8.10. proposal

`proposal` は指定したルールによって順次実行される書き方です。プログラムをご存知の方には、「発話内容をベクターやイテレータとして扱えるようにするもの」と言った方が分かりやすいかもしれません。以下は `proposal` のサンプルスクリプトで、各行のコメントは、「あ、い、う、え、お」と順に認識させた時の `Pepper` の発話内容です。

```

proposal:プロポーザル 1
proposal:プロポーザル 2

u:(あ) ^nextProposal      # 「プロポーザル 1」
u:(い) ^nextProposal      # 「プロポーザル 2」
u:(う) ^previousProposal   # 「プロポーザル 1」
u:(え) ^previousProposal   # 「プロポーザル 2」
u:(お) ^sameProposal       # 「プロポーザル 2」
    
```

図 6.8-19 proposal のサンプル

サンプルの構成要素について以下の表で説明します。

表 6.8-7 proposal のサンプルの構成要素一覧

構文	解説
proposal:プロポーザル 1 proposal:プロポーザル 2	proposal の内容を定義しています。
^nextProposal	proposal キーワードで登録されているメッセージを順に実行します。
^previousProposal	ひとつ前の発話を実行します。履歴を辿るわけではありません。
^sameProposal	直前の発話内容を再度発話します。

サンプルでは **proposal** を使用して発話内容を複数登録することはできますが、「ひとつ前」や「ひとつ後」のように扱い方が限られていました。そこで、次は特定の発話内容を指定して扱う方法を取り上げます。

表 6.8-8 proposal の応用

構文	解説
proposal:%key1 おはよう proposal:%key2 こんにちは	proposal で登録する発話内容にタグ (キー) をつけ、複数登録できます。プログラムをご存知の方には、「連想配列」や「マップ」と表現をした方が分かりやすいかもしれません。実際に扱うには、下記の構文を使用します。
^activate(key1)	key1 として保存された単語を有効化します。
^deactivate(key1)	key1 として保存された単語を無効化します。
^goto(key1)	key1 として保存された単語を発話します。次にもう一度指定されても発話しません。
^gotoReactive(key1)	key1 として保存された単語を発話します。指定されれば何度でも発話します。
^gotoRandom(key1)	key1 として保存された単語群の中から、ランダムで発話します。

※key1,key2 は任意の単語

6.8.11. コラボラティブとノンコラボラティブ

トピックファイルには、コラボラティブとノンコラボラティブを設定できます。一般販売モデルでは、Solitary 状態の際に会話が始まるありますが、トピックファイルをコラボラティブに設定しておくことで、その会話内容を拡張することができます。

Dialog ボックスの編集ウィンドウにある「コラボラティブとしてパッケージに追加」にチェックを入れるか、プロジェクトのプロパティにあるトピックファイル名にチェックを入れることでコラボラティブに設定できます。

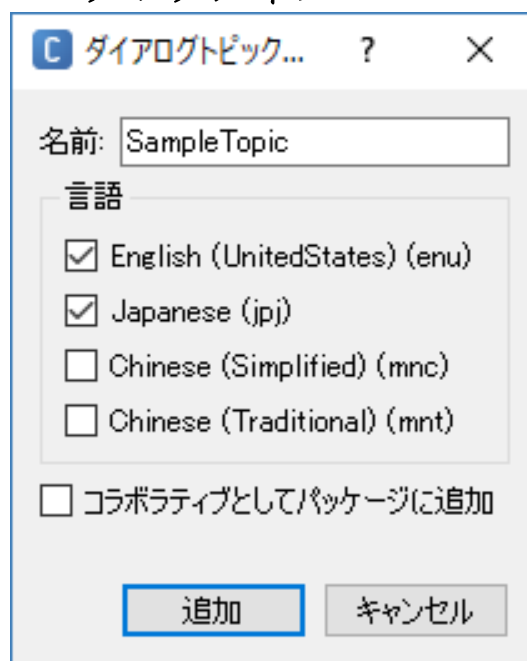


図 6.8-20 ダイアログボックスの編集

開発するロボアプリ内でしか使用しないようなトピックファイル(会話)については、ノンコラボラティブとなるように設定しておきましょう。以下のような、特定のシーンでしか使えないようなトピックファイルを間違えてコラボラティブに設定してしまうと、Solitary 状態中に何のことかよくわからない会話を Pepper が始めてしまいます。

u:(犬) 犬ですね！犬のどういったところが好きですか？この中から選んでね！
u1:(かわいい) 1番のかわいいですね！
u1:(かっこいい) 2番のかっこいいですね！
u1:(はやい) 3番のはやいですね！
u1:(かしこい) 4番の賢いですね！

図 6.8-21 コラボラティブだと意味が通じないトピックファイルの例

6.9. 会話作成時の注意点

この節では、会話を作成する際の注意点を紹介します。

6.9.1. 聞き返しとタイムアウト

特定の単語を認識すると進行するようにプログラムされていた場合、目的とする特定の単語を認識しないとロボアプリが進行しなくなってしまう。聞き取り開始から一定時間が経過（タイムアウト）した場合は、ロボアプリを終了させるか、再度利用者に質問を行い、利用者の回答を誘導するようタイムアウトルーチンをプログラムしてください。タイムアウト時間は 15 秒を推奨いたします。

6.9.2. 聞き取り精度向上

利用者の音声による回答は実に様々です。たとえば、肯定を意味する応答としては「はい、うん、そうだ……」など様々な対応が想定されます。利用者の回答の全てを想定できませんが、なるべく多くの回答を想定し様々なバリエーションの回答に反応できるようにしてください。また、どうしても聞き取りができない場合に備え、ディスプレイタッチでも回答ができるようにしておいた方が良いでしょう。

7. モーション

モーションとは **Pepper** の動作のことです。Pepper が喋っている間は動作をつけるべきであることはすでに紹介しました。[Animated Say]ボックスを使用すれば自動的に動作を付けることができますが、重要な部分や感情表現をするときには独自の動作を付けるとユーザに強い印象を与えることができます。この章では、Pepper に独自の動作をさせる方法やセリフと同期させる方法を解説します。

7.1. ポーズ関連のパネル

ポーズとは、一連の動作のある一瞬を切り出したものです。Choregraphe は、1つのモーションの中でポイントとなるポーズを複数登録すると、ポーズとポーズの間を自動補完してくれます。この節では、ポーズに関するパネルを解説します。

7.1.1. ロボットビュー

[ロボットビュー]パネルには、接続されている Pepper が表示されます。実機と接続されている場合は、Pepper を動かすと[ロボットビュー]パネル内のバーチャルロボットも同じ動きをします。[ロボットビュー]パネル上部のボタンは、情報の表示/非表示を切り替えるために使用します。



図 7.1-1 ロボットビュー

7.1.2. ポーズライブラリ

[ポーズライブラリ]パネルにはあらかじめ3種類のポーズが登録されています。Pepper の基本姿勢となるのは「Stand」です。フローダイアグラムパネルで複数のポーズを結線した場合、一つのポーズから次のポーズへと移る動作は Pepper が自動的に補完します。また、[ポーズライブラリ]パネルに Pepper がよく取るポーズを登録しておくことで、ダブルクリックするだけでいつでもそのポーズを再現させることができます。フォルダを作成し、独自のポーズを登録する手順は以下の通りです（以下の手順は、プロジェクトを保存してから行ってください）。

1. [ポーズライブラリ]パネルの任意の場所を右クリックして[フォルダの作成]を選択する
2. [フォルダ名]を入力して[OK]ボタンをクリックする
3. [ポーズライブラリ]パネルの[+]ボタンをクリックして[ポジションを新規作成]ウィンドウを表示する
4. [ポジションの名前]に任意の名前を入力する

5. [スクリーンショット]ボタンをクリックすると[ロボットビュー]パネルの背景が緑色に変化してアイコンが生成され、ファイル名が設定されたら[OK]ボタンをクリックする
6. [ポーズライブラリ]で生成されたポーズを 2 で作成したフォルダにドラッグ&ドロップする

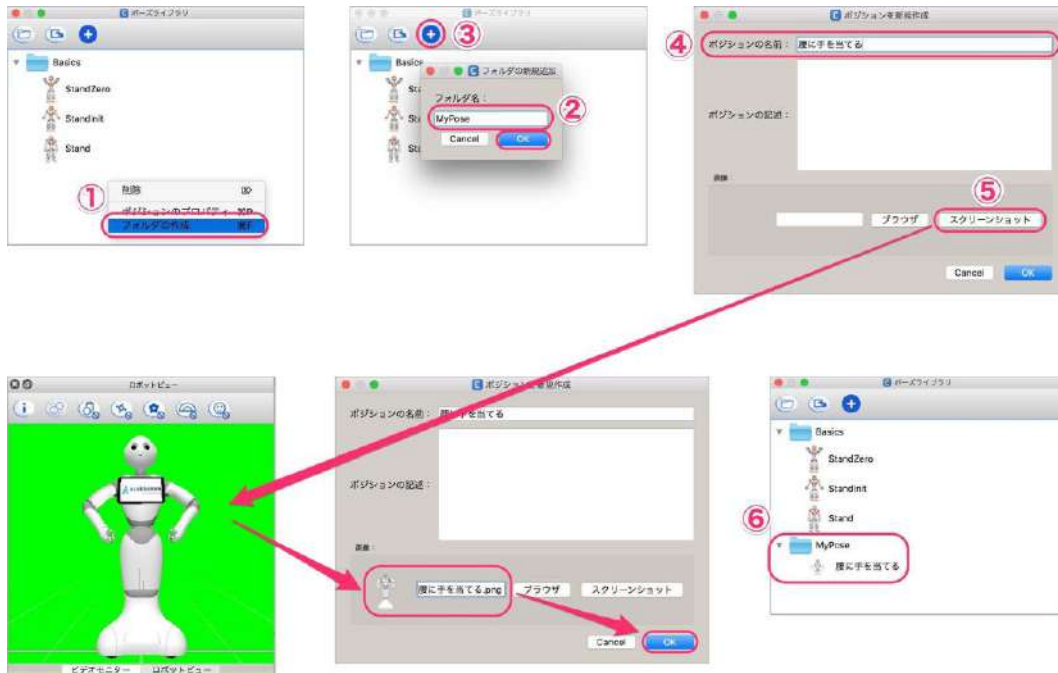


図 7.1-2 ポーズの登録

7.2. ポーズの作成

ポーズを作成するには、バーチャルロボットを使用する方法と、Pepper 実機を使用する方法があります。Pepper 実機を使用する方法は、頭と腕だけに対応していて、腰と指はバーチャルロボットで行う必要があります。この節では、ポーズの作成方法を解説します。

7.2.1. バーチャルロボットによるポーズ作成

[ロボットビュー]パネルに表示されているバーチャルロボットの頭、腕、腰をクリックすると、[インスペクタ]パネルの内容が変化します。

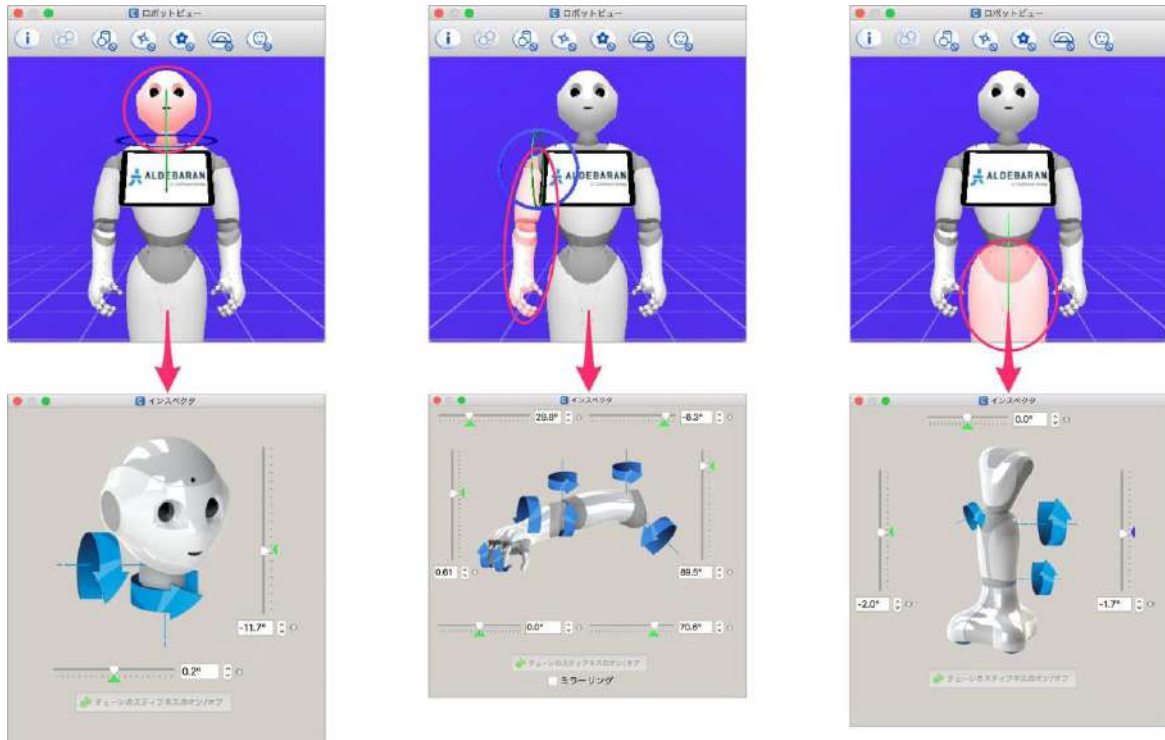


図 7.2-1 ポーズの作成 (バーチャルロボット)

[インスペクタ]パネル内のスライダーを動かすと[ロボットビュー]パネル内の Pepper の関節が連動して動きます。Choregraphe と Pepper 実機が接続されている場合は、実機も動きます。

7.2.2. 実機によるポーズ作成

[インスペクタ]パネルのスライダーを操作して思い通りのポーズを作成するのは時間がかかります。アニメーションモードを使用すると、Pepper 実機を使用してポーズを簡単に作成することができます。

7.2.2.1. アニメーションモード

Choregraphe と Pepper 実機を接続し、ツールバーの[アニメーションモード]ボタンをクリックするとボタンの色が緑から赤に変化します。また、Pepper の目がオレンジ色に変化します。この状態がアニメーションモードです。

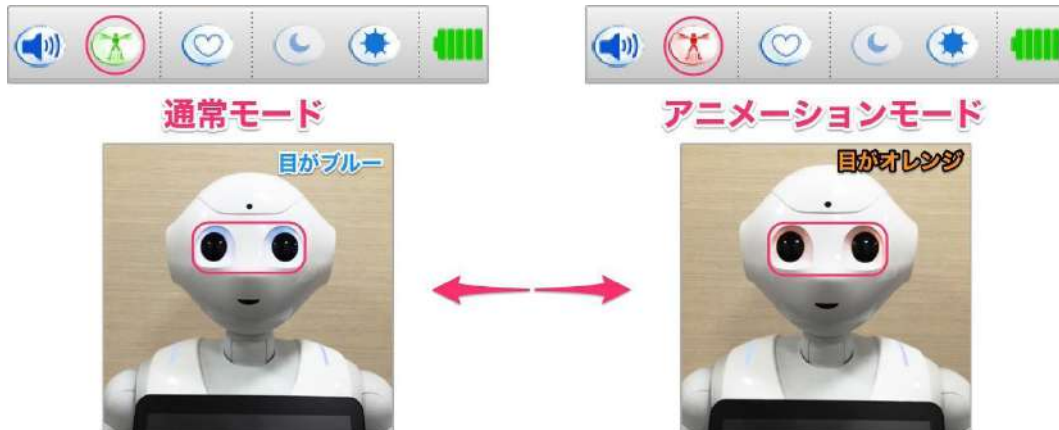


図 7.2-2 アニメーションモードの Choregraphe と Pepper の状態

アニメーションモードの状態ですらツールバーの[アニメーションモード]ボタンをもう一度クリックすると、元の状態に戻ります。

7.2.2.2. 頭の場合

アニメーションモードの状態ですら頭の角度を調整する手順は以下の通りです。

1. 頭の天辺をタッチすると目の上部が緑色になり、首の関節が緩む
2. 頭を両手で持って、ゆっくりと好みの角度に向ける
3. もう一度頭の天辺をタッチすると目全体がオレンジ色になり、首の関節がロックされる



図 7.2-3 アニメーションモードで頭の角度調整

7.2.2.3. 腕の場合

アニメーションモードの状態ですら腕の角度を調整する手順は以下の通りです。

1. 左右どちらかの手の甲に触れると、触れた側の目が緑色になり、腕の関節が緩む
2. 手の甲に触れたまま、ゆっくりと好みの腕の関節の角度を変える
3. 手の甲を離すと目がオレンジ色に戻り、腕の関節がロックされる



図 7.2-4 アニメーションモードで腕の角度調整

手の甲を離すときは、Pepper の腕を下から支えてから離すと下がらずに思った位置で固定できます。

7.3. Timeline ボックス

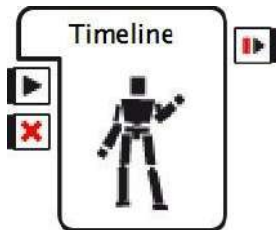


図 7.3-1 Timeline ボックス

複数のポーズから一連の動作（以下、モーション）を作成するには、[Timeline]ボックスを使用します。

7.3.1. タイムラインパネル

[Timeline]ボックスをダブルクリックすると、Choregraphe ウィンドウの上部に[タイムライン]パネルが横幅いっぱいに広がります。[タイムライン]パネル上段には[モーション]とタイトルの付いた白い帯があり、左から右へ流れる時間軸を表しています。白い帯の上には5刻みで数字が並んでいて、25フレームで1秒に設定されています。[タイムライン]パネル下段には[Behavior レイヤー]とタイトルの付いた青い帯があり、これも左から右へ流れる時間軸を表しています。

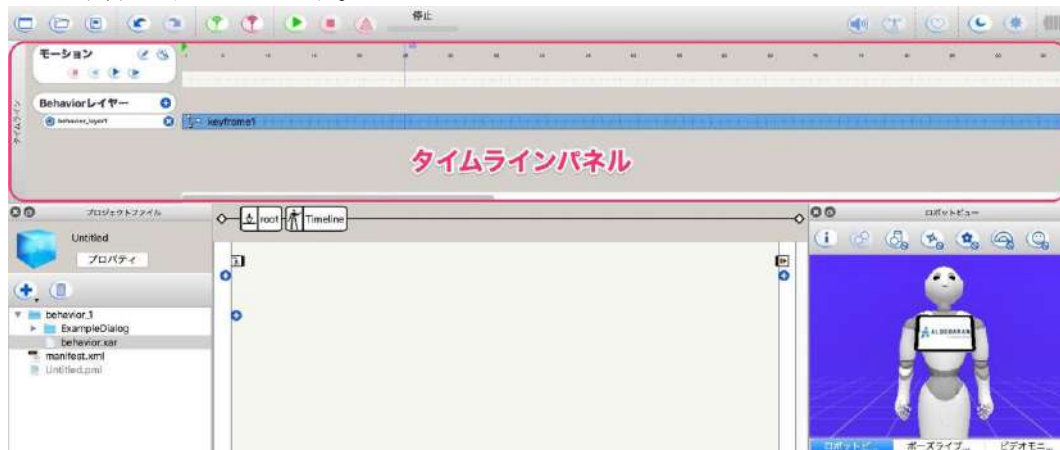


図 7.3-2 タイムラインパネル

7.3.2. モーションの作成

モーションの作成は、[タイムライン]パネルの上段の白い帯を使用します。
ここでは直立姿勢から両腕を前に出して、元の姿勢に戻るモーションを作成しながら説明します。
時間軸上のポーズは以下の通りです。

表 7.3-1 作成するモーションに登録するポーズ

経過時間	ポーズ
25 フレーム (1 秒)	[ポーズライブラリ]パネルの[StandInit]
50 フレーム (2 秒)	[ポーズライブラリ]パネルの[StandZero]
75 フレーム (3 秒)	[ポーズライブラリ]パネルの[StandZero]
100 フレーム (4 秒)	[ポーズライブラリ]パネルの[StandInit]

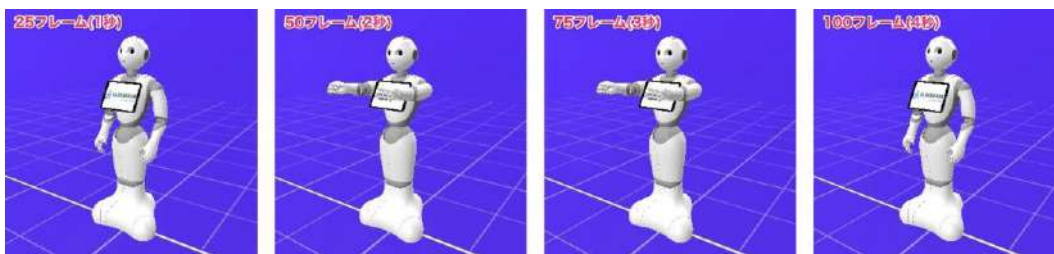


図 7.3-3 作成するモーションに登録するポーズ

初めて[Timeline]ボックスをダブルクリックすると、白い帯の 25 フレーム (1 秒) の位置に青紫色の縦線が配置されています。この縦線の位置にポーズを登録することができます。

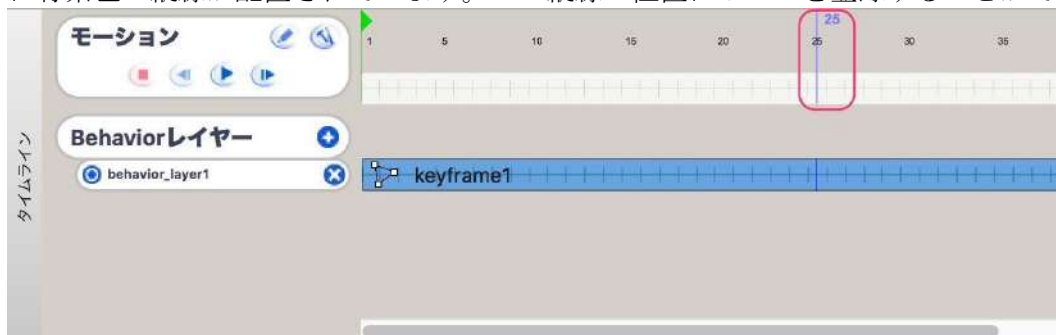


図 7.3-4 タイムラインパネルの初期状態

[ポーズライブラリ]パネルの[StandInit]をダブルクリックして、Pepper に直立姿勢を取らせます。

[タイムライン]パネルの白い帯の 25 フレームの位置を右クリックして、[キーフレーム]中に保存された関節→[全身]を選択するか F8 キーを押します (Mac の場合、システム環境設定によって[fn]+[F8]キー)。

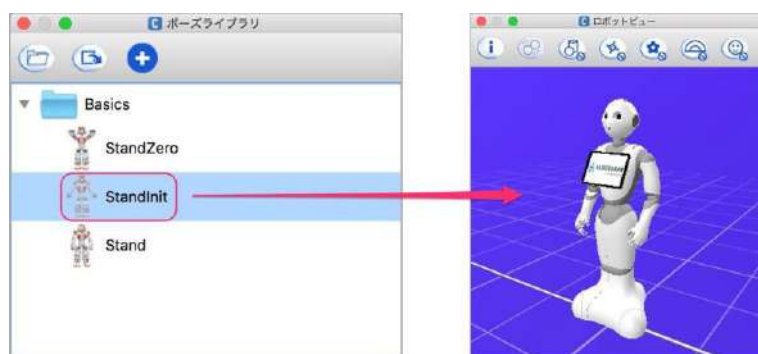


図 7.3-5 StandInit

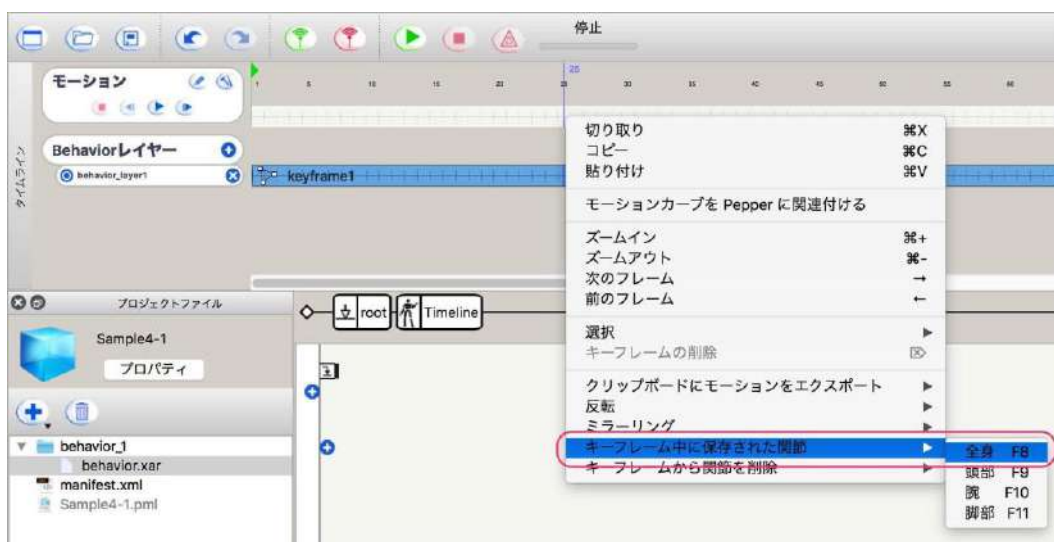


図 7.3-6 モーションのキーフレームの追加

25 フレームの位置にグレーの四角が追加されます。これをモーションのキーフレームと言います。

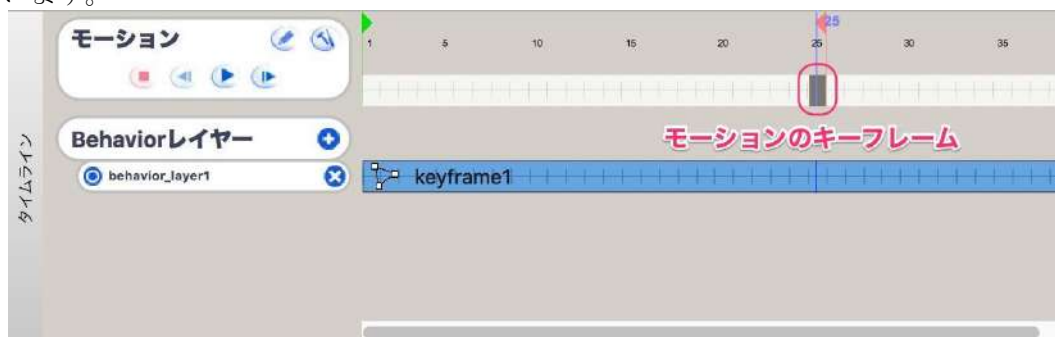


図 7.3-7 1つ目のモーションのキーフレーム

同様の手順で、2秒、3秒、4秒の位置にモーションのキーフレームを登録します。



図 7.3-8 すべてのモーションのキーフレームを登録した状態

一番右に追加したモーションのキーフレームのすぐ右側に、赤い旗のような印が移動してきます。この赤い旗を終了フレームと言い、モーションの終了点を示します。終了フレームを一番右のモーションのキーフレームより左側に配置すると、途中でモーションが止まります。また、[Timeline]ボックスも終了します。終了フレームを一番右のモーションのキーフレームよりも右に配置すると、見た目上の動きは止まっても、[Timeline]ボックス

は終了フレームまでの時間が経過しないと終了しません。

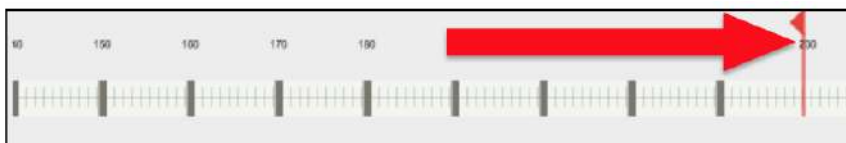


図 7.3-9 終了フレームの調整

これでモーションは完成です。モーションはアプリを実行しなくても動作を確認することができます。白い帯の左端に[モーション]と記述されているエリアがあります。その中に青い右向き三角形のボタンがあります。このボタンをクリックすると動作を確認できます。



図 7.3-10 モーションの動作確認をするボタン

実際の開発で[StandInit]や[StandZero]は使用しない

この節では、ポーズを簡単に登録していただくために、[ポーズライブラリ]パネルの[StandInit]や[StandZero]を使用しましたが、アイコンを見ていただければお分かりの通り、このポーズは NAO 用です。稀にエラーを発生させるので、実際の開発では Pepper 用のポーズを登録しておくことをお勧めします。

Sample 7-1

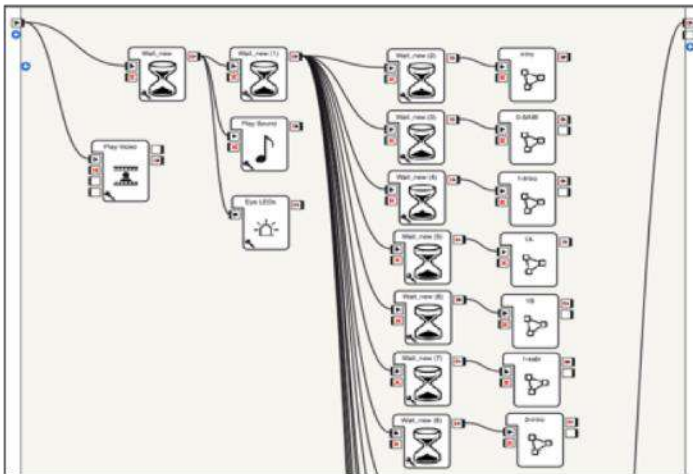
表 7.3-2 Sample7-1

項目	説明
プロジェクト名	sample7-1
ファイルパス	chapter7/sample7-1/sample7-1.pml
概要	[Timeline]ボックスでモーションを作成したサンプルアプリです。モーションのキーフレームの設定方法を確認してください。

7.3.3. モーションとセリフの同期

[タイムライン]パネルの上段の白い帯で作成したモーションに合わせて Pepper を喋らせるには、[Behavior レイヤー]の青い帯を使用します。画像・動画・音楽等とモーションを同期させて、ダンスのようなアプリを作る場合、Wait ボックスを使用して動作しない時間を作ることもできますが、消費メモリが増え、ボックスからボックスに遷移する際にタイムラグが発生する可能性があります。

× 悪い例



タイムラインの動作レイヤーを使用せず、
Waitボックスを多用して“間”を作っている

図 7.3-11 セリフとモーショントメディアの同期

7.3.3.1. Behavior レイヤーの名前変更

[タイムライン]パネルの初期状態では、1本の Behavior レイヤーが存在しています。Behavior レイヤーは名前を付けることができます。左端に"behavior_layer1"と表示されているところをクリックして"Speak"に変更してください。

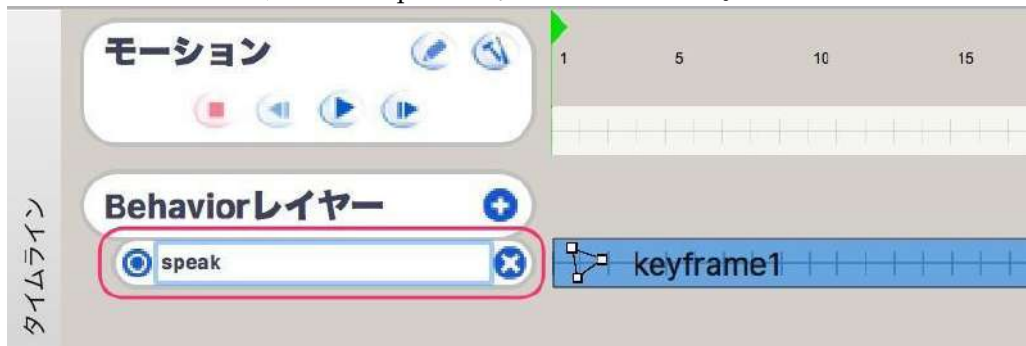


図 7.3-12 Behavior レイヤーの名前変更

7.3.3.2. セリフの追加

腕を前に出すときに「前にならえ」と言わせてみます。Behavior レイヤーの青い帯をクリックして、[フローダイアグラム]パネルに[Say]ボックスを1つ追加します。[フローダイアグラム]パネルの左端にある[onLoad]アイコンと[Say]ボックスを線でつなぎます。

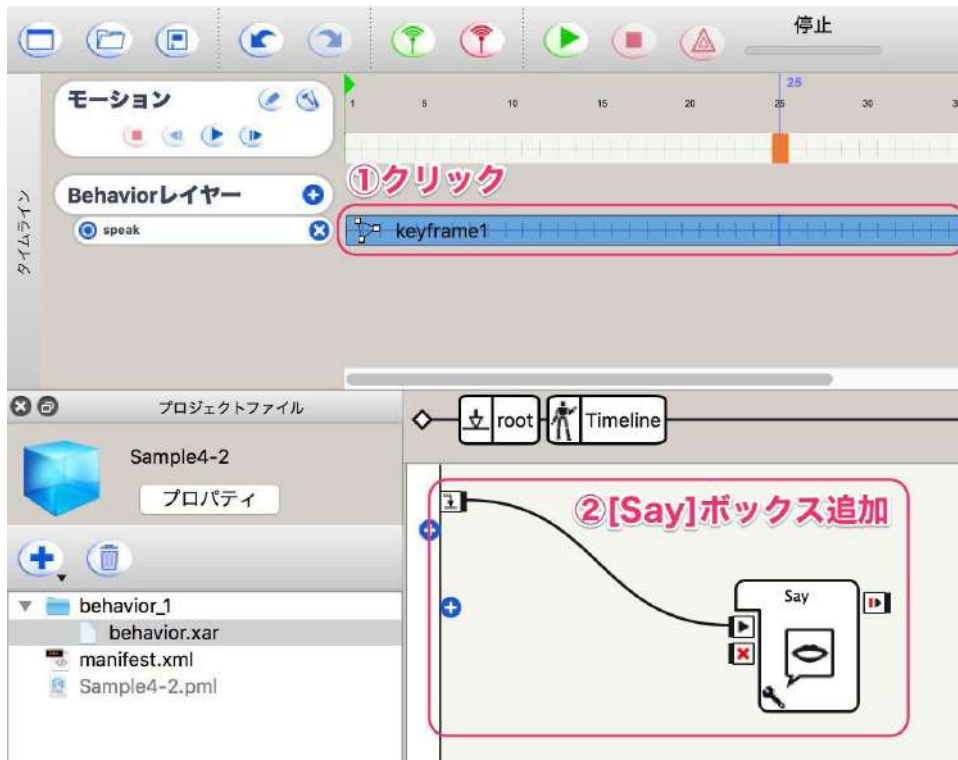


図 7.3-13 Behavior レイヤーに Say ボックスを追加

追加した[Say]ボックスを以下のように設定してください。

表 7.3-3 Say ボックスの設定 1

設定項目	値
パラメータ[Voice shaping]	135
パラメータ[Speed]	110
[Localized Text]ボックスの言語)	Japanese
[Localized Text]ボックスのセリフ	\vct=110\\vol=40\前にいい？ \vct=145\\ vol=100\なラエっ！

7.3.3.3. キーフレームの分割

今度は、腕を下ろすときに「気をつけ」と言わせませす。手を下ろし始めるのは 75 フレーム目からです。あるタイミングからセリフを始めたいときは、Behavior レイヤーを分割します。青い帯の 76 フレーム目を右クリックして[キーフレームの挿入]を選択します。

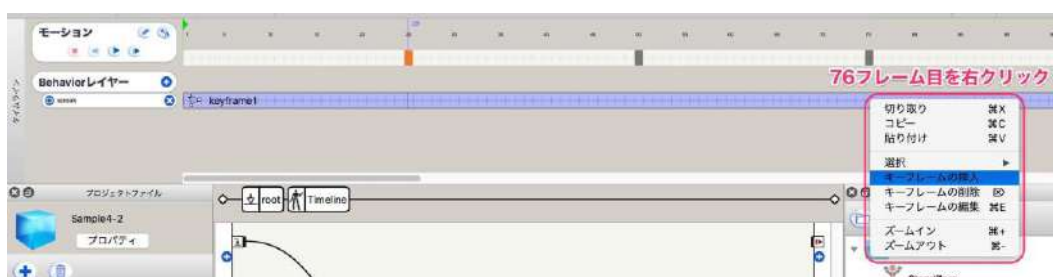


図 7.3-14 Behavior レイヤーにキーフレームを挿入 1

すると、青い帯が分割されて左が紫色、右は青くなり[keyframe75]という名前が付きます。それぞれを Behavior レイヤーのキーフレームと言います。



図 7.3-15 Behavior レイヤーにキーフレームを挿入 2

ここで注意すべきは、[フローダイアグラム]パネルが空になっている点です。先ほど追加した[Say]ボックスはどこに行ったのでしょうか。分割された Behavior レイヤーの左側（紫色の方）をクリックすると、[Say]ボックスが再び現れます。もう一度[keyframe75]と名前の付いた Behavior レイヤーをクリックすると、[Say]ボックスが消えます。[Timeline]ボックスをダブルクリックして[タイムライン]パネルを表示している時の[フローダイアグラム]パネルには、選択されている（青い）Behavior レイヤーのキーフレーム内の処理だけが表示されます。では、[keyframe75]の[フローダイアグラム]パネルに[Say]ボックスを追加して、以下のように設定してください。

表 7.3-4 Say ボックスの設定 2

設定項目	値
パラメータ[Voice shaping]	135
パラメータ[Speed]	110
[Localized Text]ボックスの言語)	Japanese
[Localized Text]ボックスのセリフ	\vct=150\きょうつけっ!\vct=135\

このように Behavior レイヤーのキーフレームを分割することで、任意のタイミングで特定の処理を開始することができます。Behavior レイヤーのキーフレーム分割位置は、マウスカーソルをドラッグすることで移動することができます。

Behavior レイヤーのキーフレームの[フローダイアグラム]パネル右端にある [onStopped]には結線しない

Behavior レイヤーのキーフレームの[フローダイアグラム]パネルの中で、[Say]ボックスの出力から右端の[onStopped]に結線していません。結線すると、そのタイミングで [Timeline]ボックスが終了してしまうので、後続のキーフレーム内の処理が実行されません。

Sample 7-2

表 7.3-5 Sample7-2

項目	説明
プロジェクト名	sample7-2
ファイルパス	chapter7/sample7-2/sample7-2.pml
概要	[Timeline]ボックスでモーションとセリフを同期させるサンプルアプリです。Behavior レイヤーの使い方を確認してください。

7.3.4. モーションとその他の処理の同期

ここまでモーションとセリフを同期させる方法を紹介しましたが、更に Pepper のディスプレイ表示やオムニホイールによる移動を同期させる場合も Behavior レイヤーを使用します。Behavior レイヤーは[+]ボタンをクリックすると増やすことができます。モーション以外の処理は、話す、ディスプレイ表示、移動など種類ごとに Behavior レイヤーを分けて実装すると管理がしやすくなります。



表 7.3-6 モーションとその他の処理の同期

7.3.5. ミラーリング/反転

左右対称のポーズを作成しようとする、微妙に左右がずれてしまいます。それを修正するのは意外と時間がかかります。ミラーリング機能を使用すれば、右（または左）の形を左（または右）に左右対称にコピーしたポーズを自動で作成できます。ミラーリングには、ポーズ作成時にリアルタイムに行う方法と、右（または左）側だけ作成して、後からミラーリングする方法があります。

7.3.5.1. インспекタパネルを使用したミラーリング

[ロボットビュー]パネルで左右どちらかの腕をクリックし、[インспекタ]パネルを表示します。[インспекタ]パネルの下部にある[ミラーリング]チェックボックスをオンにして、を使用して腕の角度を変更すると、反対側が左右対称の形になります。ただし、アニメーションモードで Pepper 実機を使用した場合、ミラーリングは使用できません。



図 7.3-16 インспекタパネルを使用したミラーリング

7.3.5.2. Timeline ボックスを使用したミラーリング

[Timeline]ボックスの[モーション]のタイムライン（白い帯）に、右（または左）だけポーズを作成して登録します。登録したポーズを右クリックして、[ミラーリング] → [選択されたキー] → [右から左]（または[左から右]）を選択すると、左右対称にポーズがコピーされます。[Timeline]ボックスの[モーション]のタイムライン（白い帯）に複数のポーズ（モーションのキーフレーム）が登録されていて、すべてのポーズをミラーリングしたい場合は、[ミラーリング] → [全タイムライン] → [右から左]（または[左から右]）を選択します。

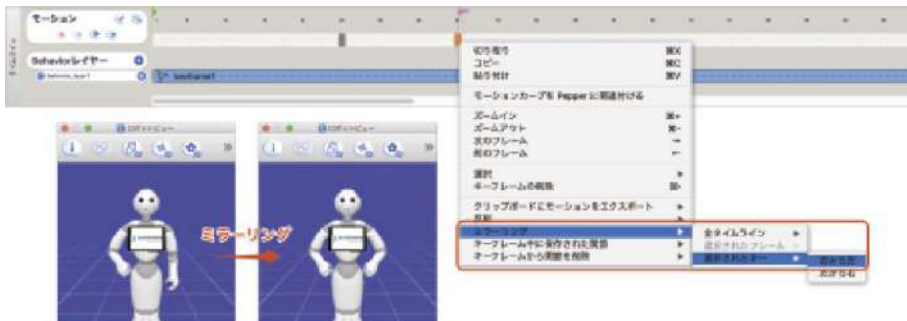


図 7.3-17 Timeline ボックスを使用したミラーリング

7.3.5.3. 反転

反転を使用すれば、左右が鏡写しのようなポーズを自動で作成することができます。

[Timeline] ボックスの[モーション]のタイムライン（白い帯）に、ポーズを作成して登録します。登録したポーズを右クリックして、[反転]→[選択されたキー]を選択すると、反転したポーズがコピーされます。ミラーリングの時と同様に、[反転]→[全タイムライン]を選択すれば、[Timeline] ボックスの[モーション]のタイムライン（白い帯）に登録されているすべてのポーズ（モーションのキーフレーム）が対象になります。

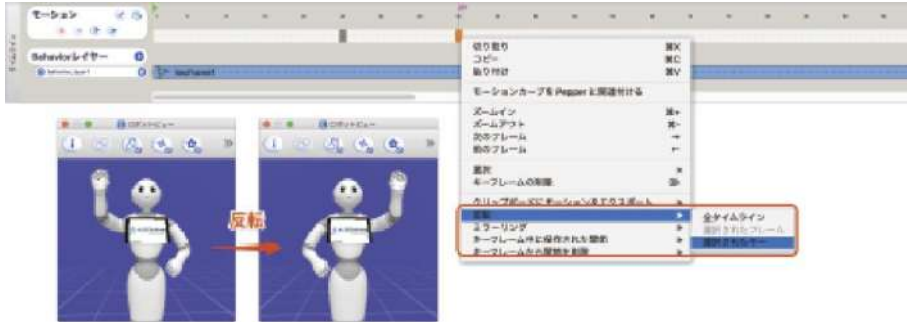


図 7.3-18 反転

7.3.6. モーションのサンプル集

ボックスライブラリとして以下のファイルを配布しているので参考にしてください。これらのライブラリの中には Pepper の代表的なモーション（動き）が含まれています。

- Motion Box Library_イベント・エンタメ用_Ver.2.1.cbl
- Motion Box Library_インタラクション用_Ver.1.1.cbl
- Pepper アトリエ秋葉原のブログで公開されているモーション

ボックスライブラリの URL : <https://www.pepper-atelier-akihabara.jp/archives/259>

各モーションの動画 : <https://www.pepper-atelier-akihabara.jp/archives/280>

7.4. 移動（平面動作）

[Move To]ボックスを使用すると、Pepper を移動できます。設定画面では、X 座標、Y 座標、角度で移動する方向と距離を設定できます。

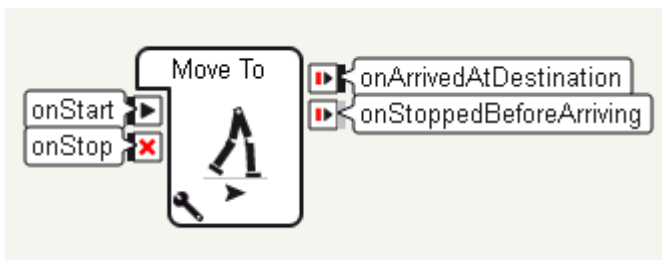


図 7.4-1 Move To ボックス

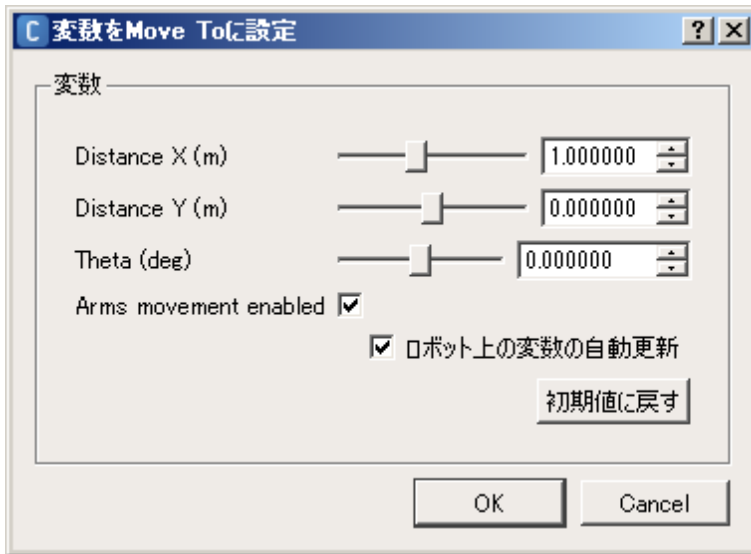


図 7.4-2 Move To ボックスのパラメータ

また、平面動作の作成機能を使用することで自由な軌道を定義することもできます。[プロジェクトファイル]パネルの[+]ボタンをクリックし、[新規プラナームーブ]をクリックして名前を入力すると、プラナームーブエディタが開きます。

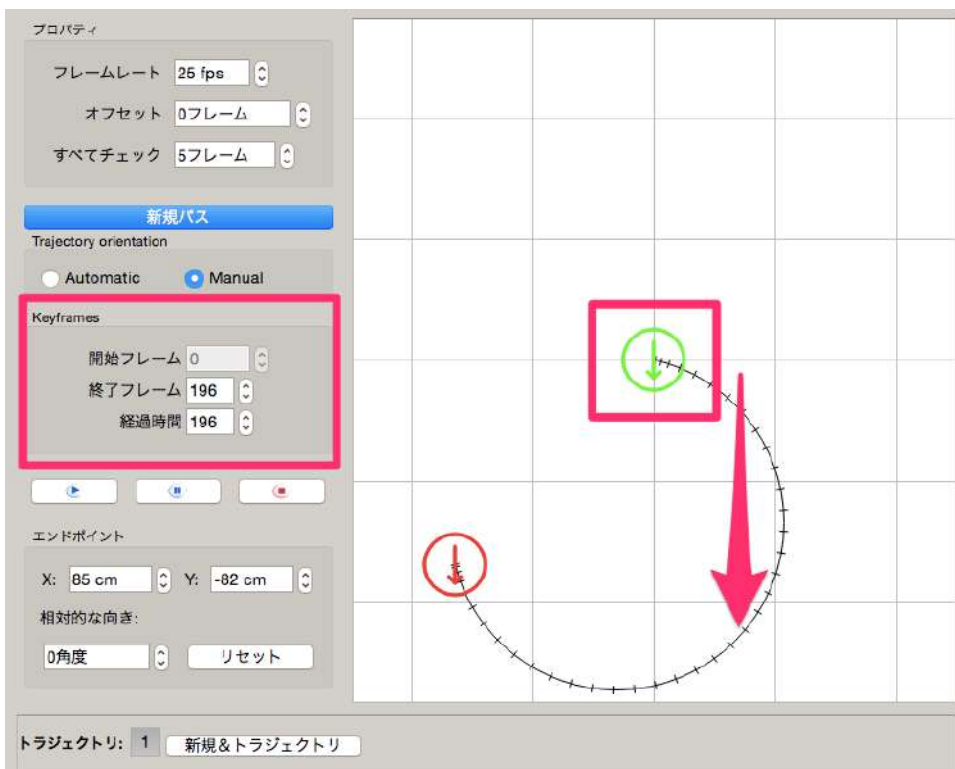


図 7.4-3 プラナームーブエディタ

画面中央の赤色の円をドラッグしてカーソルを移動することで軌道を作成します。始点は緑色の円で示される部分になり、終点は赤色の円で示される部分になります。始点と終点をつなぐ線をドラッグすることで軌道の変更ができます。エディタ左側の Keyframes に開

始フレームや終了フレームを入力して、移動時間を設定できます。作成したプラナームーブは ".pmt" という拡張子のファイルとして保存され、[フローダイアグラム]パネルにドラッグ&ドロップすることで使用できます。

7.5. 会話系ボックスとの連携

ユーザと Pepper の会話の中で独自のモーションを使用する方法を紹介します。これを実現するためには [Animated Say] ボックスの [Text] 変数、および [Dialog] ボックスと連携するトピックファイル内の Pepper のセリフと独自モーションを連携させます。

7.5.1. 新規 Behavior の作成

会話系ボックスと独自モーションを連携させるには、新規に ビヘイビア (behavior.xar) を作成します。

作成する手順は以下の通りです。

1. [プロジェクトファイル]パネルの [+]ボタンの [新規 Behavior...] を選択する
2. [新しい Behavior を追加]ウィンドウの項目を以下のように設定する

表 7.5-1 Behavior の設定値

項目	設定値
名前	任意の名前
ルートボックスタイプ	ダイアグラムまたはタイムライン
Behavior の性質	性質なし

3. [追加] ボタンをクリック



図 7.5-1 新規 Behavior の作成

[ルートボックスタイプ] の値を"ダイアグラム"にした場合、[フローダイアグラム]パネルに [Timeline] ボックスを追加して、

[onStart] と [onStopped] に結線してください。作成したタイムラインで、これまで

紹介してきた方法を用いて独自のモーションを作成します。

ラムの場合

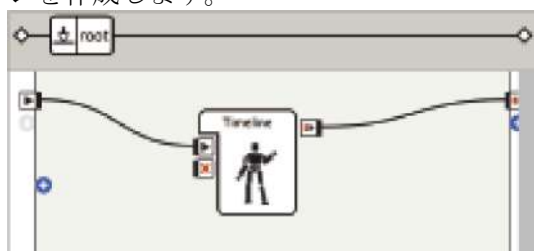


図 7.5-2 ルートボックスタイプがダイアグ

7.5.2. セリフのモーションを組み込む

セリフにモーションを組み込むには、以下のコマンドを使用します。

表 7.5-2 セリフにモーションを組み込むコマンド

コマンド	説明
^mode(モード名)	動作モード切り替える。 モード名：disabled/random/contextual
^start(ビヘイビアパス)	モーションを開始する。
^wait(ビヘイビアパス)	モーションが終了するのを待つ。

上記のコマンドを使用して、セリフにモーションを組み込む書式は以下の通りです。

```
^mode(disabled) ^start(ビヘイビアパス) セリフ ^wait(ビヘイビアパス)
```

図 7.5-3 セリフにモーションを組み込む書式

セリフの後ろに^waitを付けない場合、^startで開始したモーションが終了していても、セリフをしゃべり終えたところでモーションを中断してしまいます。

Sample 7-3

表 7.5-3 Sample7-3

項目	説明
プロジェクト名	sample7-3
ファイルパス	chapter7/sample7-3/sample7-3.pml
概要	[Timeline]ボックスでモーションとセリフを同期させるサンプルアプリです。Behavior レイヤーの使い方を確認してください。

7.6. モーション作成時の注意点

この節では、モーションを作成する際の注意点を紹介します。注意点を守らないと、短時間で関節のモーターがオーバーヒートしたり、転倒する場合があります。

7.6.1. ポーズを1フレーム目に登録しない

[Timeline]ボックスが始まる前に Pepper がどのような姿勢になっているか予想できない場合、[タイムライン]パネルの白い帯の1フレーム目にポーズ（モーションのキーフレーム）を登録すると、急激に姿勢を変化させようとして関節のモーターに高負荷をかけたり、ユーザが驚くことがあるので避けてください。



図 7.6-1 フレーム目にポーズを登録しない

7.6.2. ポーズを連続で登録しない

タイムラインでモーションのキーフレームの間隔を短くすると、動作が激しくなります。モーターがオーバーヒートして Pepper が停止する原因になるので避けてください。

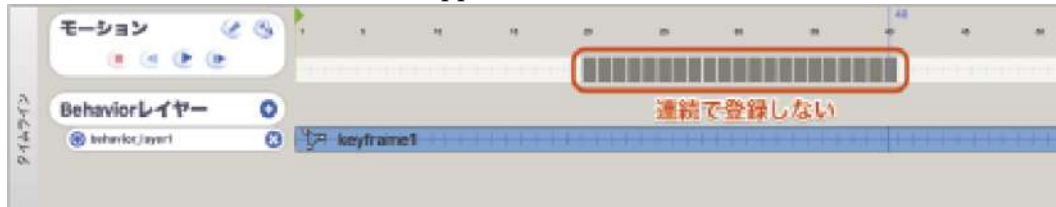


図 7.6-2 ポーズを連続で登録しない

7.6.3. 中途半端な姿勢を維持しない

腕を上げた状態など、中途半端な姿勢を長時間維持すると、肩のモーターが加熱して Pepper が停止します。特にアプリ終了時には、直立姿勢に戻るようプログラムしてください。

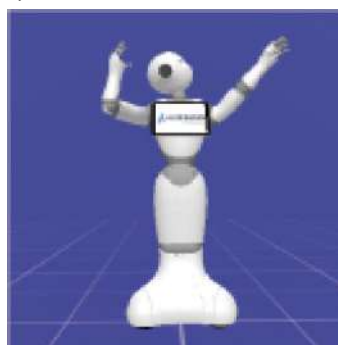


図 7.6-3 中途半端な姿勢

7.6.4. 腰の角度を深くしすぎない

腰の関節の角度を深く曲げすぎると転倒につながります。特に後方、斜め前方は注意が必要です。

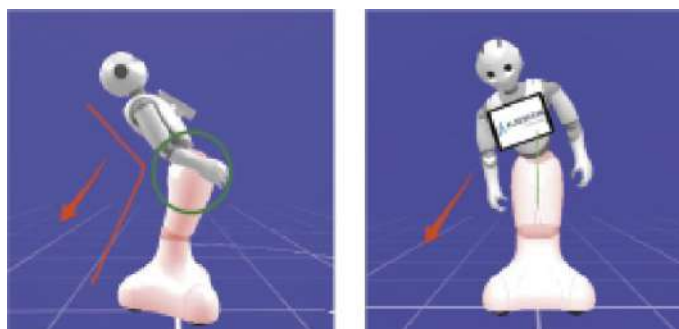


図 7.6-4 腰を深く曲げると転倒につながる

7.6.5. タイムラインは 500 フレーム以内

[Timeline] ボックスのタイムラインは、右にスクロールするといくらでも長くすることができます。しかし、保守性をなどを考慮すると、長すぎるタイムラインは非常に編集しづらくなります。1つの[Timeline]ボックスのタイムラインは 500 フレームを目安にしてください。それ以上長くなる場合は、[Timeline] ボックスを分割して、直列につなげることをお勧めします。

7.6.6. インспекタパネルを確認する

[ロボットビュー]パネルにて Pepper を選択すると、選択した場所のモーションの詳細がインспекタパネルに表示されます。インспекタパネルでモーションを設定した際に、スライダーの▼が青の場合は、設定した数値が有効化されていません。F8 キーを押して▼を緑色にして有効化してください。

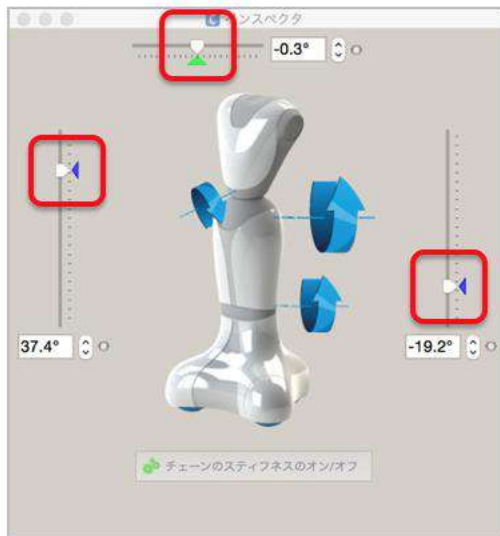


図 7.6-5 インスペクタパネルのパラメータ

インスペクタパネルにてよく発生する問題は以下の通りです。

表 7.6-1 モーション作成時のインスペクタパネルのトラブル

内容	対策
▼が緑色にならない	障害物が近くにあり、ロボットの接触回避機構（セイフティー）が動作していることが考えられます。ロボットの近くにある障害物を避けるもしくは、接触回避機構（セイフティー）を無効化してください。
F8 キーを押すと値がずれてしまう	ロボットのバランスが悪いため自動調整が行われます。そのままの数値を利用してください。

またロボット本体に Choregraphe を接続した状態で、[インスペクタ]パネルの数値を急減に変化させると危険です。

7.6.7. センサーの死角や予測できない利用者の動き

ロボットには接触回避機構（セイフティー）が搭載されていますが、あらゆる場面において接触を回避できるわけではありません。予測できない利用者の動き（子どもなど）、ロボットセンサーの死角での動作は、接触回避機構では十分にカバーできません。リスク回避の観点からスピードを落としてください。

以下のようなモーションは特に注意が必要です。

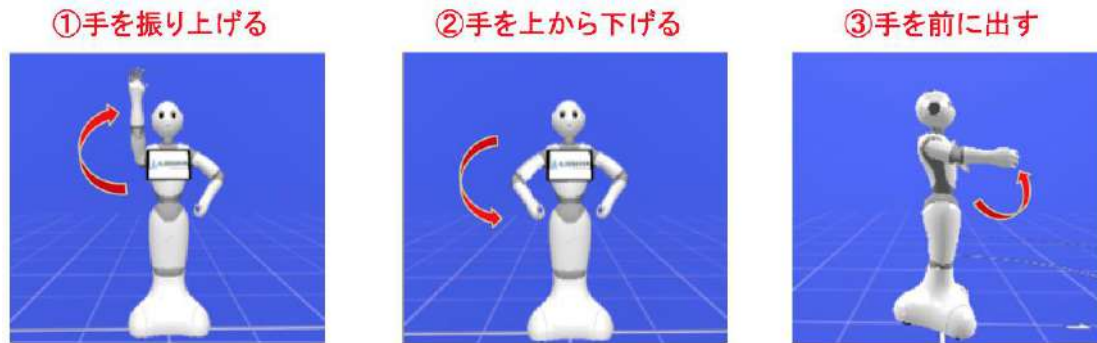


図 7.6-6 特に注意が必要なモーション

7.6.8. 人と目を合わせる

人と人同士が会話を行うときに、顔を見ながら話をしないとお互いに不自然な印象を受けてしまいます。ロボットが人と対話するロボアプリでも、ロボットの目線が利用者の顔を追わないと不自然な印象を利用者が受けてしまいます。[Basic Awareness]ボックスを利用し、利用者の顔を追いながら会話するようにします。

7.6.9. 開発時

- 開発時（Choregraphe の編集のみで動作を確認しない時）はレストモードにしておきます。通常の直立状態でも負荷がかかっています。
- 動作を確認するときは広いところで動作させます。障害物が近くにあると、Safety機能が働き、通常の動きと異なる結果になる可能性があります。
- アプリ内では、なるべく基本姿勢に戻すようにします。

7.6.10. 独自モーションとの競合回避

ロボアプリは原則として、オートノマスライフを起動した状態で起動させます。その場合、Basic Awareness と Breathing が発動したままの状態となり、ロボアプリの演出と競合してしまいます。状況によっては Basic Awareness や Breathing を一時的に停止することを検討してください。また、Pepper に発話をさせる前には必ず音による反応は解除してください。そうしないと、Pepper 自身の声に反応してしまいます。

8. イベント

Pepper の周囲の状況変化や、アプリ内の状態変化などを検知してアプリに通知してくれる仕組みです。この章では、NAOqi に標準で用意されているイベントの扱い方を解説します。

8.1. イベントを検知する方法

NAOqi には標準で多くのイベントが登録されています。例えば「頭が触られた」とか、「人を発見した」などです。アプリの中でイベントを検知して、イベントが発生したタイミングで何かしらの処理を実行するには、[フローダイアグラム]パネルの左端にイベントアイコンを作成します。イベントアイコンの作成手順は以下の通りです。

1. [フローダイアグラム]パネル左端にある背景が青い[+]ボタン（ALMemory からの

- イベントの追加) をクリックする
2. [メモリエventの選択]ウィンドウから使用したいイベントのチェックボックスをオンにする
3. [OK]ボタンをクリックする
4. [フローダイアグラム]パネル左端にイベントアイコンが作成される

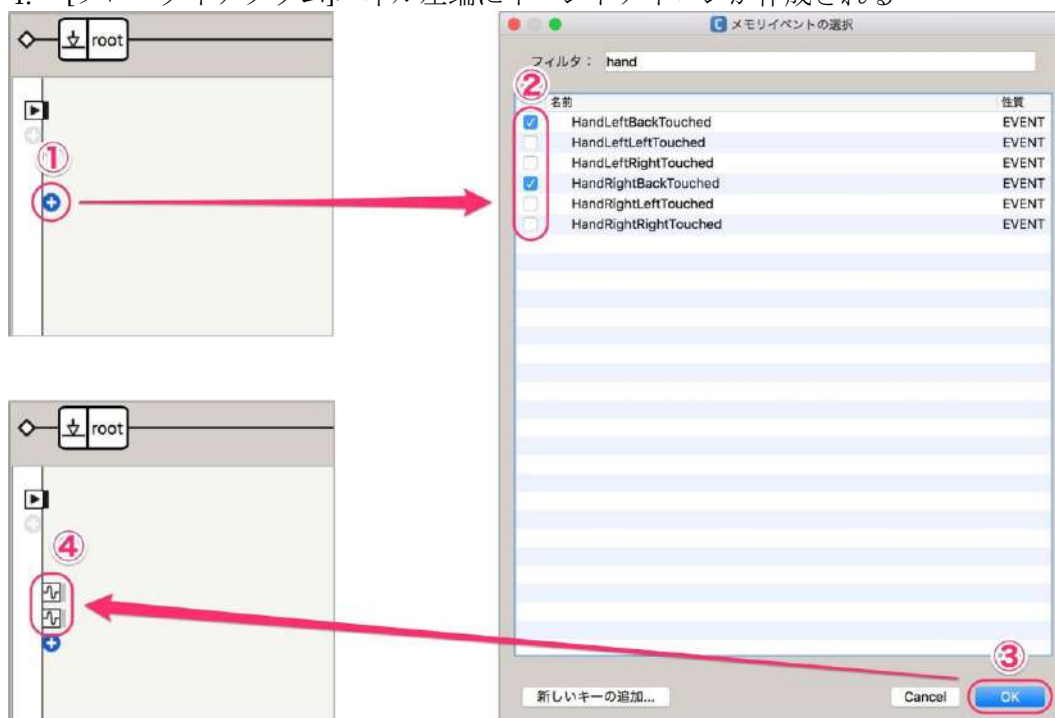


図 8.1-1 イベントアイコン作成

作成されたイベントアイコンに対応するイベントが発生すると、そのアイコンから処理が始まります。イベントアイコンからイベントに対応した処理を実装したボックスに結線してください。

8.2. タッチ系イベント

Pepper には頭に 3 つ、左右の手の甲に 1 つずつ、足元のバンパーに 3 つ、ユーザによるタッチを検知するセンサーがあります。それぞれのセンサーがタッチされた時に発生するイベントは以下の通りです。

表 8.2-1 タッチ系イベント

部位	イベント名	説明
頭	FrontTactilTouched	頭の前方。
	MiddleTactilTouched	頭の天辺。
	RearTactilTouched	頭の後方。
腕	HandLeftBackTouched	左手の甲。
	HandRightBackTouched	右手の甲。
バンパー	LeftBumperPressed	左前バンパー。
	RightBumperPressed	右前バンパー。
	BackBumperPressed	後方バンパー。

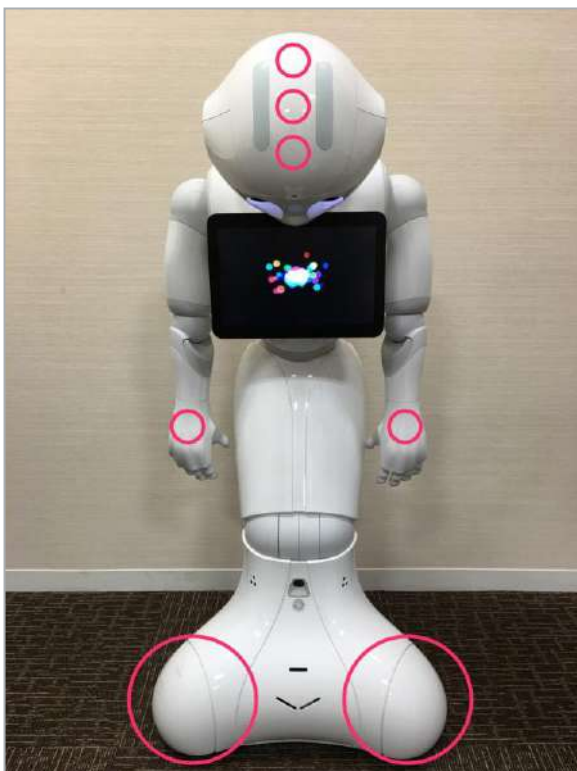


図 8.2-1 Pepper のセンサーの位置

タッチ系イベントアイコンから直接結線されたボックスは 2 回動作します。ボックスを 1 回だけ実行したい場合は、イベントアイコンと実行したい処理を含むボックスの間に [If] ボックスや [Only Once] ボックスを挟むなどの処置が必要です。

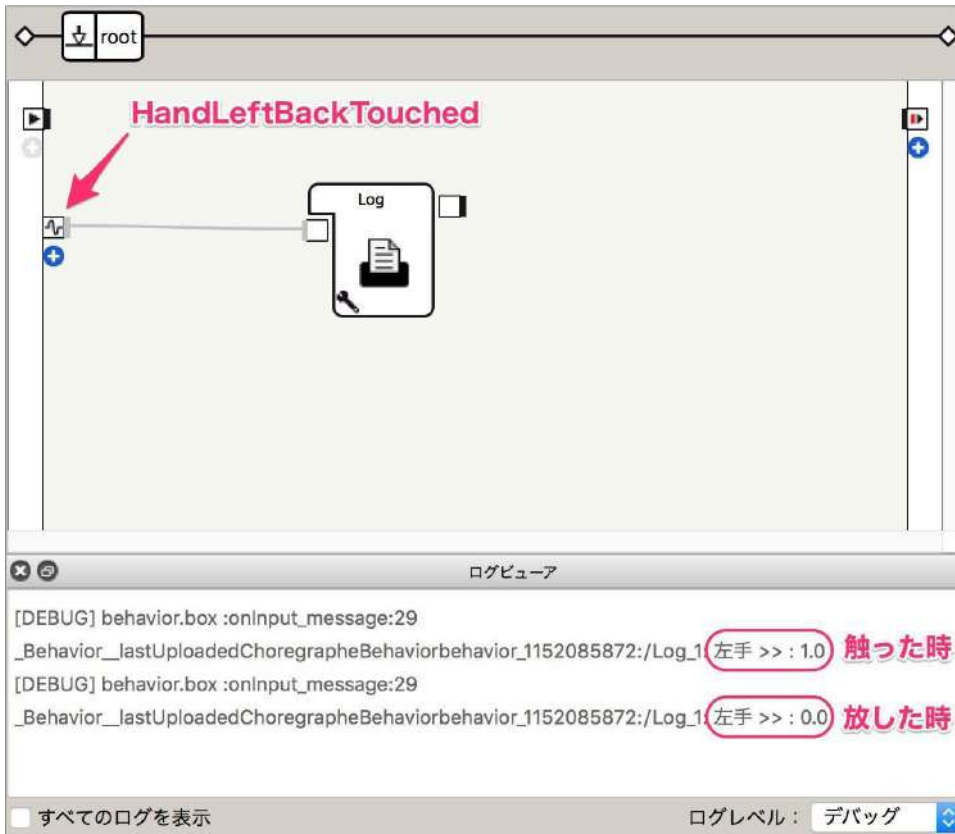


図 8.2-2 タッチ系イベントは 1 度に 2 回発生する

Sample 8-1

表 8.2-2 Sample8-1

項目	説明
プロジェクト名	sample8-1
ファイルパス	chapter8/sample8-1/sample8-1.pml
概要	手の甲のセンサーを使用したサンプルアプリです。左右の手の甲に触れた時に Pepper が喋ります。触れた時と離れた時に異なるセリフを発話します。左手に触れるとアプリは終了します。

8.3. 対人系イベント

Pepper は額のカメラや目の 3D センサーで人を認識したり、距離を測定したりします。人が近づいたり、離れたりとすると以下の様なイベントが発生します。

表 8.3-1 主な対人系イベント

種類	イベント名	説明
EngagementZones	PersonEnterdZone1	ゾーン 1 に人が入った時。
	PersonEnterdZone2	ゾーン 2 に人が入った時。
	PersonEnterdZone3	ゾーン 3 に人が入った時。
PeoplePerception	JustArrived	新しい人を見つけて ID を採番し、リストに追加した時。
	JustLeft	リストに登録されていた人がいなくなり、リストから削除された時。

表.5-2 の実際のイベント名は「イベントの種類/イベント名」になります。「ゾーン 1 に人が入った時」のイベント名は「EngagementZones/PersonEnterdZone1」となります。ゾーン 1~3 の範囲は以下のようになっています。

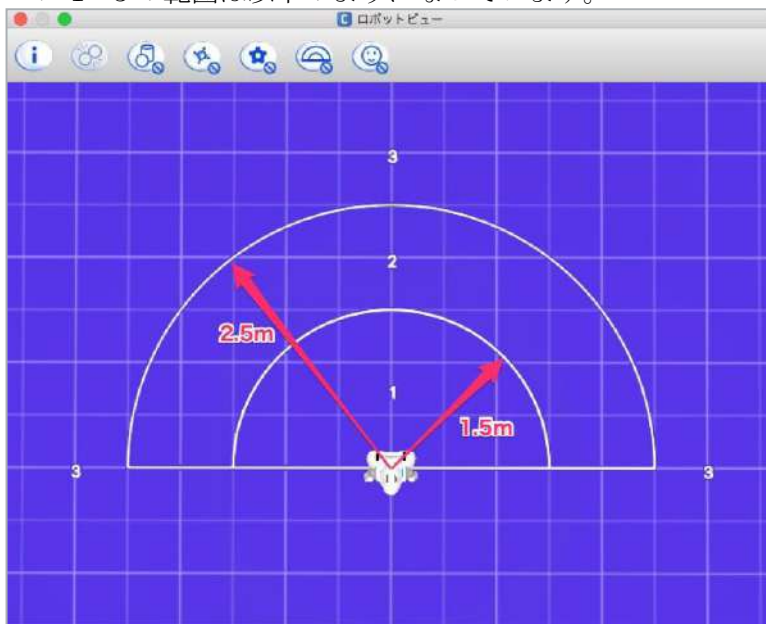


図 8.3-1 Pepper のゾーン

他にも "EngagementZones" や "PeoplePerception" で検索すると、多くの対人系イベントが見つかるので、色々試してください。

Sample 8-2

表 8.3-2 Sample8-2

項目	説明
プロジェクト名	sample8-2
ファイルパス	chapter8/sample8-2/sample8-2.pml
概要	3D センサーを使用したサンプルアプリです。ゾーン 1 に人が侵入すると Pepper が発話します。

9. ディスプレイ

Pepper の特徴の 1 つは、胸にディスプレイが付いている点です。ロボアプリのメインユーザーインターフェイスは会話です。ユーザと Pepper のすべての対話をディスプレイで行うアプリは、Pepper をプラットフォームとする必要がありません。この章では、ディスプレイと Pepper を連携させる方法を解説します。

9.1. ディスプレイ表示の基本的な仕組み

Pepper 本体には Web サーバが動作しており、ディスプレイに画像を表示する場合は、ディスプレイ側の WebView アプリを起動し、WebView アプリが Pepper 内の Web サーバにアクセスし、指定画像のダウンロードと表示を行います。ロボットウェブページがパソコンのブラウザで開けるのも同様の仕組みです。

9.2. 画像表示

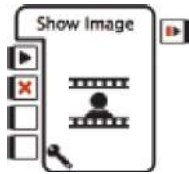


図 9.2-1 Show Image ボックス

Pepper のディスプレイいっぱい画像を表示するには、[Show Image] ボックスを使用します。ディスプレイの解像度に合わせて、表示する画像は 962 (幅) x 601 (高) ピクセルで作成してください。

9.2.1. 画像の指定

[Show Image] ボックスで表示する画像は、プロジェクトのトップレベルに [html] フォルダを作成し、その配下に配置する必要があります。画像のフォーマットは JPEG または PNG を使用してください。[Show Image] ボックスの [ImageUrl] 変数に、[html] フォルダ以下の画像ファイルのパスを設定します。例えば、[html] フォルダ → [img] フォルダ → [pepper.jpg] ファイルという構成の場合、[ImageUrl] 変数に設定する値は "img/pepper.jpg" になります。



図 9.2-2 Show Image ボックスの変数

9.2.2. 入出力

[Show Image] ボックスには 4 つの入力と、1 つの出力があります。それぞれの用途は以下の通りです。

表 9.2-1 Show Image ボックスの入出力

種類	名前	説明
入力	onStart	画像を表示する。ただし、ボックスは終了しない。
	onStop	ボックスを終了する。
	onHideImage	画像を非表示にする。ただし、ボックスは終了しない。
	onPreLoadImage	画像を表示する前に、画像を読み込んでおく。
出力	onStopped	入力[onStop]によってボックスが終了し、あとの処理に進む。

アプリ終了時に、入力 [onHideImage] に結線して画像を非表示にしてください。

Sample 9-1

表 9.2-2 Sample9-1

項目	説明
プロジェクト名	sample9-1
ファイルパス	chapter9/sample9-1/sample9-1.pml
概要	[Show Image]ボックスを使用して画像を表示するサンプルアプリです。10 秒間画像を表示した後、アプリ終了時に画像を非表示にしています。

9.2.3. 画像表示時の注意点

画像の撮影と表示をするアプリを作り、複数回実行すると、ディスプレイの Web ビューが画像をキャッシュして更新されない場合があります。その場合は、一般的なウェブブラウザでの回避方法と同じく、ファイル名の後に ? を付け、その後ろに乱数や現在時刻を付けたものを URI にします。そうすることで、直前に撮影したものとは異なるファイルとして認識されるようになります。また、NAOqi OS 2.3 からは、showImageNoCache というメソッドが用意されているので、これを使用するとキャッシュに残りません。

9.3. タッチ位置の座標取得

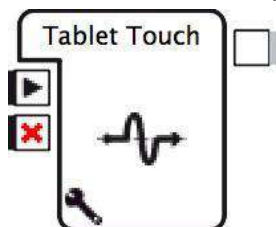


図 9.3-1 Tablet Touch ボックス

ユーザがディスプレイにタッチした時の座標を取得するには、[Tablet Touch]ボックスを使用します。

[Tablet Touch]ボックスは、タッチした位置の X 座標と Y 座標を 2 つの要素の配列として [onTouched] から出力します。

[Tablet Touch]ボックスの解像度は 1,280 x 800 ピクセルです。

[Tablet Touch]ボックスの変数[Action]は、座標を取得するタイミングを選択します。設定値の意味は以下の通りです。

表 9.3-1 Action 変数の設定値

設定値	説明
On touch move	ディスプレイ上で指が移動した時。（初期値）
On touch down	ディスプレイに指が触れた時。
On touch up	ディスプレイから指が離れた時。

Sample 9-2

表 9.3-2 Sample9-2

項目	説明
プロジェクト名	sample9-2
ファイルパス	chapter9/sample9-2/sample9-2.pml
概要	[Tablet Touch]ボックスを使用したサンプルアプリです。ディスプレイをタッチすると、その座標を Pepper が喋ります。 [Tablet Touch]ボックスの変数[Action]の設定値を"On touch down"に変更してあります。このアプリは連続して実行できるようにあえて終了しないように作成してありますので、Choregraphe のツールバーの[停止]ボタンをクリックして終了させてください。

9.4. HTML&CSS&JavaScript の使用

[Tablet Touch] ボックスを使用して、タッチされた座標から処理を分岐すると、ディスプレイ内のレイアウトが変更された場合、プログラムの大幅な修正が必要になります。ボタンやその他の UI 要素を配置して、ユーザに操作してもらうアプリの場合、ディスプレイ内のレイアウトは Web の技術を使用することをお勧めします。この節では、HTML&CSS&JavaScript を使用する場合のプロジェクトの構成について紹介します。

9.4.1. プロジェクトの構成

HTML&CSS&JavaScript を使用する場合、プロジェクトの構成を以下のようにしてください。

- html フォルダをプロジェクトのトップレベルに作成する
- html フォルダの直下に index.html ファイルを作成する
- CSS と JavaScript のファイルは html フォルダ配下に置く



図 9.4-1 HTML&CSS&JavaScript 使用時のプロジェクトの構成

9.4.2. HTML ファイルの表示/非表示

index.html ファイルを Pepper のディスプレイに表示するには [Show App] ボックスを使用します。また、Pepper のディスプレイに表示した内容を消すには [Hide Web View] ボックスを使用します。

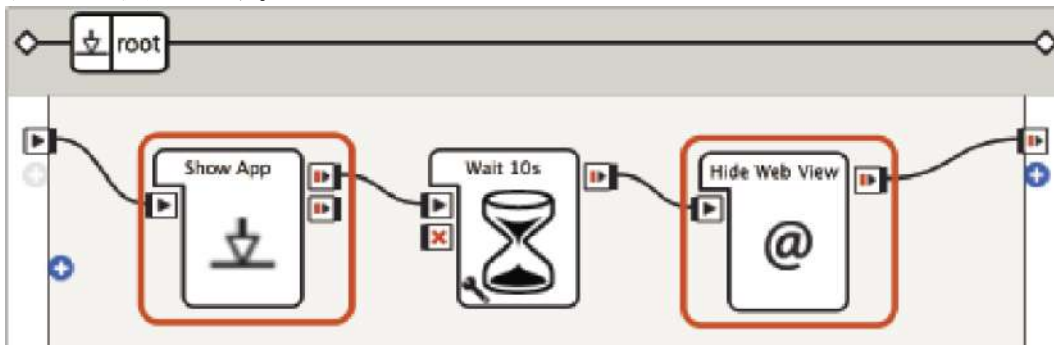


図 9.4-2 HTML ファイルの表示 / 非表示

Sample 9-3

表 9.4-1 Sample9-3

項目	説明
プロジェクト名	sample9-3
ファイルパス	chapter9/sample9-3/sample9-3.pml
概要	HTML ファイルを表示するサンプルアプリです。index.html ファイルを表示して 10 秒経過したら消して、アプリを終了します。プロジェクトのファイル構成、[Show App]、[Hide Web View]ボックスの使い方を確認してください。

9.5. Pepper とディスプレイの連携

ディスプレイを用いたアプリとして以下の様な処理を行うことがよくあります。

- ディスプレイの UI 要素を操作した結果、Pepper にリアクションさせる。
- Pepper に話しかけた言葉によってディスプレイの表示を変更する。
- Pepper のカメラで撮影した画像をディスプレイに表示する。

上記のような処理を実装する場合、Pepper とディスプレイの間で、適切なタイミングでデータを転送する必要があります。この節では Pepper とディスプレイを連携させる方法を紹介します。

9.5.1. ディスプレイ → Pepper

ディスプレイから Pepper へデータを転送するには、QiMessaging JavaScript ライブラリを使用します。

9.5.1.1. ライブラリの読み込み

QiMessaging JavaScript を使用するには、HTML ファイルの中で以下のタグを記述してライブラリを読み込みます。

```
<script src="/libs/qimessaging/2/qimessaging.js"></script>
```

図 9.5-1 QiMessaging Javascript の読み込み

9.5.1.2. 独自イベントの作成

Web アプリから ロボアプリへデータを転送する時、転送することをイベントとして通知して、同時にデータを転送します。イベントは NAOqi 標準のものではなく、アプリ独自のイベントを作成します。イベントを作成する手順は以下の通りです。

1. [フローダイアグラム]パネルの[ALMemory からのイベントの追加]ボタンをクリックする
2. [メモリイベントの選択]ウィンドウ左下の[新しいキーの追加...]ボタンをクリックする
3. [新しいメモリキー]ウィンドウの[新しいメモリキーの名前]に任意のイベント名を入力して[OK]ボタンをクリックする
4. [フローダイアグラム]パネルに独自のイベントアイコンが作成される

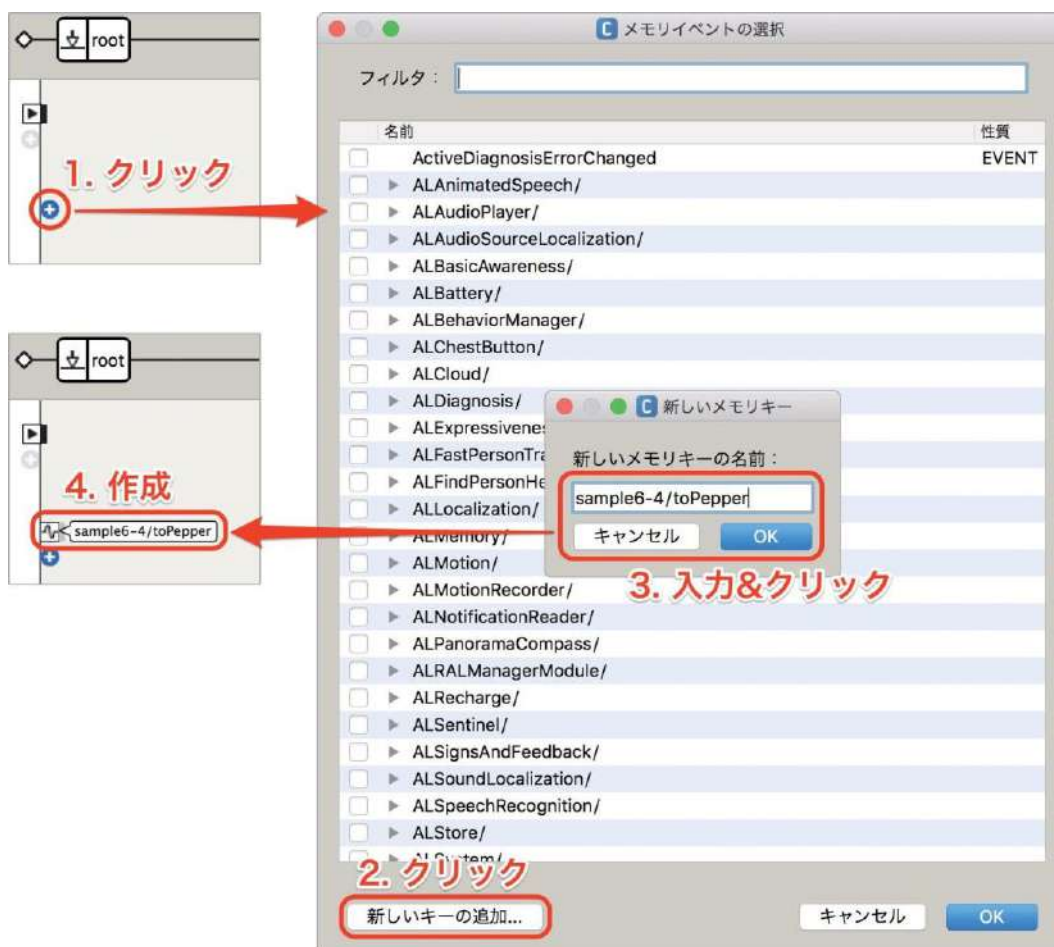


図 9.5-2 独自のイベントの作成

任意のイベント名は、途中に「/」を入れると階層構造で管理できます。

9.5.1.3. イベントの発生とデータ転送

ディスプレイから Pepper にイベントを通してデータを送るには、以下の JavaScript コードを記述します。

```

01: // QiSession オブジェクトの作成
02: var session = new QiSession();
03:
04: // Pepper にデータ送信
05: session.service("ALMemory").then(function(ALMemory) {
06:     ALMemory.raiseEvent(" 独自イベント名 ", 転送するデータ ); 07: });
    
```

図 9.5-3 イベントの発生とデータ転送

6 行目の `ALMemory.raiseEvent` 関数の第 1 引数に設定しておいた入力したイベント名を指定します。また、第 2 引数に Pepper へ転送したいデータを指定します。複数のデータを転送したい場合は、配列を使用します。Pepper 側では、イベントアイコンから実行したい処理のボックスへ結線します。JavaScript からデータが転送されて来た場合、イベントアイコンからそのデータがボックスへ渡されます。

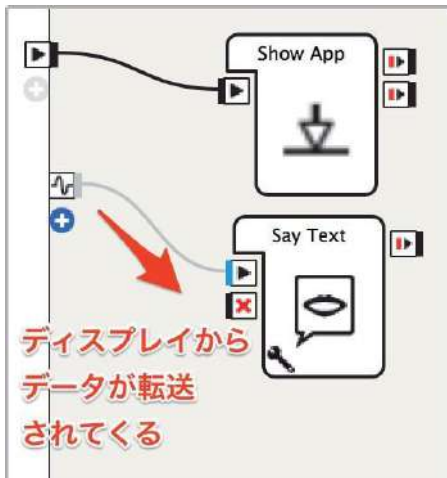


図 9.5-4 ディスプレイから転送されたデータ取得

onClick イベントは使用しない

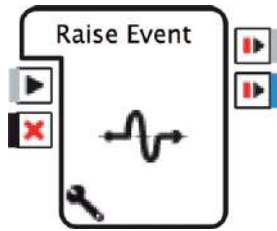
Web アプリでボタンがクリックされたことを検知するためのイベントとしてよく使用されるのは `onClick` です。 `onClick` イベントは、その後連続でクリック (ダブルクリック) されないかを判定するために少し間が空きます。 `onClick` イベントを **Pepper** のディスプレイで使用すると、反応速度が遅く感じられることがあります。その場合は、タッチデバイス用のイベントである `touchstart` や `touchend` を使用すべきです。

Sample 9-4

表 9.5-1 Sample9-4

項目	説明
プロジェクト名	sample9-4
ファイルパス	chapter9/sample9-4/sample9-4.pml
概要	ディスプレイから Pepper へデータを転送するサンプルアプリです。ボタンをタッチすると、 Pepper がどのボタンが押されたかを喋ります。5回タッチするとアプリが終了します。独自イベントの登録方法、ディスプレイからのデータ送信、 Pepper 側のデータ受信方法を確認してください。

9.5.2. Pepper → ディスプレイ



ロボアプリからディスプレイへデータを転送するには、しかるべきタイミングで [Raise Event] ボックスを使用します。

図 9.5-5 Raise Event ボックス

9.5.2.1. Raise Event ボックスによるイベントの発生とデータ転送

ディスプレイへデータを転送する際の [Raise Event] ボックスの使用方法は以下の通りです。

1. 変数 [key] に独自のイベント名を設定する。
2. 入力 [onStart] にディスプレイへ転送したいデータを渡す。

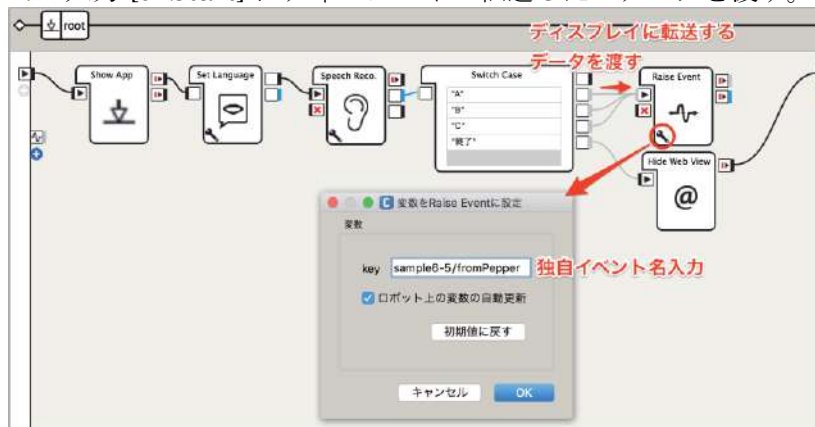


図 9.5-6 Raise Event ボックスの使用例

9.5.2.2. ディスプレイ側のイベント監視とデータ受信

ディスプレイ側で Pepper から発信されるイベントを監視して、イベント発生時にデータを受信するには、QiMessaging JavaScript ライブラリを読み込んだ上で、以下のソースコードを記述します。

```
01: // QiSession オブジェクトの作成
02: var session = new QiSession();
03:
04: // ALMemory のイベント監視
05: function startSubscribe() {
06:   session.service("ALMemory").then(function (ALMemory) {
07:     ALMemory.subscriber(" 独自イベント名").then(
08:       function(subscriber) {
09:         subscriber.signal.connect(toTabletHandler);
10:       });
11:   });
12: }
13:
14: // ディスプレイに処理を反映
15: function toTabletHandler(value) {
16:   document.getElementById("pepper").innerHTML = value;
17: }
```

図 9.5-7 ディスプレイ側のイベント監視とデータ受信

7行目の `ALMemory.subscriber` 関数の引数に、[Raise Event] ボックスの変数 [key] に設定したイベント名を指定します。9行目の `subscriber.signal.connect` 関数の引数に、データを受信するためのハンドラ関数名を指定します。上記の例では `toTabletHandler` という関数が14行目に定義してあります。関数名は任意です。15行目のハンドラ関数 `toTabletHandler` の引数 `value` で、[Raise Event] ボックスの入力 [onStart] に渡されたデータを受信することができます。

Sample 9-5

表 9.5-2 Sample9-5

項目	説明
プロジェクト名	sample9-5
ファイルパス	chapter9/sample9-5/sample9-5.pml
概要	Pepper からディスプレイへデータを転送するサンプルアプリです。Pepper に「A」「B」「C」のいずれかを話しかけると、その内容がディスプレイに表示されます。「終了」と話しかけるとアプリが終了します。[Raise Event]ボックスの使い方、ディスプレイでイベントを受信する方法を確認してください。

9.6. 異なるディスプレイの対応

1707px(W)×1067px (H) ディスプレイ向けに開発されたロボアプリを 962px(W)×601px(H)ディスプレイ搭載の Pepper (18a)で動作させると、画像が拡大されて表示され、画像全体の 4 分の 1 程度のみが表示されてしまいます。1707px(W)×1067px (H) と 962px(W)×601px(H)の両方で正しく画像表示されるようにするための対応が必要です。

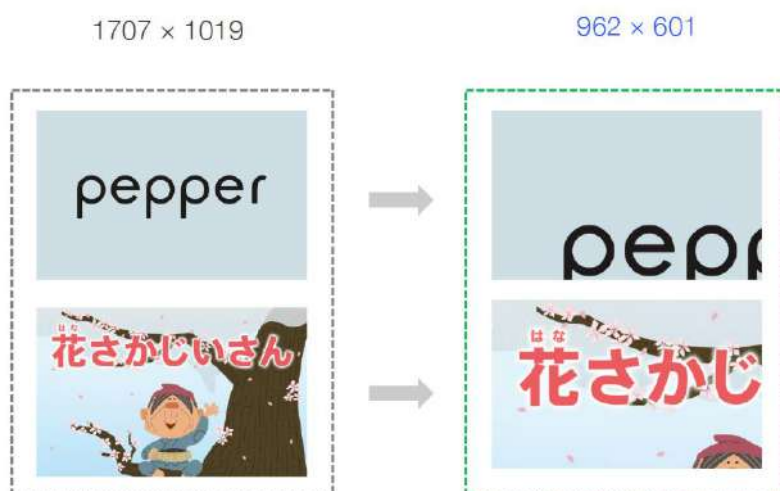


図 9.6-1 旧型向けに開発されたロボアプリを新型ディスプレイで動作させたときの見え方

9.6.1. ロボアプリの実装方法ごとの対応方法

2種類のディスプレイで正しく表示できるようにする対応は、ロボアプリの実装方法により対応方法が異なります。

9.6.1.1. viewport が 1.335 に設定されている

以下のように viewport を 1.335 に設定して開発されているロボアプリは adjust.js を読み込むことで、両方のディスプレイに対応することができます。

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

図 9.6-2 viewport が 1.335 に設定されている場合

9.6.1.2. viewport の指定がない、もしくは 1 が設定されている
 viewport の指定がない場合や以下のように viewport を 1 に設定して開発されているロボアプリは adjust23.js を読み込むことで、両方のディスプレイに対応することができます。

```
<meta name="viewport"
content="initial-scale=1, minimum-scale=1, maximum-scale=1" />
```

図 9.6-3 viewport の指定がない、もしくは 1 に設定されている場合

9.6.1.3. 画面表示使用する HTML がレスポンシブルデザインに対応している

レスポンシブデザイン (表示端末の解像度により表示が切り替わる Web デザイン手法) に対応しているロボアプリは調整の必要がありません。

9.6.2. 新型ディスプレイへの対応手順

Choregraphe プロジェクトの html/script/ ディレクトリにスクリプト (adjust.js もしくは adjust23.js) をインポートします。

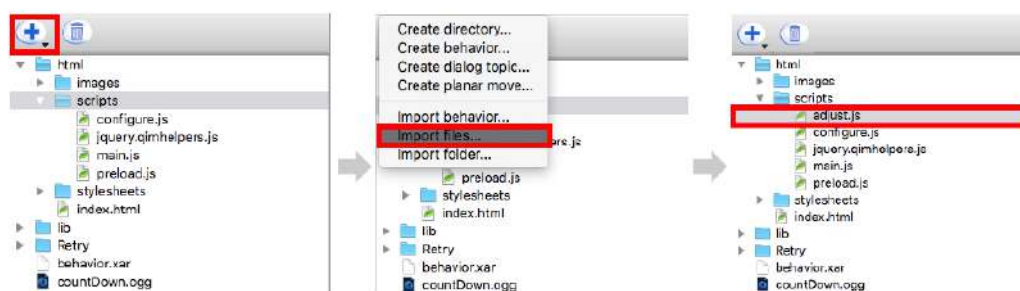


図 9.6-4 新型ディスプレイへの対応手順①

ロボアプリで読み込まれる HTML ファイル (index.html 等) にスクリプト (adjust.js もしくは adjust23.js) をインポートする記述を追加します。

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta name="viewport"
5 content="initial-scale = 1.335,
6 minimum-scale = 1.335,
7 maximum-scale = 1.335" />
8
9 <link rel="stylesheet" href="stylesheets/home.css" />
10
11 <script src="/libs/qimessaging/1.0/jquery.min.js"></script>
12 <script src="/libs/qimessaging/1.0/socket.io.min.js"></script>
13 <script src="/libs/qimessaging/1.0/qimessaging.js"></script>
14 <script src="scripts/jquery.qimhelpers.js"></script>
15
16 <script src="scripts/adjust.js"></script>
17
18 <script src="scripts/preload.js"></script>
19 <script src="scripts/main.js"></script>
20 </head>
```

図 9.6-5 新型ディスプレイへの対応手順②

黄色の部分 jquery.min.js と jquery.qimhelpers.js の後にスクリプト (adjust.js もしくは adjust23.js) をインポートする記述を追加します。

9.6.3. 例外ケース

9.6.3.1. CSS 側での制御があるロボアプリ

CSS 側で viewport の制御を行っているロボアプリは該当行を削除してください。

```
#container {
  display: none;
}

#container {
transform: scale(1.335,1.335);
  transform-origin: left top;
}
```

図 9.6-6 transform:scale()による viewport の変更

9.6.3.2. jQuery ライブラリを利用していないロボアプリ

jQuery ライブラリを利用していないロボアプリは次の記述を追加し jQuery ライブラリを読み込んでください。

```
<script src="/libs/qimessaging/1.0/jquery.min.js"></script>
```

図 9.6-7 jQuery へのリンク

9.6.3.3. 複数の HTML より構成されているロボアプリ

構成される全ての HTML に対して同様の処理を行ってください。

9.6.4. スクリプトの内容

9.6.4.1. adjust.js

```
$(function() {
  viewport = document.querySelector("meta[name=viewport]");
  if (viewport != null) {
    var legacyWidth = 1280;
    var windowWidth = window.screen.width;
    var scale = (windowWidth/legacyWidth).toFixed(3);
    init_str = "initial-scale=".concat(scale.toString());
    min_str = "minimum-scale=".concat(scale.toString());
    max_str = "maximum-scale=".concat(scale.toString());
    viewport.setAttribute("content",init_str.concat(",").concat(min_str).concat(",").concat(max_str));
  }
});
```

図 9.6-8 adjust.js の内容

9.6.4.2. adjust23.js

```
$(function() {
  viewport = document.querySelector("meta[name=viewport]");
  if (viewport != null) {
    var legacyWidth = 1707;
```

```

var windowWidth = window.screen.width;
var scale = (windowWidth/legacyWidth).toFixed(3);
init_str = "initial-scale=".concat(scale.toString());
min_str = "minimum-scale=".concat(scale.toString());
max_str = "maximum-scale=".concat(scale.toString());
viewport.setAttribute("content",
init_str.concat(",").concat(min_str).concat(",").concat(max_str));
}
});

```

図 9.6-9 adjust23.js の内容

9.7. ディスプレイ表示の注意点

ディスプレイを使用するロボアプリは、以下の点に注意して作成するように心がけてください。

9.7.1. 字はできるだけ大きくする

Pepper のディスプレイは、ユーザが立った状態で見える可能性が高いため、一般的なタブレットアプリよりディスプレイと目の距離が遠くなります。そのため、ディスプレイに表示する文字はデザインが破綻しない範囲で、できるだけ大きくすべきです。

9.7.2. UI 要素の配置が上下左右に偏らないようにする

文字、ボタンなどの UI 要素は可能な限り大きく配置して、上下左右に偏ったり、無意味な空間が存在しないようにしてください。

9.7.3. 文字と背景のコントラスト比はできるだけ高くする

文字と背景のコントラスト比が低いと、色弱者には判断できないことがあります。できるだけコントラスト比が高くなるような色使いを心がけてください。

9.7.4. ボタン連打に対応する

ディスプレイと Pepper を連携する場合、イベント発生とデータ転送のために少し時間がかかります。その間にユーザがディスプレイ内のボタンを連打すると、同じイベントが複数回発生し、同じ処理を連続で実行してしまうことがあります。

それを回避するため、ボタンは短時間で連打されても 1 回しか反応しないように設計および実装してください。

9.7.5. タッチした時に表示が上下左右にブレないようにする

縦方向（スクロール）や横方向（スワイプ）で表示が上下左右にブレないようにしてください。ユーザの操作ミスなどで表示がブレてしまうと、それを戻すなど余計な操作が必要になってしまいます。

9.7.6. ピンチイン/ピンチアウトで拡大/縮小しないよう

にする

Pepper のデバイスはマルチタッチに対応しています。2 本指を開く（ピンチアウト）または閉じる（ピンチイン）ようにタッチすると、拡大または縮小することができますが、それらを無効にしてください。

9.7.7. アプリ起動中はバブル状態に戻らないようにする

特定のアプリが起動していない時、ディスプレイはバブル状態になります。アプリが起動して終了するまでは、バブル状態にならないようにディスプレイを制御してください。



図 9.7-1 バブル状態

9.7.8. アプリが終了したら表示した内容を消す

アプリ実行中にディスプレイに表示したものは、アプリ終了時に必ず消してバブル状態に戻してください。

9.7.9. 操作は音声とディスプレイ両方でできるようにする

Pepper のメインユーザインターフェイスは会話ですが、環境音などの影響でユーザの言葉を聞き取れないことがあります。どのような環境下でもアプリを途中で止めてしまうことがないように、操作は音声とディスプレイの両方でできるようにしてください。

9.7.10. 画像作成時の注意点

Pepper のディスプレイの解像度は、1707px(W) × 1067px (H) 、もしくは 962px(W) × 601px(H) です。それ以外の解像度でも表示できますが、不必要に大きいサイズの画像はデータ量の肥大化を招き、著しく小さい画像はジャギーが目立ちロボアプリの見栄えが悪くなります。

9.7.11. ディスプレイのタイムアウト処理

ディスプレイ側で発行したイベントが Pepper 側で受信できないことがあります。イベントが受信できないことにより、ロボアプリの進行ができずフリーズしてしまうおそれがあります。ディスプレイからのイベント受信を前提とするロボアプリではタイムアウト処理を設けて、ロボアプリのフリーズを回避してください。



図 9.7-2 ディスプレイのタイムアウト処理①

次に示すフローダイアグラムのように、一定時間待った後もディスプレイからイベントが来ない場合、ロボアプリが進行できるようにします。正常にイベントを受信できた場合はタイムアウトタイマーを停止させてください。

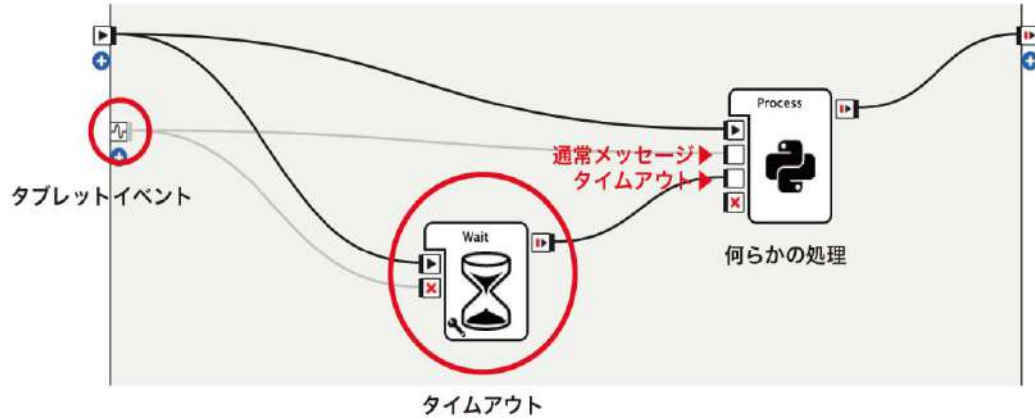


図 9.7-3 ディスプレイのタイムアウト処理②

ディスプレイタイムアウトの実装は後述する『ロボアプリ品質チェックリスト』を満たす上でも必須となります。

10. オートノマスライフ

オートノマスライフは、Pepper が稼働している間、動作し続けるモジュールです。アプリの起動や Pepper を生き物のように見せる動作などを行います。この章では、オートノマスライフの機能やプロジェクトのプロパティについて解説します。

10.1. オートノマスライフとは

Pepper の電源を入れると、オートノマスライフ (Autonomous Life : 以下、A-Life) は自動的に起動します。ロボアプリは A-Life が稼働している状態で正しく動作するように作成しなければなりません。

10.1.1. オートノマスライフの主な機能

A-Life の主な機能は以下の通りです。

- インストールされているアクティビティの自律的な起動
- Basic Awareness と Breathing Animation の自動実行
- ライフサイクル管理
- 安全確保のための反応

10.1.1.1. インストールされているアクティビティの自律的な起動

アクティビティとは、A-Life 上で動作するビヘイビア (behavior.xar) を意味します。アクティビティは、個々に起動条件を持っています。A-Life は起動条件を満たしたアクティビティを起動します。起動条件を満たしたアクティビティが複数ある場合、最も過去の時間に起動されたものを起動します。アクティビティは Solitary と Interactive の 2 種類があります。

10.1.1.2. Basic Awareness と Breathing Animation の自動実行

Basic Awareness と Breathing Animation は、Pepper が生き物のように見える動作を自動的に行います。

主な動作は以下の通りです。

- 人と目を合わせる (Basic Awareness)。
- 音がした方を向く (Basic Awareness)。
- ディ스플레이に触れられた方を向く (Basic Awareness)。
- 呼吸をしているように腕を微妙に動かす (Breathing Animation)。

10.1.1.3. ライフサイクル管理

A-Life は以下の 4 つの状態を遷移します。

表 10.1-1A-Life の状態

状態	説明
Solitary	<p>周囲の状況などを監視し、起動条件を満たした Solitary アクティビティがあれば、アクティビティを起動します。また、起動条件を満たした Interactive アクティビティがある場合は、Interactive 状態へと遷移します。</p> <p>一般販売モデルでは、くしゃみや独り言などの Solitary アクティビティが実行され、トリガーセンテンス、起動トリガー条件、ロボアプリランチャーなどで、ロボアプリが起動されると Interactive 状態へ遷移します。</p> <p>Pepper for Biz モデルでは、呼び込み用の Interactive アクティビティが起動し続けており、起動直後以外では Solitary 状態には遷移しません。</p>
Interactive	<p>Interactive アクティビティを起動し、主にユーザとのやり取りを行います。Interactive アクティビティが終了した場合は Solitary 状態へ遷移します。Interactive 状態へ遷移するのは、Solitary 状態からのみです。</p>
Safeguard	<p>関節の温度上昇やハードウェアエラーなどの危険を検知した場合に遷移します。アクティビティの起動は行わず、起動中のアクティビティがあれば停止します。また、Basic Awareness と Breathing も停止します。危険状態からの復旧を試み、復旧できれば Solitary 状態へ遷移します。</p>
Disabled	<p>A-Life 停止。電源投入直後、もしくは他の状態への遷移ができない場合に遷移します。アクティビティの起動は行わず、起動中のアクティビティがあれば停止します。また、Basic Awareness と Breathing も停止します。</p>

次の図は状態の遷移図です。基本的には **Solitary** 状態と **Interactive** 状態を交互に遷移して動作します。

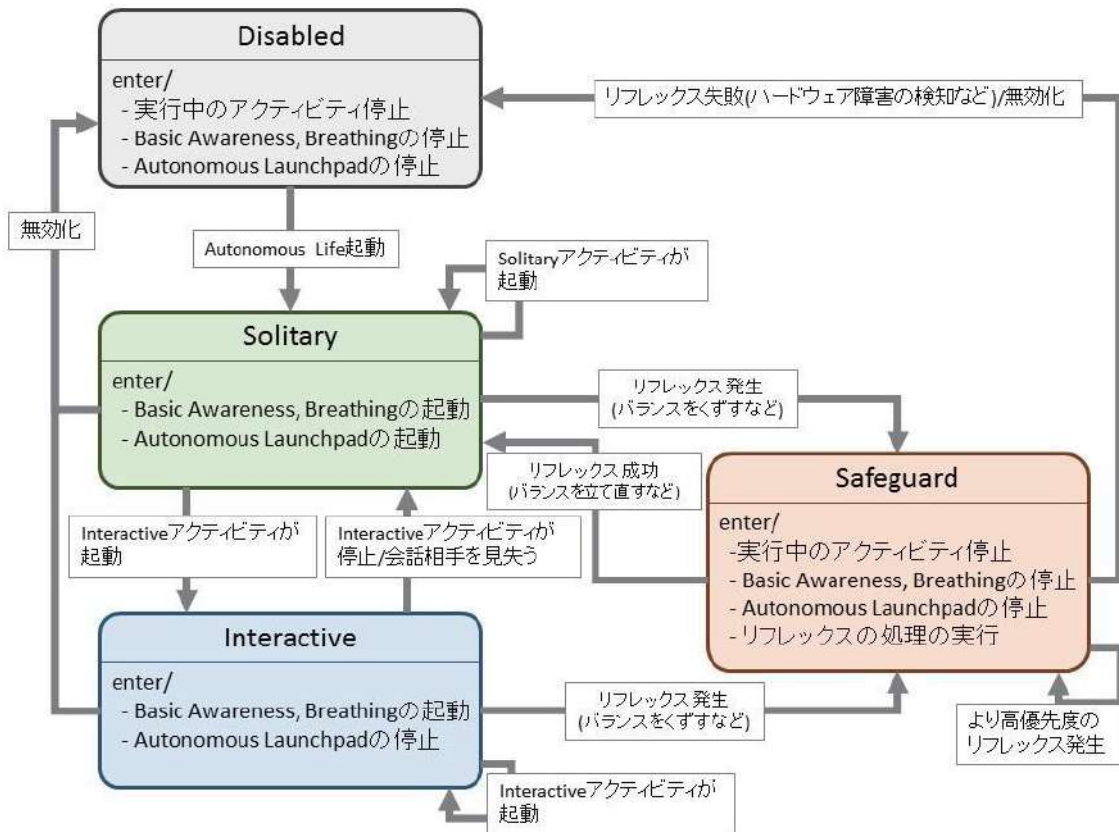


図 10.1-1 A-Life の状態遷移図

10.1.1.4. 安全確保のための反応

Pepper が転倒したり、関節のモーターがオーバーヒートした場合、安全を確保するための行動を行います。

- 実行中のアクティビティ停止
- Basic Awareness と Breathing Animation 停止
- リフレックス処理の実行（関節外す、関節温度上昇による一時停止など）

10.1.2. アクティビティ

アクティビティには2種類（性質）あり、以下の特徴を持っています。

表 10.1-2 アクティビティの性質

性質	説明
Interactive	実行中に他のアクティビティの起動条件が満たされても割り込まれない。ユーザと双方向のコミュニケーションを実装するのに適している。
Solitary	実行中に Interactive アクティビティが起動条件を満たすと割りこまれて中断する。ユーザがいないときの振る舞いを実装するのに適している。

ロボアプリは、「商品紹介をしたい」とか「アンケートを取りたい」などの目的を持っているはずですが、商品紹介やアンケートは Pepper のすぐ前にユーザがいないと成立しません。そういった本来の目的を達成するためのアクティビティは Interactive として実装します。ただし、常に Pepper のすぐ前にユーザが居るとは限りません。Pepper の前を通り過ぎる人達に注目してもらい、近くに呼び寄せ、本来の目的を達成するための Interactive アクティビティへ導く必要があります。人を惹きつけるためのアクティビティは Solitary として実装します。また、毎時 00 分に時報を行ったり、人が居なくて独り言を言って退屈な動作をするような短時間の処理も Solitary アクティビティが適しています。

10.1.3. オートノマスライフの停止と再開

A-Life は Pepper 起動後に自動的に開始されますが、停止させることもできます。Choregraphe のツールバーにある[オートノマスライフ] ボタンをクリックすると、A-Life がオンとオフが切り替わります。胸の電源ボタンを2度クリックしても A-Life が停止します。



図 10.1-2 A-Life の停止と再開

10.2. Interactive アクティビティ

Interactive アクティビティを起動する方法は以下の 3 つがあります。

表 10.2-1 Interactive アクティビティの起動方法

起動方法	説明
ロボアプリランチャー	ディスプレイをタッチすると表示されるアプリのアイコンをタッチして起動する。
トリガーセンテンス	予め設定したキーワードをユーザが話しかけて起動する。
起動トリガー条件	予め設定した条件で起動する。
お仕事かんたん生成	Pepper for Biz モデルでは、お仕事かんたん生成で設定した条件で起動する。

Interactive アクティビティは 3 つの起動方法すべてを使用することができます。

この節では、Interactive アクティビティの作成方法、トリガーセンテンスの設定方法などを紹介します。



図 10.2-1 ロボアプリランチャー

10.2.1. 性質の設定

プロジェクト内のビヘイビア (behavior.xar) ごとにアクティビティの性質を指定できます。

アクティビティの性質の設定手順は以下の通りです。

1. [プロジェクトファイル] パネルの [プロパティ] ボタンをクリックする。
2. [プロジェクトのプロパティ] ウィンドウ左側の [編集したいコンテンツを選択してください] リストから、behavior.xar ファイルを含むフォルダ名をクリックする。
3. [プロジェクトのプロパティ] ウィンドウの [性質] から [インタラクティブ] を選択して、[OK] ボタンをクリックする。



図 10.2-2 性質の設定（インタラクティブの場合）

10.2.2. トリガーセンテンス

ユーザに特定のキーワードを話しかけてもらうことでアクティビティを起動する方法です。トリガーセンテンスは**一般販売モデル**だけで有効です。トリガーセンテンスを設定する手順は以下の通りです。

1. [編集したいコンテンツを選択してください]リストから、性質がインタラクティブな behavior.xar ファイルを含むフォルダ名をクリックする。
2. [ユーザのリクエストより開始]チェックボックスをオンにする。
3. [トリガーセンテンス]領域をクリックする。
4. [トリガーセンテンス]にキーワードを入力し、[Add]ボタンをクリックする。

トリガーセンテンスは複数登録することができます。同じ発音になるキーワードをできるだけ多く登録しておくと、起動しやすくなります。発音は [Add]ボタンの右側のスピーカーアイコンをクリックすると確認できます。

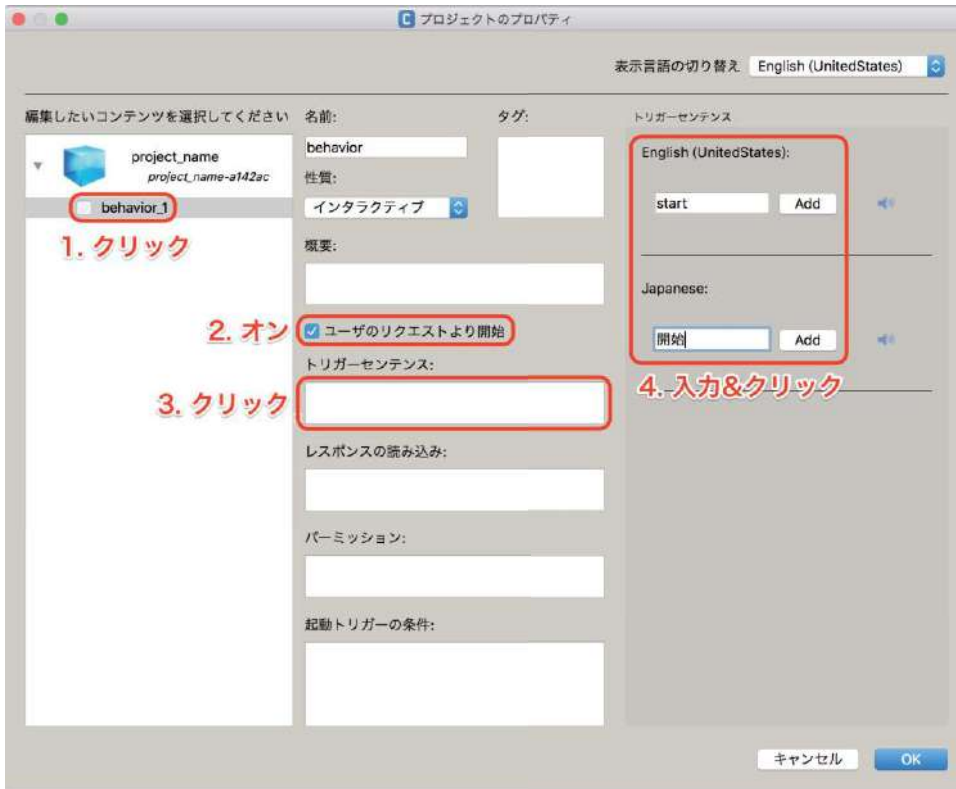


図 10.2-3 トリガーセンテンス

10.2.3. アプリアイコン

アイコンは一般販売モデルのロボアプリランチャーや Pepper for Biz モデルのロボアプリ配信管理などに表示されます。ロボアプリのイメージに相応しいデザインを心がけてください。トリガーセンテンスおよびアイコンから起動するには、プロジェクトのプロパティから[ユーザーのリクエストより起動] チェックボックスをオンにする必要があります。

10.2.3.1. アイコンの作成要領

アイコンに使用できる色は、最大6色までです（グレー系の色合いを除く）。中心となるイラストに2ピクセルの影（#a7a7a7）を入れます。背景（#d0d0cf、コーナー6.35mm）を追加し、背景の中心にイラストを配置します。以下ようになります。

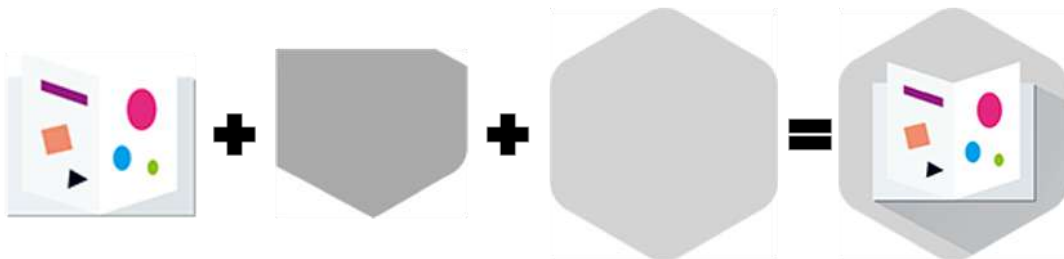


図 10.2-4 作成要領

10.2.3.2. アイコンのデザイン可能範囲

青色ゾーンが自由にデザインできるエリア "ICON DESIGN SAFE ZONE" となります。

一部であれば、青色ゾーンからデザインがはみ出ても構いませんが、赤色の破線を超えてはみ出すことはできません。また、イラストの形に制限はありません。



図 10.2-5 デザイン可能範囲

10.2.3.3. アイコンテンプレートの入手

ロボアプリアイコンテンプレートは Adobe Illustrator EPS 形式で提供されておりますので、ご活用ください。

[アイコンのテンプレート]

<http://www.softbank.jp/robot/biz/support/document/icons/>

10.2.3.4. アイコンの設定

できあがったファイルをラスターライズし、360 x 360 px、icon.png として保存し、ロボアプリのルート階層に配置します。その後、プロジェクトのプロパティ設定よりロボアプリのアイコンをクリックし、作成したアイコンを設定します。



図 10.2-6 ロボアプリのアイコン

10.3. Solitary アクティビティ

Solitary アクティビティは、Pepper のすぐ前にユーザがいない状態で実行するのに適して

います。よって、ロボットアプリラウンチャーやトリガーセンテンスのようにユーザの操作による起動方法は使用できません。この節では、Solitary アクティビティの作成方法、起動トリガー条件の設定方法を紹介します。この節の内容は、一般販売モデルだけで有効です。

10.3.1. 性質の設定

アクティビティの性質を Solitary に設定する手順は Interactive の場合と同様です。

1. [プロジェクトファイル]パネルの[プロパティ]ボタンをクリックする。
2. [プロジェクトのプロパティ]ウィンドウ左側の[編集したいコンテンツを選択してください]リストから、behavior.xar ファイルを含むフォルダ名をクリックする。
3. [プロジェクトのプロパティ]ウィンドウ右側の[性質] から[ソリタリー]を選択して、[OK]ボタンをクリックする。



図 10.3-1 性質の設定 (ソリタリーの場合)

10.3.2. 起動トリガー条件

起動トリガー条件は、Pepper のセンサーで検知した周囲の環境変化や、Pepper 内部の状態変化を条件にアプリを起動します。

10.3.2.1. 起動トリガー条件の設定方法

起動トリガー条件の設定手順は以下の通りです。

1. [編集したいコンテンツを選択してください]リストから、性質がソリタリーな behavior.xar ファイルを含むフォルダ名をクリックする
2. [起動トリガーの条件]領域に条件を入力後、[OK]ボタンをクリックする

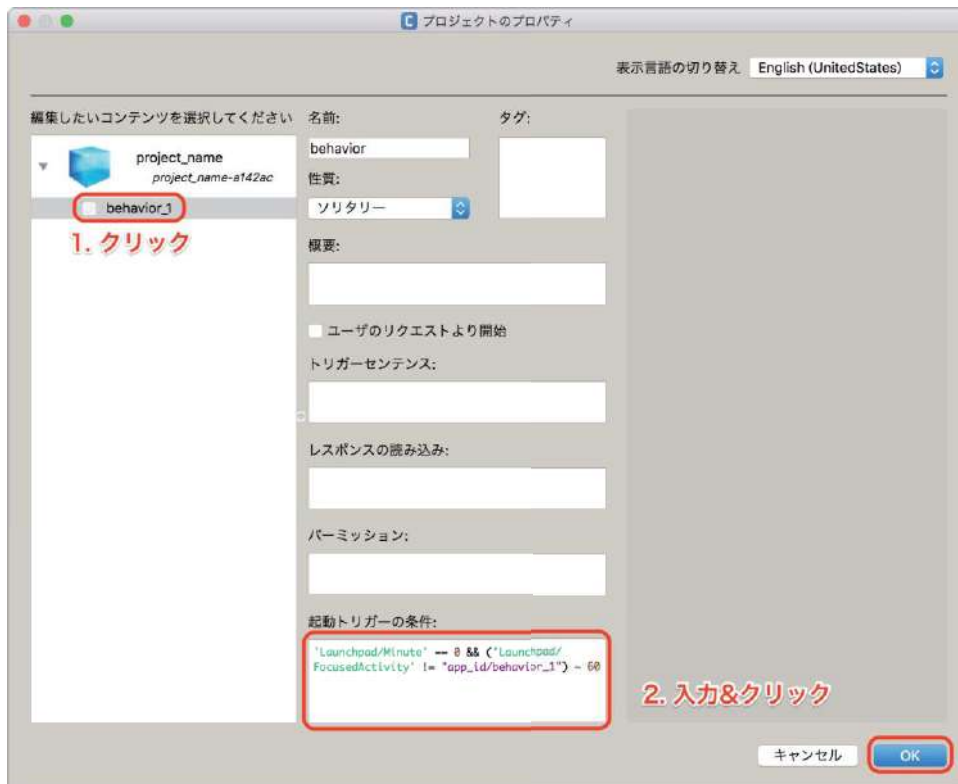


図 10.3-2 起動トリガー条件

10.3.2.2. Launchpad イベント

起動トリガー条件で使用されるイベントは"Launchpad"で始まるものが多いです。

表 10.3-1 主な Launchpad 系イベント

イベント名	説明
Launchpad/NumMotionZone1	Solitary 状態で、ゾーン 1 内で動く物体の数。
Launchpad/NumMotionZone2	Solitary 状態で、ゾーン 2 内で動く物体の数。
Launchpad/NumMotionZone3	Solitary 状態で、ゾーン 3 内で動く物体の数。
Launchpad/NumPeopleZone1	Solitary 状態で、ゾーン 1 内に存在する人の数。
Launchpad/NumPeopleZone2	Solitary 状態で、ゾーン 2 内に存在する人の数。
Launchpad/NumPeopleZone3	Solitary 状態で、ゾーン 3 内に存在する人の数。
Launchpad/NoPeopleInZones	どのゾーンにも人がいない状態。
Launchpad/Year	西暦。
Launchpad/Month	月 (1~12)。
Launchpad/MonthName	月の英語名。
Launchpad/Day	年の何日目 (1~366)。
Launchpad/DayName	曜日の英語名。
Launchpad/Date	日 (1~31)。
Launchpad/Hour	時間 (0~23)。
Launchpad/Minute	分。
Launchpad/Week	年の何週目 (0~53)。
Launchpad/FocusedActivity	現在実行中のアクティビティ。

例えば、毎時 0 分で起動するアプリの起動トリガー条件は以下のようになります。

```
'LaunchPad/Minute' == 0
```

図 10.3-3 毎時 0 分で起動するアプリの起動トリガー条件

しかし、この条件だけではアプリが 1 分未満に終了すると、また条件が満たされて同じアプリが起動してしまいます。

それを防ぐために、1 分間は同じアプリが起動しないようにする条件を追加します。

```
'Launchpad/Minute' == 0 &&
(('Launchpad/FocusedActivity' != "自身のビヘイビアパス") ~ 60)
```

図 10.3-4 1 分間は同じアプリが起動しないようにする条件

[起動トリガーの条件]領域に条件式を入力した時、イベント名やビヘイビアパスが正しく設定されていれば色が変わります。イベント名は緑色、ビヘイビアパスは紫色になります。正しく設定するには、イベント名はシングルクォーテーション、ビヘイビアパスはダブルクォーテーションで挟みます。

10.3.2.3. ビヘイビアパス

Launchpad イベントで紹介した起動トリガー条件の例の中に登場したビヘイビアパスの書式は以下の通りです。ApplicationID と behavior.xar のディレクトリパスを結合した書式 (EBNF) になります。

```
(ApplicationID)/(ディレクトリパス)
```

図 10.3-5 ビヘイビアパスの書式

ビヘイビアパス前半の「ApplicationID」はプロジェクトのプロパティに設定されている [アプリケーション ID] の値です。

ApplicationID の確認方法は以下の通りです。

1. Choregraphe でプロジェクトファイルを開きます。
2. メニュー [ファイル] - [プロジェクトのプロパティ] をクリックします。
3. アプリケーション ID 欄に ApplicationID が表示されます。

ビヘイビアパス後半の「ディレクトリパス」は、起動したいアクティビティの behavior.xar が保存されているフォルダのパスになります。Choregraphe で新規のプロジェクトを作成すると、behavior.xar ファイルは behavior_1 というフォルダに入っています。その場合、「ディレクトリパス」は "behavior_1" になります。behavior.xar ファイルがプロジェクトのトップレベルに配置されている場合、「ディレクトリパス」は "." (ドット) になります。

ディレクトリパスの確認方法は以下の通りです。

1. Choregraphe でプロジェクトファイルを開きます。
2. ロボアプリが開始される behavior.xar に注目します。

下の図では behavior.xar は "behavior_1" ディレクトリの中にあります。このような場合はディレクトリパスは "behavior_1" となります。

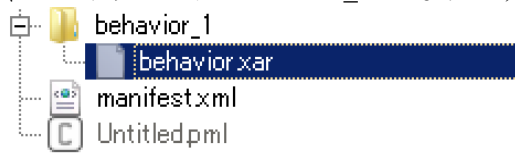


図 10.3-6 "behavior_1" ディレクトリ内

下の図では behavior.xar はルートディレクトリの中にあります。このような場合は "." がディレクトリパスとなります。

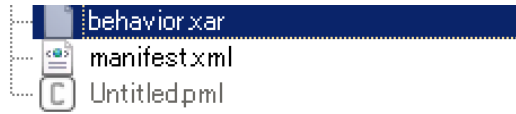


図 10.3-7 ルートディレクトリ内

Sample 10-1

表 10.3-2 Sample10-1

項目	説明
プロジェクト名	sample10-1
ファイルパス	chapter10/sample10-1/sample10-1.pml
概要	Solitary アクティビティのサンプルアプリです。毎時 0 分になると、Pepper が時間を喋ります。プロジェクトのプロパティ設定方法、起動トリガー条件設定方法を確認してください。このサンプルアプリは、一般販売モデルの Pepper にインストールして、A-Life がオンの状態で実行します。

10.4. アクティビティの連携

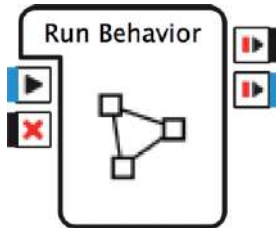


図 10.4-1 Run Behavior ボックス

Solitary アクティビティでユーザを惹きつけて (Pepper に近寄らせて)、本来の目的のための Interactive アクティビティへスムーズに移行するには、Solitary と Interactive を連携させます。Solitary アクティビティから Interactive アクティビティを呼び出すには、[Run Behavior] ボックスを使用します。入力 [onStart] に呼び出したいアクティビティのビヘイビアパスの文字列を渡します。

[Run Behavior]は、呼び出し先のアクティビティが開始されたら呼び出し元のアクティビティは一時停止して、呼び出し先のアクティビティが終了したら出力 [onStopped] から処理が先に進みます。[Run Behavior] ボックスを使用しないで、Interactive アクティビティの起動トリガー条件に「人がゾーン 1 の中に入ったら」というように設定しても実現できますが、同じ起動トリガー条件を持つ Interactive アクティビティが他に存在した場合、必ず目的の Interactive アクティビティが起動する保障はありません。

Sample 10-2

表 10.4-1 Sample10-2

項目	説明
プロジェクト名	sample10-2
ファイルパス	Chapter10/sample10-2/sample10-2.pml
概要	Solitary アクティビティから Interactive アクティビティを呼び出すサンプルです。[Run Behavior]ボックスの使用方法を確認してください。

11. 標準ボックス

この章では、Choregraphe に標準搭載されているボックスライブラリについて解説します。

11.1. Take Picture ボックス



図 11.1-1 Take Picture ボックス

[Take Picture]ボックスを使用すると、Pepper のカメラを使って画像の撮影ができます。設定画面では画像の解像度や使用するカメラ (Top または Bottom) などを選択できます。生成される画像ファイルは jpg 形式です。自動的に拡張子が付与されますので、設定画面で指定するファイル名には拡張子を含めないようにします。

11.2. Record Sound ボックス



図 11.2-1 Record Sound ボックス

[Record Sound]ボックスを使用すると、Pepper のマイクを使って音の録音ができます。設定画面では、保存形式 (wav または ogg) などが設定できます。[Record Sound]ボックスは[Get File Name]ボックスと[Wait]ボックスと[Rec. Sound File]ボックスから構成されています。

11.3. Play Sound ボックス



図 11.3-1 Play Sound ボックス

[Play Sound]ボックスを使用すると、Pepper のスピーカーを使って音の再生ができます。設定画面では音量や再生開始位置などを設定できます。[Play Sound]ボックスは[Get Attached File]ボックスと[Play Sound File]ボックスから構成されています。

11.4. Record Video ボックス

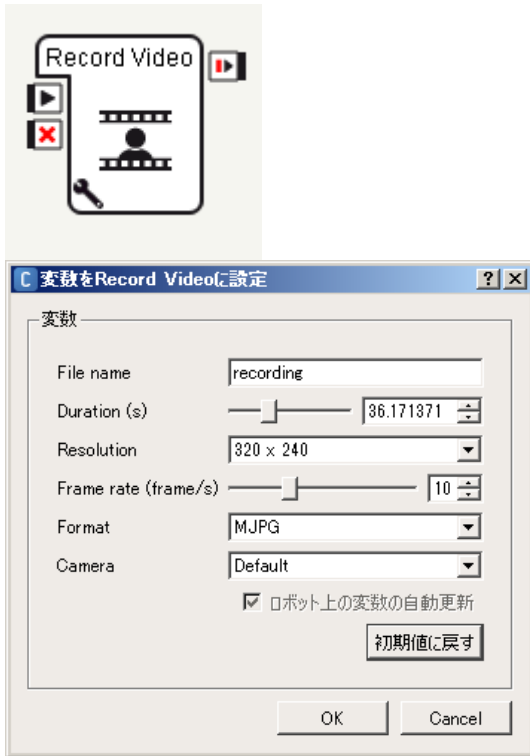


図 11.4-1 Record Video ボックス

[Record Video]ボックスを使用すると、Pepper のカメラを使って動画の撮影ができます。設定画面では動画の画面サイズやフレームレート等も設定できます。また、撮影した動画は、ディスプレイからアクセスできる html 以下に配置することで、胸部のディスプレイで再生できます。

ビデオデバイス用の API である ALVideoDevice の recordVideo メソッドを用いて動画の録画もできましたが、API の 1.2 以降では ALVideoRecorder の startRecording メソッドが推奨されています。

11.5. Play Video ボックス



図 11.5-1 Play Video ボックス

[Play Video]ボックスを使用すると、動画を再生することができます。設定画面では再生するファイルのパスを設定できます。ボックスの結線でも簡易的に動画の操作は可能ですが、スクリプトでの操作も可能です。その場合は `ALTabletService` のメソッドを使います。

表 11.5-1 動画の再生

メソッド名	概要
<code>getVideoLength</code>	動画の再生時間を取得します。
<code>getVideoPosition</code>	動画の再生位置を取得します。
<code>pauseVideo</code>	動画を停止します。
<code>playVideo</code>	動画を再生します。
<code>resumeVideo</code>	動画の再生を再開します。
<code>stopVideo</code>	動画の再生を停止します。

11.6. Basic Awareness ボックス

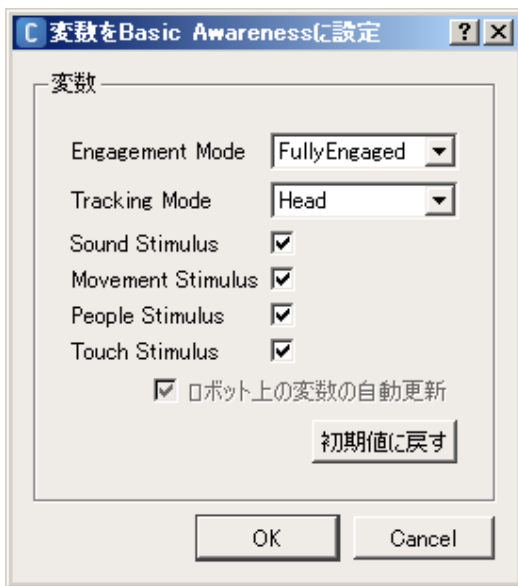


図 11.6-1 Basic Awareness ボックス

Pepper は A-Life がオンの時、周囲の情報に反応するようになっています。[Basic Awareness]ボックスを使用することで、Basic Awareness の機能のオン/オフができます。設定ウィンドウで変更可能な項目は次の通りです。

表 11.6-1 Basic Awareness ボックス

項目	設定
Engagement Mode	Pepper がユーザを認識した後、多くはそのユーザとの Interactive アクティビティ中に、外部刺激にどの程度反応するか設定します。
	Unengaged 外部刺激に反応します。
	FullyEngaged 外部刺激に反応しません。
	SemiEngaged 外部刺激に反応しますが、すぐにユーザのほうに向き直します。
Tracking Mode	Pepper がどの程度体を動かしてユーザを追跡するか設定します。
	Head 頭を動かして追跡します。
	BodyRotation 体を傾けて追跡します。
	WholeBody 全身を使って追跡しますが、旋回はしません。
Sound Stimulus	音に反応するかどうかを設定します。
Movement Stimulus	動きに反応するかどうかを設定します。
People Stimulus	人物識別をして反応するかどうかを設定します。
Touch Stimulus	タッチセンサーによる刺激に反応するかどうかを設定します。

11.6.1. Breathing

Pepper は息をしているかのようなモーションをしています。Breathing については、制御をするボックスはありませんので、ALAutonomousMoves のメソッドを使用して制御する必要があります。

11.7. Comment ボックス

Comment ボックスは Python ボックスの一種ですが、どのボックスとも結線されないボックスです。自由な場所に配置して、補足説明などを書いておきます。

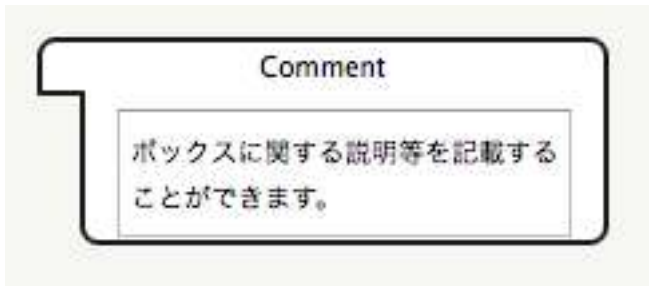


図 11.7-1 Comment ボックス

11.8. Log ボックス

Log ボックスは Python ボックスの一種ですが、ログを出力することができるボックスです。出力したログはログビューパネルで確認することができます。パラメータの設定でログレベルを設定することが可能です。

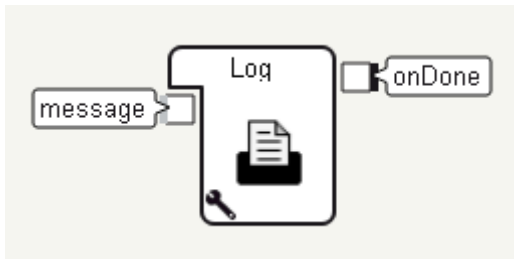


図 11.8-1 Log ボックス



図 11.8-2 Log ボックスのパラメータ

Log ボックスで設定可能なログレベルは以下の 5 種類です。

表 11.8-1 ログレベルの一覧

レベル	用途
Fatal	致命的なエラー。
Error	エラー。
Warning	警告。
Info	情報。
Debug	デバッグ。

12. NAOqi OS(API)

NAOqi OS とは、Pepper の中枢をなしているミドルウェアです。NAOqi OS が提供しているクラス群を使用することで、Pepper 内部の状態を取得したり、Pepper に直接命令を出すことができます。本項では NAOqi OS の API の中でも使用頻度が高いものを取り上げます。

12.1. API ドキュメントの見方

NAOqi OS は 1,350 以上の API を公開しており、Web サイト上で確認できます。また、Choregraphe をインストールすると、API リファレンスも同時にインストールされます。API リファレンスの確認方法は以下の通りです。

1. Choregraphe を起動します。
2. メニュー [ヘルプ] - [Choregraphe/参照 API] をクリックします。
3. インターネットブラウザが起動します。
4. ドキュメント右上にテキストボックスがあります、調べたいキーワードを入力して [Search] ボタンをクリックします。(ここでは"ALFaceCharacteristics")を検索しています。Macintosh 環境の Chrome ブラウザでは [Search] ボタン (検索機能) が正常に機能しません。Safari や Firefox を利用してください。
5. 目的のページをクリックします。

API ドキュメントは以下のように構成されています。

The screenshot shows the API documentation for 'ALFaceCharacteristics API'. It includes a search bar, a table of contents, and a detailed view of the API module. Red boxes and text highlight key features:

- ALFaceCharacteristics API** モジュールは1つのクラスです
- Method list** メソッドはモジュールのプロキシを作成して利用します
- Event list** イベントはグローバルでSubscribeできます
- ALMemory Key list** 一時的なALMemoryの値はグローバルで利用できます

図 12.1-1 API ドキュメントの使い方

サイトには、API ドキュメントを効率よく閲覧できるよう検索機能が設けられています。各 API モジュールは 1 つの python クラスからできています。モジュールに含まれる内容は、メソッド、イベント、ALMemory Key の 3 つです。

表 12.1-1 API モジュールの構成

項目	説明
メソッド(Method)	API を通して実行できる具体的な処理命令です。メソッドは Proxy を通じて使用します。
イベント(Event)	発生するイベントはグローバルで Subscribe できます。

ALMemory Key	一時的な ALMemory の値はグローバルで使用できます。
--------------	--------------------------------

次の図は API の一覧です。特に重要なものは赤枠で囲んでありますので、API リファレンスを実際にかけて確認してみてください。

Core	
ALAutonomousLife	ALPreferenceManager
ALAutonomousMoves	ALResourceManager
ALBehaviorManager	ALStore
ALConnectionManager	ALSystem
ALExtractor	ALVisionExtractors
ALMemory	ALTabletService
ALModule	ALUserSession
ALNotificationManager	ALWorldRepresentation
	PackageManager

Audio	
ALAnimatedSpeech	ALSoundDetection
ALAudioDevice	ALSoundLocalization
ALAudioPlayer	ALSpeechRecognition
ALAudioRecorder	ALTextToSpeech
ALDialog	ALVoiceEmotionAnalysis

Vision	
ALBacklightingDetection	ALPhotoCapture
ALBarcodeReader	ALRedBallDetection
ALCloseObjectDetection	ALSegmentation3D
ALColorBlobDetection	ALVideoDevice
ALDarknessDetection	ALVideoRecorder
ALLandmarkDetection	ALVisionRecognition
ALLocalization	ALVisualCompass
ALMovementDetection	ALVisualSpaceHistory

People Perception	
ALBasicAwareness	ALGazeAnalysis
ALEngagementZones	ALPeoplePerception
ALFaceCharacteristics	ALsittingPeopleDetectio
ALFaceDetection	ALWavingDetection

Diagnosis	DCM
ALDiagnosis	DCM

Sensors	
ALBattery	ALLaser
ALBodyTemperature	ALLeds
ALChestButton	ALSensors
ALFsr	ALSonar
ALInfrared	ALTouch

Motion
ALMotion
ALNavigation
ALRecharge
ALRobotPosture

Trackers
ALTracker

図 12.1-2 NAOqi API 一覧

12.2. NAOqi Vision API

Pepper にはカメラを使った、写真撮影、QR コード認識、環境光検知、Red Ball 認識などを実現する Vision API 群が用意されています。



図 12.2-1 Vision API 群

Quick search

Enter search terms or a module, class or function name.

Table Of Contents

- Site map
- What's new
- NAO Documentation
- Romeo Documentation
- Pepper Documentation
- NAOqi Developer guide
 - Getting Started
 - Programming
 - NAOqi Framework
 - Key concepts
 - **NAOqi API**
 - NAOqi Core
 - NAOqi Interaction engines
 - NAOqi Motion
 - NAOqi Audio
 - **NAOqi Vision**
 - NAOqi PeoplePerception
 - NAOqi Sensors & LEDs
 - NAOqi Trackers
 - ALDiagnosis
 - DCM
 - Types

Vision

Read also the [NAOqi Vision introduction](#).

- [ALBacklightingDetection API | overview](#)
- **[ALBarcodeReader API | overview](#)**
- [ALCloseObjectDetection API | overview](#)
- [ALColorBlobDetection API | overview](#)
- [ALDarknessDetection API | overview](#)
- [ALLandmarkDetection API | overview](#)
- [ALLocalization - API | overview](#)
- [ALMovementDetection API | overview](#)
- **[ALPhotoCapture API | overview](#)**
- [ALRedBallDetection API | overview](#)
- [ALSegmentation3D API | overview](#)
- [ALVideoDevice API | overview](#)
- [ALVideoRecorder API | overview](#)
- [ALVisionRecognition API | overview](#)
- [ALVisualCompass API | overview](#)
- [ALVisualSpaceHistory API | overview](#)

図 12.2-2 Vision モジュール群

12.2.1. ALPhotoCapture

NAOqi Vision に含まれる ALPhotoCapture を例にして具体的な API を見てみます。ALPhotoCapture は Pepper のカメラを使って、写真撮影を行うモジュールとなっています。API を使うと写真のフォーマット、解像度、デバイス等をコントロールできます。

表 12.2-1 ALPhotoCapture のメソッド

メソッド名	概要
getCameraID	設定済みのカメラ ID を取得します。
setCameraID	利用するカメラ ID を設定します。
getPictureFormat	設定済みの写真フォーマットを取得します。
setPictureFormat	写真フォーマットを設定します。
getResolution	設定済みの解像度を取得します。
setResolution	写真の解像度を設定します。
takePicture	写真を撮影し、指定パスに保存します。

12.2.2. ALPeoplePerception

目の前にいる人の顔の感情認識、顔そのものの認識を実現する ALPeoplePerception が用意されています。



図 12.2-3 PeoplePerception API

ALPeoplePerception は Pepper の目の前に立っている人を認識し、人数や感情、シャツの色など様々な情報を取得し、ALMemory の値を更新するモジュールです。

表 12.2-2 ALPeoplePerception のメソッド

メソッド名	概要
getMaximumDetectionRange	最大の検知距離を取得する。
setMaximumDetectionRange	最大の検知距離を設定する。
resetPopulation	検知した人をリセットする。
setFastModeEnabled	Fast mode を設定する。

表 12.2-3 ALPeoplePerception の Key list

ALMemory	概要
PeoplePerception/Person/[ID]/Distance	人の距離。
PeoplePerception/Person/[ID]/PresentSince	認識から経過時間。
PeoplePerception/Person/[ID]/RealHeight	人の身長。
PeoplePerception/Person/[ID]/ShirtColor	人のシャツの色。

12.2.3. ALFaceCharacteristics

ALFaceCharacteristics は人の喜怒哀楽感情を取得するモジュールです。

表 12.2-4 ALFaceCharacteristics のメソッド

メソッド名	概要
analyzeFaceCharacteristics	表情からユーザの表情から感情を取得。
getSmilingThreshold	笑顔と判定する閾値の取得。
setSmilingThreshold	笑顔と判定する閾値の設定。

表 12.2-5 ALFaceCharacteristics の Key list

ALMemory	概要
PeoplePerception/Person/[ID]/AgeProperties	年齢。
PeoplePerception/Person/[ID]/ExpressionProperties	取得した感情。 平常、怒り、喜び、驚き、悲しみの 5 種類。
PeoplePerception/Person/[ID]/GenderProperties	性別。
PeoplePerception/Person/[ID]/SmileProperties	笑顔。
PeoplePerception/Person/[ID]/FacialPartsProperties	顔、および顔のパーツの位置。

Sample 12-1

表 12.2-6 Sample12-1

項目	説明
プロジェクト名	sample12-1
ファイルパス	chapter12/sample12-1/sample12-1.pml
概要	ALPhotoCapture で写真を撮って、ディスプレイで表示するサンプルです。

Sample 12-2

表 12.2-7 Sample12-2

項目	説明
プロジェクト名	sample12-2
ファイルパス	chapter12/sample12-2/sample12-2.pml

概要	ALBarcodeReader を使った QR コード認識について解説します。ALBarcodeReader は名前こそ Barcode となっておりますが、現状 QR コード以外のバーコードを読み込むことはできません。Event "BarcodeReader/BarcodeDetected" を使います。
----	--

Sample 12-3

表 12.2-8 Sample12-3

項目	説明
プロジェクト名	sample12-3
ファイルパス	chapter12/sample12-3/sample12-3.pml
概要	表情認識で読み取った感情に応じて処理を分岐できます。 [FaceEmotionEngine]ボックスは、python のスクリプトで"平常","喜び","驚き","怒り","悲しみ"の感情を読み取る処理を行います。

12.3. NAOqi Audio API

Pepper は頭頂部のマイクを使って話しかけられた音を認識し、スピーカーから音声を再生できます。これら音声に関する機能は、Audio API として提供されています。

12.3.1. ALVoiceEmotionAnalysis

この機能は NAOqi2.5.5 では使えません。

ALVoiceEmotionAnalysis は音声の感情情報を抽出するモジュールで、EmotionRecognized イベントを使用すると、Pepper がユーザに声をかけられた時に反応させられます。

表 12.3-1 ALVoiceEmotionAnalysis の主なメソッド

メソッド名	概要
setParameter	認識するときの音声の長さを設定。
subscribe	音声の感情情報を取得する EmotionRecognized イベントを発生させる。
unsubscribe	EmotionRecognized イベントを停止する。

表 12.3-2 ALVoiceEmotionAnalysis の Key list

Event	概要
ALVoiceEmotionAnalysis/EmotionRecognized	取得した感情。 平常、怒り、喜び、悲しみ、興奮の 5 種。

12.4. NAOqi Motion API

Pepper には各種モーターを使った、モーションや移動などを実現する Motion API 群が用意されています。

12.4.1. ALNavigation

ALNavigation は周辺の空きスペースを認識し、そこまで移動させることができるモジュールです。

表 12.4-1 ALNavigation のメソッド

メソッド名	概要
navigateTo	障害物を避けて指定の位置に移動。
moveAlong	指定した経路で移動。
getFreeZone	現在のスペースの情報を取得。
findFreeZone	周辺の空きスペース情報を取得。

表 12.4-2 ALNavigation の Key list

ALMemory	概要
Navigation/AvoidanceNavigator/Status	状態。
Navigation/AvoidanceNavigator/ObstacleDetected	障害物の見地。
Navigation/AvoidanceNavigator/MovingToFreeZone	空きスペースへの移動開始、完了。
Navigation/AvoidanceNavigator/TrajectoryProgress	移動の進捗。
Navigation/AvoidanceNavigator/AbsTargetModified	移動失敗。
Navigation/MotionDetected	周辺の物体の移動を検知。

このほかに SLAM の機能も搭載されています。SLAM とは自己位置推定と環境地図作成を同時に行う機能です。この機能は B 機能につきサポートデスクなどからサポートは受けられません。

12.5. NAOqi Sensors & LEDs

Pepper に搭載されているセンサーの値を参照したり、LED の制御を行うために Sensors & LEDs API 群が用意されています。

12.5.1. ALLeds

ALLeds は LED の制御を行うことができるモジュールです。

表 12.5-1 ALNavigation のメソッド

メソッド名	概要
fade	色の強度を設定して点灯。耳の LED を使用するとき使用。
fadeListRGB	LED の色を設定して点灯。目の LED を使用するとき使用。

耳の LED は青色しか使えませんが、点灯箇所が 10 か所に分かれているため、順番に点灯させることで弧を描くような点灯のさせ方も可能です。用途としては、ユーザに向けての感情表現の一つとして使用するのが効果的でしょう。たとえば「嬉しい」と表現する場合、声だけではなくて動作もつけますが、それに加えて LED を点滅させるなどすることで、より表現を強調できます。アプリ内のエラーなどを知らせるために使用してもよいでしょう。

また、Choregraphe の標準ボックスライブラリには LED に関するボックスが用意されています。いくつかのボックスはダイアグラムの形式になっており、内部のボックスや設定ウィンドウで調整が可能です。色の変更は[Color Edit]ボックスでも行えます。[Color Edit]ボックスはカラーコードを視覚的に選びやすくなっており、アウトプットとして、数値「R,G,B」を出力します。

13. お仕事かんたん生成 2.0

「お仕事かんたん生成 2.0」は Pepper for Biz で利用可能な、Pepper のお仕事を作成することができる Web サービスです。業種ごとに分類されたテンプレートが各種用意されており、テンプレートをカスタマイズすることでロボアプリの開発経験がない方でもお仕事を作成することが可能です。

作成したお仕事は遠隔地にいる Pepper に反映したり、一括で管理編集したりすることができます。「ロボアプリ配信管理」というウェブサービスと組み合わせることで、開発したロボアプリをお仕事かんたん生成 2.0 の中に組み込むことも可能です。

Pepper がお仕事中に収集したデータは、「インタラクション分析」というウェブサービス上で閲覧・分析できます。

お仕事かんたん生成 2.0 は、Pepper をネットワークに接続した状態で利用する必要があります。

お仕事かんたん生成 2.0 と関連する上記のウェブサービスのうち、開発者が把握しておく必要のある事項について解説します。

13.1. お仕事の概要

13.1.1. お仕事について

Pepper が行う業務全体を「お仕事」と呼びます。お仕事の内容をカスタマイズし、Pepper に接客業務などをさせることができます。

お仕事をさせるには、あらかじめ「お仕事かんたん生成 2.0」でお仕事を作成し、作成したお仕事を Pepper に反映させる設定を行います。

13.1.2. お仕事かんたん生成 2.0 について

お仕事かんたん生成 2.0 では、ロボアプリの開発経験がない方でも、業務シーンに合わせて Pepper で使用したいお仕事をカスタマイズできます。テンプレートを使ってお仕事をカスタマイズできるので、業務に合わせた Pepper のお仕事をかんたんに作成することができます。

お仕事かんたん生成 2.0 を使って、作成したお仕事を離れた場所にある Pepper にも配信することができます。一度 Pepper に配信したお仕事も再度編集することができます。

13.2. お仕事かんたん生成 2.0 の基本設定

お仕事かんたん生成 2.0 の設定画面では、Pepper の機体名を新しく登録したり、修正や削除などの編集を行うことができます。機体名はインタラクション分析に利用されます。

また、複数台の Pepper を管理する場合にはお仕事作成の開始前に「Pepper 機体リスト」

を設定しておくことで、機体ごとのインタラクション分析を行うことができます。

ここでは、設定の中でも特に重要な機体リストの設定について解説します。

13.2.1. Pepper の機体リストを管理する

「Pepper 機体リスト」は、同一 SBR アカウントで複数台の Pepper を管理する際に使用する機能で、Pepper に名前をつけて管理することができます。お仕事かんたん生成 2.0 の「設定画面」から登録することができます。

Pepper 機体リストに登録しておくことで、機体ごとのインタラクション分析を行うことが可能になります。

機体の新規登録の際には一体ずつ登録することはもちろん、機体数が多い場合には CSV ファイルからの登録も可能となっています。

配信したお仕事は、機体リストの有無にかかわらず、お仕事を作成した SBR アカウントでログインしている全ての Pepper 本体へ配信されます。

13.3. お仕事の新規作成

新しくお仕事を作成する場合は、はじめにテンプレートを作成し、そのお仕事の初期設定を行います。初期設定では、Pepper の行う行動(ボックス)の設定を行います。

利用可能なテンプレートには、業界や利用シーンごとに準備されているテンプレートや、キーワード設定がなく、ボックスがない状態であるフリーのテンプレートなどがあります。いずれも、自由にカスタマイズ・編集をすることで独自性のある多様なお仕事を作成することができます。作成したお仕事はお仕事一覧に保存され、いつでも編集、削除、配信などの操作が実施可能です。

他の SBR アカウントとの間でお仕事かんたん生成 2.0 のデータのインポートやエクスポートは可能ですが、旧お仕事かんたん生成 1.0 のデータをインポートしてのお仕事作成はできません。

13.4. キーワード

キーワードは、Pepper にセリフを設定する時に活用します。

キーワードとセリフについては次の表をご参照ください。

表 13.4-1 キーワードとセリフの関係性

キーワード	セリフ作成時の定型文です。キーワード登録画面で登録できます。キーワードは細かなイントネーション調整ができます。
セリフ	Pepper が実際に話す内容です。直接文字を入力してセリフを設定したり、キーワードを活用してセリフを設定したりすることができます。

Pepper が発話する内容を「セリフ」といいます。

セリフ作成時には、キーワードという定型文を登録することができます。キーワードは細かなイントネーション調整が可能であるため、Pepper により人間らしい発話をさせることができます。また、登録したキーワードは何度でも使用できるため、開発効率を向上させることができます。

ここでは、キーワード登録で使用頻度の高い設定をご紹介します。

13.4.1. 語尾を高くする

語尾を高くする場合は語尾に次の文字を入れてください。

- ・ ?
- ・ つ
- ・ ツ

例：

- ・ 話しかけてくださいね？
- ・ Pepper ですツツツ

13.4.2. キーワードに間を入れる

間を入れるにはキーワード/セリフの間に次の文字を入れてください。

間の長さは入れる文字数により調整可能です。

- ・ つ。 つ。
- ・ ツ。 ツ。
- ・ ツ、 ツ、

例：

- ・ こんにちはー。 つ。 つ。 つ。 ペッパーですーツツ。
- ・ こんにちはー。 ツ。 ツ。 ツ。 ペッパーですーツツ。

13.4.3. キーワードを登録する

アクセントの調整方法

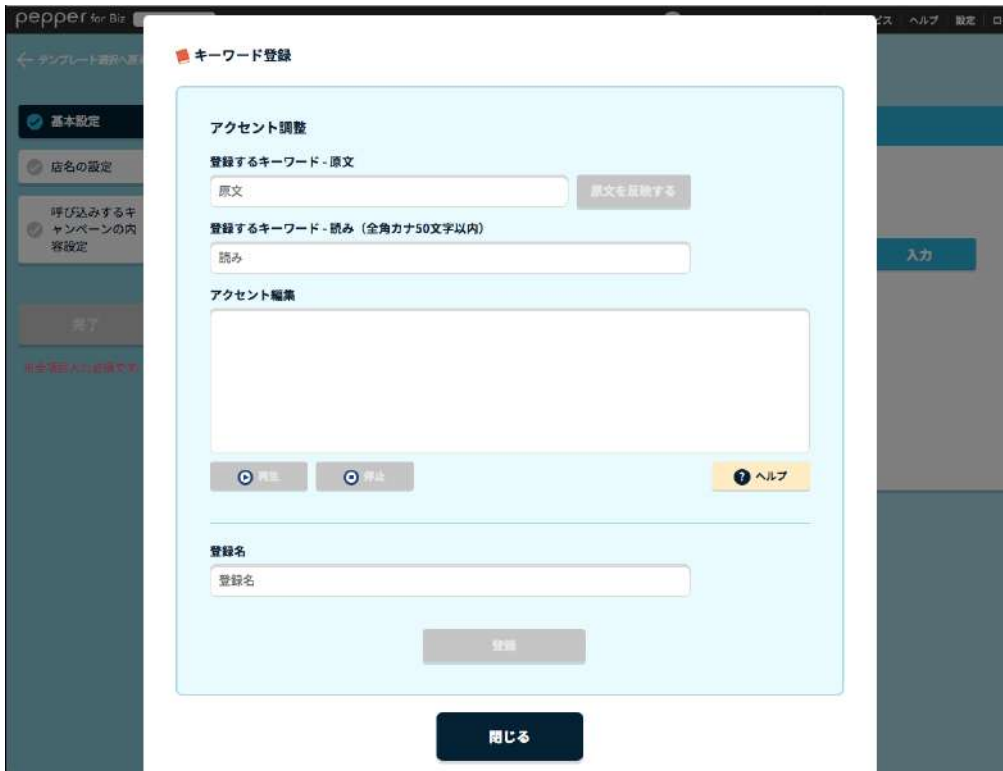


図 13.4-1 キーワード登録画面

登録するキーワードと「読み」を入力すると、アクセントの編集ができます。特に使用頻度の高い、音程の高低の編集方法には2種類あります。

文字上部のグラフをクリックすると、音程の高低を変更できます。

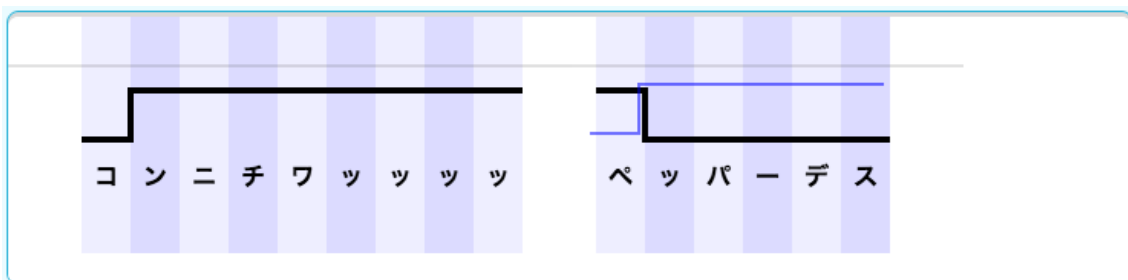


図 13.4-2 アクセント編集画面（文字上部のグラフをクリックした場合）

文字間をクリックすると、文字間に間が空き、区切り文言単位に音程の高低を変更できます。

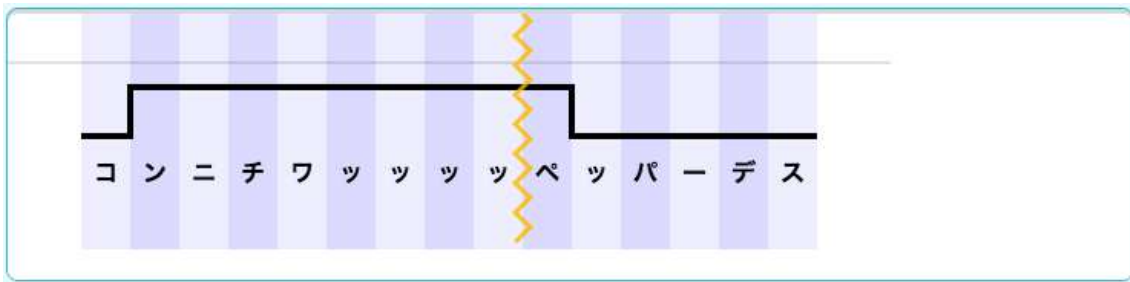


図 13.4-3 アクセント編集画面（文字間をクリックした場合）

「再生」ボタンを押下することで登録したキーワードを音声で確認することが可能です。Web サービス上での発話と実際の Pepper の発話が異なる場合がありますので、必ず実機での確認も実施してください。

キーワードの登録名を設定することができます。
登録名は自由に入力が可能で、キーワードの表記が同じになる同音異義語を判別する際に便利です。

13.5. メディアライブラリ

お仕事かんたん生成の中で使用する画像や音楽などの各種メディアファイルは、メディアライブラリに登録することで使用可能となります。ボックスの設定などの際、メディアファイルを使用する場面ではメディアライブラリ画面に遷移し、ライブラリ上にファイルを登録する、という手順が発生します。

対応するファイルの拡張子は mp4、gif、jpg、jpeg、png、ogg です。
音楽ファイルとして一般的に使用頻度の高い mp3 ファイルに対応していないことに注意が必要です。

また、動画ファイルの音量の設定はこの画面ではできません。
設定した動画ファイルの音量が小さい場合、Pepper 本体の「基本情報」画面でタブレットの音量を調節するか、動画自体の音量をあげてから再度アップロードする必要があります。

13.6. お仕事の編集

既に登録しているテンプレートの動きやお仕事の内容を更新する場合には、お仕事編集画面で編集を行います。お仕事の編集を行うには、編集権限でログインする必要があります。閲覧権限でログインしている場合には編集はできず、閲覧のみ可能です。

お仕事かんたん生成 2.0 のトップ画面で「お仕事を編集・設定する」をクリックすることで、お仕事一覧画面に遷移します。



図 13.6-1 お仕事かんたん生成 2.0 トップ画面

お仕事一覧画面では作成したお仕事のデータを一覧で確認可能です。
 お仕事名の編集やお仕事の配信、お仕事データの複製や削除、配信期間の設定などは一覧画面からでも実行できます。

ボックスの配置や設定内容など、作成したお仕事の詳細を編集する場合には、お仕事一覧画面で編集したいお仕事の「確認・編集」をクリックすることでお仕事編集画面に遷移します。

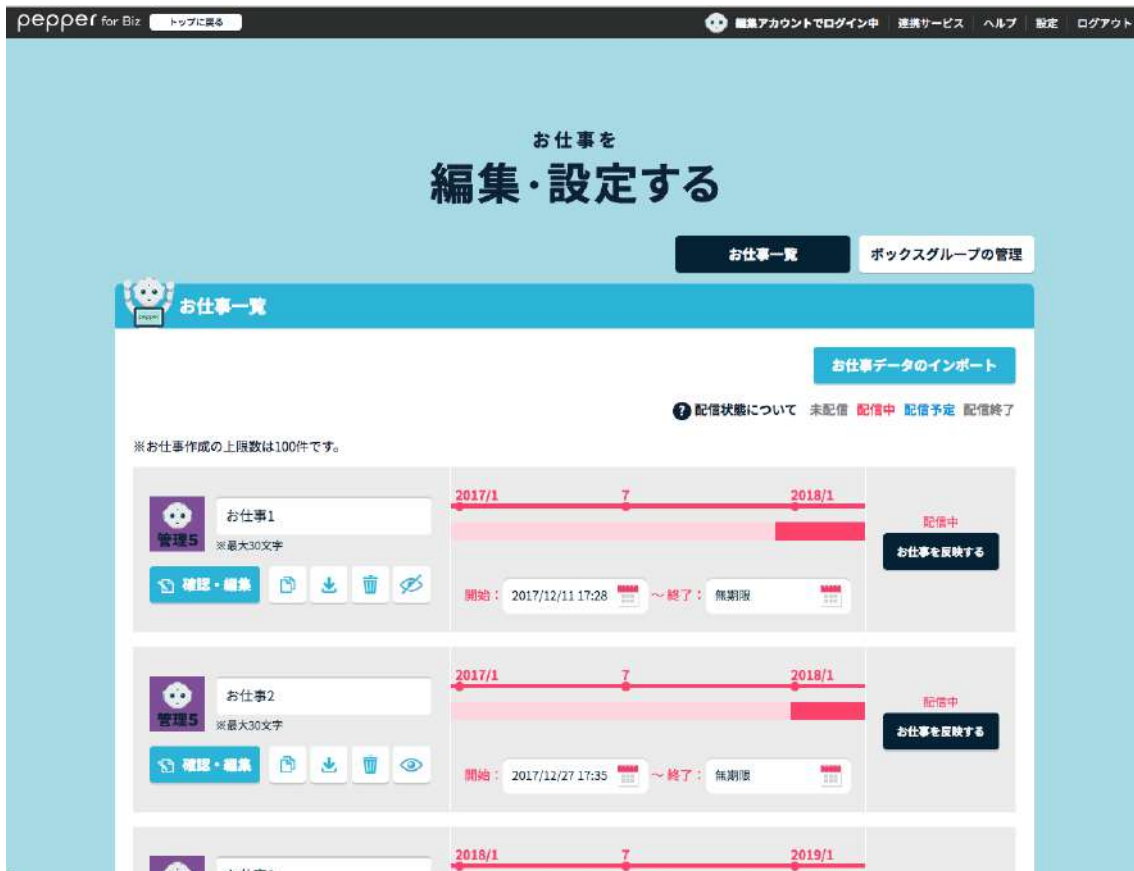


図 13.6-2 お仕事一覧画面

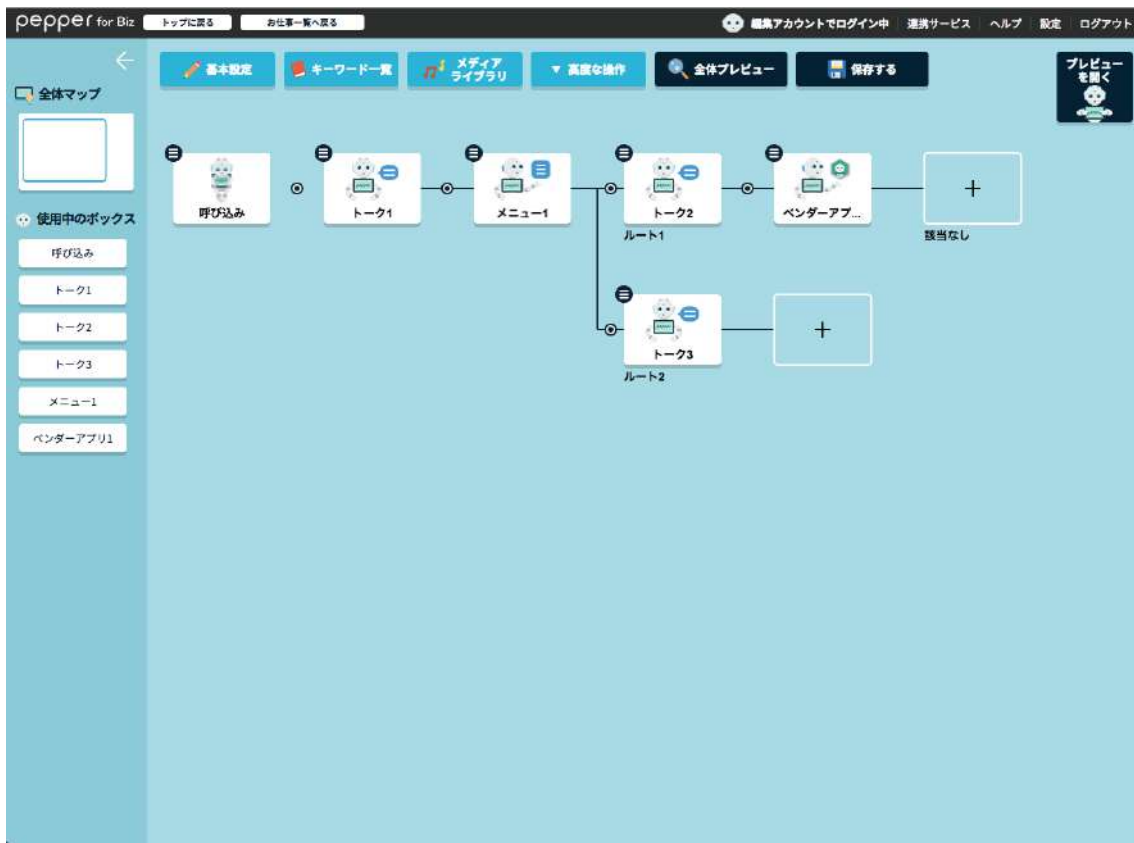


図 13.6-3 お仕事編集画面



図 13.6-4 お仕事編集画面（基本設定）

お仕事編集画面の「基本設定」画面では、「お仕事名」「お仕事アイコン」「お仕事全体の背景画像」を設定することができます。

13.7. ボックスの操作

お仕事かんたん生成 2.0 では、「ボックス」という機能の単位を自在に組み合わせることで業務に合わせて多様なお仕事を作成することができます。お仕事を作成する際には、お仕事編集画面でボックスを配置します。

ボックスには「呼び込みボックス」「基本ボックス」「応用ボックス」の 3 種類があり、「呼び込みボックス」は必ずお仕事の最初に 1 つだけ配置される決まりになっています。個別開発したロボアプリを組み込む場合には、「バンダーアプリボックス」を使用します。ここでは、複数ボックスに共通する操作に加え、各ボックスの機能の概要と把握しておくべき事項を解説します。

13.7.1. 各ボックスでの編集画面の共通操作について

ボックスは Pepper にさせることができる機能や条件分岐を設定する単位です。ランダム分岐ボックスのように、ボックスは配置するだけでも使用可能なものもありますが、ほとんどが「ボックス編集画面」にて個別に設定が必要なものとなっています。

ボックス編集画面では上部に設定項目のタブが表示され、各タブを設定してボックスを作っていきます。タブの中には複数のボックスで共通して設定が必要になる項目があります。ここでは、よく使用される設定項目について解説します。

13.7.2. ディスプレイ設定

Pepper の胸のタブレットに表示する内容を設定する項目で、タブレットを使用する動作のボックスで設定が必要になります。ディスプレイに設定できる画像/動画は1つです。

なお、横 2048px × 縦 2048px より大きな画像をタブレットに表示することはできません。



図 13.7-1 ディスプレイ設定画面

また、Pepper がお仕事をする時、胸のディスプレイに設定する内容として頻繁に使用するのが、選択肢の表示を行うケースです。メニューボックスや質問ボックスなど、お客様にディスプレイ上の選択肢をタッチしていただくことによって行動を分岐させるようなボックスの編集画面では、選択肢のディスプレイ上の表示方法も設定する必要があります。

ディスプレイ設定画面で設定可能な項目として、選択肢ボタンの背景画像や背景色、ボタンの枠線の色などがあります。選択肢の文言のフォントサイズは、選択肢のレイアウトと文言の文字数によって自動で設定されるため、自由に設定することはできません。

13.7.2.1. セリフ設定

Pepper に発話させる動作を含むボックスでは、発話するセリフについて詳細な設定を行うことができます。Pepper の最大の魅力である「人間らしさ」を表現する重要な項目です。



図 13.7-2 セリフ設定画面

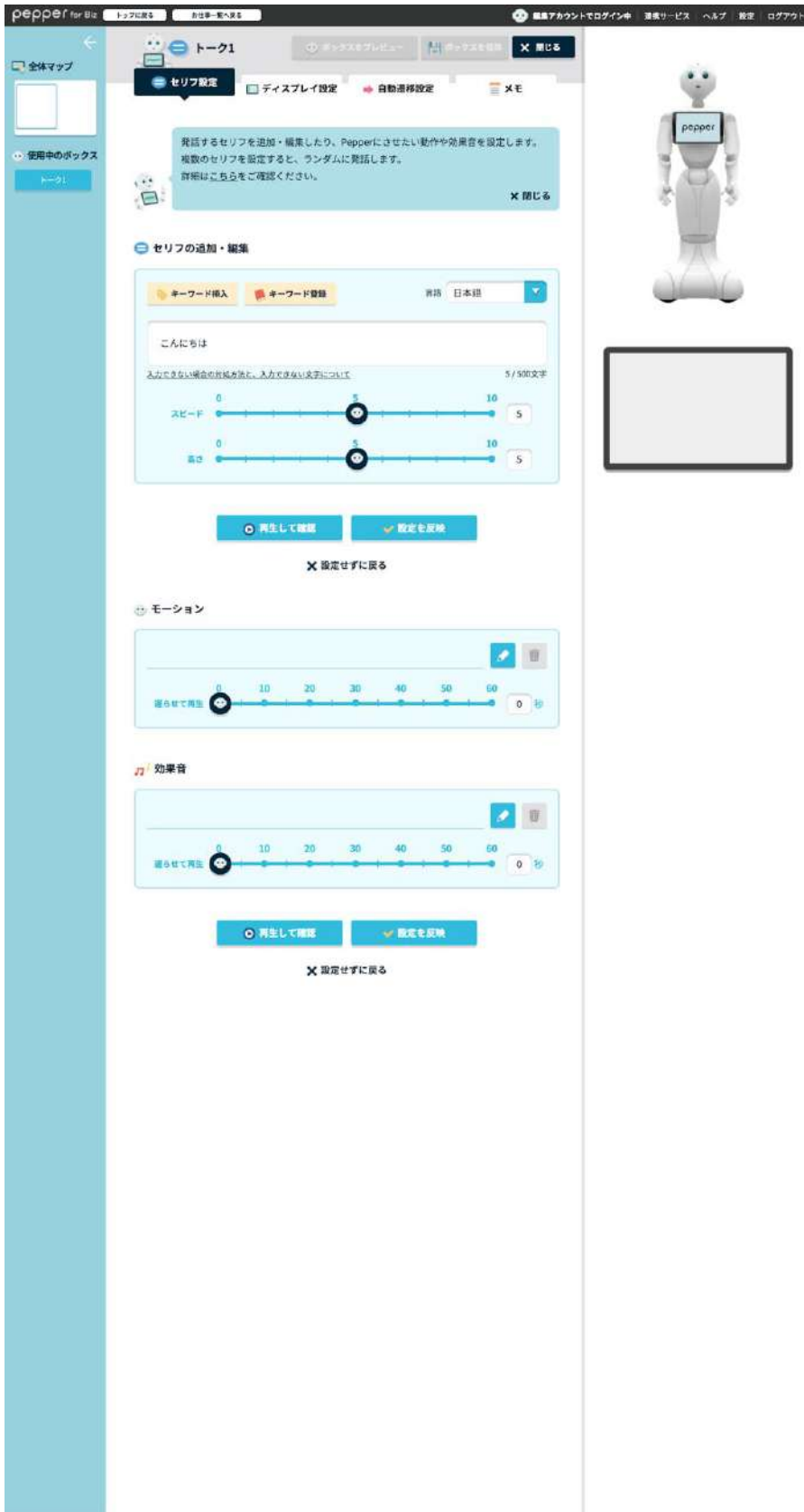


図 13.7-3 自由に入力して追加画面

セリフの設定方法はカテゴリごとに用意されているテンプレートから選択して追加する方法と、発話内容を自由に作成し、追加する方法があります。呼び込みや挨拶などの汎用的なセリフの場合にはテンプレートを利用することで設定にかかる時間を短縮することが可能です。

「自由に入力して追加」画面でセリフを編集する際には、言語設定が日本語の場合のみ、13.4 章で紹介したキーワードを挿入できます。よく使う言葉はキーワード登録しておくことで都度の設定が不要になるため、設定に係る効率を上げられますので積極的に利用すると良いでしょう。発話内容の他に声の高さやスピードもセリフ設定画面で編集することができますが、セリフの音量は設定することができません。音量は **Pepper** 本体の基本設定画面で設定した内容が適用されます。

また、セリフと合わせて **Pepper** のモーションや胸のタブレットに表示させる画像、効果音なども設定することができます。発話内容やモーションは「再生して確認」でプレビュー可能ですが、英語・中国語の発話はプレビューによる確認ができないため、**Pepper** 本体で確認する必要があります。

13.7.3. 呼び込みボックス

お仕事編集画面でボックスを配置してお仕事の作成を行う際、必ず画面の左端に配置され、最初に 1 度だけ実行されるのが呼び込みボックスです。

呼び込みボックスは、**Pepper** による呼び込み動作を設定することができます。お客様の関心を引くための重要なボックスです。

また、「誰ともコミュニケーションを取っていない状態から、呼び込みを行い後続のボックスに遷移させることが可能」という特性上、1 つのお仕事内に複数配置することはできません。後続のボックスへ遷移させず、呼び込みのみでアイキャッチの役割をさせることも可能です。

呼び込み時のディスプレイ表示やセリフ、モーションなどを設定することができます (13.7.1 章を参照)。

このボックスで特に重要な、遷移条件設定について詳述します。



図 13.7-4 呼び込みボックス（遷移条件設定）

遷移条件設定のタブでは、次のボックスに遷移するトリガーとなる動作を選択することができます。

選択肢として、以下の4つから動作を選ぶことができます。

- ・お客様の顔を認識して遷移する。
- ・ディスプレイをタッチされると遷移する。
- ・お客様の顔を認識、ディスプレイをタッチいずれかで遷移する。
- ・遷移せず、呼び込みのみを繰り返す。

13.7.4. トークボックス

Pepper からお客様へ話しかけるためのボックスです。

商品説明など、お客様からのレスポンス動作を必要としない動作を設定する場合に配置することが多いです。話しかけるセリフとやモーション、ディスプレイの画面設定、自動遷移に関する設定が可能です。

トークボックスを配置した際には、内容の設定を行わないと保存することができません。



図 13.7-5 トークボックス（自動遷移設定）

トークボックスの自動遷移設定は、呼び込みボックスの自動遷移設定と異なり「ON」「OFF」の2択となっています。

- ・「ON」にしていた場合には、ディスプレイのタッチがない状態で一定時間が経過すると、次に設定しているボックスに自動遷移します。
- ・「OFF」にしていた場合には、ディスプレイをタッチすることによって画面遷移します。

13.7.5. メニューボックス

メニューボックスは、Pepperのディスプレイにメニューボタン（選択肢ボタン）を表示し、お客様の選択によってルートを分岐させることができるボックスです。

お客様の選択は、ディスプレイ上のメニューボタン（選択肢ボタン）のタッチだけでなく、音声による選択も設定することができます。

メニューボックスでは、お客様によるディスプレイのタッチや音声によるメニューの選択がない状態で一定時間が経過すると、お仕事の先頭に遷移します。メニューの選択なしに後続のボックスへ自動遷移をすることはありませんので、注意が必要です。

13.7.6. 質問ボックス

質問ボックスは、Pepperからお客様に質問をして、回答によりルート分岐を設定することができるボックスです。

質問ボックスでは、質問内容と、回答の内容(選択肢)を設定します。ディスプレイに質問文を表示しながら Pepperが発話したり、ディスプレイタッチによる回答や音声認識による回答を受け付けたりすることが可能です。

ディスプレイタッチによる回答は、ディスプレイに回答選択肢ボタンを表示することで設定することができます

音声認識による回答は、音声認識の単語の設定により設定することができます。

その他、ボックス内での効果音も設定することが可能です。

質問ボックスの設定画面では、選択肢ボタンを 2 つ以上設定することができます。ボタンの順番を並び替えたり、ナビのボーダー色を個別に設定することができますが、ボタン設定をコピーしたり、背景画像の設定などを個別に設定することはできませんので、注意が必要です。

13.7.7. 公式アプリボックス

公式アプリボックスでは、ロボアプリ配信管理で予め配信してある状態の基本アプリやマーケットアプリなどの公式アプリを Pepper で起動させるためのボックスです。

設定は大きく分けて、起動するアプリ自体に関する設定と、アプリ終了後の分岐に関する設定の 2 段階があります。



図 13.7-6 公式アプリボックス（アプリ設定画面）

13.7.8. 印刷ボックス

Pepper に登録したプリンタで画像を印刷するためのボックスです。

画像はあらかじめ、印刷ボックスで設定しておく必要があります。

お仕事かんたん生成 2.0 で使用するプリンタは、管理メニュー画面の「プリンタ設定」から行います。プリンタを選択するとプリンタの IP アドレスが自動的に設定されます。



図 13.7-7 管理メニュー画面

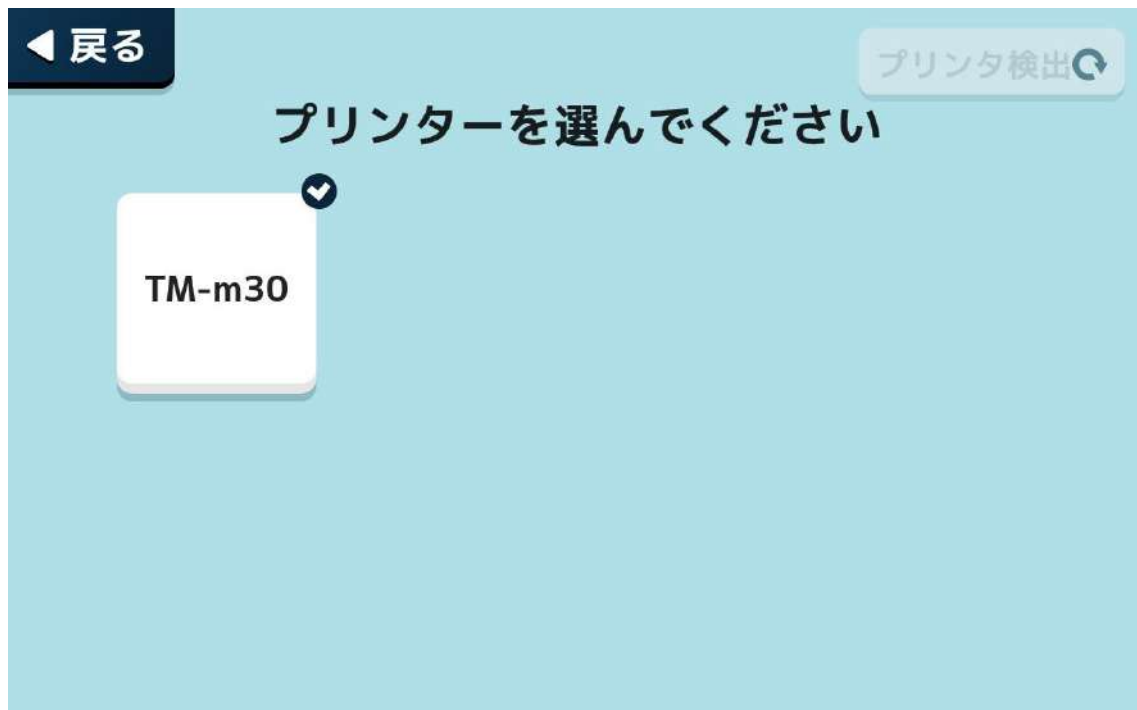


図 13.7-8 プリンタ設定画面

13.7.9. ジャンプボックス

ジャンプボックスは、指定したボックスに移動するためのボックスです。
例えば、お客様の回答結果を受けてはじめに戻ったり、2 つ先のボックスに移動したりすることができます。

13.7.10. ベンダーアプリボックス

ベンダーアプリボックスを使用することで、Choregraphe などを用いて独自開発したアプリケーション(マイアプリ)を Pepper のお仕事に組み込むことができます。

公式アプリボックスと同様、ベンダーアプリボックスにマイアプリを組み込む場合にも、事前に対象アプリをロボアプリ配信管理から配信しておく必要があります。



図 13.7-8 ベンダーアプリボックス (アプリ設定画面)

ベンダーアプリボックスでアプリを起動するために、アプリ設定画面にて設定を行う必要があります。

- ・アプリの識別は、アプリ名称ではなく、ビヘイビア名で行っています。そのため、アプリを利用するには、マイアプリのビヘイビア名(ビヘイビアパスともいいます)を登録する必要があります。ビヘイビア名を誤って入力してもエラー表示はされませんので、注意が必要です。

- ・ビヘイビア名はアプリ開発者によって設定するものです。Choregraphe 上で確認が可能ですので、自作のアプリケーションの場合には Choregraphe からの確認を行います。他者による開発アプリケーションの場合には、開発者への問い合わせが必要となります。

・ロボアプリ配信管理でアプリ配信していないアプリのビヘイビア名をベンダーアプリボックスで設定した場合、ボックスの保存時にエラー表示は行われません。そのため、そのボックスを使用したお仕事も保存が可能であり、お仕事かんたん生成 2.0 から Pepper への配信自体も実行できてしまいます。配信した上記のお仕事を実際に Pepper 本体にダウンロードした際に、エラーメッセージが表示される仕様になっている点は開発者として押さえて置くべきポイントです。

13.7.11. 属性分岐ボックス

属性分岐ボックスを使用することで、お客様の年齢や性別を判別し、後続の動作を分岐させることができます。例えば、「0歳～20歳までの男性」「31歳～50歳までの女性」など、年齢と性別を自由に組み合わせて分岐条件を作成し、お客様の年代や性別に合わせた商品やサービスの紹介などを行うお仕事を作成することが可能になります。



図 13.7-9 属性分岐ボックス（分岐設定画面）

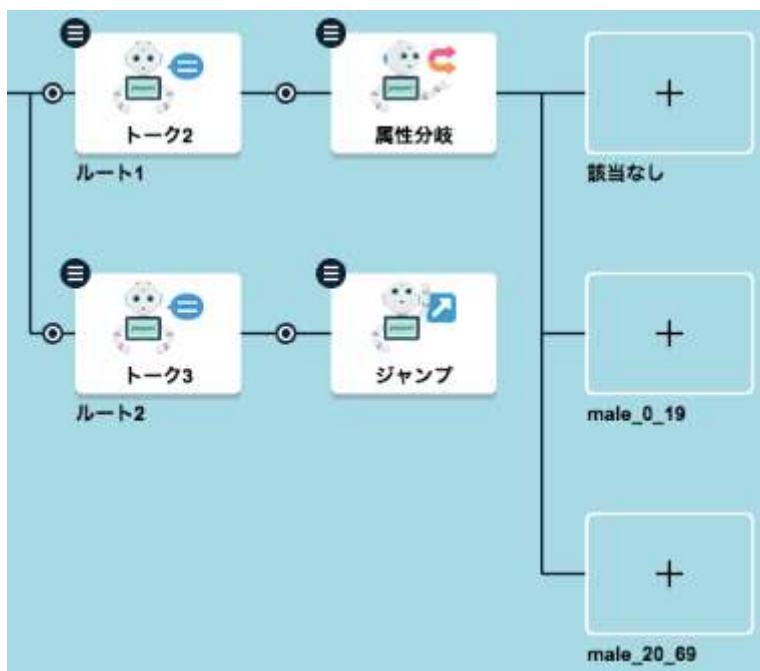


図 13.7-10 お仕事編集画面（該当無しルート）

属性分岐ボックスでルートの分岐を作成すると、作成した分岐に加えて「該当なし」というルートが作成されます。お客様の属性が作成した条件にマッチしない場合には、「該当なし」のルートに遷移する仕様になっています。対象とする条件に当てはまらないお客様の場合にも、お仕事の終了までのルートを考慮して設計することは、開発者として非常に重要なスキルであるともいえます。

13.7.12. ランダム分岐ボックス

ランダム分岐ボックスでは、任意の確率を設定してランダムに動作を分岐させることができます。複数の商品やサービスの紹介をランダムに行いたい場合や、属性にとらわれずに Pepper のトークや質問をランダムに分岐させたい場合などに配置します。ルート分岐の確率は百分率で設定することができます。

お仕事編集画面でランダム分岐ボックスを配置した際には、デフォルトで2つのルート分岐が作成されます。ボックスの編集を行わなかった場合には 50%ずつの確率で後続の分岐に進む仕様となっています。任意の分岐数及び分岐する確率を設定したい場合には、ボックスの編集と保存が必要です。

13.8. お仕事の管理

普段の店頭営業用に複数パターンのお仕事を用意しておきたい、イベント用に専用のシナリオで作成したお仕事を配信したい、などのニーズに応えることができるのがお仕事データの管理です。

作成したお仕事をストックしておき必要に応じて Pepper に配信する、お仕事データを複製して編集する形で別のお仕事を作成する、別の SBR アカウントで作成したお仕事をインポート/エクスポートする、お仕事の閲覧に制限をかけて権限を分ける、など様々な管理項目があります。

使用の用途や目的に応じて、お仕事データを適切に管理するために特に重要な項目について、下記で解説します。

13.8.1. お仕事を管理する

お仕事の管理は、「お仕事一覧画面」で行います。

お仕事一覧画面では、作成・保存したお仕事の一覧管理、編集、配信、お仕事名の変更などを行うことができます。お仕事作成の上限数は最大 100 件までとなっており、超過する場合には削除の対応が必要になります。

ひとつのお仕事の最大容量は 100MB で、Pepper 本体には最大で 300MB のお仕事データが保存可能です。300MB を超える場合には、一番古いお仕事削除されて新しいお仕事のダウンロードが開始されます。この場合、一番古いお仕事のデータは Pepper 本体から削除されていますが、お仕事一覧上には残っている状態となります。

お仕事一覧上(Web サービス上)にお仕事保存されていれば、何度でも配信することが可能です。

13.8.2. お仕事一覧画面の見かた



図 13.8-1 お仕事かんたん生成 2.0 トップ画面



図 13.8-2 お仕事一覧画面

お仕事かんたん生成のトップ画面で「お仕事を編集・設定する」メニューを選択することでお仕事一覧画面に遷移します。作成したお仕事データが一覧管理できる画面になっており、該当のお仕事の「確認・編集」ボタンを押下することでお仕事編集画面に遷移し、編集を行うことができます。お仕事作成・編集時にお仕事編集画面上で設定したお仕事名は、このお仕事一覧画面上でも変更することができます。

お仕事一覧で管理できる(作成データを保存できる)上限は 100 件です。

閲覧専用アカウントでログインしている場合には、「閲覧アカウントによる閲覧禁止」に設定されたお仕事は一覧上に表示されなくなります。

13.8.3. お仕事を複製する

既に作成してあるお仕事をコピーして新たなお仕事を作成する場合には、お仕事一覧画面で該当のお仕事を複製することができます。

お仕事一覧画面で複製ボタンを押下することで、お仕事の複製を行うことができます。複製したお仕事は「(複製元のお仕事名)のコピー」という名称になります。このお仕事名はお仕事一覧画面上で変更可能です。

13.8.4. お仕事の閲覧に制限をかける

お仕事かんたん生成の基本設定時に設定した編集権限パスワードを入力せずにログインし

た場合、閲覧アカウントでのログインとなります。閲覧アカウントではお仕事の編集はできませんが、基本的に閲覧可能となっています。

編集権限を持たない人による特定のお仕事の閲覧自体を制限したい場合、お仕事一覧画面にて「閲覧アカウントによる閲覧禁止」を設定することができます。

「閲覧アカウントによる閲覧禁止」に設定すると、閲覧アカウントでのログイン時、お仕事一覧画面に該当のお仕事が表示されなくなります。

13.8.5. お仕事の配信期間を設定する

お仕事一覧画面では、お仕事の配信期間を設定することができます。配信期間終了後のお仕事は、Pepper でお仕事を選択する時に表示されなくなります。あえてお仕事の終了日を過去の日付に設定することで、該当のお仕事を「配信停止」状態にすることも可能です。

上記を利用して、以下のように「お仕事1」の終了時間と「お仕事2」の開始時間を同じにすることで、お仕事を自動で切り替えて実行させることができます。

例：

お仕事1 → 2018/1/1 00:00~ 2018/1/1 23:59

お仕事2 → 2018/1/2 00:00~ 2018/1/2 23:59

⇒この場合「2018/1/2 00:00」に自動的に「お仕事1」→「お仕事2」に切り替わります。

13.8.6. お仕事データをエクスポート／インポートする

お仕事かんたん生成 2.0 では、お仕事データのエクスポートやインポートを行うことができます。インポートしたお仕事は作成したお仕事と同様、複製や編集を行えます。

お仕事データをエクスポートする際は、下の図 13.8-3 の通り、お仕事一覧画面でお仕事データのダウンロードボタンを押下し、お仕事データの zip ファイルを保存します。

お仕事データのインポートは、下の図 13.8-4・図 13.8-5 の通り、お仕事一覧画面で「お仕事データのインポート」を押下し、ポップアップ画面でお仕事名を入力、「ファイル参照」からインポートするデータを選択し「インポート」を押下することで実行できます。

インポート可能なお仕事データのファイル形式は「zip」形式のみです。また、旧お仕事かんたん生成 1.0 からエクスポートしたデータはインポートすることはできません。

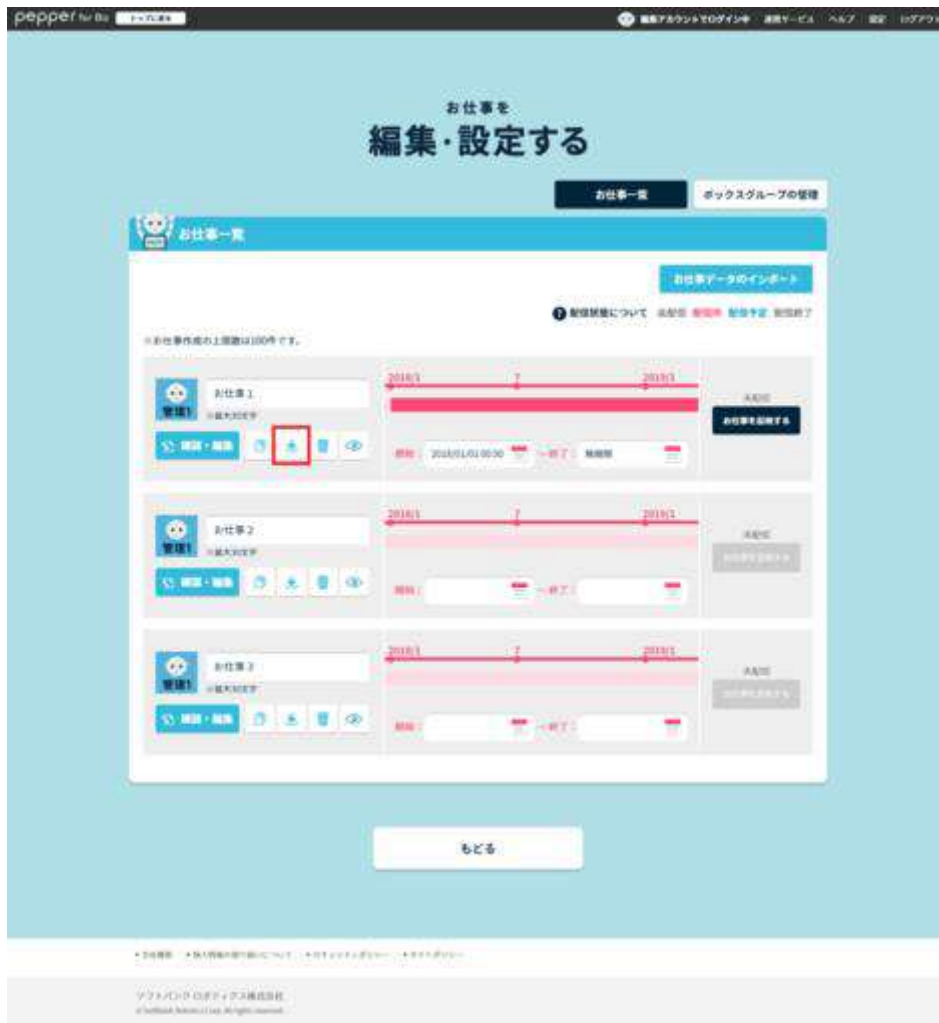


図 13.8-3 お仕事一覧画面（お仕事のエクスポート）



図 13.8-4 お仕事一覧画面（お仕事データのインポート①）



図 13.8-5 お仕事一覧画面 (お仕事データのインポート②)

13.9. ロボアプリの配信

お仕事かんたん生成 2.0 で、公式アプリボックスやベンダーアプリボックスをお仕事内に配置してロボアプリを組み込む場合、事前に「ロボアプリ配信管理」という Web サービス上で該当のロボアプリを配信しておく必要があります。

ロボアプリ配信管理には「アプリリスト」「パートナーアプリ」「マイアプリ」の 3 つのタブがあり、それぞれに配信できるアプリの種類が異なります。

アプリを使用しなくなった場合の配信停止も「ロボアプリ配信管理」から行うことができます。

13.9.1. アプリリストから配信する

ロボアプリ配信管理の「アプリリスト」タブからは、予め用意されているエンタメ向けのロボアプリを配信することができます。配信したアプリは「公式アプリボックス」や「ベンダーアプリボックス」で利用することができます。

13.9.2. パートナーアプリを配信する

ロボアプリ配信管理の「パートナーアプリ」タブでは、パートナーアプリ（パートナー認定企業から提供を受けたロボアプリ）を配信することができます。

13.9.3. マイアプリを配信する

ロボアプリ配信管理の「マイアプリ」タブでは、マイアプリ（独自開発のアプリケーション）を配信することができます。自社開発のロボアプリなどはこれにあたります。配信したアプリはベンダーアプリボックスで利用できます。

13.10. Pepper 側でのお仕事の更新

お仕事かんたん生成 2.0 上で作成、配信したお仕事は即時で Pepper に反映されるわけではありません。イベント中などに急遽お仕事内容を編集する必要がある場合など、Pepper 側のお仕事の更新タイミングを押さえておくことが肝要です。

13.10.1. Pepper 側でお仕事が更新されるタイミング

お仕事かんたん生成 2.0 で作成したお仕事が更新された場合、Pepper には次のタイミングで反映されます。

- ・ Pepper 本体のディスプレイで「お仕事の選択」から更新したお仕事を選択した時。
- ・ Pepper がお仕事実行中に、お仕事に変更があった時（更新確認は 60 秒間隔で行います）。

14. インタラクション分析

14.1. インタラクション分析

Pepper で取得したデータをクラウドに蓄積し、管理画面で「見える化」することにより、集客施策の効果測定など、データに基づいた分析が可能になります。

取得した統計情報は、以下のサイトにて確認することができます。

[インタラクション分析のサイト]

<https://softbankrobotics.com/portal/interaction-analytics/auth/login>

14.1.1. 取得できるデータについて

インタラクション分析で取得可能なデータの種類と表示粒度は以下の通りです

表 14.1-1 インタラクション分析で取得可能なデータ

種類	内容	表示粒度
接客件数	Pepper が接客した回数の合計	回 ×機体×お仕事×性別×世代×期間
平均接客時間	Pepper が接客した 1 回あたりの平均接客時間	秒 ×機体×お仕事×性別×世代×期間
採用件数	お仕事が Pepper に設定された件数	件 ×機体×お仕事×性別×世代×期間
コンバージョン件数	お仕事がコンバージョンに至った件数	件 ×機体×お仕事×コンバージョンポイント×性別×世代×期間
離脱数	仕事の途中でユーザーが離れた件数	件 ×機体×お仕事×ボックス×性別×世代×期間
質問回答数	お仕事内の質問に回答があった件数	件 ×機体×お仕事×年代×性別×期間

年代	Pepper がコミュニケーションした人の世代（認識結果）	19 歳以下/20 代/30 代/40 代/50 代/60 代以上/不明 ×機体×お仕事×性別×期間
性別	Pepper がコミュニケーションした人の性別（認識結果）	男性/女性/不明 ×機体×お仕事×世代×期間

14.1.2. インタラクション分析で表示される情報

インタラクション分析では以下のような情報を閲覧することができます。適切な分析種別を使用することで、効果的に Pepper を活用することができます。下の表 14.1-2 は非常に重要なポイントとなりますので、それぞれの分析種別の特徴と違いを押さえておきましょう。

なお、感情についてはインタラクション分析で表示することはできません。

表 14.1-2 インタラクション分析で表示される情報

種別	内容
接客分析	接客件数を性別、機体別、お仕事別で分析でき、平均接客時間も集計可能。
お仕事分析	お仕事別に採用件数、接客件数、コンバージョン件数、離脱数、平均接客時間を分析可能。
コンバージョン分析	コンバージョンポイントを設定した箇所の各分析が可能。
離脱分析	Pepper のお仕事の途中で離れてしまった状況を分析するため、離脱件数を機体別、お仕事別、お仕事をどこまでやって離脱したかを年代、性別のカテゴリから分析可能。
質問分析	Pepper のお仕事の中で設定した質問ごとに各回答の件数や回答したユーザーの属性（年代/性別）で分析が可能。質問分析は複数のお仕事のレポートを表示することができます。分析の対象ボックスは「質問ボックス」と「メニューボックス」です。「メニューボックス」では、ボックス名が円グラフのタイトルとなります。

15. Pepper for Biz

Pepper のモデルは、一般販売モデルと Pepper for Biz モデルがあります。一般販売モデルは家庭向けで、家族で楽しめるアプリが数多く用意されています。Pepper for Biz モデルは店舗向けで、Choregraphe を使用して独自アプリを作成しなくても、クラウドサービスから簡単に接客や受付アプリを作成できます。この節では、一般販売モデルと Pepper for Biz の違いをまとめます。

15.1. 一般販売モデルとの比較（仕様編）

仕様上の違いは以下の通りです。

表 15.1-1 仕様の比較

項目	Pepper for Biz	一般販売モデル
----	----------------	---------

感情エンジン (Peppe 身が感情を持つ)	△ 感情エンジンは実装されており、マイアプリで使用可能。	○ コミュニケーションの状況に応じてポジティブにもネガティブにもなる。
クラウド AI (データを集積して成長)	× for Biz では情報をより慎重に取り扱う必要があるため、現在は未使用。	○
雑談	△ 「握手」「おはよう」など 10 種類程度の呼びかけに返答。	△～○ ある程度の雑談、自由会話は可能。
マイアプリ (個別開発アプリ)	○ 「お仕事かんたん生成」に適応するように作成する必要あり。	△ 一から全てオリジナルで作成することができるが、配信は不可。
お仕事かんたん生成	○ 管理者の Web ブラウザ上で商品紹介などが簡単に作成できる。	× 個別にアプリ開発が必要。
インタラクション分析	○	×
電話サポート	○	○
故障時の交換対応	○ 通常利用の範囲内であれば何度でも無償交換が可能。	× 初期不良交換以外は自然故障も含め修理費が発生。
有償サポート	○ 環境調査、初期設定のサポート、管理者向け研修などをご用意。	×
Interactive ロボアプリ	○	○
Solitary ロボアプリ	×	○
複数のロボアプリが連携するロボアプリ	○ お仕事かんたん生成 2.0 より対応。	○
起動トリガー条件	×	○
起動センテンス	×	○

15.2. 一般販売モデルとの比較 (開発条件編)

開発条件的な違いは以下の通りです。

表 15.2-1 開発条件の比較

項目	一般販売モデル	Pepper for Biz
性質	Solitary / Interactive	Interactive

起動方法	ディスプレイ/トリガーセン テンス/起動トリガー条件	ディスプレイ
アプリ間連携	○	○ お仕事かんたん生成 2.0 より対 応。
インストール方法	PC から。	クラウドサービスから。

15.3. ロボアプリ品質チェックリスト

Pepper for Biz 向けロボアプリの開発者に対して「ロボアプリ品質チェックリスト」を公開しております。「ロボアプリ品質チェックリスト」では、ロボアプリを正しく登録・動作させること、利用者にとって使いやすいアプリとなるための基本指針などが掲載されており、「マイアプリ開発ガイドライン」、「マイアプリ開発ガイドライン（解説）」に代わるものとして用意された資料となります。ロボアプリパートナー (Basic) 認定試験では、「マイアプリ開発ガイドライン」、「マイアプリ開発ガイドライン（解説）」と内容に相違がある場合は、「ロボアプリ品質チェックリスト」の内容を優先とします。

基本指針については主に以下の項目があります。

- 全分岐網羅試験
- 負荷耐久試験
- 実環境試験
- UX 観点レビュー
- ソースレビュー
- manifest 確認

関連情報は下記 URL にてご確認ください。

表 15.3-1 ドキュメントの URL

ドキュメント名	URL
ロボアプリ品質チェックリスト	https://www.softbank.jp/robot/developer/dev-support/documents/

15.4. ロボアプリ配信管理

Pepper for Biz モデルでは、ロボアプリ配信管理でマイアプリとしてアップロードを行い、ロボアプリをクラウド経由で配信することができます。

ロボアプリ配信管理でのアプリの配信する方法は以下の通りです。

1. ロボアプリ配信管理にアクセスします。
2. 登録された SBR アカウントでログインします。
3. [マイアプリ] タブをクリックします。
4. [アプリを登録] ボタンをクリックします。

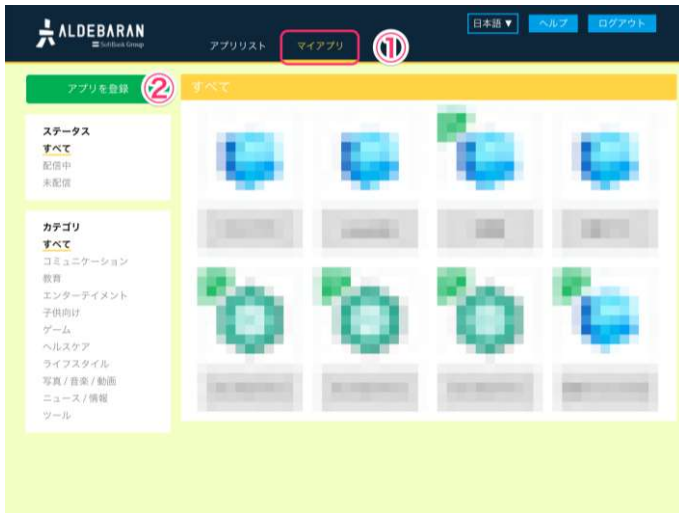


図 15.4-1 アプリを登録

5. アップロードファイルを指定し、任意の[カテゴリ]を選択し、任意の[コメント]を記載し、[登録]ボタンをクリックします。
6. アップロードされたロボアプリをクリックします。
7. [配信開始]ボタンをクリックします。



図 15.4-2 [配信開始]ボタン

詳細な手順については、『メニューで利用するロボアプリを配信する（管理者）』（<http://help.mb.softbank.jp/robot/pepper-for-biz/pc/06-04.html>）を参照してください。

15.5. ビヘイビアパス

ビヘイビアパスは、お仕事かんたん生成に登録する際に必要となります。マイアプリロボアプリ配信管理にアップロード後、ビヘイビアパスと呼ばれるロボアプリ実行のための情

報が必要です。ビヘイビアパスはマイアプリを開発したベンダー様がロボアプリを利用するお客様に伝えてください。

15.6. ソフトウェアの違い

Pepper for Biz ではお仕事かんたん生成のクライアントアプリがインストールされているため、マイアプリはクライアントアプリ（下図水色）と協調して動作させる必要があります。

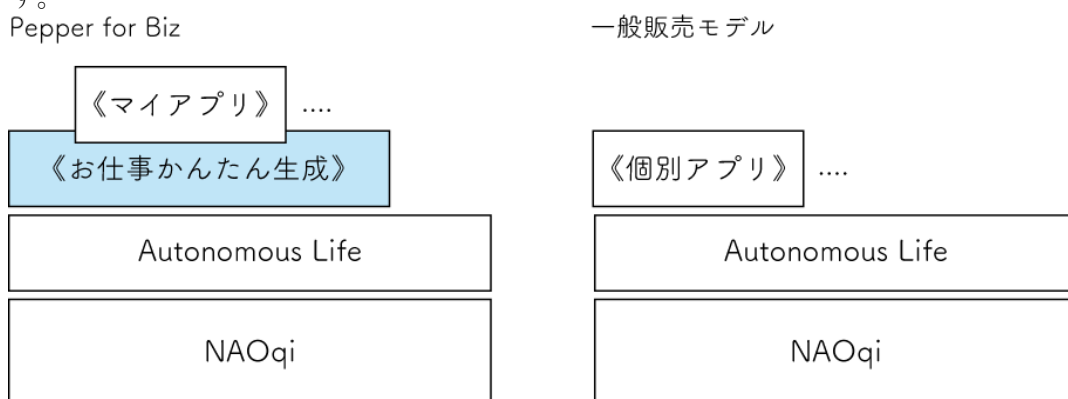


図 15.6-1 Pepper for Biz と一般販売モデル

マイアプリとはお仕事かんたん生成のクライアントアプリの中で動作させる、アプリプラグインのことをいいます。マイアプリとして開発されたロボアプリをお仕事かんたん生成に登録すると、独自開発したロボアプリが実行できます。

マイアプリはユーザ様が開発する場合がありますが、多くは専門知識を持ったベンダー様が開発されることとなります。何れの場合も「ロボアプリ品質チェックリスト」に準拠してください。

15.7. ロボアプリ公開申請

本プログラムによりパートナー認定された企業様は、ロボアプリ公開申請を受けることができます。ロボアプリ公開申請では、前述した「ロボアプリ品質チェックリスト」を全て満たし、且つロボアプリの内容に問題がないことが合格条件となります。

マイアプリは一般的なロボアプリとほぼ同じ構成をしていますが、マイアプリ用にロボアプリを修正する必要があります。ここでは、特別な事項について解説します。

15.7.1. manifest.xml の修正

manifest.xml はロボアプリの基本情報や実行条件などが記載されている重要なファイルです。Choregraphe にて自動生成されますが、不要な XML エレメントが生成されることがあるため、以下に示すファイルのようになっているか必ず確認してください。

表 15.7-1 テンプレートファイル (manifest.xml)

```
[?xml version='1.0' encoding='UTF-8?']
```

```

<package version="1.0.0" uuid="biz_softbank_pepper-sample">
  <names>
    <name lang="en_US">Pepper Sample Application</name>
    <name lang="ja_JP">サンプルアプリ</name>
  </names>
  <descriptions>
    <description lang="ja_JP">マイアプリ実装のためのサンプルアプリです。</description>
  </descriptions>
  <supportedLanguages>
    <language>ja_JP</language>
  </supportedLanguages>
  <descriptionLanguages>
    <language>en_US</language>
    <language>ja_JP</language>
  </descriptionLanguages>
  <contents>
    <behaviorContent path=".">
      <nature>interactive</nature>
      <names>
        <name lang="en_US">Pepper Sample Application</name>
        <name lang="ja_JP">サンプルアプリ</name>
      </names>
      <triggerSentences/>
      <loadingResponses/>
      <autonomous/>
      <permissions/>
    </behaviorContent>
  </contents>
  <requirements>
    <naoqiRequirement maxVersion="2" minVersion="2.5"/>
    <robotRequirement model="JULIETTE_Y20"/>
  </requirements>
</package>

```

15.7.2. ロボアプリバージョンの説明

ロボアプリバージョン任意の数字およびドット(.)で構成され次の書式に従って下さい。但し、<MajorNumber>は1以上の整数を使って下さい。

(MajorNumber).(MinorNumber).(RevisionNumber)

図 15.7-1 ロボアプリバージョンの書式

ロボアプリバージョンは1.0.0以上にしてください。ロボアプリバージョン"1.0.0"として

登録するには以下のようにします。

```
<package version="1.0.0" uuid="biz_softbank_pepper-sample">
```

図 15.7-2 ロボアプリバージョン 1.0.0

15.7.3. manifest.xml のエレメント

manifest.xml 内の各エレメントについて説明します。

表 15.7-2manifest.xml のエレメント

エレメント名	説明
names	names エレメントの子エレメントの name エレメントは、日本語(ja_JP)と英語(en_US)が必須です。ロボアプリ名は、「ロボアプリ配信管理画面」で表示されます。
supportedLanguages	supportedLanguages エレメントの子エレメントの language エレメントは、日本語(ja_JP)が必須です。対応する言語によって中国語(zh_CN)、英語(en_US)等を設定してください。
descriptionLanguages	descriptionLanguages エレメントの子エレメントの language エレメントについては、日本語(ja_JP)と英語(en_US)が必須です。
behaviorContent	behaviorContent エレメントの path 属性は、ロボアプリを開始起動させる behavior.xar が格納されているフォルダによって変更してください。基本的には Choregraphe が自動で設定する項目です。
nature	nature エレメントは"interactive"のみ有効です。以下のようになります。
userRequestable	userRequestable エレメントは含めないでください。 <userRequestable />タグは削除してください。
triggerSentences	triggerSentences エレメントは、何も記入しないでください。Pepper for Biz モデルはトリガーセンテンスによる起動はできません。
loadingResponses	loadingResponses エレメントは、何も記入しないでください。 以下のように空タグ<loadingResponses />のみになります。
permissions	permissions エレメントは、何も記入しないでください。以下のように空タグ<permissions />のみになります。
autonomous	autonomous エレメントは、何も記入しないでください。Pepper for Biz モデルは起動トリガー条件による起動はできません。
naoqiRequirement	最小バージョン (minVersion 属性) は"2.5"、最大バージョン (maxVersion 属性) は"2"としてください。2.5~2.x という意味になります。上記のように記述し、Choregraphe にてパッケージ化を行うと、警告が出る場合がありますが、無視してパッケージ化をしてください。最小バージョンと最大バージョンは必ず指定してください。
robotRequirement	Model 属性を"JULIETTE_Y20"としてください。

16. 付録

この章では、開発者が知っておくべき Pepper に関する情報を提供します。

16.1. 演出

Pepper の振る舞いの自然さや面白さと、その動作をさせているアプリの質は必ずしも比例しません。とても複雑な動作が可能でアプリだとしても、それが不自然であったり、あるいは使用する人々にとって不親切であったりすれば、ユーザからするとそれはよくないアプリとなります。Pepper のアプリを開発する場合は、そうした Pepper とユーザとの関係を確認し、どのようにアプリやユーザ体験を与えられるのか、綿密に設計・デザインする必要があります。

16.1.1. 人を惹きつける方法

Pepper で動作するアプリは、ユーザの要求(操作・声かけ)に応じて動作させるものだけではありません。Pepper 自身が周辺にいる人に対して声をかけ、認識した人の状態を判断して適切なアプリを起動するなど、Pepper 側から能動的にユーザに働きかけを行うアプリを作成することもできます。例えば、店舗に Pepper を配置した場合は、定期的に店内を巡回し、認識した顧客に対して声かけを行うようなアプリで注目を集められます。また、子供が集まるような場所であれば、Pepper の目の部分の LED を光らせたり、音楽を再生したりすると楽しんでもらえるでしょう。このようなアプリは、ユーザの行動によって起動する Interactive アクティビティではなく、定期的に動作する Solitary アクティビティとして作成します。Solitary アクティビティで人を惹きつけ、集まってきた人が Pepper に話しかけることで Interactive アクティビティを起動させるといった工夫を行うことで、Pepper を使ったコミュニケーションはより広がります。

16.1.2. 安全性の確保

人を惹きつけるアプリを作成する際に注意しなければならないことは、安全性を確保することです。特に Pepper が移動するようなアプリを使用したい場合は、周囲半径 90cm 以内に障害物がない場所に配置することが望ましいです。また、インパクトのある動きをさせることで注目を集めようとして、突然大きな声で話したり、人が近づいてきたところで突然大きな音楽を再生したりすることはトラブルの原因になりますので避けましょう。Pepper に派手なアクションをさせようと無理な姿勢を長時間つづけるようなことも故障の原因となります。ロボアプリの開発は、人を惹きつけることと同時に安全性を確保するという、相反する要求を同時に達成するための工夫が求められると言えるでしょう。

16.2. 再起動方法

Pepper の動作が不安定になったり、Choregraphe との連携がうまく行かなくなった場合、Pepper を再起動すると改善されます。この節では、幾つかの停止、起動方法を紹介します。

16.2.1. 通常再起動

トラブル時だけでなく、通常の再起動方法です。胸の電源ボタンを 3 秒長押しして電源オ

フの後、電源ボタンを 1 回押して起動します。

16.2.2. NAOqi のみ再起動

比較的軽度の問題 (Choregraphe との連携ができなくなったなど) の場合に適した再起動方法です。電源をオフにすると、起動まで時間がかかり、面倒なこともあります。時間がかかる原因は、基礎 OS である Gentoo Linux と NAOqi の両方を再起動するからです。SSH 接続して「`nao restart`」コマンドを実行した場合は、NAOqi のみを再起動するので、比較的早く再起動できます。

16.2.3. 強制再起動

Pepper に深刻な問題が発生すると、電源ボタン 3 秒長押しでは終了しないことがあります。その場合は、そのまま 8 秒長押しして強制終了した後、胸の電源ボタンを 1 回押して起動します。電源がオフになった瞬間、Pepper は硬直して不安定になるので、解除キーを用意して、すぐにセーフレスト状態にできるように準備しておいてください。

16.2.4. 緊急停止

何か特別な事象が発生して緊急停止したいときは、首の後ろにあるゴム素材で覆われた緊急停止ボタンを強く押して強制終了してください。緊急停止ボタンで終了した場合、ボタンが押し込まれた状態になり電源が入らなくなります。電源ボタンを右に回して、カチッと音がして元の状態に戻してから電源を入れてください。

16.2.5. 自己修復モードで起動

電源を入れる時、電源ボタンを 4 秒長押しして、肩の LED が青く点滅し始めたらすぐに離します。この方法は、各部位にある基盤のファームウェアの再書き込みを行います。この方法で起動しても問題が改善されない場合は、ハードウェア的な故障を疑ってください。

16.3. qicli コマンド

SSH 接続すると、コマンドで Pepper を操作することができます。Choregraphe より qicli コマンドで操作したほうが便利な場合もあります。

16.3.1. SSH 接続の方法

SSH (Secure SHell) 接続は、安全にリモートコンピュータに接続する通信プロトコルです。ロボアプリの開発は基本的に Choregraphe で行いますが、Choregraphe ではできないことを SSH 接続して命令することができます。

SSH 接続の方法は、Mac の場合はターミナルから以下のコマンドを入力します。

```
ssh nao@[IP address]
```

図 16.3-1 SSH 接続のコマンド

- ※ 環境やクライアントアプリの設定によって操作手順が異なりますので、それに応じた方法で接続してください。
- ※ Windows の場合は、SSH 接続クライアントアプリを用意する必要があります。
- ※ Tera Term を使用する場合、「チャレンジレスポンス認証を使う(キーボードイン

タラクティブ)」設定で接続してください。

SSH 接続していれば、Linux の tail コマンドを使って、直接ログファイルを参照することも可能です。

```
tail -f /var/log/naoqi/tail-naoqi.log
```

図 16.3-2 Linux の tail コマンド

16.3.2. セーフティー機能解除

Pepper は A-Life のオン/オフに関わらずセーフティー機能が働いています。安全が確保できない方向に手を出すときは、ぶつかっても Pepper が壊れたり、人がケガをしない速度でゆっくり動きます。移動のときは障害物があれば自動で停止します。アプリ開発中にモーションを作成している時、テストをすると腕がうまく動かない時があります。その場合は、セーフティー機能が原因であることが考えられます。正しくモーションが作成できているかを確認する場合、以下のコマンドで腕のセーフティ機能を解除することができます。

```
qicli call ALMotion.setExternalCollisionProtectionEnabled "Arms" 0
```

図 16.3-3 腕のセーフティ機能を解除するコマンド

ただし、A-Life がオンだとセーフティー機能がすぐに自動復帰するので、セーフティー機能を解除してモーションのテストをする際は、A-Life をオフにしてください。

16.3.3. イベント発生

開発中アプリのビヘイビアの起動トリガー条件が「ゾーン 1 に 2 人いたら」という設定の場合、テストの度に Pepper に向かって 2 人で近づいていくのは煩わしいです。以下のコマンドを使用すると、イベントを発生させることができます。

```
qicli call ALMemory.raiseEvent "[EventName]" [param]
```

図 16.3-4 コマンドからのイベント発行

「ゾーン 1 に 2 人いたら」という条件の場合、以下のようになります。

```
qicli call ALMemory.raiseEvent "Launchpad/NumPeopleZone1" 2
```

図 16.3-5 ゾーン 1 に 2 人いるときのイベント

Choregraphe の[メモリウォッチャー]パネルでも同様のことができますが、qicli コマンドを用いたほうが、より確実にイベントの発行を行うことが可能です。

16.3.4. ログを確認する

ログを確認する qicli コマンドがあります。

```
qicli log-view
```

図 16.3-6 ログを確認する qicli コマンド

発生した問題によっては Choregraphe 上で確認できない状態であることも考えられます。その問題解決のために必要になることもあります

16.4. ロボットウェブページ (advanced)

Pepper に接続可能な環境で、Web ブラウザから下記アドレスへアクセスすると、Pepper の詳細設定画面 (旧ロボットウェブページ) が表示されます。ダンスなどモーターの温度が上がりやすいロボアプリを長時間使用する場合は、定期的にモーターの温度を確認する必要があります。詳細設定画面を使えば、各モーターの温度を一覧で確認することができます。

[Pepper の詳細設定画面の URL]

[http://\(Pepper の IP アドレス\)/advanced/](http://(Pepper の IP アドレス)/advanced/)



図 16.4-1 Pepper の詳細設定画面

詳細設定画面で確認できる主な内容は以下の通りです。

表 16.4-1 advanced ページ

ページ	項目	主な内容
HOME	NAO qi	バージョンやバッテリーの残量、Head&Body ID、現在の設定言語。
	NetWork	IP アドレス。
	Build	Build Date、Build ID。
Setting	Security	セーフティー機能の OFF。
Hardware		デバイスの接続形式や関節部分の温度。
Memory		内部メモリの状態。
Tethering		Wi-Fi によるテザリングの設定。

16.5. 運用時の注意点

Pepper を実運用する際の注意点を説明します。

16.5.1. ネットワーク環境

多くのロボアプリはインターネットに接続されていないと正しく動作することができません。Pepper の設置場所のネットワーク環境について、運用開始前に確認しておきましょう。特にイベント会場などの場合は Wi-Fi の電波が届かないケースやセキュリティ設定で Pepper 本体と Choregraphe が接続できないようなケースもあります。

来場者のスマートフォンなどの接続により Wi-Fi ルーターやネットワークが重くなるようなことも考えられますので、モバイル Wi-Fi などを準備しておくとい良いでしょう。

16.5.2. 設置場所

強い逆光があるような場所だと顔認識や表情認識などカメラを使った機能が正しく動作しません。また、狭い場所だとセーフティ機能が頻繁に動作してしまいよくわからないモーションになってしまうことがあります。実際に使用する位置に Pepper を設置後、一連の動作が正しく動作するか確認する必要があります。Pepper が壁に反射する光を顔だと認識してしまうと、ずっと壁の方を向いてしまう場合があります。電源フラップを開き、移動機能を停止してしまうことで、問題を回避することができる場合もあります。

16.5.3. 複数台体制の検討

ダンス系のロボアプリを連続して使用する場合やイベント会場でのプレゼンなどで失敗が許容されにくい場合では、Pepper を複数用意することを検討します。万が一、モーターの温度が上昇して動作できなくなる場合や急に動作が不調になったさいに、スワップ機の準備があればリスクを減らすことができます。

16.6. Pepper の商標・キャラクターについての制限

Pepper の商標・キャラクターの権利はソフトバンクロボティクス株式会社に帰属します。Pepper を使用した著作物、コンテンツ、Pepper そのものの使用には制限があります。詳

細については、「Pepper のメディア掲載、公共での稼働をお考えの方に」
 (<http://www.softbank.jp/robot/biz/support/character/>) を参照してください。
 以下具体的な制限事項の一部抜粋を記述致します。

16.7. メディア掲載・公共環境での稼働について

一部のメディア掲載、公共での稼働については、事前にソフトバンクロボティクス株式会社の許諾が必要です。

16.7.1. 禁止事項

- TV 番組へのタレント出演
- CM 出演
- ウェブ広告
- 交通広告
- 屋外広告

16.7.2. 事前申請が必要

- TV 番組・新聞・ウェブ・ラジオなどの取材
- プレスリリース
- 発表会への登壇
- 新聞・チラシ・雑誌・ラジオやこれらに準じる各メディアでの広告
- 広告以外のウェブ掲載
- イベント・店頭・各種施設での展示・稼働（メディア露出がある場合のみ要申請）

16.7.3. キャラクターに関する制限

Pepper のキャラクターイメージを損なうことは禁止されております。Pepper を使って他者の名誉を毀損する行為も同様に禁止されております。詳細につきましては、「商標・著作物・Pepper キャラクターに関するガイドライン」

(http://cdn.softbank.jp/mobile/set/data/static/robot/legal/pepper_character_guideline.pdf) がありますので参照してください。以下具体的な制限事項の一部抜粋を記述致します。

16.7.4. 商標について

Pepper 商標等のソフトバンクロボティクス社のロゴは使用できません。

文字表記の Pepper 商標に関してはガイドライン掲載の条件を満たすことにより使用が許可されています。ただし、表示方法については以下の通り制限があります。

16.7.5. 名称（Pepper）に関する制限

開発したソリューション・製品の名前に、商標 Pepper を連続して使用することを禁止致します。但し、Pepper との間に「for～」や「Pepper 用～」の文言を挿入することは制限しません。

- ○○販売促進 Pepper (→Pepper と商品名が連続して使用されているため×)

- ◯販売促進 for Pepper (→for が Pepper との間にあるため○)
- Pepper 用◯販売促進アプリ (用が Pepper との間にあるため○)

また、Pepper の商標は P (大文字) epper (小文字) の表記になります。
以下に掲載する例は全て誤りです。

- pepper (小文字表記)
- PEPPER (大文字表記)
- pEPPER (大文字と小文字の箇所が誤っている)
- ペッパー (カタカナ表記)
- Pepper's (Pepper から何らかの変形を伴った形式)

Pepper の名前を含んだ、商標・企業名・インターネットドメイン・SNS アカウント等の登録は禁止しております。

16.7.6. Pepper の画像について

広告、告知物に Pepper の画像を使用する場合はご自身で撮影された画像をご使用ください。

無断でソフトバンクロボティクス社およびソフトバンクグループ会社のホームページ等から写真画像、Pepper ロゴ、イラストなどのコンテンツを使用することは禁止されています。Pepper for Biz をご使用で、公式画像が必要な場合は営業担当もしくは Pepper for Biz お問い合わせ窓口へ連絡する必要があります。

Pepper パートナープログラムへの加入後は Pepper パートナー会員事務局へご相談ください。

Pepper パートナープログラム

ロボアプリパートナー (Basic)

学習用ワークブック

発行 ソフトバンクロボティクス株式会社
改訂 2018/10/31

本書の一部または全部を、ソフトバンクロボティクス株式会社から正式な許諾を得ずに、いかなる方法(転載・転用・送信・上映)においても無断で複写、複製することを禁ずる。

◇免責事項◇

本書において記載されている会社名、製品名は、それぞれの会社の商標もしくは登録商標の場合がある。本書では®、TM マークを明記しない。