プラグインの仕様に関する詳細資料［01］

基本

- コードはUTF8 で記述してください。
- プラグインの仕様は、 追加・　　変更される可能性があります。

プラグインの概要

プラグインの概要を説明します。

プラグインは、 スクリプトプラグインを想定しています。
スクリプトプラグインはエディターで管理され、 エディターUI 上でパラメータ指定ができます。 (オートタイルプラグイン、 独自アクションコマンドプラグイン、 独自リンク条件プラグインなど。)
スクリプトの対応言語は JavaScript とします。 スクリプトプラグインを CoffeeScript で記述することもできます。
CoffeeScript は自動的に JavaScript に変換されて使用されます。

スクリプトプラグインはエディタのプラグイン画面で追加することができます。
追加されたプラグインはプロジェクトフォルダ内の plugins フォルダに保存されます。
JavaScript ファイルの拡張子は js、 CoffeeScript の拡張子は .coffee とします。
CoffeeScript は読込や更新の時に自動的に JavaScript に変換され、 拡張子が js のファイルとして plugins フォルダに保存されます。

スクリプトプラグインの多言語対応のため Editor のロケールがsetLocale( ) 関数で指定されます。スクリ
プトプラグインの内部処理で戻り値の言語を切り替えることで多言語対応が可能になります。
※「ja_JP」 のように地域識別文字列が付いていますが、 地域が不要であれば先頭の 2 文字を切り出して利用してください。

スクリプトプラグインはスクリプトファイルロード時に JavaScript オブジェクト (スクリプトプラグインオブジェクト) を返す必要があります。
ロードに失敗 (JavaScript の解析に失敗) した場合や、 getInfo で情報を取得できない場合は、 破棄されます。 グローバル領域を汚さないようにしてください。
スクリプトプラグインオブジェクトはグローバルオブジェクト Agtk.plugins に格納されます。 (スクリプトプラグインオブジェクトは Agtk.plugins[< プラグイン ID>] でアクセスできます。)

現時点でスクリプトプラグインオブジェクトは次の関数を提供する必要があります。

setLocale	関数
getInfo	関数
initialize	関数
finalize	関数
call	関数
update	関数

※省略可能

スクリプトプラグインでは cocos2d-x の JSB API を呼び出すことができます。
http://www.cocos2d-x.org/docs/api-ref/js/v3x/

ゲームのプレビュー時にスクリプトの実行ログが player.exe と同じディレクトリーのscript.log に出力されます。

スクリプトプラグイン IF

関数名	説明	パラメータ	戻り値	備考	エディター実装	プレイヤー実装
getInfo	プラグイン情報を取得する	arg1: (string)category			✓	✓
		name	プラグイン名	文字列型を返す。		
		description	プラグイン説明	文字列型を返す。		
		author	作者名	文字列型を返す。		
		help	プラグインヘルプ	文字列型を返す。		
		parameter	プラグインパラメータ定義のJavaScriptオブジェクト	プラグイン詳細画面でプラグインパラメータの設定を行うことができる。 ※「UIパラメータについて」を参照。 [{id: <パラメータID1>, name: <パラメータ名1>, type: "String"/"Number"/"Json"/"ImageId"/"TextId"/… /"Custom"/"Embedded"/"EmbeddedEditable", decimals: <Numberの時のみ。小数点の桁数>, minimumValue: <Numberの時のみ。最小値>, maximumValue: <Numberの時のみ。最大値>, customParam: [{id: <カスタムID1>, name: <カスタム名1>}, ...], defaultValue: <デフォルト値1>, …}]		
		internal	プラグイン状態を表すデータ	プラグインの内部情報を返す。フォーマットはJSON.stringifyで文字列化できる情報のみ次回のinitializeやsetInternalで引き渡される。 ゲームプレイのセーブ・ロードの際、保 存・復帰される。		
		actionCommand	独自アクションコマンドの定義	アクションコマンド選択画面に、ここで定義した独自アクションが出てくる。 独自アクションコマンドを追加するとその設定画面でparameterで指定した設定項目が表示される。 ※parameter部については「UIパラメータについて」を参照。 [{id: <独自アクションコマンドID>, name: "<独自アクションコマンド名>", description: "<独自アクションコマンド説明>", parameter: <記述はプラグインパラメータ定義と同様>, …}, …}		
		linkCondition	独自リンク条件の定義	リンク条件選択画面に、ここで定義した独自リンク条件が出てくる。 独自リンク条件を追加するとその設定画面でparameterで指定した設定項目が表示される。 ※parameter部については「UIパラメータについて」を参照。 [{id: <独自リンク条件ID>, name: "<独自リンク条件名>", description: "<独自リンク条件説明>", parameter: <記述はプラグインパラメータ定義と同様>, …}, …}		
		autoTile	オートタイルパラメータ定義	タイルセット設定画面のプラグイン選択項目の下に、ここで指定した設定項目が表示される。 記述はプラグインパラメータ定義と同様		
initialize	プラグインを初期化する	getInfo('internal')で返されたデータ	–	システムから呼び出される。 初回はundefinedが渡される。 プレイヤーでは、エディターのプロジェクトセーブ時にgetInfo('internal')で返されたデータが渡される。	✓	✓
finalize	プラグインを終了化する	–	–	システムから呼び出される。 プラグインの終了化処理を行う。必要が無ければ何もしなくても良い。 プラグインの内部状態は、getInfo('internal')で取り出され、次回のinitializeの呼び出しのパラメータとして渡される。 プラグインの内部状態は、エディターとプレイヤーで独立して記録される。	✓	✓
update	更新処理を呼び出す	arg1: <delta時間>	–	システムから呼び出される。 毎フレーム呼び出される。 <delta時間>に前のフレームからの経過時間が入る。		✓
setLocale	プラグインに動作して欲しいロケールを設定する。	arg1: (string)locale	–	システムから呼び出される。 locale="en"へのは必須。	✓	✓
setParamValue	プラグインパラメータに設定されたデータを設定する。	arg1: パラメータ設定のJSONをパースしたオブジェクト	–	システムから呼び出される。 エディターのプラグイン設定画面でパラメータが更新されるたびに呼び出される。	✓	–
setInternal	内部データを設定する	プレイデータのセーブ時にgetInfo('internal')で取り出されたデータ	–	システムから呼び出される。 プレイデータのロード時に呼び出される。		✓
call	各種機能を呼び出す				✓	
	タイルー覧の横タイル個数を返す	arg1: <object: オートタイルパラメータ設定値> arg2: "getHorzCount"	横タイル個数	システムから呼び出される。 オートタイルプラグインでは定義する必要あり。		
	タイルー覧の縦タイル個数を返す。	arg1: <object: オートタイルパラメータ設定値> arg2: "getVertCount"	縦タイル個数	システムから呼び出される。 オートタイルプラグインでは定義する必要あり。		
	タイルー覧のタイル情報リストを返す。	arg1: <object: オートタイルパラメータ設定値> arg2: "getFrontTileList"	タイル情報リスト	システムから呼び出される。 オートタイルプラグインでは定義する必要あり。		
	タイルー覧の各タイルが取りまとめているタイルの範囲情報のリストを返す。	arg1: <object: オートタイルパラメータ設定値> arg2: "getBackTileList"	タイルの範囲情報のリスト	システムから呼び出される。 オートタイルプラグインでは定義する必要あり。		
	このオートタイルプラグインが扱う属性のリストを返す。	arg1: <object: オートタイルパラメータ設定値> arg2: "getAttributeList"	属性のリスト	システムから呼び出される。 オートタイルプラグインでは定義する必要あり。		
	指定範囲のオートタイルを更新する。	arg1: <object: オートタイルパラメータ設定値> arg2: "updateArea" arg3: [<int: タイル開始座標X>, <int: タイル開始座標Y>, <int: タイル横個数>, <int: タイル縦個数>]	–	システムから呼び出される。 オートタイルプラグインでは定義する必要あり。		
	パラメータの変更を通知する	arg1: <object: オートタイルパラメータ設定値> arg2: "onParamChanged" arg3: <前回の設定値>	–	システムから呼び出される。 オートタイルプラグインでは定義する必要あり。		
getAttribute	指定タイル座標の属性を返す。	arg1: <object: オートタイルパラメータ設定値> arg2: <int: タイル座標X> arg3: <int: タイル座標Y> ret: <string: 属性文字列>	属性	他のオートタイルプラグインが設定する属性を使って自身のオートタイルで使用する タイルを切り替えるために使用する。	✓	–
getBorderType	指定タイル座標の境界情報を返す。	arg1: <object: オートタイル情報> arg2: <int: タイル座標X> arg3: <int: タイル座標Y> ret: <string: 境界文字列>	境界情報	他のオートタイルプラグインが設定する境界情報を使って自身のオートタイルで使用するタイルを切り替えるために使用する。	✓	–
execActionCommand	アクションコマンドを実行する。	arg1: <アクションコマンドインデックス> arg2: <設定パラメータ> arg3: <オブジェクトID> arg4: <インスタンスID> arg5: <アクションID> arg6: <コマンドID> arg7: <コモンアクションのとき1、そうでなければ0> arg8: <配置後も変更可能なアクションの場合配置シーンID、そうでなければ-1>	コマンド戻り値 「アクションボックスの「その他の実行アクション」の処理の流れ」を参照。	システムから呼び出される。 プラグインで複数のアクションコマンドを定義した場合に、<アクションコマンドインデックス>で区別する。 アクションコマンドで設定したパラメータが<設定パラメータ>でオブジェクトとして引き渡される。	–	✓
execLinkCondition	リンク条件を評価する。	arg1: <リンク条件インデックス> arg2: <設定パラメータ> arg3: <オブジェクトID> arg4: <インスタンスID> arg5: <アクションリンクID> arg6: <コモンアクションの入るリンクの場合0、出ていくリンクの場合1、コモンアクションでない場合-1>	リンクが成立するかの真偽値	システムから呼び出される。 リンク条件評価時に呼び出される。 プラグインで複数のリンク条件を定義した場合に、<リンク条件インデックス>で区別する。 リンク条件で設定したパラメータが<設定パラメータ>でオブジェクトとして引き渡される	–	✓

システム提供スクリプト API［1］

名前空間	名前	パラメータ	説明	備考	エディター実装	プレイヤー実装
Agtk.version			実行エンジンのバージョンを表す文字列	PGMMV〈バージョン番号〉 player 〈バージョン番号〉 runtime 〈バージョン番号〉	✓	✓
Agtk.log		arg1: 〈string: 出力する文字列〉	ログを出力する。 プレイヤーでは「実行ログコンソール」に出力される。		✓	✓
Agtk.reset		引数なし	ゲームをリセットする。F5キーによるリセットと同様の動作を行う。			✓
Agtk.settings	tileWidth		タイル幅(単位：ドット)		✓	✓
	tileHeight		タイル高さ(単位：ドット)		✓	✓
	screenWidth		画面幅（単位：ドット）		✓	✓
	screenHeight		画面高さ（単位：ドット）		✓	✓
	playerCharacters		playerCharactersのオブジェクト			✓
	projectPath		プロジェクトパス			✓
Agtk.playerCharacters	getCount	–	プレイヤーキャラクター管理の枠の数を返す。 ※現状4で固定。			✓
	get	arg: 〈インデックス〉	指定枠のplayerCharacterを返す。			✓
Agtk.playerCharacter	getCount		プレイヤーキャラクター管理枠に設定されたオブジェクトの数を返す。			✓
	get	arg: 〈インデックス〉	Object IDを返す。			✓
Agtk.scenes	getCount/getIdByIndex/getIdByName/getById				✓	
	getIdList/getIdByName/get					✓
Agtk.scene	id/name				✓	✓
	getLayerByIndex				✓	
	horzScreenCount		シーンサイズ幅		✓	✓
	vertScreenCount		シーンサイズ高さ		✓	✓
	getLayerIdList/getLayerIdByName/getLayerIndexById/getLayerById					✓
Agtk.layers	getCount/getIdByIndex/getIdByName/getById				✓	
Agtk.layer	getTileInfo	arg1: x arg2: y	シーンに設定されたタイルの情報を取得する。	{tilesetId: 〈タイルセットID〉, x: 〈X位置〉, y: 〈Y位置〉}	✓	✓
	setSubtileInfo	arg1: sx arg2: sy arg3: subtileX arg4: subtileY	シーンに設定されたサブタイルの情報を設定する。		✓	
	getSlopeIdList/getSlopeById					✓
Agtk.slope	name/startX/startY/endX/endY					✓
Agtk.ui					✓	–
Agtk.ui.editSceneId					✓	–
Agtk.ui.editLayerIndex					✓	–
Agtk.sceneInstances					–	✓
	getCurrent	–	カレントシーンを取得		–	✓
Agtk.sceneInstance	.getLayerByIndex	arg1: 〈レイヤーインデックス〉	指定番目のレイヤーを取得する。 番号は0スタート。 cc.Nodeが返される。		–	✓
	getMenuLayerById	arg1: 〈メニューレイヤーID〉	指定IDのメニューレイヤーを取得する。 cc.Nodeが返される。 Agtk.constants.systemLayers.HudLayerIdで（暗黙に作成される）最前面メニューレイヤーを指定できる。			
	getHudLayer	–	カメラの影響を受けないcc.Nodeが返される。			
	sceneId		シーンID		–	✓
	getCurrentCameraTargetPos	–	カレントカメラの注視点を返す。	{x: 〈X座標〉, y: 〈Y座標〉}		✓
	getCurrentCameraDisplayScale		カレントカメラの画面拡大率を返す。	{x: 〈X拡大率〉, y: 〈Y拡大率〉}		✓
Agtk.plugins	isValid	arg1: 〈プラグインオブジェクト〉 arg2: 〈プラグインID〉 arg3: 〈ロケール〉	プラグインオブジェクトが基本的な関数を実装しているか確認する。	システムからのみ使用する想定。	✓	✓
	reload	arg1: 〈プラグインオブジェクト〉 arg2: 〈プラグインID〉 arg3: 〈ロケール〉 arg4: 〈内部データ〉	ロード済みプラグインを破棄し、プラグインを再設定する。プラグインは〈内部データ〉を使って状態を復帰する。	システムからのみ使用する想定。	✓	✓
	unload	arg: 〈プラグインID〉	プラグインをアンロードする。	システムからのみ使用する想定。	✓	✓
	[]	〈インデックス〉	Agtk.plugins[〈インデックス〉]の形式で呼び出して、指定〈インデックス〉のプラグインオブジェクトを返す。	プラグインオブジェクトについては「スクリプトプラグインIF」シートを参照。	✓	✓
	length		ロード済みプラグインの数を返す。		✓	✓
	getIndexById	arg: 〈プラグインID〉	プラグインのインデックスを返す。		✓	✓
	getById	arg: 〈プラグインID〉	プラグインオブジェクトを返す。		✓	✓
Agtk.objects	getIdByName/get					✓
Agtk.object	id/name		readonly			✓
	operatable		bool: 入力デバイスで操作可能			✓
	bullets		「弾の設定」で設定した弾データ			✓
	actions		アクションプログラムのアクションボックスデータ			✓
	animationId		「アニメーションを選択」で設定したアニメーションID			✓
Agtk.object.bullets	getIdList/getIdByName/get		「弾の設定」で設定した弾データ			✓
Agtk.object.bullet	id/name		ID/名前			✓
Agtk.object.actions	getIdList/getIdByName/get		アクションプログラムのアクションボックスデータ			✓
Agtk.object.action	id/name		ID/名前			✓
Agtk.object.viewports	getIdList/getIdByName/get		「視野、照明」で設定した視野、照明データ			✓
Agtk.object.viewport	id/name		ID/名前			✓
Agtk.object.switches	getIdList/getIdByName/get		オブジェクトのスイッチ情報	オブジェクトスイッチのプリセットIDをAgtk.constants.switchesで定義しています。 ※Agtk.objectInstance.switchsのプリセットIDの定義と同内容。		✓
Agtk.object.switch	id/name		ID/名前			✓
Agtk.object.variables	getIdList/getIdByName/get		オブジェクトの変数情報	オブジェクトスイッチのプリセットIDをAgtk.constants.variablesで定義しています。 ※Agtk.objectInstance.variablesのプリセットIDの定義と同内容。		✓
Agtk.object.variable	id/name		ID/名前			✓
Agtk.animations	getIdList/getIdByName/get		アニメーションデータ			✓
Agtk.animation	id/name		ID/名前			✓
	type		アニメーションタイプ Agtk.constants.animations.Motion/Agtk.constants.animations.Effect/Agtk.constants.animations.Particle			✓
	motions		モーションデータ type != Agtk.constants.animations.Motionのときは未定義。			✓
	getResourceSetIdList		素材セットIDの配列			✓
	getResourceSetNameList		素材セット名の配列※各要素は、getResourceSetIdList()の配列のそれと対応している。			✓
Agtk.animation.motions	getIdList/getIdByName/get		モーションデータ			✓
Agtk.animation.motion	id/name		ID/名前			✓
	directions		表示方向データ			✓
Agtk.animation.motion.directions	getIdList/getIdByName/get		表示方向データ			✓
Agtk.animation.motion.direction	id/name		ID/名前			✓
	tracks		トラックデータ			✓
Agtk.animation.motion.direction.tracks	getIdList/getIdByName/get		トラックデータ			✓
Agtk.animation.motion.direction.track	id/name		ID/名前			✓
	timelineType		トラック種別 Agtk.constants.tracks.TimelineWall/TimelineHit/TimelineAttack/TimelineConnect			✓
Agtk.images	getCount/getIdByIndex/getIdByName/getById				✓	
	getIdList/getIdByName/get					✓
Agtk.image	id/name				✓	✓
	width				✓	
	height				✓	
	filename					✓
Agtk.tilesets	getCount/getIdByIndex/getIdByName				✓	
	getById	arg: 〈タイルセットID〉	Tilesetを返す。		✓	
	getIdList/getIdByName/get					✓
Agtk.tileset	id/name				✓	✓
	imageId		タイルセット画像ID		✓	✓
	getPluginId				✓	
	getPluginParam				✓	
	clearAnimation	arg1: x 2: y arg3: width arg4: height	アニメーションをクリアする。		✓	
	appendAnimation	arg1: x arg2: y arg3: width arg4: height arg5: ax arg6: ay arg7: duration300	アニメーションを追加する。		✓	
	getWallBits	arg1: x arg2: y	タイルの各方向壁判定情報を取得する。 次の定数との論理和で壁判定が設定されているかを判定できる。 Agtk.constants.tile.WallTopBit Agtk.constants.tile.WallLeftBit Agtk.constants.tile.WallRightBit Agtk.constants.tile.WallBottomBit			✓

システム提供スクリプト API [2]

名前空間	名前	パラメータ	説明	備考	エディター実装	プレイヤー実装
Agtk.switches	プロジェクト共通スイッチを参照					
	プリセット参照用ID定義					
	InitId	初期化			✓	
	SaveFileId	ファイルをセーブ			✓	
	LoadFileId	ファイルをロード			✓	
	CopyFileId	ファイルをコピー			✓	
	DeleteFileId	ファイルを削除			✓	
	InitialCameraId	初期カメラ用			✓	
	LoadingSceneId	ロード画面表示			✓	
	QuitTheGameId	ゲームを終了			✓	
	メソッド					
	get	arg1: <int: スイッチID>	Agtk.switchオブジェクトを返す。		✓	
	getIdByName	arg1: <string: スイッチ名> ret: <int: スイッチID>	名前でスイッチIDを取得する。 見つからなければ-1を返す。 ※プリセットの変数名は言語依存となるため使用を避けるのが望ましい。		✓	
Agtk.switch	getValue	ret: <bool: 値>	スイッチの値を取得する。		✓	
	setValue	arg1: <bool: 設定値>	スイッチに値を設定する。		✓	
Agtk.variables	プロジェクト共通変数を参照					
	プリセット参照用ID定義					
	PlayerCountId	プレイ人数				✓
	_1PObjectId	1Pオブジェクト				✓
	_2PObjectId	2Pオブジェクト				✓
	_3PObjectId	3Pオブジェクト				✓
	_4PObjectId	4Pオブジェクト				✓
	_1PInstanceId	1Pインスタンス				✓
	_2PInstanceId	2Pインスタンス				✓
	_3PInstanceId	3Pインスタンス				✓
	_4PInstanceId	4Pインスタンス				✓
	_1PControllerId	1Pコントローラー ID				✓
	_2PControllerId	2Pコントローラー ID				✓
	_3PControllerId	3Pコントローラー ID				✓
	_4PControllerId	4Pコントローラー ID				✓
	PortalMoveStartTimeId	ポータルの移動開始時間				✓
	FileSlotId	ファイルスロット				✓
	CopyDestinationFileSlotId	コピー先ファイルスロット				✓
	MouseXId	マウス座標X				
	MouseYId	マウス座標Y				
	メソッド					
	get	arg1: <int: 変数ID>	Agtk.variableオブジェクトを返す。			✓
	getIdByName	arg1: <string: 変数名> ret: <int: 変数ID>	名前で変数IDを取得する。 見つからなければ-1を返す。 ※プリセットの変数名は言語依存となるため使用を避けるのが望ましい。			✓
Agtk.variable	getValue	ret: <double: 値>	変数の値を取得する。			✓
	setValue	arg1: <double: 設定値>	変数に値を設定する。			✓
Agtk.actionCommands	objectCreate	arg1: <int: オブジェクトID> arg2: <int : 生 成 位 置 X> arg3 : <int : 生成位置Y> arg4 : <int : レイヤー番号> ret: <int: インスタンスID>	<レイヤー番号>: 手前から0, 1, …			✓
	objectDestroy	arg1: <int: インスタンスID>				✓
Agtk.portals	getIdList/getIdByName/get					✓
Agtk.portal	id/name/a/b		ポータルA/B			✓
Agtk.portal.a/b	sceneId		シーンID			✓
	x/y/width/height		X/Y/幅/高さ			✓
	movable		もう一方に移動可能か			✓
Agtk.controllers	MaxControllerId	16				✓
	getOperationKeyPressed	arg1: <int: コントローラID> arg2: <操作キーID> ret: <bool: 押されているか>	コントローラID: 0～16 ※0はキーボード・マウス 操作キーiD: 1～	<操作キーID>: Agtk.constants.controllers.OperationKeyA, …, Agtk.constants.controllers.OperationKeyCancel		✓
	getOperationKeyValue	arg1: <int: コントローラID> arg2: <操作キーID> ret: <double: 値>	コントローラID: 0～16 ※0はキーボード・マウス 操作キーiD: 1～ 返却値は-1～1の範囲。	<操作キーID>: Agtk.constants.controllers.OperationKeyA, …, Agtk.constants.controllers.OperationKeyCancel		
	getKeyValue	arg1: <int: コントローラID> arg2: <キーコード> ret: <double: 値>	コントローラID: 0～16 ※0はキーボード・マウス キーコード: 0～ キーコード、返却値はデバイス依存。	コントローラID=0のときの、キーコード: Agtk.constants.controllers.ReservedKeyCodePc_W～ Agtk.constants.controllers.ReservedKeyCodePc_MousePointer		✓
	isConnected	arg1: <int: コントローラID> ret: <bool: 接続されているか>	コントローラiD: 0～16 ※0はキーボード・マウス			✓
Agtk.objectInstances	get	arg1: <int: インスタンスID>	Agtk.objectInstanceオブジェクトを返す。			✓
	getIdByName	arg1: <int: オ ブ ジ ェ ク ト ID> arg2: <string: インスタンス名> ret: <int: インスタンスID>	名前でインスタンスIDを取得する。 見つからなければ-1を返す。 ※動的に生成したオブジェクトインスタンスの場合はオブジェクトの名前で一致チェックします。			✓
Agtk.objectInstance	Id		readonly			✓
	objectId		readonly int: <オブジェクトID>			✓
	layerId		オブジェクトが配置されているレイヤーID			✓
	layerIndex		オブジェクトが配置されているレイヤーのインデックス			✓
	getAttackerInstanceIdList	ret: <array: インスタンスID配列>	このオブジェクトインスタンスを攻撃したオブジェクトインスタンスのインスタンスIDリストを返す。 ※1フレーム前の情報が得られます。			
Agtk.objectInstance.switches	オブジェクトスイッチを参照					✓
	プリセット参照用ID定義					
	InvincibleId	無敵	属性管理用の変数です。1～8はプリセット用[火、水、地、風、雷、氷、光、闇]として使用されています。			✓
	FreeMoveId	自由移動				✓
	LockTargetId	ロック対象				✓
	PortalTouchedId	ポータルに触れた				✓
	CriticalDamagedId	クリティカルを受けた				✓
	DisabledId	無効				✓
	SlipOnSlopeId	坂ですべる				✓
	AffectOtherObjectsId	他の物理演算オブジェクトに影響を与える				✓
	AffectedByOtherObjectsId	他の物理演算オブジェクトから影響を受ける				✓
	FollowConnectedPhysicsId	接続されている物理オブジェクトの動作を優先				✓
	メソッド					
	get	arg1: <int: スイッチID>	Agtk.objectInstance.switchオブジェクトを返す。			✓
	getIdByName	arg1: <string: スイッチ名> ret: <int: スイッチID>	名前でスイッチIDを取得する。 見つからなければ-1を返す。 ※プリセットの変数名は言語依存となるため使用を避けるのが望ましい。			✓
Agtk.objectInstance.switch	getValue	ret: <bool: 値>	スイッチの値を取得する。			✓
	setValue	arg1: <bool: 設定値>	スイッチに値を設定する。			✓

システム提供スクリプト API [3]

名前空間	名前	パラメータ	説明	備考	エディター実装	プレイヤー実装
Agtk.objectInstance.variables	オブジェクト変数を参照					✓
	プリセット参照用ID定義					
	ObjectIDId	オブジェクトID				✓
	HPId	体力				✓
	MaxHPId	最大体力				✓
	MinimumAttackId	最小攻撃力				✓
	MaximumAttackId	最大攻撃力				✓
	DamageRatioId	被ダメージ率				✓
	AttackAttributeId	攻撃属性				✓
	AreaAttributeId	エリア判定				✓
	XId	X座標位置				✓
	YId	Y座標位置				✓
	VelocityXId	X方向速度				✓
	VelocityYId	Y方向速度				✓
	PlayerIDId	プレイヤー判定				✓
	DamageValueId	被ダメージ数値				✓
	CriticalRatioId	クリティカル倍率（%）				✓
	CriticalIncidenceId	クリティカル発生率(%)				✓
	InvincibleDurationId	無敵時間				✓
	FixedAnimationFrameId	アニメの表示を指定フレームで固定				✓
	InstanceIDId	インスタンスID				✓
	InstanceCountId	シーンに配置された同インスタンス数				✓
	SingleInstanceIDId	単体インスタンスID				✓
	ControllerIDId	コントローラーID				✓
	HorizontalMoveId	左右の移動量				✓
	VerticalMoveId	上下の移動量				✓
	HorizontalAccelId	左右の加速量				✓
	VerticalAccelId	上下の加速量				✓
	HorizontalMaxMoveId	左右の最大移動量				✓
	VerticalMaxMoveId	上下の最大移動量				✓
	HorizontalDecelId	左右の減速量				✓
	VerticalDecelId	上下の減速量				✓
	DurationToTakeOverAccelerationMoveSpeedId	加速移動切り替え時間				✓
	ScalingXId	スケールX(%)				✓
	ScalingYId	スケールY(%)				✓
	DispPriorityId	表示優先度				✓
	InitialJumpSpeedId	ジャンプの初速	Agtk.objectInstance.variableオブジェクトを返す。			✓
	メソッド					
	get	arg1: <int: 変数ID>	Agtk.objectInstance.variableオブジェクトを返す。			✓
	getIdByName	arg1: <string: 変数名> ret: <int: 変数ID>	名前で変数IDを取得する。 見つからなければ-1を返す。 ※プリセットの変数名は言語依存となるため使用を避けるのが望ましい。			✓
Agtk.objectInstance.variable						✓
	getValue	ret: <double: 値>	変数の値を取得する。			✓
	setValue	arg1: <double: 設定値>	変数に値を設定する。			✓
Agtk.fonts	getIdList/getIdByName/get					✓
Agtk.font	id/name					✓
	imageFontFlag	bool	真の時、画像をフォントとして使用			✓
	imageId	int	画像ID			✓
	fontName	string	フォントファイル名			✓
	ttfName	string	TTF名			✓
	fontSize	int	フォントサイズ			✓
	antialiasDisabled	bool	アンチエイリアスを無効			✓
	aliasThreshold	int	閾値			✓
	fixedWidth	bool	真の時、固定幅			✓
	hankakuWidth	int	半角幅			✓
	zenkakuWidth	int	全角幅			✓
	letterLayout	string	文字配置			✓
Agtk.texts	getIdList/getIdByName/get					✓
Agtk.text	id/name					✓
	fontId	int	フォントID			✓
	letterSpacing	int	文字間隔			✓
	lineSpacing	int	行間隔			✓
	localeText	object	localeをキー、テキストを値とするプロパティーが格納されている。			✓
	getText	arg1: <string: locale> ret: <string: テキスト>	指定localeのテキストのタグ(¥O, ¥T, ¥V)を展開したテキストを返す。			
Agtk.movies	getIdList/getIdByName/get					✓
Agtk.movie	id/name					✓
	filename					✓
Agtk.bgms	getIdList/getIdByName/get					✓
Agtk.bgm	id/name					✓
	filename					✓
Agtk.ses	getIdList/getIdByName/get					✓
Agtk.se	id/name					✓
	filename					✓
Agtk.voices	getIdList/getIdByName/get					✓
Agtk.voice	id/name					✓
	filename					✓
Agtk.databases	get	arg: <int: データベースID> ret: <object: Agtk.database>	データベースをIDから取得 見つからなければnull			✓
	getIdByName	arg: <string: データベース名> ret: <int: データベースID>	データベースIDをデータベース名から取得 見つからなければ-1			✓
	getIdList	ret: <array: データベースIDの配列>	データベースIDの配列を取得 見つからなければnull			✓
Agtk.database	getColumnList	ret: <array: カラム名の配列>	カラム名の配列を取得 見つからなければnull			✓
	getColumnByIndex	arg: <int: カラム番号> ret: <array: カラム値の配列>	カラム値の配列を番号から取得 見つからなければnull			✓
	getColumnByName	arg: <string: カラム名> ret: <array: カラム値の配列>	カラム値の配列をカラム名から取得 見つからなければnull			✓
	getRowList	ret: <array: レコード名の配列>	レコード名の配列を取得 見つからなければnull			✓
	getRowByIndex	arg: <int: レコード番号> ret: <array: レコード値の配列>	レコード値の配列を番号から取得 見つからなければnull			✓
	getRowByName	arg: <string: レコード名> ret: <array: レコード値の配列>	レコード値の配列をレコード名から取得 見つからなければnull			✓
	getFieldByIndex	args: <int: カラム番号, int: レコード番号> ret: <string: フィールド値の文字列>	フィールド値を番号から取得 見つからなければnull			✓

アクションコマンド [1]

「スクリプトを実行」で指定したスクリプトから、「その他の実行アクション」を呼び出す際は注意が必要です。

※参照：「アクションボックスの「その他の実行アクション」の処理の流れ」

特に、 次の２つを呼び出す際は、 スクリプトを使わない「その他の実行アクション」と動作を合わせるために、 次のような処理を組む必要があります。

- 単体で、 execCommandObjectPushPull を呼び出す場合、「スクリプトを実行」の戻り値として、 Agtk.constants.actionCommands.commandBehavior.CommandBehaviorLoop を指定します。

  （「oneTimeEffect: true」を指定しない場合。)

- 単体で、 execCommandActionExec を呼び出す場合、「スクリプトを実行」の戻り値として、 execCommandActionExec の戻り値を指定します。

※単体以外で使用する場合は、 深い理解のうえでご使用ください。

名前空間	名前	パラメータ	説明	備考(指定可能プロパティー)	プレイヤー実装
Agtk.objectInstance	execCommandTemplateMove	arg1: <パラメータオブジェクト > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「テンプレート移動の設定」を実行する。	moveType: int *Agtk.constants.actionCommands.templateMove.MoveHorizontal等。 *Agtk.constants.actionCommands.templateMoveの定義のうち、MoveNearPlayer, MoveApartNearPlayerを廃止予定。代わりに、MoveNearObject, MoveApartNearObjectを使用すること。 horizontalMoveStartRight: bool horizontalMoveDuration300: int horizontalInfinite: bool verticalMoveStartDown: bool verticalMoveDuration300: int verticalInfinite: bool randomMoveDuration300: int randomMoveStop300: int nearPlayerGroup: int nearObjectLockedObjectPrior: bool nearPlayerLockedPlayerPrior: bool *廃止予定 apartNearObjectGroup: int apartNearPlayerLockedPlayerPrior: bool apartNearObjectLockedObjectPrior: bool *廃止予定 fallFromStep: bool * エディターと逆 ignoreOtherObjectWallArea: bool ignoreWall: bool * duration300には、300が1秒とした数値を指定する。(他同様。)	✓
	execCommandObjectLock	arg1: <パラメータオブジェクト > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「オブジェクトをロック」を実行する。	lockTouchedObject: bool lockViewportLightObject: bool lockObjectOnScreen: bool objectType: int *Agtk.constants.actionCommands.objectLock.SetByObjectGroup等。 objectGroup: int objectId: int useType: int *Agtk.constants.actionCommands.objectLock.UseSwitch等。 switchId: int switchCondition: int variableId: int compareVariableOperator: int compareValueType: int *Agtk.constants.actionCommands.objectLock.CompareValueConstant等。 compareValue: double compareVariableObjectId: int compareVariableQualifierId: int compareVariableId: int * qualifierIdには、オブジェクトインスタンスIDか次のいずれかを指定する: Agtk.constants.qualifier.QualifierSingle Agtk.constants.qualifier.QualifierWhole * objectGroupIdにはオブジェクトグループ管理で設定したオブジェクトグループのインデックスを指定する。システムで用意されているオブジェクトグループはAgtk.constants.objectGroup.ObjectGroupPlayer, Agtk.constants.objectGroup.ObjectGroupEnemyで指定可能。	✓
	execCommandObjectCreate	arg1: <パラメータオブジェクト > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「オブジェクトを生成」を実行する。	objectId: int actionId: int createPosition: int useConnect: bool connectId: int adjustX: int adjustY: int probability: double childObject: bool useRotation: bool lowerPriority: bool gridMagnet: bool	✓
	execCommandObjectChange	arg1: <パラメータオブジェクト > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「オブジェクトを変更」を実行する。	objectId: int actionId: int takeOverVariables: bool takeOverSwitches: bool takeOverParentChild: bool createPosition: int useConnect: bool connectId: int adjustX: int adjustY: int probability: double	✓
	execCommandObjectMove	arg1: <パラメータオブジェクト > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「オブジェクトを移動させる」を実行する。	moveType: int *Agtk.constants.actionCommands.objectMove.MoveWithDirection等。 angle: double moveDistance: int posX, posY: int moveInDisplayCoordinates: bool followCameraMoving: bool centerObjectId: int centerQualifierId: int centerAdjustX: int centerAdjustY: int connectId: int useObjectParameter: bool changeMoveSpeed: double moveDuration300: int targettingType: int *Agtk.constants.actionCommands.objectMove.TargettingByGroup等。 targetObjectGroup: int targetObjectId: int targetQualifierId: int excludeObjectGroupBit: int fitDispDirToMoveDir: bool dispWhileMoving: bool　＊エディターと逆 stopActionWhileMoving: bool stopAnimWhileMoving: bool finalGridMagnet: bool * excludeObjectGroupBitには、対象外としたいオブジェクトグループに対して、そのオブジェクトグループのインデックス分だけ1を左シフトした値の論理和を指定する。例えば、プレイヤーグループとエネミーグループの両方を対象外としたい場合は、(1 << Agtk.constants.objectGroup.ObjectGroupPlayer) | (1 << Agtk.constants.objectGroup.ObjectGroupEnemy)を指定する。	✓

アクションコマンド [2]

名前空間	名前	パラメータ	説明	備考(指定可能プロパティー)	プレイヤー実装
Agtk.objectInstance	execCommandObjectPushPull	arg1: <パラメータオブジェクト> > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorLoop	「オブジェクトを押す・引く」を実行する。	effectRangeBaseConnect: bool effectRangeBaseConnectId: int rectangle: bool rectangleDistance: int rectangleHeight: int circleDistance: int arcAngle: int effectDirectionType: int *Agtk.constants.actionCommands.objectPushPull.EffectDirectionAngle等。 effectDirection: double directionType: int *Agtk.constants.actionCommands.objectPushPull.DirectionAngle等。 angle: double connectId: int effectValue: int distanceEffect: bool nearValue: double farValue: double oneTimeEffect: bool targettingType: int *Agtk.constants.actionCommands.objectPushPull.TargettingByGroup等。 targetObjectGroup: int targetObjectId: int targetQualifierId: int excludeObjectGroupBit: int	✓
	execCommandLayerMove	arg1: <パラメータオブジェクト> > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「レイヤーを移動」を実行する。	layerIndex: int	✓
	execCommandAttackSetting	arg1: <パラメータオブジェクト> > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「攻撃の設定」を実行する。	attackChange: double hitObjectFlag: bool objectGroupBit: int hitTileFlag: bool tileGroupBit: int attributeType: int *Agtk.constants.actionCommands.attackSetting.AttributeNone等。 attributePresetId: int *Agtk.constants.attackAttributes.Fire等。 attributeValue: int * objectGroupBitには、対象としたいオブジェクトグループに対して、そのオブジェクトグループのインデックス分だけ1を左シフトした値の論理和を指定する。例えば、プレイヤーグループとエネミーグループの両方を対象としたい場合は、(1 << Agtk.constants.objectGroup.ObjectGroupPlayer) | (1 << Agtk.constants.objectGroup.ObjectGroupEnemy)を指定する。	✓
	execCommandBulletFire	arg1: <パラメータオブジェクト> > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「弾を発射」を実行する。	bulletId: int connectId: int	✓
	execCommandDisappear	arg1: <パラメータオブジェクト> > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「オブジェクトを消滅する」を実行する。	パラメーター無し ※arg1は省略可能。	✓
	execCommandDisappearObjectRecover	arg1: <パラメータオブジェクト> > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「消滅状態のオブジェクトを復活させる」を実行する。	objectId: int	✓
	execCommandDisable	arg1: <パラメータオブジェクト> > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「オブジェクトを無効にする」を実行する。	パラメーター無し ※arg1は省略可能。	✓
	execCommandDisableObjectEnable	arg1: <パラメータオブジェクト> > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「無効状態のオブジェクトを有効にする」を実行する。	objectId: int	✓
	execCommandObjectFilterEffect	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「オブジェクトにフィルター効果を設定」を実行する。	effectType: int *Agtk.constants.filterEffects.EffectNoise等。 noise: int mosaic: int monochrome: int sepia: int negaPosiReverse: int defocus: int chromaticAberration: int darkness: int transparency: int imageId: int imageTransparency: int fillA: int fillR: int fillG: int fillB: int duration300: int	✓
	execCommandObjectFilterEffectRemove	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「オブジェクトのフィルター効果を削除」を実行する。	removeBit: int duration300: int ※removeBitには、Agtk.constants.filterEffectsから指定する値(value)に対して、(1 << value)としたものを指定する。OR演算で複数指定可能。	✓
	execCommandSceneEffect	arg1: <パラメータオブジェクト> > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「シーンに画面効果を設定」を実行する。	layerIndex: int filterEffect: <FilterEffectパラメーターオブジェクト> FilterEffectパラメーターオブジェクトのプロパティー: effectType: int *Agtk.constants.filterEffects.EffectNoise等。 noise: int mosaic: int monochrome: int sepia: int negaPosiReverse: int defocus: int chromaticAberration: int darkness: int imageId: int imageTransparency: int imagePlacement: int *Agtk.constants.filterEffects.PlacementCenter等。 fillA: int fillR: int fillG: int fillB: int duration300: int	✓

アクションコマンド [3]

名前空間	名前	パラメータ	説明	備考(指定可能プロパティー)	プレイヤー実装
Agtk.objectInstance	execCommandSceneEffectRemove	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「シーンの画面効果を削除」を実行する。	layerIndex: int removeBit: int duration300: int	✓
	execCommandSceneGravityChange	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「シーンの重力効果を変更する」を実行する。	gravity: double direction: double duration300: int durationUnlimited: bool	✓
	execCommandSceneRotateFlip	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「シーンを回転・ 反転」を実行する。	type: int rotationFlag: bool rotation: double absoluteRotation: bool flipX: bool flipY: bool duration300: int	✓
	execCommandCameraAreaChange	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「カメラの表示領域を変更する」を実行する。	xRate: double yRate: double duration300: int ※xRate, yRateは1.0が等倍	✓
	execCommandSoundPlay	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「音の再生」を実行する。	soundType: int *Agtk.constants.actionCommands.soundPlay.SoundSe等。 seId: int voiceId: int bgmId: int loop: bool fadein: bool specifyAudioPosition: bool audioPositionVariableObjectId: int audioPositionVariableQualifierId: int audioPositionVariableId: int duration300: int volume: int pan: int pitch: int	✓
	execCommandMessageShow	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「テキストを表示」を実行する。	textFlag: bool textId: int variableObjectId: int variableQualifierId: int variableId: int fontId: int digitFlag: bool digits: int zeroPadding: bool comma: bool withoutDecimalPlaces: bool duration300: int durationUnlimited: bool colorA: int colorR: int colorG: int colorB: int windowWidth: int windowHeight: int windowType: int *Agtk.constants.actionCommands.messageShow.WindowNone等。 templateId: int *Agtk.constants.messageShow.TemplateWhiteFrame等。 imageId: int windowTransparency: int positionType: int *Agtk.constants.actionCommands.messageShow.PositionCenter等。 useConnect: bool connectId: int adjustX: int adjustY: int topBottomMargin: int leftRightMargin: int horzAlign: int *Agtk.constants.actionCommands.messageShow.HorzAlignLeft等。 vertAlign: int *Agtk.constants.actionCommands.messageShow.VertAlignTop等。 actionChangeHide: bool closeByKey: bool keyId: int objectStop: bool gameStop: bool priorityMostFront: bool(削除予定) priority: bool priorityType: int ※priorityTypeには、 Agtk.constants.actionCommands.priorityType.PriorityBackground～ PriorityMostFrontWithMenuを指定する。	✓
	execCommandScrollMessageShow	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「テキストをスクロール表示」を実行する。	textId: int scrollSpeed: double scrollUp: bool colorA: int colorR: int colorG: int colorB: int backgroundWidth: int backgroundHeight: int backgroundType: int *Agtk.constants.actionCommands.scrollMessageShow.BackgroundNone等。 templateId: int *Agtk.constants.actionCommands.scrollMessageShow.TemplateBlack等。 imageId: int backgroundTransparency: int topBottomMargin: int leftRightMargin: int horzAlign: int *Agtk.constants.actionCommands.scrollMessageShow.HorzAlignLeft等。 positionType: int *Agtk.constants.actionCommands.scrollMessageShow.PositionCenter等。 useConnect: bool connectId: int adjustX: int adjustY: int actionChangeHide: bool speedUpByKey: bool keyId: int objectStop: bool gameStop: bool priorityMostFront: bool(削除予定) priority: bool priorityType: int ※priorityTypeには、 Agtk.constants.actionCommands.priorityType.PriorityBackground～ PriorityMostFrontWithMenuを指定する。	✓
	execCommandEffectShow	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「エフェクトを表示」を実行する。	effectId: int positionType: int *Agtk.constants.actionCommands.effectShow.PositionCenter等。 useConnect: bool connectId: int adjustX, adjustY: int duration300: int durationUnlimited: bool	✓

アクションコマンド [4]

名前空間	名前	パラメータ	説明	備考(指定可能プロパティー)	プレイヤー実装
Agtk.objectInstance	execCommandEffectRemove	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「エフェクトを非表示」を実行する。	effectId: int targettingType: int *Agtk.constants.actionCommands.effectRemove.TargettingByGroup等。 targetObjectGroup: int targetObjectId: int	✓
	execCommandParticleShow	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「パーティクルを表示」を実行する。	particleId: int positionType: int *Agtk.constants.actionCommands.particleShow.PositionCenter等。 useConnect: bool connectId: int adjustX, adjustY: int duration300: int durationUnlimited: bool	✓
	execCommandParticleRemove	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「パーティクルを非表示」を実行する。	particleId: int targettingType: int *Agtk.constants.actionCommands.particleRemove.TargettingByGroup等。 targetObjectGroup: int targetObjectId: int	✓
	execCommandMovieShow	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「動画を再生」を実行する。	movieId: int loop: bool volume: int defaultSize: bool width, height: int positionType: int *Agtk.constants.actionCommands.movieShow.PositionCenter等。 useConnect: bool connectId: int vertAlign: int *Agtk.constants.actionCommands.movieShow.VertAlignCenter等。 horzAlign: int *Agtk.constants.actionCommands.movieShow.HorzAlignCenter等。 adjustX, adjustY: int hideOnObjectActionChange: bool stopObject: bool stopGame: bool fillBlack: bool priority: bool priorityMostFront: bool(削除予定) priorityType: int ※priorityTypeには、 Agtk.constants.actionCommands.priorityType.PriorityBackground～ PriorityMostFrontWithMenuを指定する。	✓
	execCommandImageShow	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「画像を表示」を実行する。	imageId: int duration300: int durationUnlimited: bool defaultSize: bool width: int height: int positionType: int *Agtk.constants.actionCommands.imageShow.PositionCenter等。 useConnect: bool connectId: int vertAlign: int *Agtk.constants.actionCommands.imageShow.VertAlignCenter等。 horzAlign: int *Agtk.constants.actionCommands.imageShow.HorzAlignCenter等。 adjustX: int adjustY: int hideOnObjectActionChange: bool closeByOk: bool stopObject: bool stopGame: bool priority: bool priorityMostFront: bool(削除予定) priorityType: int ※priorityTypeには、 Agtk.constants.actionCommands.priorityType.PriorityBackground～ PriorityMostFrontWithMenuを指定する。	✓
	execCommandSwitchVariableChange	arg1: <パラメータオブジェクト > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「スイッチ・変数を変更」を実行する。	swtch: bool switchObjectId: int switchQualifierId: int switchId: int switchValue: int *Agtk.constants.assignments.SwitchAssignOn等。 variableObjectId: int variableQualifierId: int variableId: int variableAssignOperator: int *Agtk.constants.assignments.VariableAssignOperatorSet等。 variableAssignValueType: int *Agtk.constants.assignments.VariableAssignValue等。 assignValue: double assignVariableObjectId: int assignVariableQualifierId: int assignVariableId: int randomMin: int randomMax: int assignScript: string 参照: Agtk.constants.assignments * variableAssignValueTypeに、 Agtk.constants.assignments.VariableAssignScriptを指定してはいけない。	✓
	execCommandSwitchVariableReset	arg1: <パラメータオブジェクト > ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「スイッチ・変数を初期値に戻す」を実行する。	配列で指定する。 各要素オブジェクトで次のプロパティーが設定可能。 swtch: bool objectId: int itemId: int * objectId == 0の場合、プロジェクト共通を意味する。	✓
	execCommandGameSpeedChange	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「ゲームスピードを変更する」を実行する。	gameSpeed: double duration300: int durationUnlimited: bool targetObject: bool targetEffect: bool targetTile: bool targetMenu: bool targettingType: int *Agtk.constants.actionCommands.gameSpeedChange.TargettingByGroup等。 targetObjectGroup: int targetObjectId: int targetQualifierId: int excludeObjectGroupBit: int	✓

プラグインの仕様に関する詳細資料［10］

アクションコマンド [5]

名前空間	名前	パラメータ	説明	備考(指定可能プロパティー)	プレイヤー実装
Agtk.objectInstance	execCommandMenuShow	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「メニュー画面を表示」を実行する。	layerId: int useEffect: bool effectType: int *Agtk.constants.actionCommands.menuShow.None等。 fadein: bool duration300: int	✓
	execCommandMenuHide	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「メニュー画面を非表示」を実行する。	hideExceptInitial: bool useEffect: bool effectType: int *Agtk.constants.actionCommands.menuHide.None等。 fadeout: bool duration300: int disableObjects: bool	✓
	execCommandDisplayDirectionMove	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「表示方向と同じ方へ移動」を実行する。	moveDistance: int reverse: bool distanceOverride: bool	✓
	execCommandFileLoad	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「ファイルをロード」を実行する。	projectCommonVariables: bool projectCommonSwitches: bool sceneAtTimeOfSave: bool objectsStatesInSceneAtTimeOfSave: bool effectType: int *Agtk.constants.actionCommands.fileLoad.None等。 duration300: int ※effectTypeには、Agtk.constants.actionCommands.fileLoadの定数を指定する。	
	execCommandSoundPositionRemember	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「音の再生位置を保存」を実行する。	soundType: int *Agtk.constants.actionCommands.soundPositionRemember.SoundSe等 variableObjectId: int variableQualifierId: int variableId: int	
	execCommandObjectUnlock	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「ロックを解除」を実行する。	objectType: int *Agtk.constants.actionCommands.objectUnlock.SetByObjectGroup等。 objectGroup: int objectId: int * objectGroupにはオブジェクトグループ管理で設定したオブジェクトグループのインデックスを指定する。システムで用意されているオブジェクトグループはAgtk.constants.objectGroup.ObjectGroupPlayer, Agtk.constants.objectGroup.ObjectGroupEnemyで指定可能。	
	execCommandResourceSetChange	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「アニメーションの素材セットを変更」を実行する。	objectId: int qualifierId: int resourceSetId: int ※resourceSetIdに指定可能な値は、Agtk.animation.getResourceSetIdList()で取得できる。	
	execCommandTimer	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「タイマー機能」を実行する。	start: bool timerVariableObjectId: int timerVariableQualifierId: int timerVariableId: int countUp: bool secondType: int *Agtk.constants.actionCommands.timer.SecondByValue等。 second300: int secondVariableObjectId: int secondVariableQualifierId: int secondVariableId: int	✓
	execCommandSceneTerminate	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「このシーンを終了」を実行する。	パラメーター無し ※arg1は省略可能。	✓
	execCommandDirectionMove	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「移動方向を指定して移動」を実行する。	direction: double directionId: int *Agtk.constants.actionCommands.directionMove: 移動方向を指定して移動。(Match Move Direction) moveDistance: int moveDistanceEnabled: bool	✓
	execCommandForthBackMoveTurn	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「前後移動と旋回」を実行する。	moveType: int *Agtk.constants.actionCommands.forthBackMoveTurn.MoveNone等。 turnType: int *Agtk.constants.actionCommands.forthBackMoveTurn.TurnNone等。 directionId: int	✓
	execCommandActionExec	arg1: <パラメータオブジェクト> ret: <Agtk.constants.actionCommands.commandBehavior .CommandBehaviorNext or Agtk.constants.actionCommands.commandBehavior.CommandBehaviorBreak>	「オブジェクトのアクションを実行」を実行する。 「自身のオブジェクト」のアクションを実行した場合に Agtk.constants.actionCommands.commandBehavior. CommandBehaviorBreak が返される。	objectId: int qualifierId: int actionId: int	✓
	execCommandSceneShake	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「シーンを揺らす」を実行する。	duration300: int fadeIn: bool fadeOut: bool interval300: int height: int heightDispersion: int width: int widthDispersion: int	✓
	execCommandLayerHide	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「レイヤーの表示をOFF」を実行する。	layerIndex: int exceptFlag: bool	✓
	execCommandLayerShow	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「レイヤーの表示をON」を実行する。	layerIndex: int exceptFlag: bool	✓
	execCommandLayerDisable	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「レイヤーの動作をOFF」を実行する。	layerIndex: int exceptFlag: bool	✓
	execCommandLayerEnable	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「レイヤーの動作をON」を実行する。	layerIndex: int exceptFlag: bool	✓
	execCommandDatabaseReflect	arg1: <パラメータオブジェクト> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	「データベースを反映」を実行する。	columnIndex: int columnIndexFromName: bool columnNumberFromValue: bool columnVariableId: int columnVariableObjectId: int columnVariableQualifierId: int databaseId: int fromObject: bool fromRow: bool objectId: int reflectObjectId: int reflectQualifierId: int reflectVariableAssignOperator: int *Agtk.constants.assignments.VariableAssignOperatorSet 等。 reflectVariableId: int rowIndexFromName: bool rowNumberFromValue: bool rowIndex: int rowVariableObjectId: int rowVariableQualifierId: int rowVariableId: int	✓

アクションボックスの「その他の実行アクション」の処理の流れ

※「その他の実行アクション」を「コマンド」と呼称します。

アクションボックスが切り替わると、 切り替わり先のアクションボックスのコマンドリストが処理されます。

基本的には、 同一フレーム内でリスト内のすべてのコマンドが処理されますが、 コマンドの戻り値により、 コマンドの処理の流れを変更できます。

a.次のコマンドに処理を進める。 （戻り値に Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext を指定。）

b.次のコマンドに処理を進めるが、 次のフレームにこのコマンドを再度実行する。 （戻り値に Agtk.constants.actionCommands.commandBehavior.CommandBehaviorLoop を指定。）

※再実行の際は c, d の戻り値は意味が無くなることに注意。

c.次のコマンドの処理を遅延し、 次のフレームにこのコマンドを再度実行する。 （戻り値に Agtk.constants.actionCommands.commandBehavior.CommandBehaviorBlock を指定。）

d.次のコマンド以降の処理を行わない。 （戻り値に Agtk.constants.actionCommands.commandBehavior.CommandBehaviorBreak を指定。）

大部分のコマンドは a タイプです。

b タイプのコマンドは「オブジェクトを押す・　　引く」です。 アクションボックスが切り替わるまで「押す・ 引く」の動作が継続します。 （「効果は対象となった時一度だけ受ける」のチェックが無い場合。）

c タイプのコマンドは「ウェイトを入れる」です。 ウェイトが終わってから次のコマンドが処理されます。

d タイプのコマンドは「オブジェクトのアクションを実行」で「自身のオブジェクト」を指定した場合です。 以降のコマンドは処理されません。

「スクリプトを実行」は、 コマンドリストの処理の中で呼び出されるため、「スクリプトを実行」の戻り値により、 処理の流れをコントロールできます。

リンク条件判定[1]

名前空間	名前	パラメータ	説明	備考(指定可能プロパティー)	プレイヤー実装
Agtk.objectInstance	isWallTouched	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「タイルの壁判定に接触」を判定する	wallBit: int useTileGroup: bool tileGroup: int 次から判定したい接触する方向をOR演算した値を指定する。 Agtk.constants.tile.WallTopBit Agtk.constants.tile.WallLeftBit Agtk.constants.tile.WallRightBit Agtk.constants.tile.WallBottomBit ＊ tileGroupにはタイルグループ管理で設定したタイルグループのインデックスを指定する。システムで用意されているタイルグループは Agtk.constants.tileGroup.TileGroupDefaultで指定可能。	✓
	isWallAhead	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「1タイル進んだ先で、タイルの壁判定と接触」を判定する	wallBit: int useTileGroup: bool tileGroup: int	✓
	isObjectWallTouched	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「他オブジェクトの壁判定に接触」を判定する	wallBit: int objectType: int *Agtk.constants.linkCondition.objectWallTouched.SetByObjectGroup等。 objectGroup: int objectId: int ※objectTypeには、 Agtk.constants.actionCommands.ObjectByTypeか Agtk.constants.actionCommands.ObjectByIdを指定する。 ※objectTypeByTypeには、 Agtk.constants.objectType.ObjectTypeAll/ObjectTypePlayer/ObjectTypeEnemyのいずれかを指定する。	✓
	isObjectHit	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「他オブジェクトの当たり判定に接触」を判定する	wallBit: int objectType: int *Agtk.constants.linkCondition.objectHit.SetByObjectGroup等。 objectGroup: int objectId: int	✓
	isAttackAreaTouchd	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「攻撃判定に当たる」を判定する	wallBit: int objectType: int *Agtk.constants.linkCondition.attackAreaTouched.SetByObjectGroup等。 objectGroup: int objectId: int attributeType: int *Agtk.constants.linkCondition.attackAreaTouched.AttributeNone等。 attributePresetId: int attributeEqual: bool attributeValue: int	✓
	isAttackAreaNear	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「攻撃判定との距離」を判定する	otherDirections: bool objectDirection: bool directionBit: int distanceType: int *Agtk.constants.linkCondition.attackAreaNear.DistanceNone等。 distance: int objectType: int *Agtk.constants.linkCondition.attackAreaNear.SetByObjectGroup等。 objectGroup: int objectId: int attributeType: int *Agtk.constants.linkCondition.attackAreaNear.AttributeNone等。 attributePresetId: int *Agtk.constants.attackAttributes.Fire等。 attributeEqual: bool attributeValue: int	✓
	isObjectNear	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「他オブジェクトとの距離」を判定する	otherDirections: bool objectDirection: bool directionBit: int distanceType: int *Agtk.constants.linkCondition.objectNear.DistanceNone等。 distance: int objectType: int *Agtk.constants.linkCondition.objectNear.SetByObjectGroup等。 objectGroup: int objectId: int	✓
	isObjectFacingEachOther	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「他オブジェクトと向かい合っている」を判定する	objectType: int *Agtk.constants.linkCondition.objectFacingEachOther.SetByObjectGroup等。 objectGroup: int objectId: int	✓
	isObjectFacing	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「他オブジェクトの方向を向いている」を判定する	objectType: int *Agtk.constants.linkCondition.objectFacing.SetByObjectGroup等。 objectGroup: int objectId: int	✓
	isObjectFound	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「他オブジェクトを発見した」を判定する	viewportId: int objectType: int *Agtk.constants.linkCondition.objectFound.SetByObjectGroup等。 objectGroup: int objectId: int	✓
	isObjectFacingDirection	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「他オブジェクトが指定方向を向いている」を判定する	otherDirections: bool objectDirection: bool directionBit: int objectType: int *Agtk.constants.linkCondition.objectFacingDirection.SetByObjectGroup等 objectGroup: int objectId: int	✓
	isHpZero	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「体力が0」を判定する	objectId: int	✓
	isCameraOutOfRange	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「カメラの範囲外にでた」を判定する	objectId: int distanceFlag: bool distance: int	✓
	isLocked	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「ロックした/された」を判定する	lockingObjectId: int lockedObjectType: int *Agtk.constants.linkCondition.locked.SetByObjectGroup等。 lockedObjectGroup: int lockedObjectId: int	✓
	isProbability	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「確率を使用する」を判定する	probability: double	✓
	isSwitchVariableChanged	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「スイッチ・変数が変化」を判定する	swtch: bool switchObjectId: int switchQualifierId: int switchId: int switchCondition: int *Agtk.constants.conditions.SwitchConditionOn等。 variableObjectId: int variableQualifierId: int variableId: int compareVariableOperator: int *Agtk.constants.conditions.OperatorLess等。 compareValueType: int *Agtk.constants.conditions.CompareValue等。 compareValue: double compareVariableObjectId: int compareVariableQualifierId: int compareVariableId: int	✓
	isAnimationFinished	arg1: <パラメータオブジェクト> ret: <bool: 値 >	「モーションの表示が全て終わった」を判定する	パラメーター無し	✓

リンク条件判定[2]

名前空間	名前	パラメータ	説明	備考(指定可能プロパティー)	プレイヤー実装
	isObjectActionChanged	arg1: <パラメータオブジェクト > ret: <bool: 値 >	「指定オブジェクトのアクションが変化」を判定する *アクションの比較はアクション名で行われる。	objectId: int * Agtk.constants.actionCommands.SelfObject, ChildObject, LockedObject, ParentObjectも指定可能。 actionObjectId: int * objectIdがAgtk.constants.actionCommands.ChildObject, LockedObject, ParentObjectのいずれかの場合に、actionIdがどのオブジェクトに対して指定されているかを指定する。 actionId: int otherActions: bool	✓
	isJumpTop	arg1: <パラメータオブジェクト > ret: <bool: 値 >	「ジャンプが頂点になった」を判定する	パラメーター無し	✓
	isSlopeTouched	arg1: <パラメータオブジェクト > ret: <bool: 値 >	「坂に接触」を判定する	directionType: int *Agtk.constants.linkCondition.slopeTouched.DirectionUpper等。 downwardType: int *Agtk.constants.linkCondition.slopeTouched.DownwardLeft等。	✓
	isBuriedInWall	arg1: <パラメータオブジェクト > ret: <bool: 値 >	「壁判定に埋まった」を判定する	objectId: int	✓

プラグインの仕様に関する詳細資料［14］

スクリプト定数

名前空間	備考	エディター実装	プレイヤー実装
Agtk.constants.actionCommands	詳細は次のファイルを参照。 <プレイヤーディレクトリー>/Resources/plugins/prepare.js		✓
Agtk.constants.actionCommands.commandBehavior			✓
Agtk.constants.actionCommands.priorityType			✓
Agtk.constants.actionCommands.templateMove			✓
Agtk.constants.actionCommands.objectLock			✓
Agtk.constants.actionCommands.objectCreate			✓
Agtk.constants.actionCommands.objectChange			✓
Agtk.constants.actionCommands.objectMove			✓
Agtk.constants.actionCommands.objectPushPull			✓
Agtk.constants.actionCommands.sceneRotateFlip			✓
Agtk.constants.actionCommands.soundPlay			✓
Agtk.constants.actionCommands.messageShow			✓
Agtk.constants.actionCommands.scrollMessageShow			✓
Agtk.constants.actionCommands.effectShow			✓
Agtk.constants.actionCommands.effectRemove			✓
Agtk.constants.actionCommands.particleShow			✓
Agtk.constants.actionCommands.particleRemove			✓
Agtk.constants.actionCommands.movieShow			✓
Agtk.constants.actionCommands.imageShow			✓
Agtk.constants.actionCommands.gameSpeedChange			✓
Agtk.constants.actionCommands.timer			✓
Agtk.constants.actionCommands.directionMove			✓
Agtk.constants.actionCommands.forthBackMoveTurn			✓
Agtk.constants.actionCommands.menuShow			✓
Agtk.constants.actionCommands.menuHide			✓
Agtk.constants.actionCommands.soundStop			✓
Agtk.constants.actionCommands.soundPositionRemember			✓
Agtk.constants.actionCommands.fileLoad			✓
Agtk.constants.actionCommands.soundPositionRemember			✓
Agtk.constants.actionCommands.objectUnlock			✓
Agtk.constants.linkCondition.objectWallTouched			✓
Agtk.constants.linkCondition.objectHit			✓
Agtk.constants.linkCondition.attackAreaTouched			✓
Agtk.constants.linkCondition.attackAreaNear			✓
Agtk.constants.linkCondition.objectNear			✓
Agtk.constants.linkCondition.objectFacingEachOther			✓
Agtk.constants.linkCondition.objectFacing			✓
Agtk.constants.linkCondition.objectFound			✓
Agtk.constants.linkCondition.objectFacing			✓
Agtk.constants.linkCondition.locked			✓
Agtk.constants.linkCondition.slopeTouched			✓
Agtk.constants.conditions			✓
Agtk.constants.assignments			✓
Agtk.constants.attackAttributes			✓
Agtk.constants.filterEffects			✓
Agtk.constants.systemLayers			✓
Agtk.constants.qualifier			✓
Agtk.constants.objectType(削除予定)			✓
Agtk.constants.objectGroup			✓
Agtk.constants.tileGroup			✓
Agtk.constants.tile			✓
Agtk.constants.controllers			✓
Agtk.constants.animations			✓
Agtk.constants.tracks			✓
Agtk.constants.objects			✓
Agtk.constants.switchVariableObjects			✓
Agtk.constants.databaseTemplateTypes			✓

エディターが管理するプラグイン情報

プロジェクトデータにはplugins[]に格納する。

格納される情報：

id	プラグインID	
filename	プラグインファイル名	拡張子を見て、JavaScriptかCoffeeScriptかを判定する。 CoffeeScriptであれば拡張子がjsのJavaScriptファイルが自動生成される。
enabled	有効化されているか	
parameters	パラメータ情報	[{id: <ID1>, name: <名前1>, value: <値1>}, …]

UI パラメータについて

プラグインでエディターに追加できるパラメータについて、 数値入力、 文字列入力ができます。

パラメータの記述

JSONで記述	[{id: <パラメータID1>, name: <パラメータ名1>, type: String/Number/Json/ImageId/TextId/…/Custom, decimals: <Numberの時のみ。小数点の桁数>, minimumValue: <Numberの時のみ。最小値>, maximumValue: <Numberの時のみ。最大値>, customParam: [{id: <カスタムID1>, name: <カスタム名1>}, ...], defaultValue: <デフォルト値1>, …}]

id	パラメータID	1以上のプラグイン内で重ならない整数
name	パラメータ名	
type	パラメータ種別	String/Number/Json/ImageId/TextId/…/Custom
defaultValue	デフォルト値	

種別ごとの指定

type=String　文字列を入力できる

type=MultiLineString　改行を含む文字列を入力できる

type=Number　数値を入力できる（数値は倍精度浮動小数点数で格納されます。）

decimals	小数点の桁数	デフォルト値: 0
minimumValue	最小値	デフォルト: 制限なし
maximumValue	最大値	デフォルト: 制限なし

type=Json　JSON 文字列を入力できる。

type=ImageId　画像ID を選択できる。

type=TextId　テキストID を選択できる。

type=SceneId　シーン ID を選択できる。

type=TilesetId　タイルセットID を選択できる。

type=AnimationId　アニメーション ID を選択できる。

type=ObjectId　オブジェクトID を選択できる。

type=FontId　フォント ID を選択できる。

type=MovieId　動画ID を選択できる。

type=BgmId　BGM ID を選択できる。

type=SeId　SE ID を選択できる。

type=VoiceId　音声ID を選択できる。

type=VariableId　オブジェクトの変数IDを選択できる。
referenceId: <SwitchVariableIdが指定されているid>
withNewButton: true/false
※referenceIdが指定され、SwitchVariableIdにプロジェクト共通が選択された場合は、プロジェクト共通のスイッチIDを選択可能。

type=SwitchId　オブジェクトのスイッチIDを選択できる。
referenceId: <SwitchVariableIdが指定されているid>
withNewButton: true/false
※referenceIdが指定され、SwitchVariableIdにプロジェクト共通が選択された場合は、プロジェクト共通のスイッチID
を選択可能。

type=AnimOnlyId　アニメ専用 ID を選択できる。

type=PortalId　ポータル ID を選択できる。

type=CustomId　コンボボックスから選択できる。

customParam	コンボボックスの定義	省略不可 記述例: [{id: <カスタムID1>, name: <カスタム名1>}, ...]

type=Embedded　他のパラメータの内容を埋め込み表示する。

type=EmbeddedEditable　他のパラメータの内容を埋め込み表示、編集可能にする。

sourceId: < 他の素材のパラメータ ID>, reference: < 他の素材の参照する箇所 > の追加指定が必須。
width: < 表示幅>, height: < 表示高さ >を追加指定可能。

sourceId: <他の素材のパラメータID>	現在<他の素材のパラメータID>に指定可能なのは、次のパラメータIDのみ。 Embeddedの場合 : type が "ImageId"/"TextId" のパラメータID EmbeddedEditableの場合 : typeが"TextId"のパラメータID
reference: <他の素材の参照する箇所>	"ImageId"では指定しない。"TextId"の場合はfontId/textが指定できる。
width: <表示幅>	Embedded項目の表示幅を指定
height: <表示高さ>	Embedded項目の表示高さを指定

type=SwitchVariableObjectId　スイッチまたは変数のオブジェクトIDを選択できる。

option	特殊なオブジェクトIDの指定を追加。文字列要素を含む配列で指定。（その他の実行アクション、その他の条件設定でのみ有効。）	
	“SelfObject”	自身のオブジェクト
	“LockedObject”	ロックされたオブジェクト
	“ParentObject”	親オブジェクト

type=DatabaseId　データベースIDを選択できる。

JSB API

cocos2d-x の JSB API を利用可能です。

JSB API を利用することで、 アクションゲームツクール MV で提供されない機能を追加することができるようになります。

アクションゲームツクール MV では、 独自のシーン、 レイヤー管理をしています。

アクションゲームツクール MV 独自のシーン、 レイヤーに合わせて表示するには、 独自シーンレイヤーを取得して利用する必要があります。

※参照： Agtk.sceneInstances.getCurrent().getLayerByIndex()

JSB API の呼び出しテスト

Editor で適当なプロジェクトを開きます。

シーンタブにて、 シーンがあるのを確認します。 （無ければ新規作成してください。）

オブジェクトタブに切り替え、 オブジェクト一覧からオブジェクトを選択します。 （無ければ新規作成してください。）

アクションプログラムタブでなければ切り替えてください。

アクションボックスを選択します。 （無ければ作成してください。） そ

の他の実行アクションで「＋」ボタンをクリックします。

3 ページ目の「スクリプトを実行」を選択します。

「スクリプトを実行」の部分に次を貼り付けます。

```
var layer = Agtk.sceneInstances.getCurrent().getLayerByIndex(0);
if(layer != null){
  var tag = 0x1234;
  layer.removeChildByTag(tag);
  var rect = cc.DrawNode.create();
  rect.drawRect(cc.p(16, 32), cc.p(64, 128), cc.color(128, 255, 0, 128));
  layer.addChild(rect, 0, tag);
}
```

簡易実行をクリックすると、 プレビュー画面の左下に白く縁取りされた薄緑色の四角形が表示されます。

※プラグイン間で tag 値が衝突しないようにしてください。

tag 値 (pluginId << 16) | <16bit の自由に使える数値 >

ゲームデータ内のスクリプト指定

ゲームデータ内にスクリプトを指定して、ゲームのプレイ時に実行することができます。

タイル画面で指定する

本ソフトの「タイル画面」でギミックタイルを選択し、 詳細設定の「ギミック設定」からスクリプトの指定が可能です。

●ギミックタイル ＞ ギミック設定 ＞ スイッチ・変数を変更

変数を変更：スクリプト

タイルのタイルセットID, タイルX位置、タイルY位置、インデックスを参照可能。(tilesetId, tileX, tileY, index)

スクリプトを評価した結果が数値になるようにしてください。

単一の式の例：

```
Math.random()
```

関数を使って値を返す例 ：

```
(function(){
    var  a = 1.23;
    var b = 4.56;
    return a + b;
})()
```

オブジェクト画面で指定する

本ソフトの「オブジェクト画面」で「アクションプログラム」のタブを選び、 アクションやリンクの詳細設定からスクリプトの指定が可能です。

●アクション ＞ その他の実行アクション ＞ スイッチ・変数を変更

変数を変更：スクリプト

自身のオブジェクトのオブジェクト ID, インスタンス ID を参照可能。 (objectId, instanceId)

自身のアクション ID, コマンド ID を参照可能。 (actionId, commandId)

※isCommonActionが1の場合は、actionIdにコモンアクションIDが入る。

追加パラメータ:	
	isCommonAction: 0: コモンアクションでない、1: コモンアクション
	sceneId: 配置後も変更可能なアクションの場合配置シーンID、そうでなければ-1

スクリプトを評価した結果が数値になるようにしてください。

単一の式の例：

```
Math.random()
```

関数を使って値を返す例 ：

```
(function(){
    var  a = 1.23;
    var b = 4.56;
    return a + b;
})()
```

●アクション ＞ その他の実行アクション ＞ スクリプトを実行

自身のオブジェクトのオブジェクト ID, インスタンス ID を参照可能。 (objectId, instanceId)

自身のアクション ID, コマンド ID を参照可能。 (actionId, commandId)

※commonActionLinkIndexが0か1の場合は、linkIdにはコモンアクションIDが入る。

例：

追加パラメータ:	
	isCommonAction: 0: コモンアクションでない、1: コモンアクション
	sceneId: 配置後も変更可能なアクションの場合配置シーンID、そうでなければ-1

※戻り値を明示的に指定することで、「スクリプトを実行」を処理した後の挙動を制御可能です。 (「アクションコマンド」を参照。)

```
Agtk.log("Action  executed: objectId: " + objectId + ", instanceId: " + instanceId)
```

● アクションリンク ＞ その他の条件設定 ＞ スクリプトを実行
自身のオブジェクトのオブジェクト ID, インスタンス ID を参照可能。 (objectId, instanceId)
自身のリンクID, 条件IDを参照可能。(linkId, conditionId)
※commonActionLinkIndexが0か1の場合は、linkIdにはコモンアクションIDが入る。
追加パラメータ：commonActionLinkIndex: コモンアクションの入るリンクの場合0、出ていくリンクの場合1、コモンアクションでない場合-1
「スイッチ・変数を変更」と同様にスクリプトを評価した結果が真偽値になるようにしてください。

単一の式の例

```
Math.random() < 0.1
```

関数を使って値を返す例 ：

```
(function(){
    var threshold = 0.1;
    var value = Math.random();
    return (value < threshold);
})()
```

シーンで指定する

● その他＞コース(直線)、コース(曲線)、コース(円) ＞ スイッチ・変数を変更
変数を変更：スクリプト
シーンiD、パーツID, ポイントID、インデックスを参照可能。(sceneId, scenePartId, pointIndex, index)
スクリプトを評価した結果が数値になるようにしてください。

遷移で指定する

● 画面フロー＞ 遷移リンク ＞ 切り替え後演出 ＞ スイッチ・変数を変更
変数を変更：スクリプト
遷移リンクID、インデックスを参照可能。(transitionLinkId, index)
スクリプトを評価した結果が数値になるようにしてください。

● ポータル移動 ＞ ポータル ＞ A→Bの設定、B→Aの設定
■移動前（プレイヤーが触れたときの設定） ＞ スイッチ・変数を変更
変数を変更：スクリプト
ポータルID、A/B ID、インデックスを参照可能。(portalId, abId, index)
スクリプトを評価した結果が数値になるようにしてください。

■移動後 ＞ スイッチ・変数を変更
変数を変更：スクリプト
ポータルID、A/B ID、インデックスを参照可能。(portalId, abId, index)
スクリプトを評価した結果が数値になるようにしてください。