SV アシスタント試作品 v0.1

設計書

1. アセット

本プロジェクトでは下記の外部アセットを利用している。

アセット名	有料／無料	説明
Cartoon FX	有料	パーティクル効果。 呼び出し中の敵エフェクトや、攻撃エフェクトに利用。
CustomizableSword_Free	無料	剣モデル。
dpt_zippermouth	無料	敵モデル及びアニメーション。
Fantasy Skybox FREE	無料	スカイボックスに利用。
girlWarrior155	不明	出処不明の為、商用利用はかなりグレー。 待機、走る、攻撃等のアニメーションに利用。
MecanimExample	無料	子熊(TeddyBear)モデルを利用。
NassacGamesShieldsPack	無料	盾モデル。
NGUI	無料	NGUI。2D 文字表示やスプライト表示等など。 有名アセットの無料版なので使い方は調べれば出てきます。
Nomads Music	無料	サウンド集。 待ち呼閾値超え警告音に利用。
SOUND ATELIER	無料	サウンド集。未使用。
SU3DJPFont	無料	日本語フォント。
iTween	無料	position、scale、rotation などのプロパティを自由にアニメーションさせる Asset。 使い方は下記参照。 www40.atwiki.jp/spellbound/pages/1604.html
UnityChan	無料	ユニティちゃん

2.　UnityProject フォルダ構成

Assets 配下に下記のフォルダ構成をとる。

第１階層	第２階層	第３階層	保存アイテム
3rdAssets			外部 Asset
_Animations			アニメーションクリップ
_ Animattors			アニメーションコントローラー
_Atlases			NGUI 用アトラス
_Editor			エディタスクリプト
_Materials			マテリアル
_Prefs			プレハブ
_Scripts			ロジックスクリプト
	_Manager		マネージャ関連スクリプト
		Asterisk	アスタリスク関連マネージャスクリプト
		Queue	アスタリスクキュー関連マネージャスクリプト
	Asterisk		アスタリスク関連スクリプト
	Game		ゲームロジック関連スクリプト
	GUI		GUI 関連スクリプト
	Library		その他スクリプト
	Tweet		Tweet 関連スクリプト
_Textures			テクスチャ
Editor			エディタスクリプト
	Image Effects		ImageEffects 用エディタスクリプト(標準 Asset)
Plugins			プラグイン
	Npgsql		PostgresSQL アクセスライブラリ
	Pixelplacement		iTween のプラグイン
Scenes			シーン
Standard Assets			標準 Asset

3. UnityProject ヒエラルキー構成

ヒエラルキー内に下記の構成をとる。

Light				ライトグループ
	Directional light			メインライト
	Directional light for UnityChan			ユニティちゃん専用ライト
Main Camera				メインカメラ
Managers				マネージャ郡
	AsteriskManager			Asterisk マネージャ
	ConfigManager			設定マネージャ
	GameManager			ゲームマネージャ
	GUIManager			2DGUI マネージャ
	QueueManager			AsteriskQueue マネージャ
	TweetManager			Tweet マネージャ
SpawnArea				オペレータ（ファイター）及びコール（敵）の出現範囲
SplashLogo				スプラッシュロゴ
	Camera			スプラッシュロゴ用カメラ
UIRoot(2D)				2DGUI Root　(NGUI)
	2D Main Camera			2DGUI 用カメラ
		2D Main Panel		2DGUI 用パネル
			CallsInfoGrp ・sign_o_S ・sign_x_S ・WaitCount ・WaitLables ・WarningLabel ConfigButtonGrp ・ConfigButton ・ShadowToggleButton TweetGroup ・TweetBox ・TweetDateLabel	呼情報グループ ・ユニティちゃん○ ・ユニティちゃん× ・待ち呼数 ・Wait の文字 ・WARNNING の文字 設定グループ ・設定（ギア）ボタン ・影の ON/OFF ボタン Tweet グループ ・Tweet 表示エリア背景 ・Tweet 日付表示用ラベル
View UI				2DGUI Root　(NGUI)

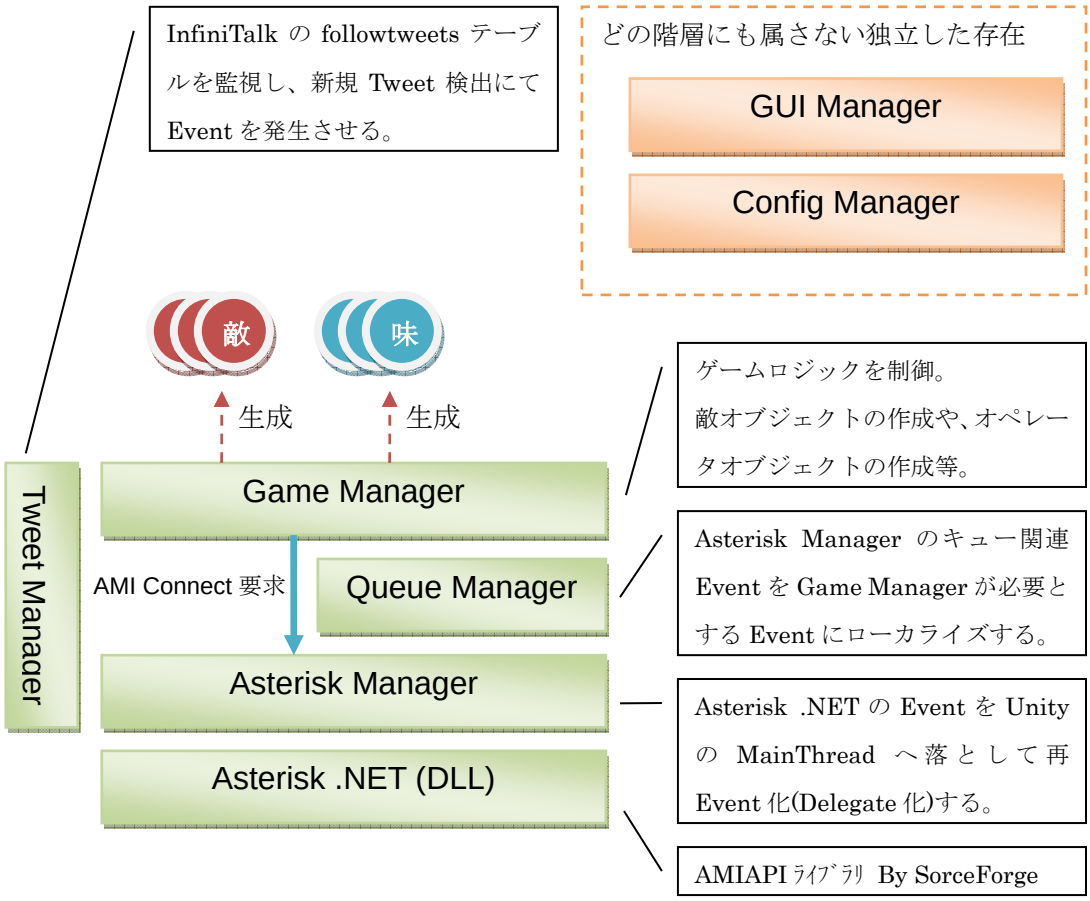
	Anchor			2DGUI 用アンカー
		Offset		2DGUI 用パネル
			TweetLabel	ツイート表示用ラベル
	View Camera			2DGUI 用カメラ（Tweet 用）
World				世界グループ
	Plane			地面

4.　スクリプト基本構成

ロジックの基本となるスクリプト郡を図と共に説明する。

各ブロックは上位階層を極力意識しなくても良い構成を取っており、必須となる Public 関数以外はいくつかのイベントデリゲートを用意するのみである。

「通知が必要な人は勝手にイベント（デリゲート）登録して使ってね！」というスタンス。



5.　Game Manager

GameManager は画面への敵・味方を含めたキャラクター生成／廃棄とキャラクター間の通信を担当します。

本アプリの改修を行う場合は本ソースからトレースするのが一番簡単かと思います。

6.　Queue Manager

AsteriskManager からの AsteriskEvent を受け取り、GameManager が処理し易いイベントに変換します。

例えば、Bridge イベントを受け取ったら対象キューのブリッジイベントか判定し Queue

Manager 内に保持した Call 一覧から該当 Call 情報を取得し、GameManager へ「メンバー情報」「呼情報」をイベントにて渡します。
QueueManager がいるおかげで、GameManager は余計なロジックを気せず処理を行うことが出来ます。

7. AsteriskManager

AsteriskManager は Asterisk.net からのイベント Unity メインプロセスに落とした上で再イベント化を行います。
これを行わないと何が起こるかというと、そのイベントをそのまま処理すると UnityAPI にアクセスできず、ほぼ何も出来ない状態になります。

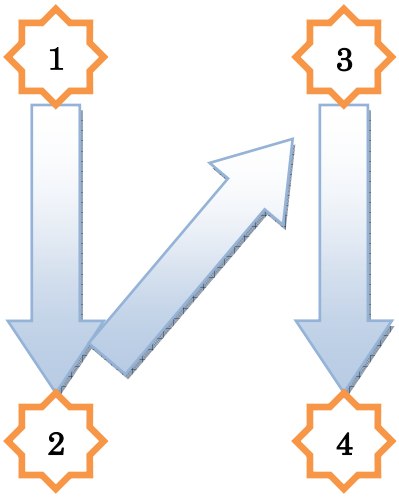
8. TweetManager

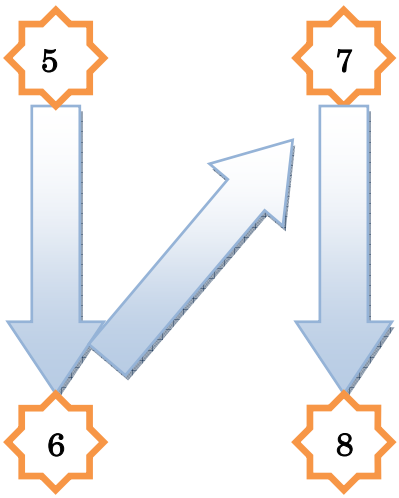
ConfigManager の ConfigUpdate イベントを受け取り PostgreSQL へ接続します。
接続後は周期的に新規ツイート有無をチェックし、新規ツイートがある場合は DetectNewTweet イベントを発行します。

9. 主要動作フロー

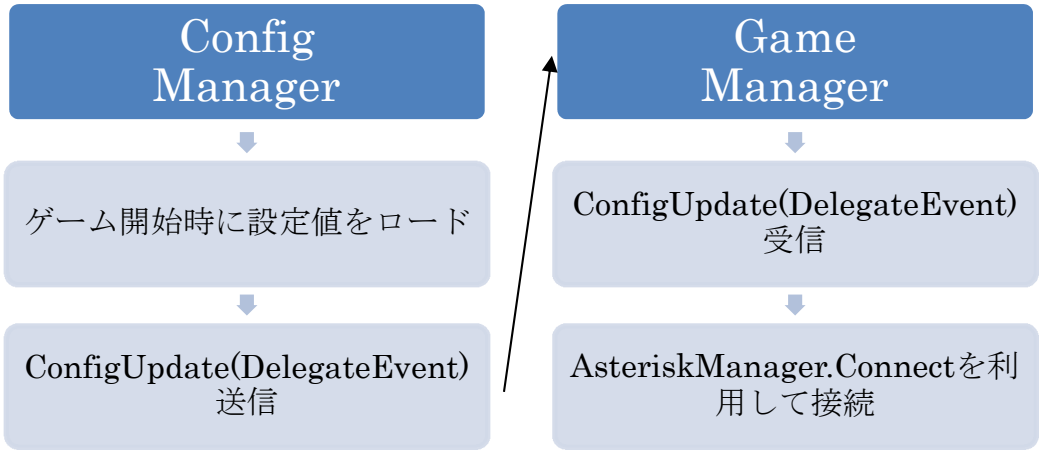
本章では主要動作のフローを記述します。

フローは下記の通りの順番で遷移する記述をとっていますので、そのつもりで読んでください。

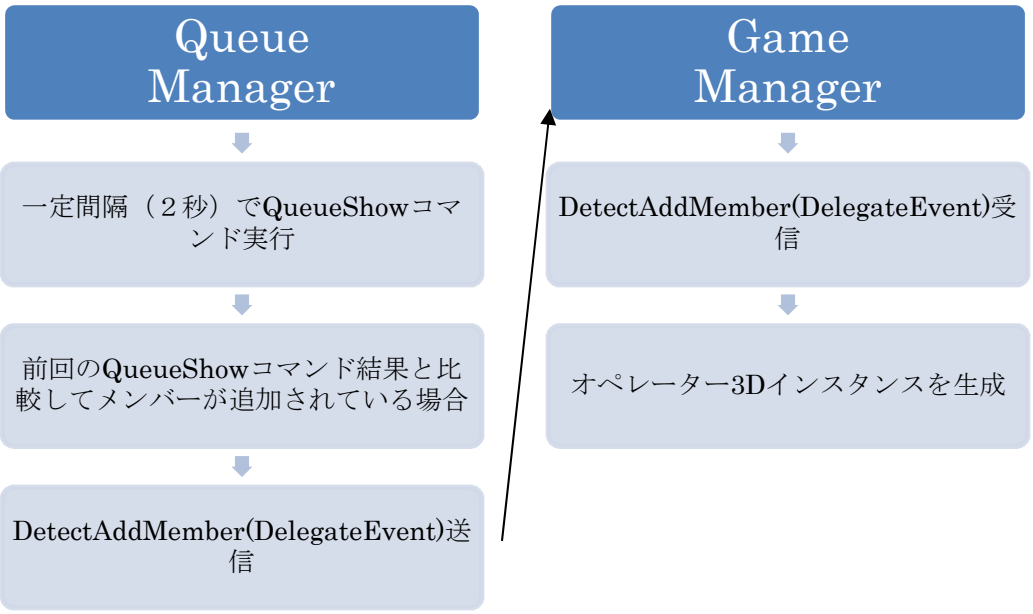




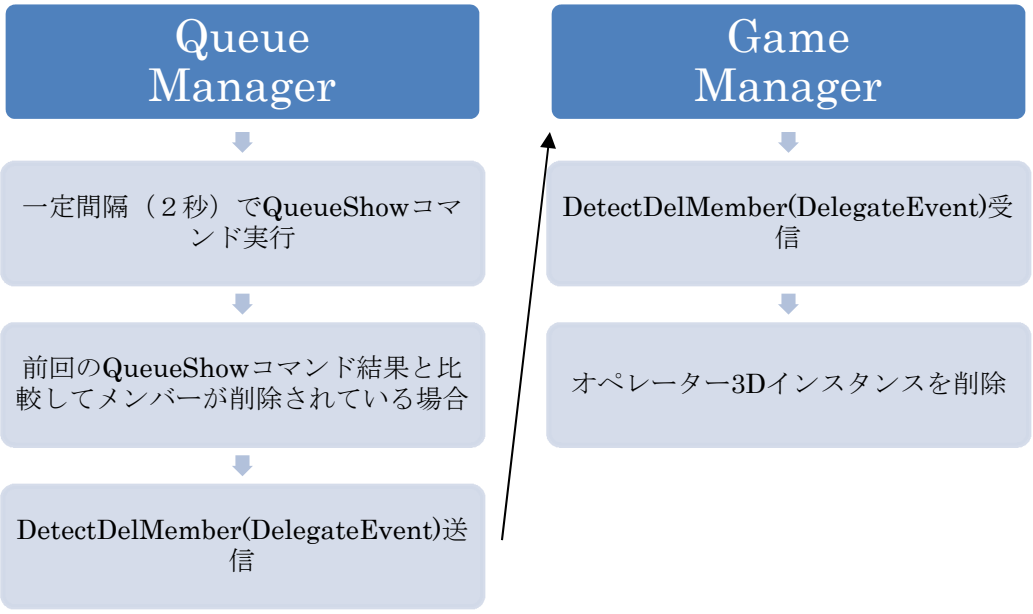
9.1 Asterisk への接続



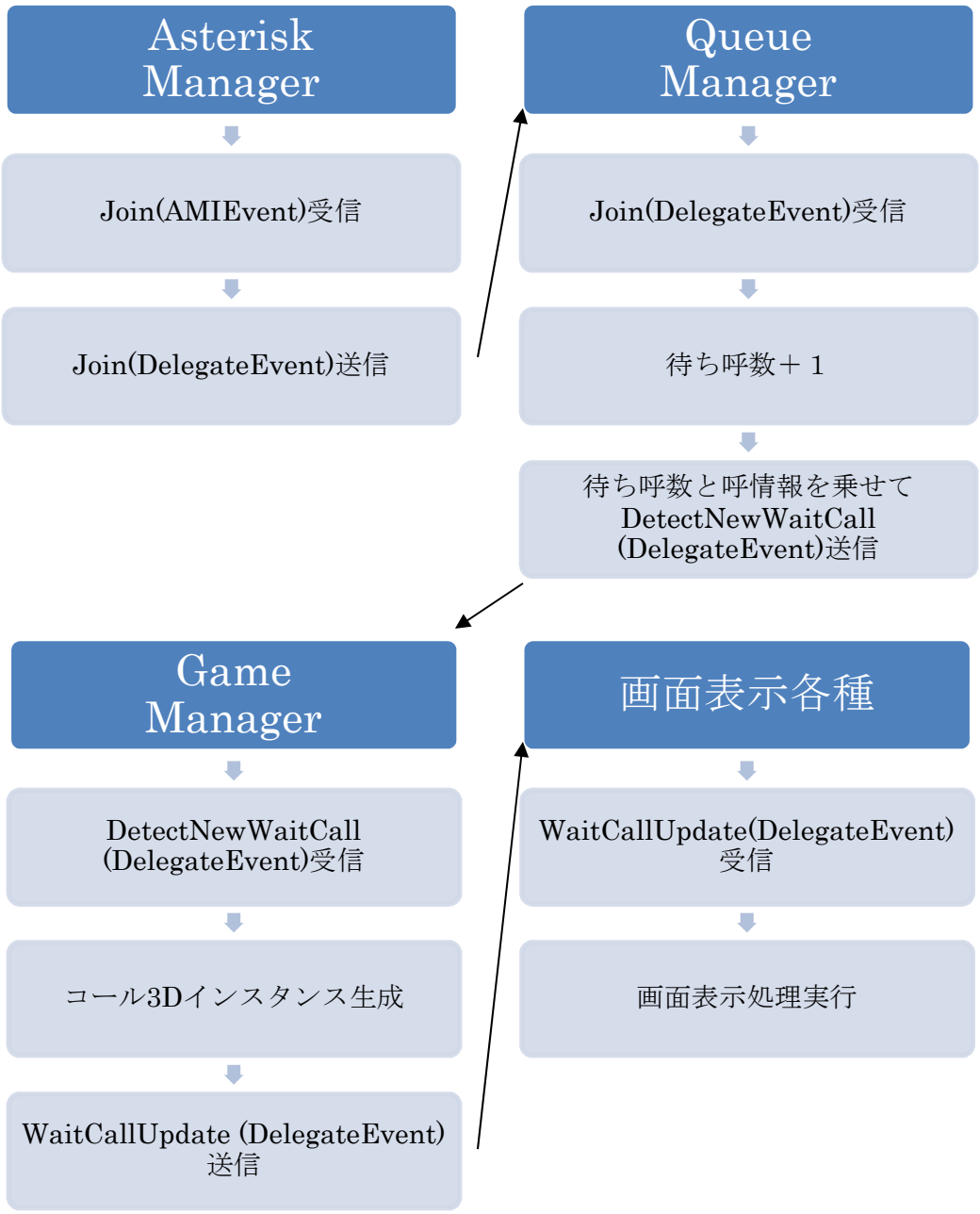
9.2 Queue メンバーの追加



9.3 Queue メンバーの削除



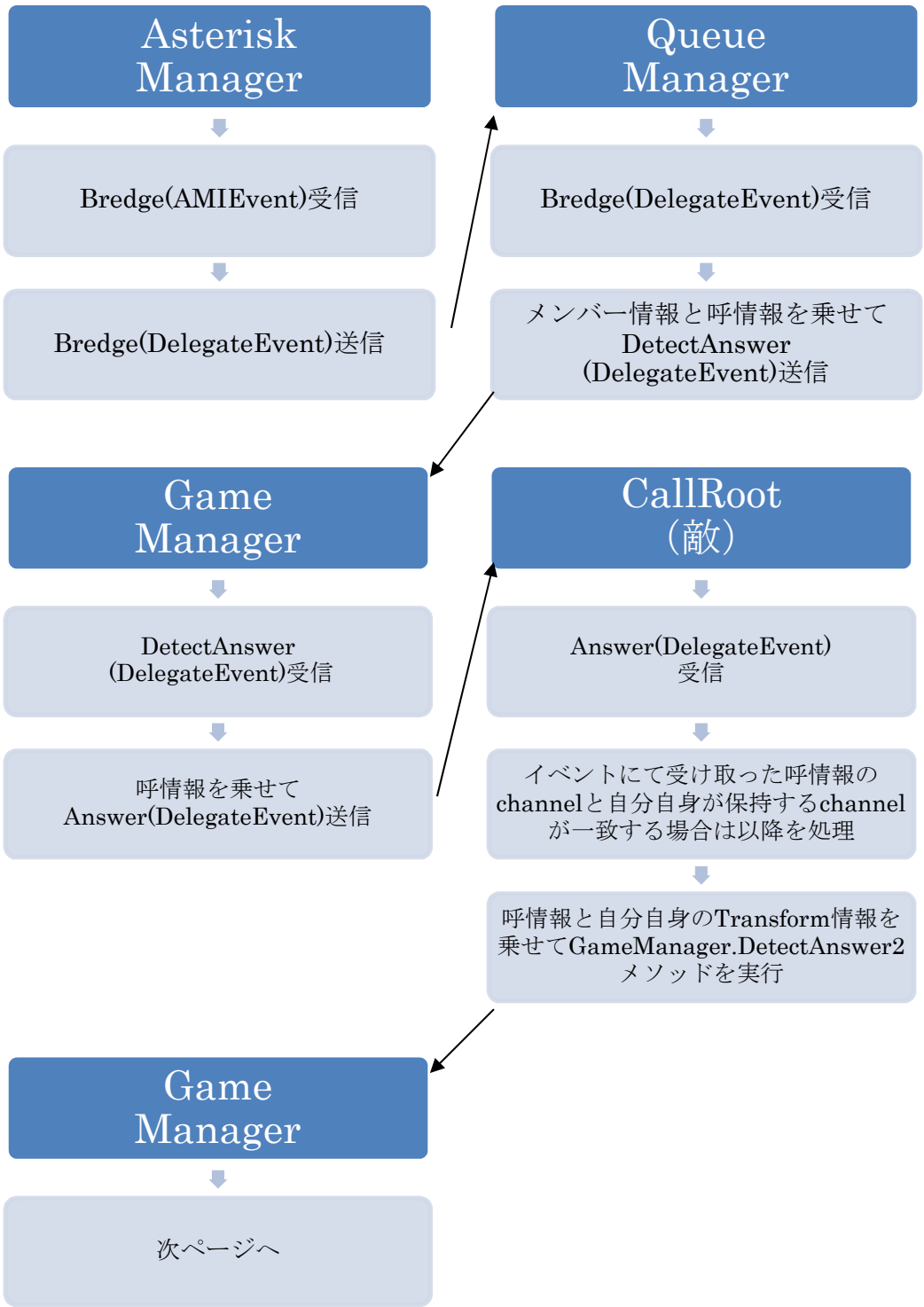
9.4 待ち呼の追加検出

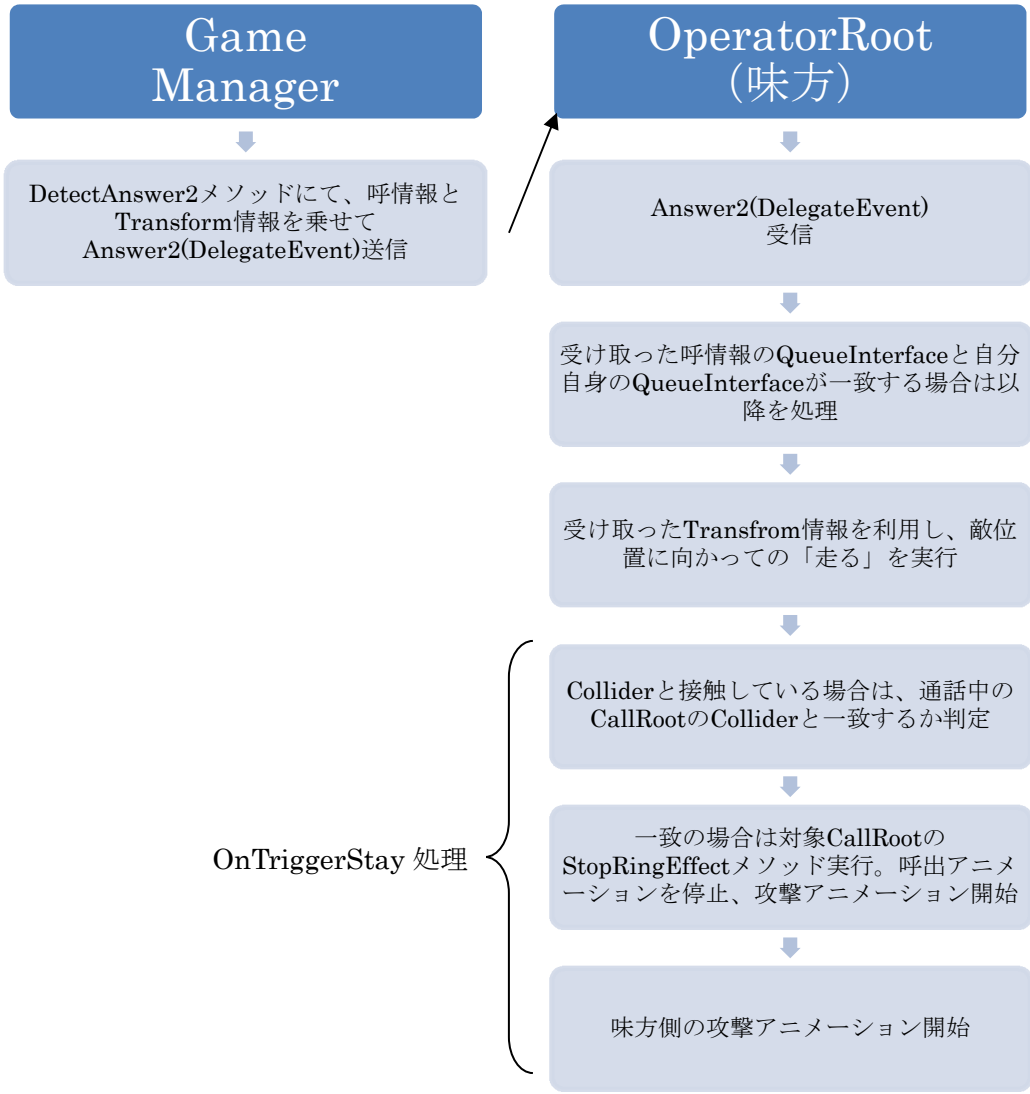

Asterisk Manager
Join(AMIEvent)受信
Join(DelegateEvent)送信
Queue Manager
Join(DelegateEvent)受信
待ち呼数＋１
待ち呼数と呼情報を乗せて DetectNewWaitCall (DelegateEvent)送信
Game Manager
DetectNewWaitCall (DelegateEvent)受信
コール3Dインスタンス生成
WaitCallUpdate (DelegateEvent) 送信
画面表示各種
WaitCallUpdate(DelegateEvent) 受信
画面表示処理実行

9.5 待ち呼の削除検出

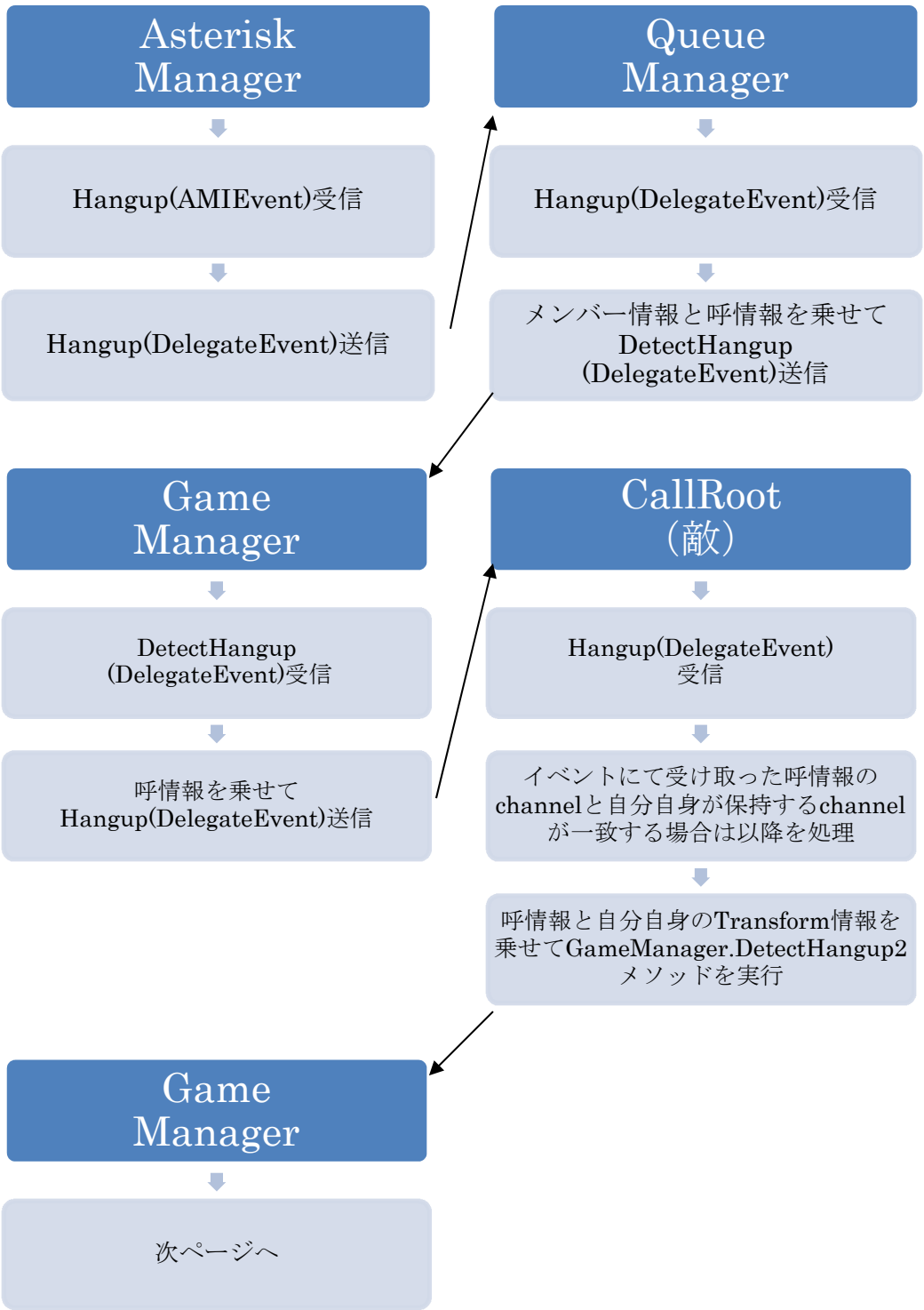

Asterisk Manager
Leave(AMIEvent)受信
Leave(DelegateEvent)送信
Queue Manager
Leave(DelegateEvent)受信
待ち呼数－１
待ち呼数と呼情報を乗せて DetectDelWaitCall (DelegateEvent)送信
Game Manager
DetectDelWaitCall (DelegateEvent)受信
コール3Dインスタンス削除
WaitCallUpdate (DelegateEvent) 送信
画面表示各種
WaitCallUpdate(DelegateEvent) 受信
画面表示処理実行

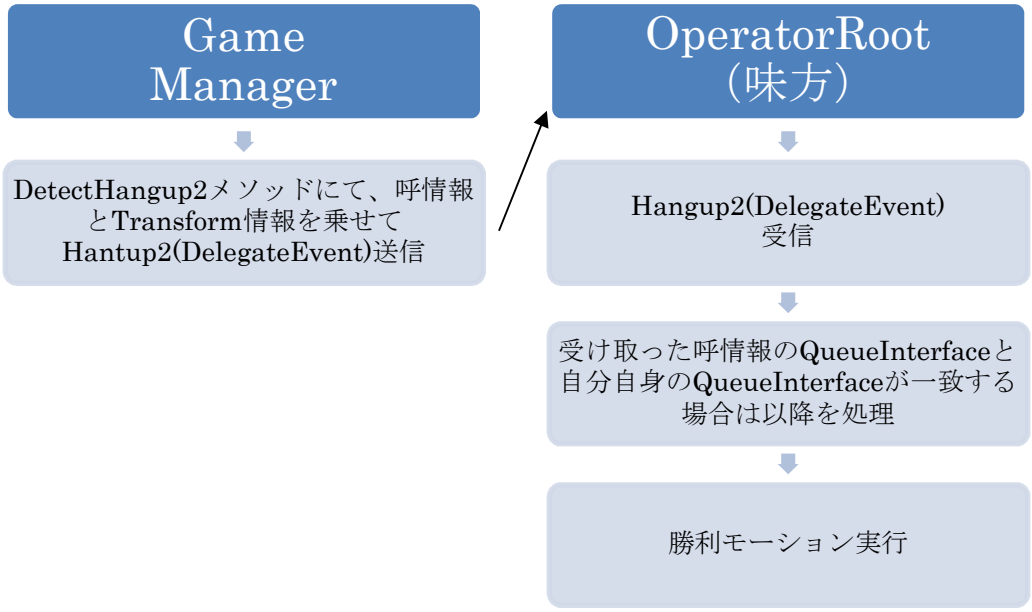
9.6 通話開始（Answer）を検出




Game Manager
DetectAnswer2メソッドにて、呼情報とTransform情報を乗せてAnswer2(DelegateEvent)送信
OperatorRoot（味方）
Answer2(DelegateEvent)受信
受け取った呼情報のQueueInterfaceと自分自身のQueueInterfaceが一致する場合は以降を処理
受け取ったTransfrom情報を利用し、敵位置に向かっての「走る」を実行
OnTriggerStay 処理
Colliderと接触している場合は、通話中のCallRootのColliderと一致するか判定
一致の場合は対象CallRootのStopRingEffectメソッド実行。呼出アニメーションを停止、攻撃アニメーション開始
味方側の攻撃アニメーション開始

9.7 通話終了（Hangup）を検出　（作成中）

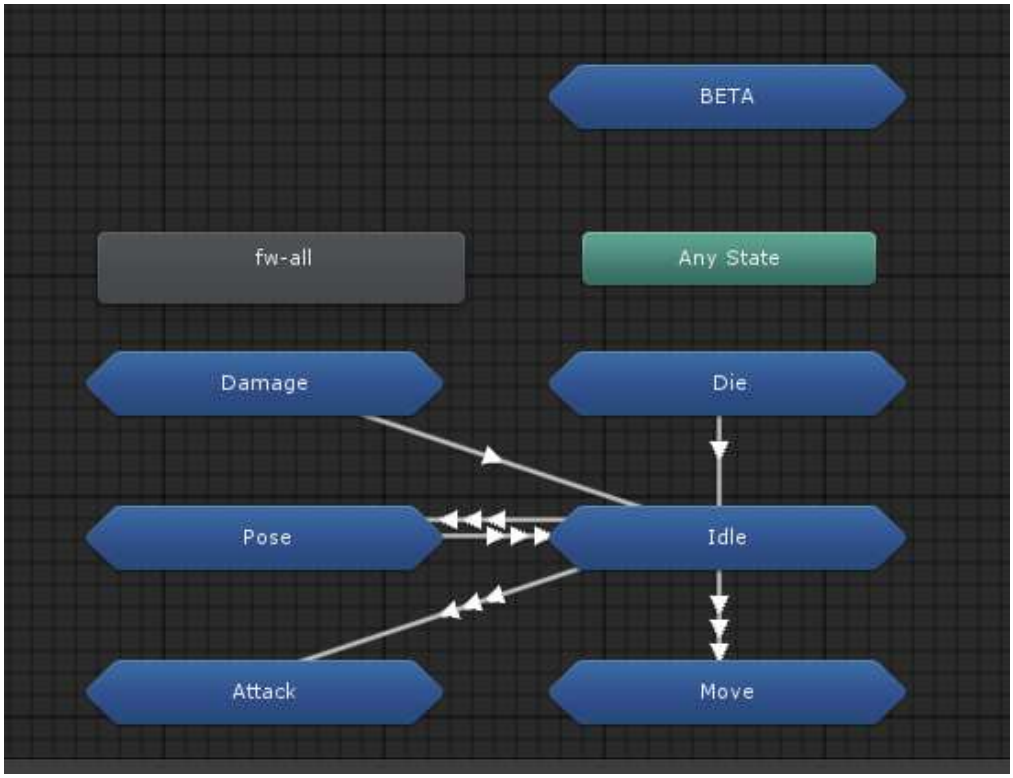



Game Manager
DetectHangup2メソッドにて、呼情報とTransform情報を乗せてHantup2(DelegateEvent)送信
OperatorRoot（味方）
Hangup2(DelegateEvent)受信
受け取った呼情報のQueueInterfaceと自分自身のQueueInterfaceが一致する場合は以降を処理
勝利モーション実行

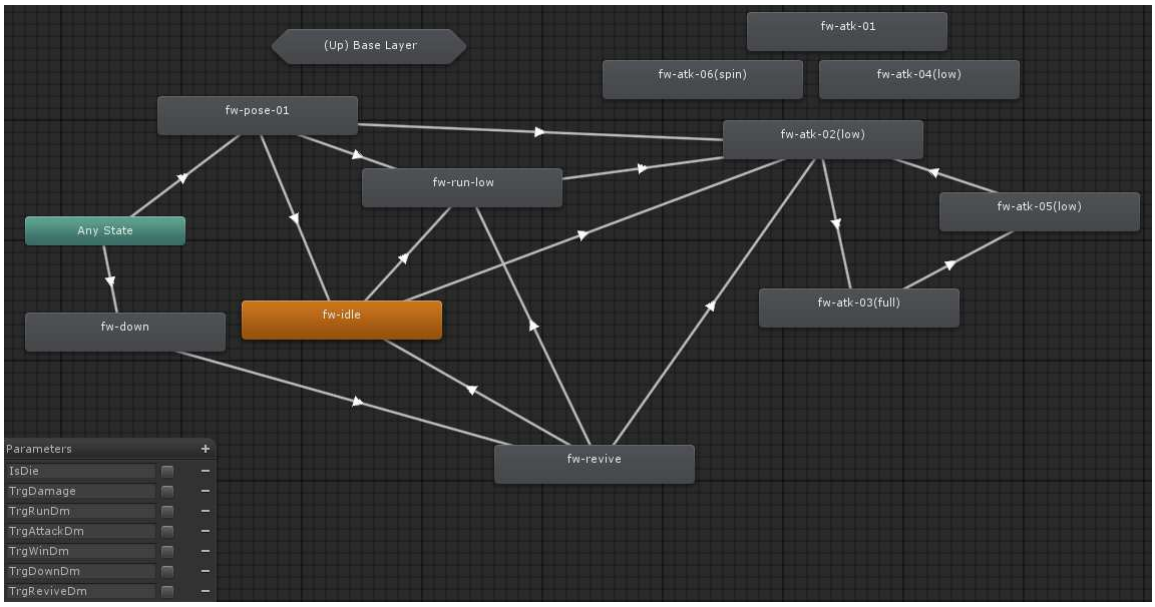
10. オペレータのアニメーションパターン

Assets/_Animators/OperatorFemaleWarrior に従います。

OperatorFemaleWarrior の中身をみると以下のイメージになっているかと思います。



<BETA>以外は使っていません。BETA をダブルクリックして下さい。



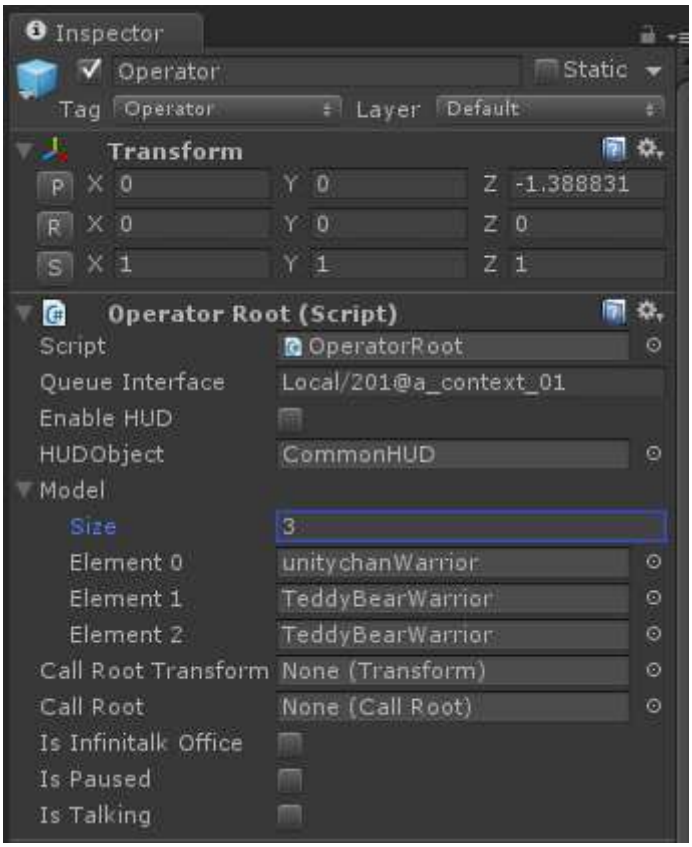
これがキャラクターのアニメーション遷移パターンです。特別なことはやっていないので Unity の Animator（Mecanim）を別途勉強してください。

11. オペレータ（味方）とコール（敵）の出現パターンと行動ロジック

11.1 出現パターン

Operator（若しくは Call）プレファブにアタッチされた OperatorRoot スクリプト（若しくは CallRoot スクリプト）の Model 配列により出現パターンが決定されています。本配列に設定された順にキャラクタープレファブが決定されます。

下記の Operator の例では、Element0 に unitychanWarrior が設定され、Element1 と 2 に TeddyBearWarrior が設定されているため、一人目のオペレータはユニティちゃん、二人目、三人目のオペレータはテディベアとなります。4 人目は Element0 にもどりユニティちゃんです。



11.2 行動ロジック

先の章にて説明した Model の Element に設定するプレファブに OperatorChiled（コール（敵）の場合は CallChiled）を継承したスクリプトをアタッチする必要があります。

OperatorChiled（若しくは CallChiled）には各イベント時にコールされるメソッドが用意されています。継承先のクラスにて本メソッドをオーバーライドし、希望の処理を記述することで、キャラクターに特徴を持たせることが可能です。

OperatorChiled

メソッド名	実行契機
DetectAwake	インスタンス化された時
DetectAnswer	通話状態になった時
DetectHangup	通話が終了した時
DetectPaused	Asterisk の Queue 状態が Pause になった時
DetectParentDestroy	OperatorRoot が Destory する直前

CallChiled

メソッド名	実行契機
DetectAwake	着信時（インスタンス化された時）
DetectAnswerd	通話状態になった時
DetectLeave	Asterisk の Queue 管理から外れた時（通話開始や切断）
DetectHangup	通話が終了した時
DetectParentDestroy	CallRoot が Destroy する直前