

## 目次

1. お好み焼き協調料理の概要.....	2
1. 1. お好み焼き協調料理の全体像.....	2
1. 2. お好み焼き協調料理のソフトウェア構成.....	4
2. お好み焼き協調料理の詳細仕様.....	5
2. 1. お好み焼き協調料理の実現方式.....	5
2. 2. お好み焼き GUI 及び CUI の画面仕様.....	6
2. 4. お好み焼き協調料理のクラス仕様.....	12
2. 5. お好み焼き協調料理の状態遷移.....	15
2. 6. お好み焼き協調料理のモジュール仕様.....	18
2. 7. お好み焼き協調料理の物理配置.....	18
2. 8. お好み焼き協調料理のファイル仕様.....	19

## 1. お好み焼き協調料理の概要

本資料は、お好み焼き協調料理のプログラムの構成及び処理の流れの仕様を説明する資料である。主としてお好み焼き協調料理が持つ機能と SIGVerse-API との関連、プログラムとしての実現方法を記す。

### 1. 1. お好み焼き協調料理の全体像

お好み焼き協調料理は、SIGVerse の3機能を元に、大きく7機能に分類する。これら全ての機能は SIGVerse に依存しており、SIGVerse なしには動作しない。以下にその機能の一覧を記す。

機能名	関連する SIGVerse の能力	機能の概要
見た目の変更	知覚	エージェントの見た目を切り替える
回転	力学	三次元空間上のエージェントの向きを変える(回転させる)
移動		三次元空間上のエージェントの位置を変える(移動させる)
身体動作		人体の間接を元にした複雑な動作を行う
エージェント・アバタ間対話	対話	利用者の要求をアバタを介して他のエージェントに伝える
エージェント間相互対話		エージェント間の要求(発言)を自分を含むエージェントに伝える
状態遷移	本プログラム特有機能	状態遷移の「情報」「条件」「処理」を制御する

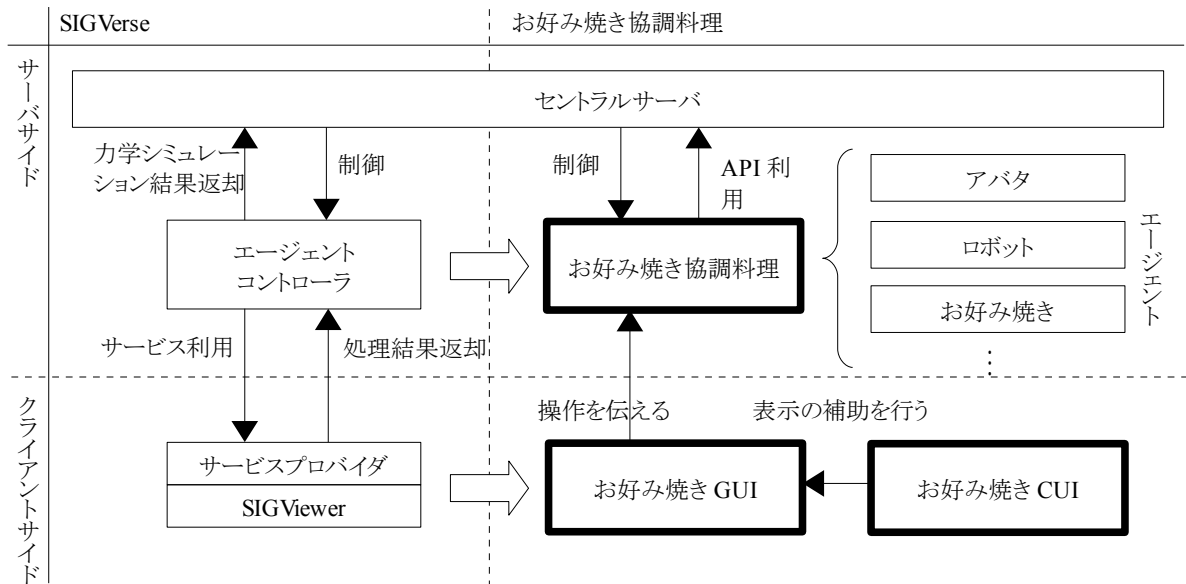
SIGVerse 及びお好み焼き協調料理は、実装言語として C++言語を用いる。以下にその C++文法に従い定義されている SIGVerse-API の代表例を記す。

API の名前	SIGVerse の能力	API のインターフェイス	API の例
見た目の変更	知覚	setAttrValue( const char, /* 対象名 */ char) /* 設定値 */	setAttrValue(“visual”, “koge”); /* 見た目を焦げに変える*/
回転	力学	setAxisAndAngle( double, /* x 軸 */ double, /* y 軸 */ double, /* z 軸 */ double) /* 角度 */	SetAxisAndAngle(1, 0, 0, 45 * (PI / 180)); /* x 軸に+45 度回転する */
移動		setPosition( double, /* x 座標 */ double, /* y 座標 */ double) /* z 座標 */	setPosition(100, 80, 100); /* 座標(x=100, y=80, z=100)に移動する */
身体動作		setJointAngle( const char, /* 対象名 */ double, /* x 軸 */ double, /* y 軸 */ double, /* z 軸 */ double) /* 角度 */	setJointAngle(“LARM_JOINT1”, 1, 0, 0, 45); /* 左肩を 45 度回転する */
エージェント・アバタ間対話	対話	sendMessage( const char, /* 対象名 */ int, /* 文言行数 */ char**) /* 送信文 */	char *text = “Hello, SIGVerse”; sendMessage(“Avator”, 1, (char**) &text); /* ロボットに「Hello, SIGVerse」と送信する */
エージェント間相互対話			char *command = “Move=0:0:0”;



## 1. 2. お好み焼き協調料理のソフトウェア構成

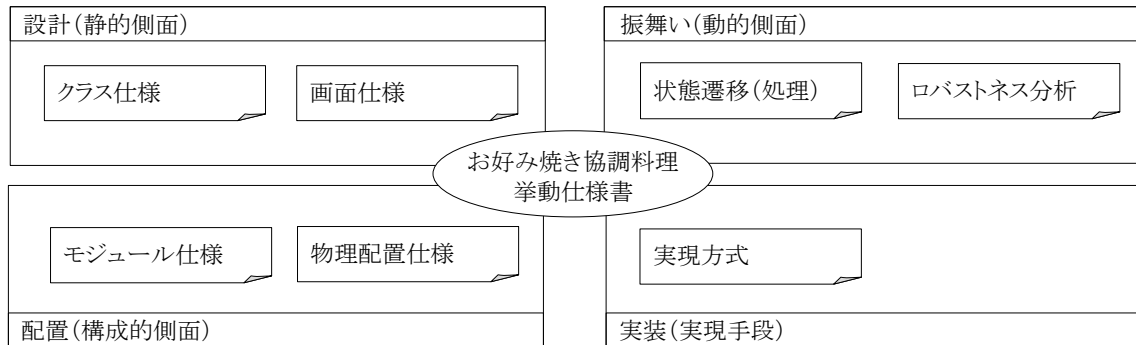
お好み焼き協調料理は、サーバサイド・クライアントサイドに跨る SIGVerse の基本機能を拡張して実現する。例えば、代表的なエージェントであるアバタやロボットはサーバサイドの SIGVerse の力を借りて実装と動作を実現する。同様にクライアントサイドでもシミュレーション状況の視覚化を行う SIGViewer と同列の扱いで、要求をお好み焼き協調料理に関連するエージェントに送信するお好み焼き GUI、表示の補助を行うお好み焼き CUI からなる。以下にそれらソフトウェアの全体図と一覧を記す。



ソフトウェア名	稼動層	ソフトウェアの役割
セントラルサーバ	サーバサイド	セントラルサーバは力学の計算と対話を制御する SIGVerse の本体サーバであり、お好み焼き協調料理を含むユーザアプリケーションは、このソフトウェア上で動作する。
お好み焼き協調料理	サーバサイド	お好み焼き協調料理本体を示す。実際には各種エージェントの集合体となる。
サービスプロバイダ (SIGViewer)	クライアントサイド	エージェントの視覚情報や音声情報を提供する。例えば、ロボットが見ているもののリストや視界のピクセル単位での情報提供を行う。サービスプロバイダは、シミュレーションリアルタイムビューワとしての SIGViewer に内包される。
お好み焼き GUI	クライアントサイド	お好み焼き GUI は、SIGVerse 仮想空間内における利用者の代理人であるアバタの操作を行う GUI である。例えば、「生地を混ぜる」釦を押下し仮想空間上のアバタに生地を混ぜる行為を行わせる
お好み焼き CUI	クライアントサイド	お好み焼き CUI は、ロボット及びアバタの発言を見やすく、わかりやすくするために、テロップ表示 (スーパーインポーズ) や音声出力を行う

## 2. お好み焼き協調料理の詳細仕様

以下にお好み焼き協調料理を構成するプログラムの構造と処理の具体的な方式を記す。大きく分けて実現方式、画面・クラス・シーケンス(処理)・モジュール・物理配置・依存設定ファイル仕様からなる。以下に本章の俯瞰図と項目の一覧を記す。



(\*1)...上図は、お好み焼き協調料理「挙動」仕様書を元に、各種の設計(静的側面)、振る舞い(動的側面)、配置(構成的側面)、実現手段からお好み焼き協調料理が実現される様を示す。

分類名	章名	記述内容
設計	画面詳細仕様	お好み焼き GUI に関する画面仕様を定める
	クラス仕様	挙動仕様書を元にしたお好み焼き協調料理の「静的」構造を定める
振舞い	状態遷移(処理)	挙動仕様書を元にしたお好み焼き協調料理の「動的」構造を定める
配置	モジュール仕様	再利用性を意識した本向上PGのモジュール構成を定める
	物理配置仕様	お好み焼き協調料理の実際の物理配置と関連を定める
実装	実現方式	実装言語、実装環境、利用製品などを定める

### 2. 1. お好み焼き協調料理の実現方式

お好み焼き協調料理は、SIGVerse の利用を大前提に以下のソフトウェアを用いて実現する。以下に依存するソフトウェアの一覧を記す。

利用ソフトウェア名	稼働層	用途
RetHat Enterprise Linux V4 64Bit	サーバサイド	SIGVerse (セントラルサーバ) 及びお好み焼き協調料理 (エージェント類) の稼働環境
SIGVerse Platform	サーバサイド	SIGVerse (セントラルサーバ) 本体
お好み焼き協調料理	サーバサイド	お好み焼き協調料理本体
Windows XP	クライアントサイド	SIGViewer 及びお好み焼き GUI、お好み焼き CUI の稼働環境
SIGViewer	クライアントサイド	SIGViewer (サービスプロバイダ) 本体
お好み焼き GUI	クライアントサイド	お好み焼き協調料理の操作 GUI
お好み焼き CUI	クライアントサイド	お好み焼き協調料理に関する表示補助
Microsoft DirectX 8.0	クライアントサイド	シミュレーション環境 3D 描画
Microsoft SAPI5.0 音声合成	クライアントサイド	アバタ及びロボットの発話の読み上げ

また、お好み焼き協調料理(サーバサイド)、お好み焼き GUI、お好み焼き CUI に関しての前提とする開発環境と稼働環境について以下に記す。

対象方式名	開発対象	対象製品名	バージョン	備考
実装言語	お好み焼き協調料理	C++	“3.4.6.”	
	お好み焼き GUI/CUI	VisualStudio Express edition	“2008”	.NET は 3.5 を利用する
稼働環境	お好み焼き協調料理	Red Hat Enterprise Linux v4	“V4”	標準添付 Boost を利用する
	お好み焼き GUI/CUI	Windows XP	—	※SP3 までを対象

## 2. 2. お好み焼き GUI 及び CUI の画面仕様

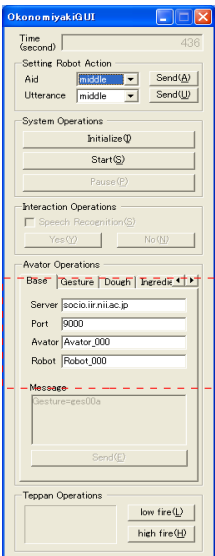
お好み焼き GUI 及びお好み焼き CUI は、直接利用者が利用する為、画面対話方式で実現する。但し、上記お好み焼き協調料理の全ての機能は、これら画面だけでは実現せず、お好み焼き協調料理本体で実現する。この画面類は、利用者の操作をお好み焼き協調料理本体に要求するだけの機能を持つ。以下に、それら画面の仕様を記す。

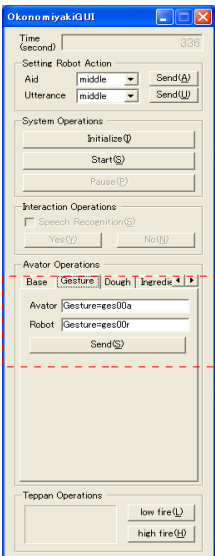
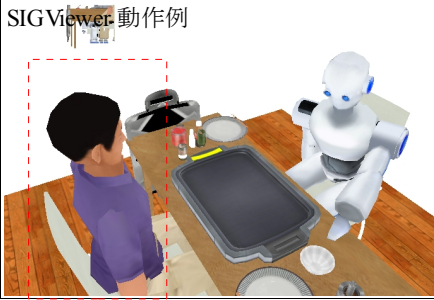
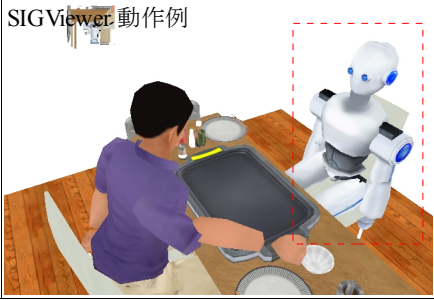
### 2. 2. 1. お好み焼き GUI

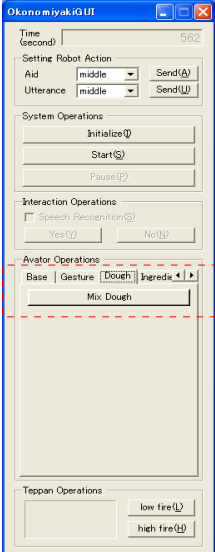


お好み焼き GUI は、以下の画面構成で、挙動仕様書上取り扱えるアバタの全ての操作を実現する。基本的に本 GUI よりも SIGViewer の処理結果の閲覧を重視し、本 GUI は出来る限りレイアウト領域を小さくする前提を置く。

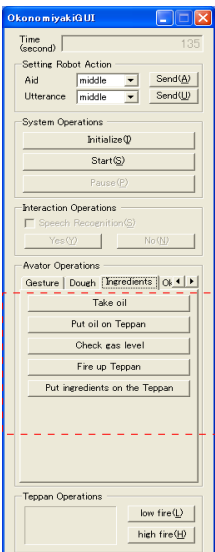

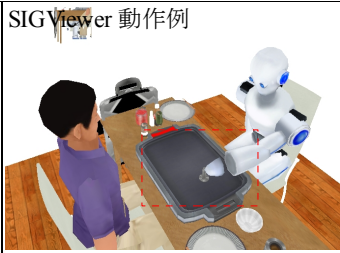
画面名	お好み焼き GUI 基本画面							
画面概要	1. 利用者が利用したいロボットの補助度、発言度をお好み焼き協調料理本体に送信する 2. お好み焼き協調料理の動作を初期化し、初期状態に戻す 3. お好み焼き協調料理上のロボットの動作を開始する							
画面レイアウト								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	補助度	cmbAid	文字	Combo	—	1	{0,1,2,3}
	2	発言度	cmbUttr	文字	Combo	—	1	{0,1,2}
	3	補助度送信	btnAid	—	Button	—	—	Aid=?(*1)
	4	発言度送信	btnUttr	—	Button	—	—	Uttr=?(*2)
	5	初期化	btnInit	—	Button	—	—	Clear=?(*3)
	6	開始	btnStart	—	Button	—	—	Attr=1(*4)

(\*1)... 補助度送信は、文言 Aid={0, 1, 2, 3} をお好み焼き協調料理に送信する  
 (\*2)... 発言度送信は、文言 Uttr={0, 1, 2} をお好み焼き協調料理に送信する  
 (\*3)... 初期化は全エージェントに文言 Clear=Robot\_000 等の初期化要求を送信する  
 (\*4)... 開始はロボットに文言 Attr=1 を送信する、ロボットは Attr=1 で状態遷移を開始する

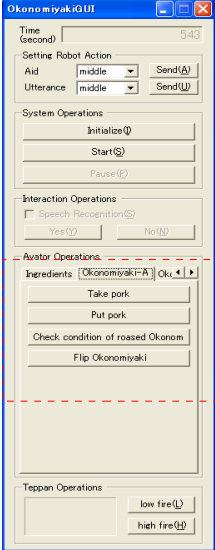
画面名	お好み焼き GUI 接続先設定画面							
画面概要	1. お好み焼き協調料理(セントラルサーバ)への接続情報を設定する							
画面レイアウト(Baseタブに相当)								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	サーバ名	txtServer	文字	Text	24	socio.iir.nii.ac.jp	自由入力
	2	ポート番号	txtPort	数値	Text	24	9000	—
	3	アバタ名	txtAvator	文字	Text	24	Avator_000	自由入力(*1)
	4	ロボット名	txtRobot	文字	Text	24	Robot_000	自由入力(*2)
(*1)... アバタ名には、アバタエージェントの実名を入力する (*2)... ロボット名には、ロボットエージェントの実名を入力する								

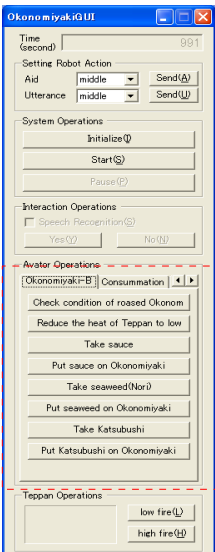
画面名	お好み焼き GUI 初期身体動作設定画面							
画面概要	1. アバタ及びロボットの初期姿勢を設定する							
画面レイアウト(Gestureタブに相当)								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	アバタ姿勢	txtAGestre	文字	Text	24	Gesture=ges00a	自由入力(*1)
	2	ロボット姿勢	txtRGestre	文字	Text	24	Gesture=ges00r	自由入力(*2)
	3	送信	btnGSend	—	Button	—	—	—
(*1)... アバタ姿勢には、アバタの初期の身体動作(椅子に座る)名を指定する (*2)... ロボット姿勢には、ロボットの初期の身体動作(椅子に座る)名を指定する								
 SIGViewer 動作例				 SIGViewer 動作例				

画面名	お好み焼き GUI お好み焼き生地状態時操作画面1							
画面概要	1. お好み焼きが生地状態である場合の料理操作を行う							
画面レイアウト(Dough タブに相当)								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	MixDough	txtState1	—	Button	—	—	—(*1)
(*1)... 生地を混ぜる要求「State=1」をアバタに送信する								
								

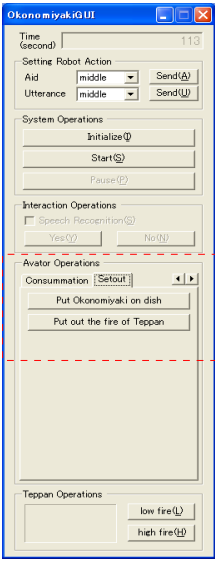


画面名	お好み焼き GUI お好み焼き生地状態時操作画面2							
画面概要	1. お好み焼きが生地状態である場合の料理操作を行う							
画面レイアウト(Ingredients タブに相当)								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	Take oil	btnState2	—	Button	—	—	—(*1)
	2	Put oil on Teppan	btnState3	—	Button	—	—	—(*2)
	3	Check gas level	btnState4	—	Button	—	—	—(*3)
	4	Fire up Teppan	btnState5	—	Button	—	—	—(*4)
	5	Put ingredients on Teppan	btnState6	—	Button	—	—	—(*5)
(*1)... 油をとる要求「State=2」をアバタに送信する								
(*2)... 油を塗る要求「State=3」をアバタに送信する								
(*3)... ガス残量を確認する要求「State=4」をアバタに送信する								
(*4)... 鉄板を点火する要求「State=5」をアバタに送信する								
(*5)... 生地を鉄板にのせる要求「State=6」をアバタに送信する								
								

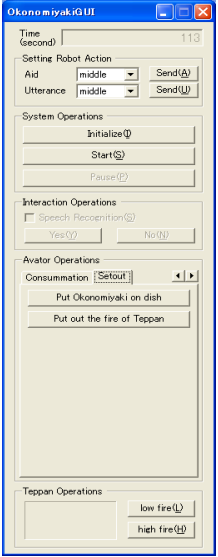




画面名	お好み焼き GUI お好み焼き A 面調理中操作画面							
画面概要	1. お好み焼きの A 面を焼いている状態の料理操作を行う							
画面レイアウト(Okonomiyaki-A タブに相当)								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	Take pork	btnState7	—	Button	—	—	—(*1)
	2	Put pork	btnState8	—	Button	—	—	—(*2)
	3	Check condition of roased Okonomiyaki	btnState9	—	Button	—	—	—(*3)
	4	Flip Okonomiyaki	btnState10	—	Button	—	—	—(*4)
(*1)... 豚肉をとる要求「 State=7 」をアバタに送信する (*2)... 豚肉をお好み焼きにのせる要求「 State=8 」をアバタに送信する (*3)... お好み焼きの焼け加減を確認する要求「 State=9 」をアバタに送信する (*4)... お好み焼きを裏返す要求「 State=10 」をアバタに送信する								

画面名	お好み焼き GUI お好み焼き B 面調理中操作画面							
画面概要	1. お好み焼きの B 面を焼いている状態の料理操作を行う							
画面レイアウト(Okonomiyaki-B タブに相当)								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	Take sauce	btnState12	—	Button	—	—	—(*1)
	2	Put sauce on Okonomiyaki	btnState13	—	Button	—	—	—(*2)
	3	Take seaweed(Nori)	btnState14	—	Button	—	—	—(*3)
	4	Put seaweed on Okonomiyaki	btnState15	—	Button	—	—	—(*4)
	5	Take Katsuobushi	btnState16	—	Button	—	—	—(*5)
	6	Put Katsuobushi on Okonomiyaki	btnState17	—	Button	—	—	—(*6)
(*1)... ソースをとる要求「 State=12 」をアバタに送信する (*11 は空き番号とする) (*2)... ソースをお好み焼きにかける要求「 State=13 」をアバタに送信する (*3)... 海苔をとる要求「 State=14 」をアバタに送信する (*4)... 海苔をお好み焼きにかける要求「 State=15 」をアバタに送信する (*5)... 鰹節をとる要求「 State=16 」をアバタに送信する (*6)... 鰹節をお好み焼きにかける要求「 State=17 」をアバタに送信する								

画面名	お好み焼き GUI お好み焼き盛り付け操作画面1							
画面概要	1. お好み焼きを切り分ける操作を行う							
画面レイアウト(Consummation タブに相当)								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	Cut Okonomiyaki	btnState18	—	Button	—	—	—(*1)
(*1)... お好み焼きをきる要求「 State=18 」をアバタに送信する								
SIG Viewer 動作例				SIG Viewer 動作例				
								

画面名	お好み焼き GUI お好み焼き盛り付け操作画面1							
画面概要	1. お好み焼きを切り分ける操作を行う							
画面レイアウト(Consummation タブに相当)								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	Put Okonomiyaki on dish	btnState19	—	Button	—	—	—(*1)
2	Put out the fire of Teppan	btnState20	—	Button	—	—	—(*2)	
(*1)... お好み焼きを更によそう要求「 State=19 」をアバタに送信する								
(*2)... 鉄板の火を消す要求「 State=20 」をアバタに送信する								
SIG Viewer 動作例				SIG Viewer 動作例				
								

画面名	お好み焼き GUI 鉄板操作画面							
画面概要	1. 鉄板の火力操作を行う							
画面レイアウト(Consummation タブに相当)								
	項番	論理項目名	物理項目名	種別	型	桁	初期値	入力
	1	LowFire	btnState21	—	Button	—	—	—(*1)
	2	HighFire	btnState22	—	Button	—	—	—(*2)
(*1)... 鉄板の火力を弱くする要求「 State=21 」をアバタに送信する (*2)... 鉄板の火力を強くする要求「 State=22 」をアバタに送信する								
								

2. 2. 2. お好み焼き CUI

お好み焼き CUI は主にロボットまたはアバタの発話が利用者にわかりやすくなる様に、テロップ表示と音声出力する。その際に、SIGViewer に送られているメッセージを参照して文言を表示する。このメッセージ参照はパケットキャプチャにて実現し、利用者との対話としては対象とする NIC の選択のみとなる。以下に画面イメージを記す。

C:\Documents and Settings\錠 尚史\Desktop\Disambiguation...

```

1. #Device#NPF_{CC2F0F32-E4CD-478C-85C6-AA003BA691DB}
   (Dell 藤ヶ 1490 対応・ミ・ WLAN Mini-Card (Microsoft's Packet Scheduler) )
2. #Device#NPF_{A9232BC1-B33B-4120-A276-1B557A18C18D}
   (Broadcom 440x 10/100 Integrated Controller (Microsoft's Packet Scheduler) )
)
Enter the interface number (1-2):
```

NIC の一覧表示

---

NIC の番号選択



テロップ表示、及び音声読み上げ

お好み焼き CUI は選択された NIC に対するメッセージを元に、テロップ表示と音声出力を行う。



以下に各概念クラスの一覧を記す。

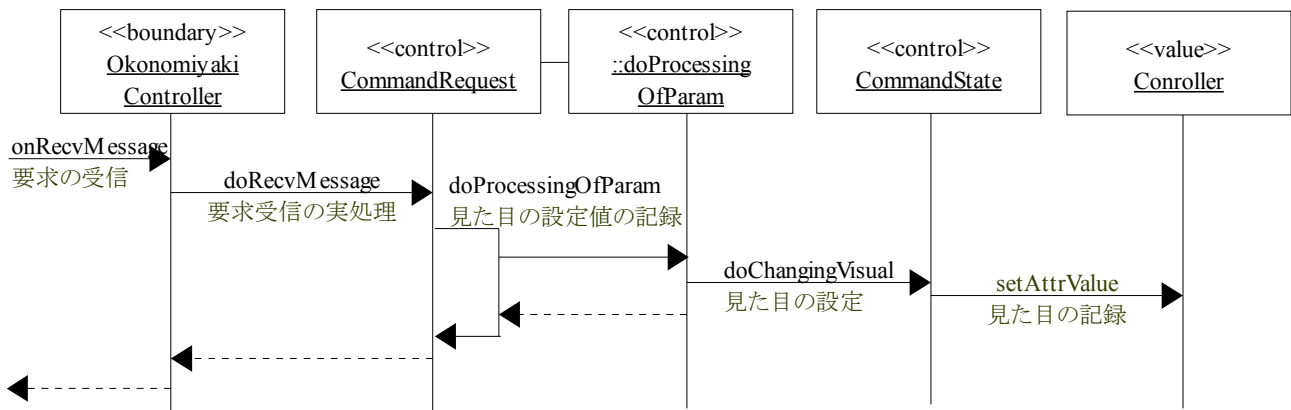
クラス名	パッケージ	クラスの役割
CommandBase	core	お好み焼き協調料理で行う共通処理を提供する。例えば、getObj の補助処理、setAttrValue の取得の補助処理を行う。
Supplier		お好み焼き協調料理で行う共通の情報に依存した処理を提供する。例えば、CSV ファイルの読み込みを行う。
Define		お好み焼き協調料理で使われる大域変数を定義。例えば、CSV ファイルのカレントディレクトリ、間接角動作の段階的な動作の回数、1 ユニット時間の実時間など。
CommandMotion	motion	力学に関する共通処理を提供。例えば、自身が次に行うべき身体動作の履歴記録など。
CommandMove		「移動」に関する処理を提供。例えば、お好み焼き協調料理では、移動は目標「座標」まで段階的に徐々に移動。その様な制御を SIGVerse-API を用いて行う。
CommandRotation (CommandMove)		「回転」に関する処理を提供。例えば、お好み焼き協調料理では、回転は目標「角度」まで段階的に徐々に移動。その様な制御を SIGVerse-API を用いて行う。
CommandRolling (CommandMove)		移動しながら回転する処理の補助処理を行う。主処理は CommandWalk。
CommandGesture (CommandMove)		身体動作を提供。指定部位に指定角度を目標角度とし、段階的に徐々に動かす。
CommadWalk (CommandMotion)		歩く動作と実際の移動、及び目標座標までの間に中間地点がある場合、その中間地点で方向転換する。現在、この処理はお好み焼き強調料理で未使用。
SupplierGesture		身体動作で使用する各種の設定ファイルを読み込み、身体動作に関する情報を提供。
CommandState		state
FilterState	状態遷移時の「状態遷移条件」の処理を提供。例えば、状態遷移条件「State」がその制約条件を満たしているかの判定を行う。また、全ての状態遷移条件に指定可能な文言はここで定義する。	
SupplierState	状態遷移に関する全ての CSV ファイルの取得を行う。それらのファイルは「状態遷移情報」「状態遷移条件」「状態遷移処理」。	
CommandTransit	transit	状態遷移に関する中で「状態遷移処理」に特化した処理を提供する。
TransitAngle		状態遷移処理「Angle」、つまり回転に関する処理を提供する。
TransitMotion		状態遷移処理「Motion」、つまり身体動作に関する処理を提供する。
TransitReject		状態遷移処理「Reject」、例えば既にその状態に遷移していた場合、それ以降の状態遷移処理を実行しない処理を提供する。
TransitState		状態遷移処理「State」、つまり状態に関する処理を提供する。
TransitUttrAndAction		状態遷移処理「UttrAndAction」、つまり一定時間の経過後、それ以降の処理を行う処理を提供する。
TransitVisual		状態遷移処理「Visual」、つまり見た目を変える処理を提供する。
TransitAttr		状態遷移処理「Attr」、つまり属性を変える処理を提供する。
TransitMove		状態遷移処理「Move」、つまり移動に関する処理を提供する。
TransitSend		状態遷移処理「Send」、つまり文言の他エージェントに対して送信に関する処理を提供する。(*現在、この処理は非推奨)
TransitUtterance		状態遷移処理「Utterance」、つまり発言の処理を提供する。

クラス名	パッケージ	クラスの役割
TransitUttrAndReject		状態遷移処理「UttrAndReject」、つまり条件を満たす場合、指定の発言を行い、それ以降の処理を行わない処理を提供する。
CommandRequest	request	「要求」に関する処理を提供する。例えば、sendMessage で送信した内容の受信、それに伴い実行すべき処理を実行。これは要求に関する共通的な処理を提供する。
RequestAttr		「Attr」要求の処理を提供する、つまり指定された属性に自身の属性を設定。
RequestGesture		「Gesture」要求の処理を提供する、指定された身体動作を実行。
RequestJudgement		この処理は、内部的な処理であり、例えば、エージェントは現在、回転中に突然身体動作を行えない為、受信した要求が行えるか判定する。その判定の処理を提供する。
RequestParam		「Aid」「Uttr」「Visual」の要求をまとめて処理する。例えば、Aid の場合は指定された補助度に自身を設定する。また、一部現在利用していない要求も含まれる。
RequestReqState		現在の状態を送信して欲しい要求を受け付けて、送信元のエージェントに返す処理を提供する。現在、この処理は正常に動作出来ない。
RequestStat		エージェントの現在の状態を始めとする、補助度、発言度を表示する。(※この機能は直接お好み焼き協調料理に関係せず、開発用の機能となる。)
RequestCatch		エージェントの前方 100 座標以内で最も距離が近いエージェント(物体)を掴む。この機能は、お好み焼き強調料理では利用していない。物を掴む場合は、身体動作機能を利用する。この機能は現在、非推奨。
RequestInitialize		これは内部処理です。直接お好み焼き協調料理で利用しない。 例えば、物を動かしているまたは移動している最中に、状態が変化して新たな移動を行う必要が発生する場合、移動や回転に関する情報を初期化しなければならず、その判定と初期化を実行する。
RequestMessage		「Message=」以降の文言を取得する試験用の処理。お好み焼き協調料理では直接利用しない。
RequestPosition		段階的な移動を行わずに、直接指定の座標に移動する要求。例えば、物を持ち運ぶ場合、所持者と持たれるものの座標にずれが生じない様、この要求を利用して持ち運んでいる様に見せる。(※将来、ものを掴むと言う正式な機能が実装される可能性がある)
RequestRequiment		現在位置や方向の要求を受け付ける。ですが、現在この機能は使われていない。そして非推奨。
RequestState		要求された状態に自身の状態を設定する。もし、指定の状態に遷移する条件が満たされていない場合、この要求は無視される。
RequestAngle		要求された角度に回転する。これは操作や状態遷移処理で行う回転と同等。
RequestClear		エージェントを初期の状態に戻す。例えば、状態、属性など。
RequestJoint		指定の間接を指定の角度に直接設定する。この機能はお好み焼き協調料理では利用されていない。開発用の処理。
RequestMove		指定の x,y,z 座標に段階的に移動する。これは、操作と状態遷移処理で利用した Move と同等。
RequestRelease		掴んでいるものを離す要求。現在のお好み焼き協調料理では利用されていない。
RequestResState		要求された「現在の状態の提供」などに返答を返す。この機能は現在、正常に動作していない。

## 2. 5. お好み焼き協調料理の状態遷移

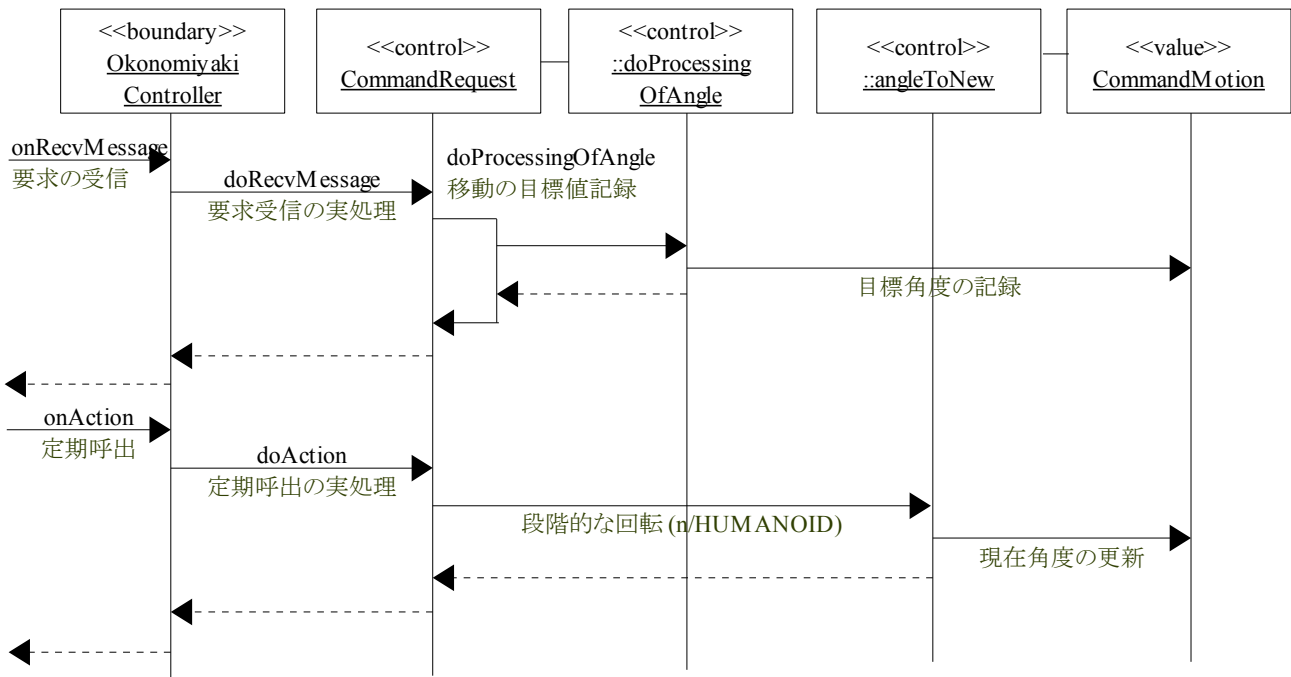
お好み焼き協調料理の全てのエージェントは、外部インターフェイスに相当する「onAction」と「onRecvMessage」メソッドを持つ。「onAction」は SIGVerse から定期的と呼ばれるメソッドであり、動画アプリケーションにおける1フレームに近い処理形態である。「onRecvMessage」は自身に対してメッセージが送られた場合、そのメッセージと共に SIGVerse から呼ばれるメソッドであり、全てのエージェントはこの2つのメソッドを基本に全ての動作を実現する。以下に、「onAction」、「onRecvMessage」を窓口とした処理の流れを記す。

### 2. 5. 1. 見た目変化



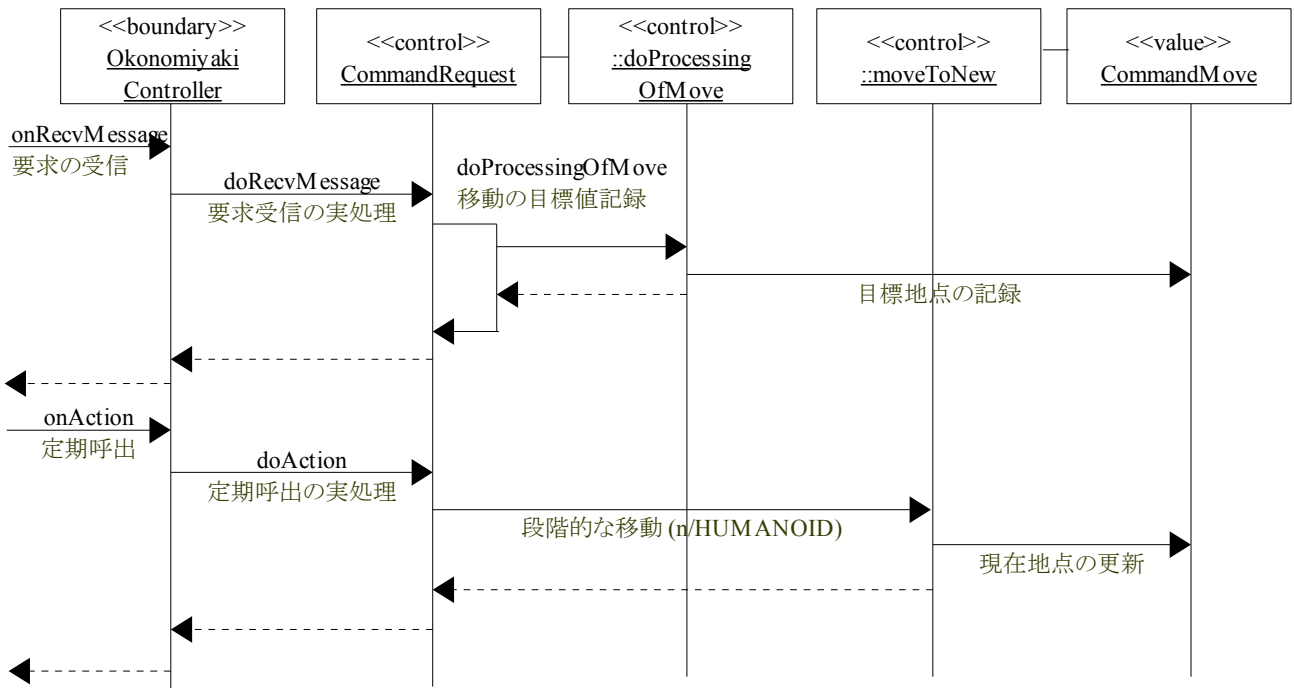
見た目設定は、見た目設定要求により、SIGVerse-API の setAttrValue を用いて見た目を変更する。

### 2. 5. 2. 回転



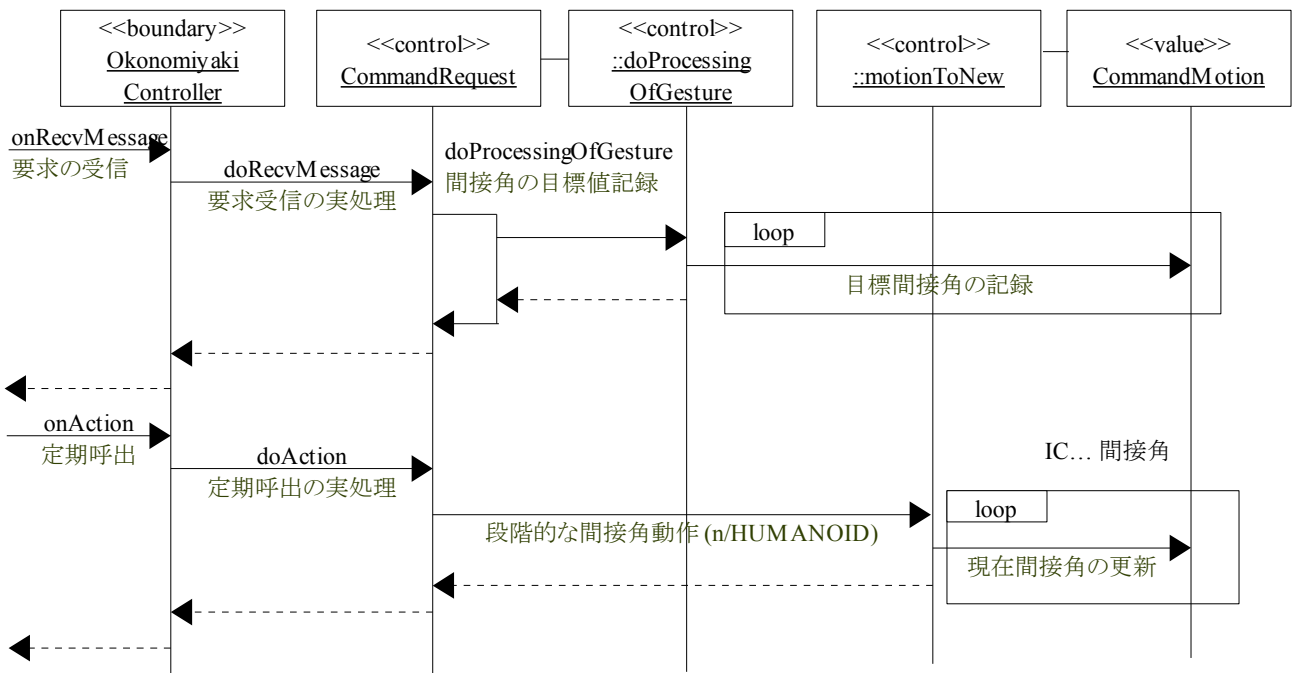
回転は、回転要求により、目標角度を保持する。その後、onAction のタイミングで逐次目標角度まで段階的に5段階を経て回転する。回転は、AIGVerse-API の setAxisAndAngle を用いる。

2.5.3. 移動



移動は、移動要求の受信により、目標点を保持する。その後、onAction のタイミングで逐次目標点まで段階的に5段階を経て移動する。移動は、SIGVerse-API の setPosition を用いる。

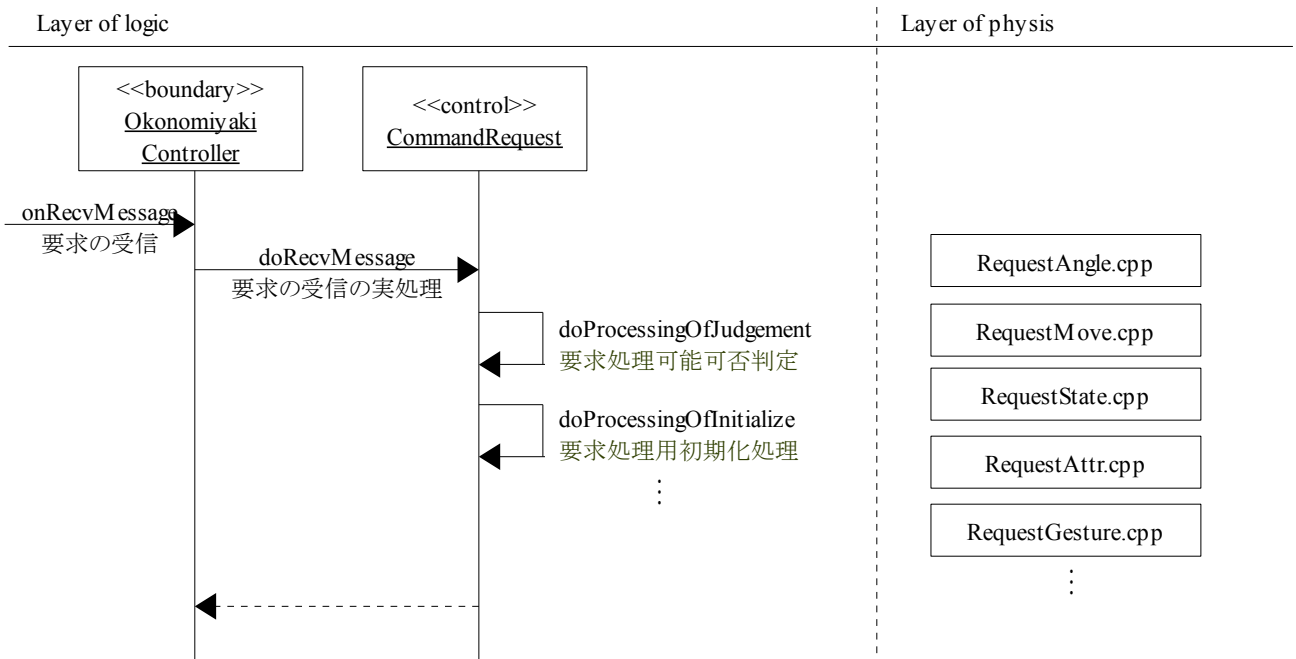
2.5.4. 身体動作



身体動作は身体動作の要求により各間接角の目標角度を保持する。その後、onAction のタイミングで「全間接角」を逐次目標角度まで段階的に5段階を経て回転する。身体動作は SIGVerse-API の setJointAngle を用いる。

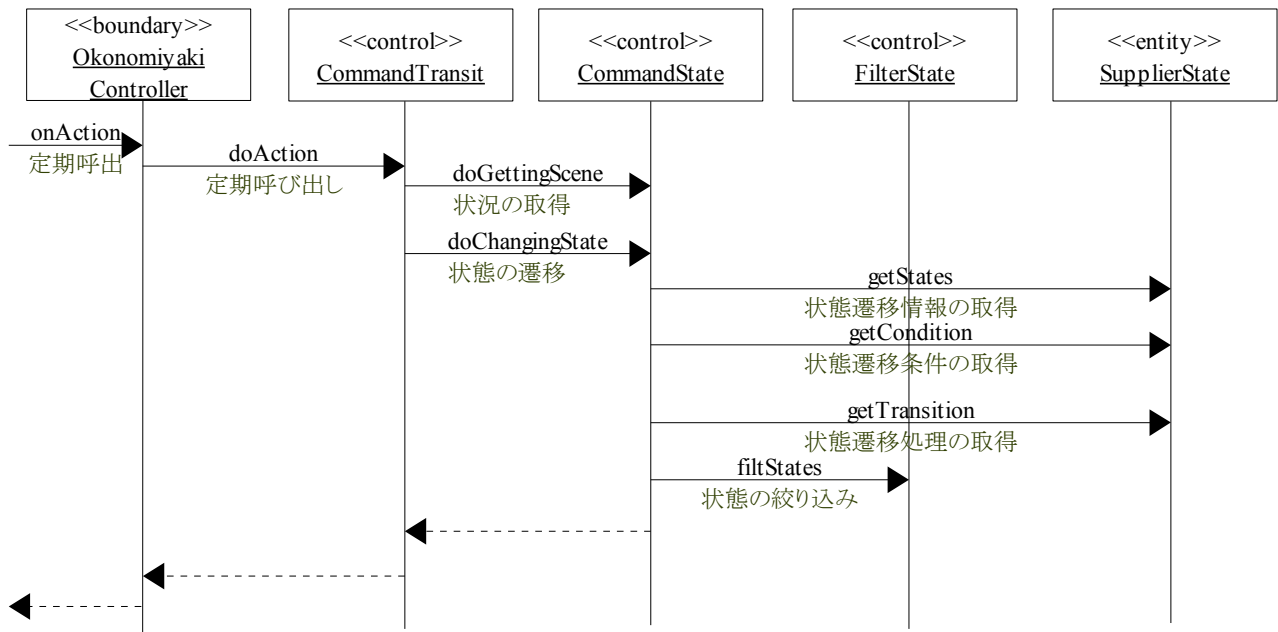


2.5.5. エージェント・アバタ間対話/エージェント間相互対話



エージェント・アバタ間対話、エージェント間相互対話は、要求の受信に関して onRecvMessage で全ての文言を受け付ける。その受信文言の解釈は、各細分化した処理に委譲する。対話の送信にあたる発話は、SIGVerse-API の sendMessage か broadcastMessage で送信する。

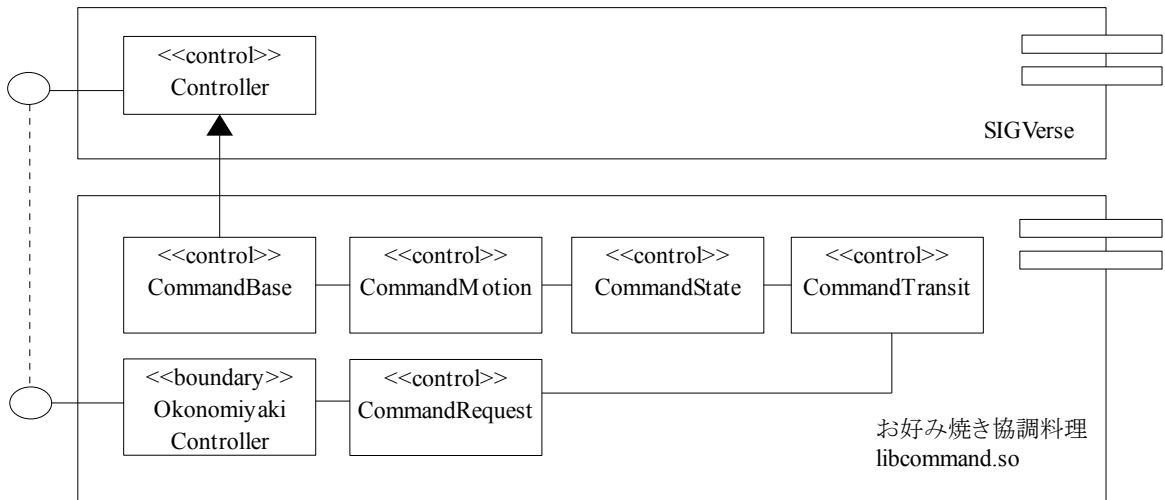
2.5.6. 状態遷移



状態遷移は、状態遷移情報、条件、処理を取得し、回りの状況(他エージェントの状態)から次状態を決定する。次に遷移可能な状態がある場合、状態遷移し、その際に状態遷移処理を実行する。

## 2. 6. お好み焼き協調料理のモジュール仕様

SIGVerse では全てのエージェントは、SharedObject で実装する。SIGVerse は SharedObject で実装されたエージェントを、上記「onAction」と「onRecvMessage」の契機で dlopen してメソッドコールする。その為、お好み焼き協調料理のエージェントは全て SharedObject である必要がある。但し、エージェント毎に1 SharedObject を提供するのではなく、1つの SharedObject で全エージェントをまかなうメタクラスとして SharedObject を提供する方式とする。以下にそのモジュール仕様を記す。

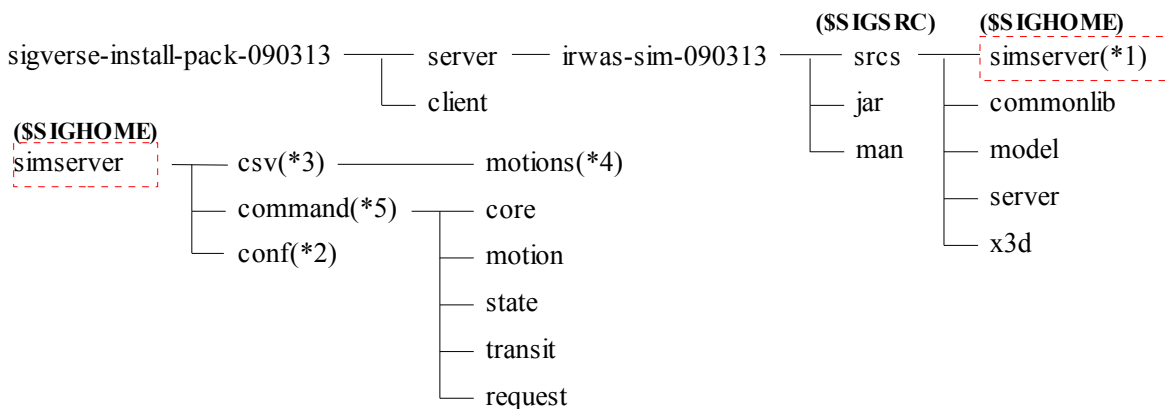


## 2. 7. お好み焼き協調料理の物理配置

お好み焼き協調料理では、提供するプログラムは、上記の通り、SharedObject 1つのみとなる。また、それはサーバサイドの話であり、クライアントサイドの場合は、お好み焼き GUI 用実行ファイル(\*.exe)、お好み焼き CUI 用実行ファイル(\*.exe)を提供し、合計で3ファイルのみとなる。以下にそれらプログラムファイルの物理配置を記す。

### 2. 7. 1. サーバサイド

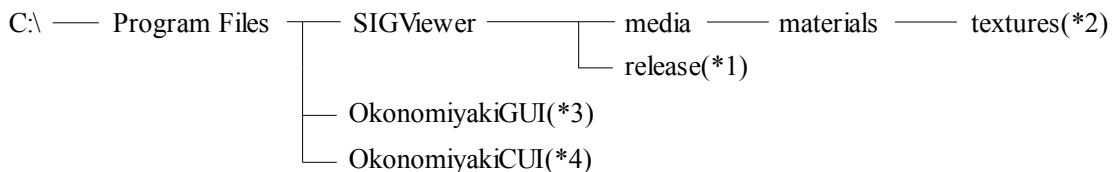
お好み焼き協調料理は、サーバサイドで全てのエージェントの実働と各種設定ファイルの格納を想定する。以下にそれらエージェントに関する実行ファイルと各種定義ファイルの物理配置を記す。尚、SIGVerse のインストールを前提とし、そのインストール先を\$SIGHOME、ソースの所在を\$SIGSRCとして、お好み焼き協調料理に関する物理配置を説明する。



項番	ディレクトリ名	ディレクトリの役割	主な格納ファイル	ファイルの役割
(*1)	simsrver	セントラルサーバを格納します	simsrver.sh	セントラルサーバ(お好み焼き協調料理)を起動します
(*2)	conf	仮想空間の定義、エージェントの定義を格納します	MyWorld.xml	仮想世界の定義
			Robot-x3d.xml	ロボットの変数及びモデル定義
			nii_robot.x3d	ロボットの身体モデル定義
(*3)	csv	お好み焼き協調料理の状態遷移に関わる CSV ファイルを格納します	Robot_000_state.csv	ロボットの状態遷移情報の定義
			Robot_000_condition.csv	ロボットの状態遷移条件の定義
			Robot_000_transition.csv	ロボットの状態遷移処理の定義
(*4)	motions	お好み焼き協調料理の身体動作に関わるデータファイルを格納します	ges92r	ロボットの油に手を伸ばす身体動作の定義
			ges92a	ロボットの油に手を伸ばす身体動作の定義
(*5)	command	お好み焼き協調料理のエージェントの C++ソースを格納します	libcommand *.cpp, *.h	お好み焼き協調料理用のエージェントプログラム(SharedObject)

2. 7. 2. クライアントサイド

クライアントサイドには、SIGViewer と SIGViewer の表示に関するテクスチャ(表示画像)、お好み焼き GUI、お好み焼き CUI を格納する。以下にクライアントに関する物理配置を記す。



項番	ディレクトリ名	ディレクトリの役割	主な格納ファイル	ファイルの役割
(*1)	release	SIGViewer 本体と各種定義ファイルを含む	startup.cfg	SIGViewer の動作設定を保持する
(*2)	texture	SIGViewer 上で表示する物体の見たい目画像を含む	oknomiyaki_katame nn_c01.jpg	お好み焼きの見たい目を提供する
(*3)	OkonomiyakiGUI	お好み焼き GUI を含む	Okonomiyaki.cfg	お好み焼き GUI 画面項目を保持する
(*4)	OkonomiyakiCUI	お好み焼き CUI を含む	Utterance.cfg	読み上げ及びテロップ表示対象とする文言の設定を行う

2. 8. お好み焼き協調料理のファイル仕様

お好み焼き協調料理は、3つのプログラムファイル(実行ファイル)の他に、各エージェント用に様々な設定ファイルを必要とする。そのファイルは主に状態遷移の定義に関するファイルである。状態遷移の定義に関するファイルは、大きく分けて「状態遷移情報」、「状態遷移条件」、「状態遷移処理」、「属性定義」の4つとその他に分けられる。以下にそれらのファイルの一覧と書式仕様を記す。

ファイル名	種別	定義の用途	定義の内容
*_state.csv	state	状態遷移情報	各エージェントの状態遷移「情報」を CSV 形式で定義する
*_condition.csv	condition	状態遷移条件	各エージェントの状態遷移「条件」を CSV 形式で定義する
*_transition.csv	transition	状態遷移処理	各エージェントの状態遷移「処理」を CSV 形式で定義する
*_attribute.csv	attribute	エージェント属性情報	各エージェントの属性を CSV 形式で定義する
*_visual.csv	visual	見た目定義	各エージェントの見た目を定義する
*_texts.csv	texts	対話用補助定義	自然言語を状態と結びつける為の定義である

2. 8. 1. 状態遷移情報

全エージェントの状態遷移情報の内容は、挙動仕様書の通り。状態遷移情報は CSV 形式で記述し、縦は前状態、横は後状態として記述する。以下に書式と記述例を記す。

前後	S <sub>1</sub>	S <sub>2</sub>	...	S <sub>n</sub>
S <sub>1</sub>	10	20	...	ε
S <sub>2</sub>	ε	ε	...	30
...	...	...	...	...
S <sub>n</sub>	ε	10	...	ε

上記、表中 S は状態名を示す。最も左上の欄は、前状態 S<sub>1</sub> から後状態 S<sub>1</sub> の遷移可能を示す。これは同状態で定期的に状態遷移を行う。その右隣の欄は、前状態 S<sub>1</sub> から後状態 S<sub>2</sub> の遷移可能を示す。表中「ε」は状態遷移不可を示す。「...」は不特定多数の指定可能を示す。

```

""",0,1,2,3,4,5,6,7,8,9,10,12,13,14,15,16,17,18,19,20,21,22
0,0,10,""
1,"",0,10,"",10,30,""
2,"",0,30,10,30,""
3,"",0,10,30,""
4,"",0,30,""
5,"",0,30,"",10,10,10
6,"",0,60,"",30,10,"",10,10,10
7,"",0,10,30,10,"",10,10,10
8,"",0,30,10,"",10,10,10
9,"",0,10,"",10,10,10
10,"",0,10,"",10,"",10,10,60,"",10,10,10
12,"",0,30,10,"",10,"",60,"",10,10,10
13,"",10,0,10,"",10,"",60,"",10,10,10
14,"",10,"",0,30,10,"",60,"",10,10,10
15,"",10,"",10,0,10,"",60,"",10,10,10
16,"",10,"",10,"",0,30,60,"",10,10,10
17,"",0,60,"",10,10,10
18,"",0,30,10,10,10
19,"",0,10,10,10
20,"",0,0,0,0,0,0,0,0,0,0,0,10
21,"",0,0,0,0,0,0,0,0,0,0,0,10
22,"",0
    
```

上記は実際の CSV 形式でのアバタの状態遷移情報を示す。0～22 までの状態を定義し、かつ状態遷移可能な接点を数値で表現、状態遷移不可は「""(空文字)」で表す。

また、状態遷移情報は、そのファイル名に補助度を付加する場合がある。その際のファイル名の命名規則は以下の通り。もし、補助度を省略した場合は、補助度を「」（空文字）」としたファイル名を利用する。

ファイル名	種別	ファイル名命名規則	定義例
*_state.csv	state	"Agent name" + "_state" + "Supplementary level" + ".csv"	Robot_000_state2.csv

2. 8. 2. 状態遷移条件

全エージェントの状態遷移条件の内容は、挙動仕様書の通り。状態遷移条件は CSV 形式で記述し、縦は前状態、横は後状態として記述する。以下に書式と記述例を記す。

前後	S <sub>1</sub>	S <sub>2</sub>	...	S <sub>n</sub>
S <sub>1</sub>	State=1@Avator_000	Attr=2@Teppan_000	...	ε
S <sub>2</sub>	ε	ε	...	Time=6<Okonomi_000
...	...	...	...	...
S <sub>n</sub>	ε	State=4@Avator_000	...	ε

上記、表中 S は状態名を示す。最も左上の欄は、前状態 S<sub>1</sub> から後状態 S<sub>1</sub> の遷移に条件「State=1@Avator\_000 (アバタが状態"1"の場合に遷移可能)」を付加する。表中「ε」は状態遷移条件がなく、状態遷移情報で状態遷移が定義されている場合、無条件で状態遷移を許す。「...」は不特定多数の指定可能を示す。

```

"",0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22
0,,,,,,,,,State=1@Avator_000,,,,,,,,,
1,,,,Attr=2@Teppan_000,,,,,,,,,
2,,,,,,,,,Time=10<Okonomi_000,,,,,,,,,
3,,,,,,,,,
4,,,,,,,,,
5,,,,,,,,,
    :
    
```

上記は実際の CSV 形式でのアバタの状態遷移条件を示す。0~22 までの状態を定義し、かつ状態遷移可能な接点を状態遷移条件の命令で表現、状態遷移条件なし(無条件)は「」（空文字）」で表す。

以下に状態遷移条件の命令の一覧を記す。

名前	条件の内容	記述の例
State	自分の状態が指定された状態の場合、状態遷移を許可する	State=2
	他のエージェントの状態が指定された状態の場合、状態遷移を許可する	State=2@Avator_000
	2つのエージェントのどちらかの状態が指定された状態である場合に遷移を許可する	State=2@Avator_000 Robot_000
	状態が State で指定した値未満の場合に遷移を許可する ※この場合、状態名は「数値」のみが有効	State=2>Avator_000
	状態が State で指定した値より大きい場合に遷移を許可する ※この場合、状態名は「数値」のみが有効	State=0<Avator_000
Time	その状態に遷移してから指定時間未満の場合に遷移を許可する ※この時間は、状態遷移の度に0に設定	Time=100>Okonomi_000
	その状態に遷移してから指定時間より大きい場合に遷移を許可する ※この時間は、状態遷移の度に0に設定	Time=500<Okonomi_000

名前	条件の内容	記述の例
Attr	自分自身の属性値が指定値の場合に遷移を許可する	Attr=1
	指定エージェントの属性値が指定値の場合に遷移を許可する	Attr=2@Teppan_000
	自分自身の属性が指定値以外の場合に遷移を許可する	NotAttr=2
Aid	自分自身の補助度が指定値の場合に遷移を許可する	Aid=2
	自分自身の補助度が指定値以外の場合に遷移を許可する	NotAid=2
Uttr	自分自身の発言度が指定値の場合に遷移を許可する	Uttr=1
	自分自身の発言度が指定値以外の場合に遷移を許可する	NotUttr=2

また、状態遷移情報は、そのファイル名に補助度を付加する場合がある。その際のファイル名の命名規則は以下の通り。もし、補助度を省略した場合は、補助度を「」（空文字）としたファイル名を利用する。

ファイル名	種別	ファイル名命名規則	定義例
*_condition.csv	condition	"Agent name" + "_condition" + "Supplementary level" + ".csv"	Robot_000_condition2.csv

### 2. 8. 3. 状態遷移処理

全エージェントの状態遷移処理の内容は、挙動仕様書の通り。状態遷移処理は CSV 形式で記述し、1 列目を状態名、2 列目は状態遷移処理内容として記述する。以下に書式と記述例を記す。

状態名	状態遷移処理
S <sub>1</sub>	State=1@Robot_000
S <sub>2</sub>	ε(*1)
...	...
S <sub>n</sub>	Move=100:80:100@Okonomi_000

上記、表中 S は状態名を示す。状態遷移処理の最も上の行は、状態 S<sub>1</sub> に遷移した場合に「State=1@Robot\_000 (ロボットの状態を 1 に遷移する)」の実行を示す。表中「ε」は状態遷移処理がなく、「…」は不特定多数の指定可能を示す。

(\*1)...何かしらの状態遷移処理の定義を推奨。例えば、状態 1 の場合、「State=1」を指定する等。

"1","Utterance=Please_mix_dough&20"
"2","Utterance=I_take_oil,Motion=ges82r,Motion=ges83r,State=1@Abura_000"
"3","Utterance=Please_put_oil_on_Teppan&20"
⋮

上記は実際の CSV 形式でのアバタの状態遷移処理を示す。

以下に状態遷移処理の命令の一覧を記す。

名前	処理の内容	記述の例
State	他のエージェントの状態を指定した状態に遷移します。但し、他のエージェントが状態の遷移の条件を満たさない場合、その要求は無視する。	State=1@Abura_000 (油を使用済みにする)
Attr	他のエージェントの属性を指定した属性に設定します。例えば、鉄板の火力を設定する場合に使用する。	Attr=2@Teppan_000 (鉄板の火力を「強」にする)
Move	他のエージェントを指定した座標に移動します。座標は X:Y:Z 形式で記述する。	Move=100:80:100@Sauce_000 (ソースを鉄板の上に移す)

名前	処理の内容	記述の例
	他のエージェントを指定した座標を経由しながら移動する。座標は「X:Y:Z X:Y:Z形式」で記述します。座標はいくつでも指定可能。	Move=100:70:100 100:80:100@Sauce_000 (ソースを鉄板の上に移動して持ち上げる)
Angle	他のエージェントを指定した向きに回転する。「X方向(0,1):Y方向(0,1):Z方向(0,1):開始角度:終了角度:増分角度」形式で記述。これは Operation で操作した形式と同等。	Angle=1:0:0:0:180:10@Okonomi_000 (お好み焼きを裏返す)
Visual	他のエージェントの見た目を変えます。例えばお好み焼きにソースをかけた場合など。	Visual=5@Okonomi_000 (お好み焼きの見た目を、「ソースがかけられた見た目」に変える。)
Motion	(指定の動作を行う。詳細は、「身体動作の定義」を参照。これは身体動作の指定です。)	Motion=ges92r (ボウルに手を伸ばす)
Utterance	指定の文言を発話する。但し、スペースは「_」に指定してください。	Utterance=Hello (「Hello」と発言する)
	指定の文言を発話する。指定した文言は、指定ユニット時間経過の後、繰り返して発言。そして、この指定の後の Transition は全て無視する。	Utterance=Hello&20 (「Hello」と20ユニット時間間隔で繰り返して発言)
	状態遷移した初回のみ、指定の文言を発話する。	Utterance=Hello&first (その状態で一度のみ「Hello」と発言)
	指定ユニット時間の間、待つ。指定ユニット時間が過ぎた場合、指定の文言を発話して、続きの Transition を実行する。	UttrAndAction=GoodBye&10 (10ユニット時間が経過した場合、「さようなら」と発言する)
	指定の条件を満たす場合、発言する。その後の Transition は行わない。(アバタが1(油をとった)の場合、お礼を言う。自分はその後の Transition (油をとる)を行わない。)	UttrAndReject=Thanks&1@Avator
Reject	既に行われた状態遷移の場合、処理を実行しない。	Reject=AlwasState

また、状態遷移処理は、そのファイル名に補助度を付加する場合がある。その際のファイル名の命名規則は以下の通り。もし、補助度を省略した場合は、補助度を「'''(空文字)」としたファイル名を利用する。

ファイル名	種別	ファイル名命名規則	定義例
*_transition.csv	transition	"Agent name" + "_transition" + "Supplementary level" + ".csv"	Robot_000_trnsition2.csv