Plug-in Specifications & Information［01］

Basics

- Write code in UTF-8.

- Plug-in specifications may be added to or changed.

Plug-Ins

Here's a brief explanation of plug-ins.

Plug-ins here refer to script plug-ins.

Script plug-ins are managed in the editor, and plug-in parameters can be specified in the editor's UI. (Auto-tile plug-in, unique action command plug-ins, unique link condition plug-ins, etc.)

The language supported is JavaScript. Script plug-ins can also be written in CoffeeScript.

CoffeeScript will be automatically converted to JavaScript.

Add script plug-ins from the editor's plug-in screen.

Added plug-ins are saved in the plugins folder within the project folder itself.

The JavaScript file extension should be .js, and the CoffeeScript file extension should be .coffee.

CoffeeScript will be automatically converted into JavaScript when loaded or updated, and will be saved in the plugins folder as a .js Javascript file.

To support multiple languages for script plug-ins, the editor's locale is defined with the setLocale() function.

By changing the display language according to the function's returned value, the plug-in can support multiple languages.

* Although region ID strings (such as ja_JP) are included, the first two characters can be removed if the region is unnecessary.

Script plug-ins must return a JavaScript object (script plug-in object) when the script file is loading.
If the load fails (failed to parse the JavaScript), or getInfo data retrieval fails, this object is destroyed. Try not to pollute the global environment.

Script plug-in objects are stored in the global object Agtk.plugins. (Script plug-in objects can be accessed with "Agtk.plugins[<Plug-in ID>]".)

At the moment, script plug-in objects need to provide the following functions.

setLocale	Function	
getInfo	Function	
initialize	Function	
finalize	Function	
call	Function	
update	Function	* Optional

You can call cocos2d-x JSB API in script plug-ins.

http://www.cocos2d-x.org/docs/api-ref/js/v3x/

When previewing the game, the script execution log will be output to script.log in the same directory player.exe is located.

Script Plug-in Interface

Function Name	Description	Parameter	Return Value	Notes	Editor Implementation	Player Implementation
getInfo	Retrieve plug-in information	arg1: (string)category			✓	✓
		name	Plug-in name	Returns string type.		
		description	Plug-in description	Returns string type.		
		author	Creator name	Returns string type.		
		help	Plug-in help	Returns string type.		
		parameter	JavaScript object of plug-in parameter	Set plug-in parameters on the plug-in details screen. * See "UI Parameters". [{id: <Parameter ID 1>, name: <Parameter Name 1>, type: "String"/"Number"/"Json"/"ImageId"/"TextId"/… /"Custom"/"Embedded"/"EmbeddedEditable", decimals: <Only used when type is Number. Number of Decimal Places>, minimumValue: <Only used when type is Number. Minimum Value>, maximumValue: <Only used when type is Number. Maximum Value>, customParam: [{id: <Custom ID 1>, name: <Custom Name 1>}, ...], defaultValue: <Default Value 1>, …}]		
		internal	Data that displays plug-in status	Returns plug-in's internal information. Format is limited to information that can be converted to text strings by JSON.stringify and handed off by the next initialize and setInternal. Saved and restored on gameplay save/load.		
		actionCommand	Defines unique action command	Unique actions defined here will appear on the action command selection screen. Add a unique action command and the settings configured with "parameter" will be shown on the settings screen. * See the "UI Parameter" for information about the parameter element. [{id: <Unique Action Command ID>, name: "<Unique Action Command Name>", description: "<Unique Action Command description>", parameter: <Description is the same as plug-in parameter definition>, …}, …}		
		linkCondition	Defines unique link condition	Unique link conditions defined here will appear on the link condition selection screen. Add a unique link condition and the settings configured with "parameter" will be shown on the settings screen. * See the "UI Parameter" for information about the parameter element. [{id: <Unique Link Condition ID>, name: "<Unique Link Condition Name>", description: "<Unique Link Condition description>", parameter: <Description is the same as plug-in parameter definition>, …}, …}		
		autoTile	Defines auto-tile parameters	The settings specified here will be displayed beneath the plug-in selection item on the tileset settings screen.		
initialize	Initializes plug-in	Data returned with getInfo('internal')	–	Called from the system. Initially passes undefined. In the player, the data returned by getInfo('internal') will be passed when the project is saved in the editor.	✓	✓
finalize	Finalizes plug-in	–	–	Called from the system. Processes plug-in finalization. No need to do anything if it's not needed. Internal plug-in status is fetched with getInfo('internal') and passed as a call parameter on the next initialize. Internal plug-in status is recorded independently by both the editor and player.	✓	✓
update	Calls update process	arg1:<delta time>	–	Called from the system. Called with each frame. The time elapsed from the previous frame is contained in <delta time>.		✓
setLocale	Sets the locale in which to run the plug-in.	arg1: (string)locale	–	Called from the system. Required for setting to locale="en".	✓	✓
setParamValue	Sets data configured in plug-in parameters.	arg1: Object that parsed the JSON in parameter settings	–	Called from the system. Called whenever parameters are updated on the editor's plug-in settings screen.	✓	–
setInternal	Sets internal data	Data fetched using getInfo('internal') when play data is saved	–	Called from the system. Called when play data is loaded.		✓
call	Calls various functions				✓	
	Returns number of horizontal tiles from tile list	arg1: <object: Auto-tile parameter setting value> arg2: "getHorzCount"	Horizontal tile count	Called from the system. Needs to be defined in auto-tile plug-in.		
	Returns number of vertical tiles from tile list.	arg1: <object: Auto-tile parameter setting value> arg2: "getVertCount"	Vertical tile count	Called from the system. Needs to be defined in auto-tile plug-in.		
	Returns tile information list from tile list.	arg1: <object: Auto-tile parameter setting value> arg2: "getFrontTileList"	Tile information list	Called from the system. Needs to be defined in auto-tile plug-in.		
	Returns range information list of all tiles in the tile list.	arg1: <object: Auto-tile parameter setting value> arg2: "getBackTileList"	Tile range information list	Called from the system. Needs to be defined in auto-tile plug-in.		
	Returns list of attributes handled by this auto-tile plug-in.	arg1: <object: Auto-tile parameter setting value> arg2: "getAttributeList"	Attribute list	Called from the system. Needs to be defined in auto-tile plug-in.		
	Updates auto-tiles in the specified range.	arg1: <object: Auto-tile parameter setting value> arg2: "updateArea" arg3: [<int: Tile starting X coordinate>, <int: Tile starting Y coordinate>, <int: Horizontal tile count>, <int: Vertical tile count>]	–	Called from the system. Needs to be defined in auto-tile plug-in.		
	Reports parameter change	arg1: <object: Auto-tile parameter setting value> arg2: "onParamChanged"	–	Called from the system. Needs to be defined in auto-tile plug-in.		
getAttribute	Returns attribute of specified tile coordinates.	arg1: <object: Auto-tile parameter setting value> arg2: <int: Tile X coordinate> arg3: <int: Tile Y coordinate> ret: <string: Attribute string>	Attribute	Used for switching tiles used in this auto-tile using the attribute set by other auto-tile plug-ins.	✓	–
getBorderType	Returns border information for specified tile coordinates.	arg1: <object: Auto-tile information> arg2: <int: Tile X coordinate> arg3: <int: Tile Y coordinate> ret: <string: Border string>	Border information	Used for switching tiles used in this auto-tile using the border information set by other auto-tile plug-ins.	✓	–
execActionCommand	Executes action command.	arg1: <Action command index> arg2: <Settings parameter> arg3: <Object ID> arg4: <Instance ID> arg5: <Action ID> arg6: <Command ID> arg7: <1 if common action, 0 if not> arg8: <Placement scene ID if changeable after placement, -1 if not>	Command return value Refer to the "Process Flow of Other Runtime Actions" in "Action Box".	Called from the system. When multiple action commands are defined by the plug-in, they are differentiated by <Action command index>. Parameters set by action commands are passed as objects using <Settings parameter>.	–	✓
execLinkCondition	Evaluates link condition.	arg1: <Link condition index> arg2: <Settings parameter> arg3: <Object ID> arg4: <Instance ID> arg5: <Action link ID> arg6: <0 if incoming link of common action, 1 if outgoing link of common action, -1 if not common action>	Boolean indicating whether link was established	Called from the system. Called during evaluation of link conditions. If multiple link conditions are defined by a plug-in, they are differentiated by <Link index>. Parameters set by link conditions are passed as objects using <Settings parameter>.	–	✓

Shared Script API [1]

Namespace	Name	Parameter	Description	Notes	Editor Implementation	Player Implementation
Agtk.version			Text string that displays execution engine version.	PGMMV <Version Number> player <Version Number> runtime <Version Number>	✓	✓
Agtk.log		arg1: <string: Output string> ...	Outputs log. In the player, it is output in the "Runtime Log Console".		✓	✓
Agtk.reset		No parameters	Resets the game. Performs same operation as resetting with F5 key.			✓
Agtk.settings	tileWidth		Tile width (unit: dots)		✓	✓
	tileHeight		Tile height (unit: dots)		✓	✓
	screenWidth		Screen width (unit: dots)		✓	✓
	screenHeight		Screen height (unit: dots)		✓	✓
	playerCharacters		playerCharacters object			✓
	projectPath		Project path			✓
Agtk.playerCharacters	getCount	–	Returns number of slots for managing player characters. * Currently fixed at 4.			✓
	get	arg: <Index>	Returns playerCharacter of the specified slot.			✓
Agtk.playerCharacter	getCount		Returns number of objects set in the player character slot.			✓
	get	arg: <Index>	Returns Object ID.			✓
Agtk.scenes	getCount/getIdByIndex/getIdByName/getById				✓	
	getIdList/getIdByName/get/getLayerById					✓
Agtk.scene	id/name				✓	✓
	getLayerByIndex				✓	
	horzScreenCount		Scene size (width)		✓	✓
	vertScreenCount		Scene size (height)		✓	✓
	getLayerIdList/getLayerIdByName/getLayerIndexById/getLayerById					✓
Agtk.layers	getCount/getIdByIndex/getIdByName/getById				✓	
Agtk.layer	getTileInfo	arg1: x arg2: y	Retrieves tile information set in scene.	{tilesetId: <Tileset ID>, x: <X position>,y: <Y position>}	✓	✓
	setSubtileInfo	arg1: sx arg2: sy arg3: subtileX arg4: subtileY	Configures subtile information set in scene.		✓	
	getSlopeIdList/getSlopeById					✓
Agtk.slope	name/startX/startY/endX/endY					✓
Agtk.ui					✓	–
Agtk.ui.editSceneId					✓	–
Agtk.ui.editLayerIndex					✓	–
Agtk.sceneInstances					–	✓
	getCurrent	–	Retrieves current scene		–	✓
Agtk.sceneInstance	getLayerByIndex	arg1: <Layer Index>	Retrieves layer corresponding to the specified number. Numbering starts from 0. Returns cc.Node.		–	✓
	getMenuLayerById	arg1: <Menu Layer ID>	Retrieves menu layer corresponding to specified ID. Returns cc.Node. The foremost menu layer (created implicitly) can be specified in Agtk.constants.systemLayers.HudLayerId.			
	getHudLayer	–	A cc.Node unaffected by the camera will be returned.			
	sceneId		Scene ID		–	✓
	getCurrentCameraTargetPos	-	Returns current camera target point.	{x: <X coordinates>, y: <Y coordinates>}		✓
	getCurrentCameraDisplayScale		Returns current camera screen enlargement ratio.	{x: <X enlargement ratio>, y: <Y enlargement ratio>}		✓
Agtk.plugins	isValid	arg1: <Plug-in object> arg2: <Plug-in ID> arg 3: <Locale>	Checks whether a plug-in object has basic functions implemented.	Presumably only used by the system.	✓	✓
	reload	arg1: <Plug-in object> arg2: <Plug-in ID> arg3: <Locale> arg4: <Internal data>	Destroys loaded plug-in and resets it. Plug-in status will be restored using <Internal data>.	Presumably only used by the system.	✓	✓
	unload	arg: <Plug-in ID>	Unloads plug-in.	Presumably only used by the system.	✓	✓
	□	<Index>	Calls in the format of Agtk.plugins[<Index>] and returns the plug-in object of the specified <Index>.	Refer to "Script Plug-in IF" sheet for details about plug-in objects.	✓	✓
	length		Returns number of loaded plug-ins.		✓	✓
	getIndexById	arg: <Plug-in ID>	Returns plug-in index.		✓	✓
	getById	arg: <Plug-in ID>	Returns plug-in object.		✓	✓
Agtk.objects	getIdByName/get					✓
Agtk.object	id/name		readonly			✓
	operatable		bool: Operable via input device			✓
	bullets		Bullet data set in "Bullet Settings"			✓
	actions		Action program action box data			✓
	animationId		Animation ID set by "Select Animation"			✓
Agtk.object.bullets	getIdList/getIdByName/get		Bullet data set in "Bullet Settings"			✓
Agtk.object.bullet	id/name		ID/Name			✓
Agtk.object.actions	getIdList/getIdByName/get		Action program action box data			✓
Agtk.object.action	id/name		ID/Name			✓
Agtk.object.viewports	getIdList/getIdByName/get		Field of vision, lighting data set in "Field of Vision, Lighting"			✓
Agtk.object.viewport	id/name		ID/Name			✓
Agtk.object.switches	getIdList/getIdByName/get		Object switch information	The object switch's preset ID is defined in Agtk.constants.switches. * The content is identical to the preset ID defined in Agtk.objectInstance.switches.		✓
Agtk.object.switch	id/name		ID/Name			✓
Agtk.object.variables	getIdList/getIdByName/get		Object variable information	The object switch's preset ID is defined in Agtk.constants.variables. * The content is identical to the preset ID defined in Agtk.objectInstance.variables.		✓
Agtk.object.variable	id/name		ID/Name			✓
Agtk.animations	getIdList/getIdByName/get		Animation data			✓
Agtk.animation	id/name		ID/Name			✓
	type		Animation type Agtk.constants.animations.Motion/Agtk.constants.animations.Effect/Agtk.constants.animations.Particle			✓
	motions		Motion data Undefined when type != Agtk.constants.animations.Motion			✓
	getResourceSetIdList		Resource Set Name Array			✓
	getResourceSetNameList		Resource Set Name Array *Each element is compatible with the respective getResourceSetIdList() array.			✓
Agtk.animation.motions	getIdList/getIdByName/get		Motion data			✓
Agtk.animation.motion	id/name		ID/Name			✓
	directions		Display direction data			✓
Agtk.animation.motion.directions	getIdList/getIdByName/get		Display direction data			✓
Agtk.animation.motion.direction	id/name		ID/Name			✓
	tracks		Track data			✓
Agtk.animation.motion.direction.tracks	getIdList/getIdByName/get		Track data			✓
Agtk.animation.motion.direction.track	id/name		ID/Name			✓
	timelineType		Track type 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: <Tileset ID>	Returns tileset.		✓	
	getIdList/getIdByName/get					✓

Shared Script API [2]

Namespace	Name	Parameter	Description	Notes	Editor Implementation	Player Implementation
Agtk.tileset	id/name				✓	✓
	imageId		Tileset image ID		✓	✓
	getPluginId				✓	
	getPluginParam				✓	
	clearAnimation	arg1: x arg2: y arg3: width arg4: height	Clears an animation.		✓	
	appendAnimation	arg1: x arg2: y arg3: width arg4: height arg5: ax arg6: ay arg7: duration300	Adds an animation.		✓	
	getWallBits	arg1: x arg2: y	Acquires wall detection data for all valid tile directions. Check the constants listed below to determine if wall detection is configured. Agtk.constants.tile.WallTopBit Agtk.constants.tile.WallLeftBit Agtk.constants.tile.WallRightBit Agtk.constants.tile.WallBottomBit			✓
Agtk.switches	References project common variables					
	Preset reference ID definition					
	InitId	Reset				✓
	SaveFileId	Save File				✓
	LoadFileId	Load File				✓
	CopyFileId	Copy File				✓
	DeleteFileId	Delete File				✓
	InitialCameraId	Default Camera				✓
	LoadingSceneId	Show Loading Screen				✓
	QuitTheGameId	End Game<				✓
	Method					
	get	arg1: <int: SwitchID>	Return objects with Agtk.switch.			✓
	getIdByName	arg1: <string: Switch Name> ret: <int: Switch ID>	Acquire switch ID by name. If not found, return −1. *Since preset variable identifiers are language dependent, Retrieves switch value using them.			✓
Agtk.switch	getValue	ret: <bool: Value>				✓
	setValue	arg1: <bool: Settings value>	Assigns value to the switch.			✓
Agtk.variables	References project common variables					
	Preset reference ID definition					
	PlayerCountId	Number of players				✓
	_1PObjectId	1P object				✓
	_2PObjectId	2P object				✓
	_3PObjectId	3P object				✓
	_4PObjectId	4P object				✓
	_1PInstanceId	1P instance				✓
	_2PInstanceId	2P instance				✓
	_3PInstanceId	3P instance				✓
	_4PInstanceId	4P instance				✓
	_1PControllerId	1P controller ID				✓
	_2PControllerId	2P controller ID				✓
	_3PControllerId	3P controller ID				✓
	_4PControllerId	4P controller ID				✓
	PortalMoveStartTimeId	Portal transfer start time				✓
	FileSlotId	File slot				✓
	CopyDestinationFileSlotId	Destination file slot				✓
	MouseXId	Mouse X coordinate				
	MouseYId	Mouse Y coordinate				
	Method					
	get	arg1: <int: Variable ID>	Returns Agtk.variable object.			✓
	getIdByName	arg1: <string: Variable name> ret: <int: Variable ID>	Retrieves variable ID using variable name. Returns −1 if ID is not found. * Preset variable may change according to language, so avoid using this method if possible.			✓
Agtk.variable	getValue	ret: <double: Value>	Retrieves variable value.			✓
	setValue	arg1: <double: Setting value>	Assigns value to variable.			✓
Agtk.actionCommands	objectCreate	arg1: <int: Object ID> arg2: <int : X position> arg3 : <int : Y position> arg4 : <int : Layer number> ret: <int: Instance ID>	<Layer number>: Starting from the front: 0, 1, ...			✓
	objectDestroy	arg1: <int: Instance ID>				✓
Agtk.portals	getIdList/getIdByName/get					✓
Agtk.portal	id/name/a/b		Portal A/B			✓
Agtk.portal.a/b	sceneId		Scene ID			✓
	x/y/width/height		X/Y/width/height			✓
	movable		Whether it can be moved in other directions			✓
Agtk.controllers	MaxControllerId	16				✓
	getOperationKeyPressed	arg1: <int: Controller ID> arg2: <Control key ID> ret: <bool: Whether pressed>	Controller ID: 0−16 *0 = keyboard/mouse Control key ID: 1~	<Control key ID>: Agtk.constants.controllers.OperationKeyA, ..., Agtk.constants.controllers.OperationKeyCancel		✓
	getOperationKeyValue	arg1: <int:ControllerID> arg2: <Operationkey ID> ret: <double:Value>	Controller ID: 0~16 *0 is keyboard/mouse Operation key ID: 1~ Return value is within range of −1~1.	<Operation key ID>: Agtk.constants.controllers.OperationKeyA, ⋯, Agtk.constants.controllers.OperationKeyCancel		
	getKeyValue	arg1: <int: Controller ID> arg2: <Key code> ret: <double: Value>	Controller ID: 0−16 *0 = keyboard/mouse Key code: 0~ Key codes, return values depend on device.	Key code when controller ID = 0: Agtk.constants.controllers.ReservedKeyCodePc_W − Agtk.constants.controllers.ReservedKeyCodePc_MousePointer		✓
	isConnected	arg1: <int: Controller ID> ret: <bool: Whether connected>	Controller ID: 0−16 *0 = keyboard/mouse			✓
Agtk.objectInstances	get	arg1: <int: Instance ID>	Returns Agtk.objectInstance object.			✓
	getIdByName	arg1: <int: Object ID> arg2: <string: Instance name>	Retrieves instance ID via name. Returns −1 if not found. * Checks for consistency with object name if object instance is dynamically generated.			✓
Agtk.objectInstance	id		readonly			✓
	objectId		readonly int: <Object ID>			✓
	layerId		Object location layer ID			
	layerIndex		Object location layer index			
	getAttackerInstanceIdList	ret: <array: Instance ID array>	Returns instance ID list of object instance that attacked this object instance. *Information for 1 frame before can be acquired.			
Agtk.objectInstance.switches	References object switches					✓
	Preset reference ID definition					
	InvincibleId	Invincibility	Variable for managing attributes. 1−8 are used for presets [Fire, Water, Earth, Wind, Lightning, Ice, Light, Dark].			✓
	FreeMoveId	Free movement				✓
	LockTargetId	Lock target				✓
	PortalTouchedId	Portal touched				✓
	CriticalDamagedId	Critical received				✓
	DisabledId	Disable				✓
	SlipOnSlopeId	Slide on slope				✓
	AffectOtherObjectsId	Influence other physics objects				✓
	AffectedByOtherObjectsId	Influenced by other physics objects				✓
	FollowConnectedPhysicsId	Prioritizes movement of connected physics object				✓
	Method					
	get	arg1: <int: Switch ID>	Returns Agtk.objectInstance.switch object.			✓
	getIdByName	arg1: <string: Switch Name> ret: <int: Switch ID>	Retrieves switch ID using switch name. Returns −1 if ID is not found. * Preset variable names may change according to language, so avoid using this if possible.			✓
Agtk.objectInstance.switch	getValue	ret: <bool: Value>	Retrieves switch value.			✓
	setValue	arg1: <bool: Settings value>	Assigns value to the switch.			✓

Shared Script API [3]

Namespace	Name	Parameter	Description	Notes	Editor Implementation	Player Implementation
Agtk.objectInstance.variables	References object variables					✓
	Preset reference ID definition					
	ObjectIDId	Object ID				✓
	HPId	HP				✓
	MaxHPId	Max HP				✓
	MinimumAttackId	Min attack				✓
	MaximumAttackId	Max attack				✓
	DamageRatioId	Received Damage Rate				✓
	AttackAttributeId	Attack Attribute				✓
	AreaAttributeId	Area Detection				✓
	XId	X Coordinate Position				✓
	YId	Y Coordinate Position				✓
	VelocityXId	X Direction Speed				✓
	VelocityYId	Y Direction Speed				✓
	PlayerIDId	Player Detection				✓
	DamageValueId	Received Damage Amount				✓
	CriticalRatioId	Critical Boost (%)				✓
	CriticalIncidenceId	Critical Rate (%)				✓
	InvincibleDurationId	Invincibility duration				✓
	FixedAnimationFrameId	Fixes animation display on a specific frame				✓
	InstanceIDId	Instance ID				✓
	InstanceCountId	No. of same instances placed in scene				✓
	SingleInstanceIDId	Single unit instance ID				✓
	ControllerIDId	Controller ID				✓
	HorizontalMoveId	Horizontal Displacement				✓
	VerticalMoveId	Vertical Displacement				✓
	HorizontalAccelId	Horizontal Acceleration				✓
	VerticalAccelId	Vertical Acceleration				✓
	HorizontalMaxMoveId	Horizontal max displacement				✓
	VerticalMaxMoveId	Vertical max displacement				✓
	HorizontalDecelId	Horizontal deceleration				✓
	VerticalDecelId	Vertical deceleration				✓
	DurationToTakeOverAccelerationMoveSpeedId	Acceleration changeover time				✓
	ScalingXId	Scale X (%)				✓
	ScalingYId	Scale Y (%)				✓
	DispPriorityId	Display Priority				✓
	InitialJumpSpeedId	Initial jump speed	Returns Agtk.objectInstance.variable object.			✓
	Method					
	get	arg1: <int: Variable ID>	Returns Agtk.objectInstance.variable object.			✓
	getIdByName	arg1: <string: Variable name> ret: <int: Variable ID>	Retrieves variable ID using variable name. Returns -1 if ID is not found. * Preset variable may change according to language, so avoid using this method if possible.			✓
Agtk.objectInstance.variable						✓
	getValue	ret: <double: Value>	Retrieves variable value.			✓
	setValue	arg1: <double: Setting value>	Assigns value to variable.			✓
Agtk.fonts	getIdList/getIdByName/get					✓
Agtk.font	id/name					✓
	imageFontFlag	bool	If TRUE, uses image as font			✓
	imageId	int	Image ID			✓
	fontName	string	Font file name			✓
	ttfName	string	TTF name			✓
	fontSize	int	Font size			✓
	fixedWidth	bool	If TRUE, fixes width			✓
	antialiasDisabled	bool	Disable antialiasing			✓
	aliasThreshold	Int	Threshold			✓
	hankakuWidth	int	Half-width			✓
	zenkakuWidth	int	Full-width			✓
	letterLayout	string	Character placement			✓
Agtk.texts	getIdList/getIdByName/get					✓
Agtk.text	id/name					✓
	fontId	int	Font ID			✓
	letterSpacing	int	Character spacing			✓
	lineSpacing	int	Line spacing			✓
	localeText	object	Stored properties: locale as key, text as value.			✓
	getText	arg1: <string: locale> ret: <string: Text>	Returns text with expanded text tags of specified locale (¥O, ¥T, ¥V).			
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: Database ID> ret: <object: Agtk.database>	Get Database from ID null if not found			✓
	getIdByName	arg: <string: Database Name> ret: <int: Database ID>	Get Database ID from Database Name -1 if not found			✓
	getIdList	ret: <array: Array of the Database ID>	Get an array of Database ID null if not found			✓
Agtk.database	getColumnList	ret: <array: Array of the Column Name>	Get an array of Column Name null if not found			✓
	getColumnByIndex	arg: <int: Column Number> ret: <array: Array of Column Values>	Get an array of Column Value from Column Number null if not found			✓
	getColumnByName	arg: <string: Column Name> ret: <array: Array of Column Values>	Get an array of Column Value from the Column Name null if not found			✓
	getRowList	ret: <array: Array of Row Name>	Get an array of Row Name null if not found			✓
	getRowByIndex	arg: <int: Row Number> ret: <array: Array of Row Values>	Get an array of Row Value from Row Number null if not found			✓
	getRowByName	arg: <string: Row Name> ret: <array: Array of Row Values>	Get an array of Row Value from the Row Name null if not found			✓
	getFieldByIndex	args: <int: Column Number, int: Row Number> ret: <string: Field Value's String>	Get Field Value from Number null if not found			✓

Action Commands [1]

Caution is required when calling "Other Runtime Actions" from scripts specified in "Execute Script".

* See: "Process Flow of Other Runtime Action in Action Box"

In particular, processing must be performed as below so that when the following two items are called, their behavior will match with "Other Runtime Actions" not using scripts.

- When calling execCommandObjectPushPull on single units, specify Agtk.constants.actionCommands.commandBehavior.CommandBehaviorLoop as the return value of "Execute script".
  (If oneTimeEffect: true is not specified.)
- When calling execCommandActionExec on single units, specify execCommandActionExec as the return value of "Execute script".

* If using on more than a single unit, please proceed only after thoroughly understanding the mechanics involved.

Namespace	Name	Parameter	Description	Notes (Properties that can be specified)	Player Implementation
Agtk.objectInstance	execCommandTemplateMove	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Template movement settings"	moveType: int *Agtk.constants.actionCommands.templateMove.MoveHorizontal, etc. *MoveNearPlayer and MoveApartNearPlayer are scheduled to be removed from Agtk.constants.actionCommands.templateMove definitions. MoveNearObject and MoveApartNearObject will be used in their place. horizontalMoveStartRight: bool horizontalMoveDuration300: int horizontalInfinite: bool verticalMoveStartDown: bool verticalMoveDuration300: int verticalInfinite: bool randomMoveDuration300: int randomMoveStop300: int nearPlayerGroup: int nearObjectLockedObjectPrior: bool nearPlayerLockedPlayerPrior: bool *Scheduled for removal apartNearObjectGroup: int apartNearPlayerLockedPlayerPrior: bool apartNearObjectLockedObjectPrior: bool *Scheduled for removal fallFromStep: bool *Opposite of Editor ignoreOtherObjectWallArea: bool ignoreWall: bool * 300 specifies a numerical value equivalent to 1 second in duration300. (Same for other instances of '300'.)	✓
	execCommandObjectLock	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Lock Object"	lockTouchedObject: bool lockViewportLightObject: bool lockObjectOnScreen: bool objectType: int *Agtk.constants.actionCommands.objectLock.SetByObjectGroup, etc. objectGroup: int objectId: int useType: int *Agtk.constants.actionCommands.objectLock.UseSwitch, etc. switchId: int switchCondition: int variableId: int compareVariableOperator: int compareValueType: int *Agtk.constants.actionCommands.objectLock.CompareValueConstant, etc. compareValue: double compareVariableObjectId: int compareVariableQualifierId: int compareVariableId: int * For the qualifierID, use either an Object's instance ID or one of the following Agtk.constants.qualifier.QualifierSingle Agtk.constants.qualifier.QualifierWhole For objectGroup, specify theindex of the object group set The object groups prepared by the system are Agtk.constants.objectGroup.ObjectGroupPlayer, Can be specified with Agtk.constants.objectGroup.ObjectGroupEnemy	✓
	execCommandObjectCreate	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Generate Object".	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: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Change Object".	objectId: int actionId: int takeOverVariables: bool takeOverSwitches: bool takeOverParentChild: bool createPosition: int useConnect: bool connectId: int adjustX: int adjustY: int probability: double	✓
	execCommandObjectMove	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Move Object".	moveType: int *Agtk.constants.actionCommands.objectMove.MoveWithDirection, etc. 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, etc. targetObjectGroup: int targetObjectId: int targetQualifierId: int excludeObjectGroupBit: int fitDispDirToMoveDir: bool dispWhileMoving: bool *Opposite of Editor stopActionWhileMoving: bool stopAnimWhileMoving: bool finalGridMagnet: bool * excludeObjectGroupBit is the 1 left shift number of the target group's index value. Use a logical OR to allow multiple groups. For example, to exclude the player and enemy groups, the code is (1 << Agtk.constants.objectGroup.ObjectGroupPlayer) | (1 << Opposite of Can be specified with Agtk.constants.objectGroup.ObjectGroupEnemy).	✓

Action Commands [2]

Namespace	Name	Parameter	Description	Notes (Properties that can be specified)	Player Implementation
Agtk.objectInstance	execCommandObjectPushPull	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorLoop	Executes "Push/Pull Object".	effectRangeBaseConnect: bool effectRangeBaseConnectId: int rectangle: bool rectangleDistance: int rectangleHeight: int circleDistance: int arcAngle: int effectDirectionType: int *Agtk.constants.actionCommands.objectPushPull.EffectDirectionAngle, etc. effectDirection: double directionType: int *Agtk.constants.actionCommands.objectPushPull.DirectionAngle, etc. angle: double connectId: int effectValue: int distanceEffect: bool nearValue: double farValue: double oneTimeEffect: bool targettingType: int *Agtk.constants.actionCommands.objectPushPull.TargettingByGroup, etc. targetObjectGroup: int targetObjectId: int targetQualifierId: int excludeObjectGroupBit: int	✓
	execCommandLayerMove	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Move Layer"	layerIndex: int	✓
	execCommandAttackSetting	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Attack Settings".	attackChange: double hitObjectFlag: bool objectGroupBit: int hitTileFlag: bool tileGroupBit: int attributeType: int *Agtk.constants.actionCommands.attackSetting.AttributeNone, etc. attributePresetId: int *Agtk.constants.attackAttributes.Fire,etc. attributeValue: int *objectGroupBit is the 1-left shift number of the targetgroup's index value. Use a logical OR to allow multiple groups.For example, to allow the objectGroupBit: int player and enemy groups to tileAttackArea: bool be affected, the code is (1 << Agtk.constants.objectGoup.ObjectGroupPlayer) | (1 << Can be specified with Agtk.constants.objectGroup.bjectGroupEnemy).	✓
	execCommandBulletFire	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Fire Bullet".	bulletId: int connectId: int	✓
	execCommandDisappear	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Destroy Object".	No parameters * arg1 is optional.	✓
	execCommandDisappearObjectRecover	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Restore Destroyed Object".	objectId: int	✓
	execCommandDisable	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Disable Object".	No parameters * arg1 is optional.	✓
	execCommandDisableObjectEnable	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Enable Disabled Object".	objectId: int	✓
	execCommandObjectFilterEffect	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Apply Filter Effects on Object".	effectType: int *Agtk.constants.filterEffects.EffectNoise, etc. 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: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Delete Filter Effects from Objects".	removeBit: int duration300: int * In removeBit, (1 << value) is assigned to value (value) specified from Agtk.constants.filterEffects. Use OR operators to specify multiples.	✓
	execCommandSceneEffect	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Apply Screen Effect to Scene".	layerIndex: int filterEffect: <FilterEffect Parameter object> FilterEffect Parameter object properties: effectType: int *Agtk.constants.filterEffects.EffectNoise, etc. noise: int mosaic: int monochrome: int sepia: int negaPosiReverse: int defocus: int chromaticAberration: int darkness: int imageId: int imageTransparency: int imagePlacement: int fillA: int fillR: int fillG: int fillB: int duration300: int	✓

Action Commands [3]

Namespace	Name	Parameter	Description	Notes (Properties that can be specified)	Player Implementation
Agtk.objectInstance	execCommandSceneEffectRemove	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Delete Screen Effects from Scene".	layerIndex: int removeBit: int duration300: int	✓
	execCommandSceneGravityChange	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Change Scene Gravity Effect".	gravity: double direction: double duration300: int durationUnlimited: bool	✓
	execCommandSceneRotateFlip	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Rotate/Flip Scene".	type: int rotationFlag: bool rotation: double absoluteRotation: bool flipX: bool flipY: bool duration300: int	✓
	execCommandCameraAreaChange	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Change Camera Display Area".	xRate: double yRate: double duration300: int * For xRate, yRate, 1.0 is actual size	✓
	execCommandSoundPlay	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Audio Playback".	soundType: int *Agtk.constants.actionCommands.soundPlay.SoundSe, etc. 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: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Show Text".	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, etc. templateId: int *Agtk.constants.messageShowTempleteWhiteFrame, etc. imageId: int windowTransparency: int positionType: int *Agtk.constants.actionCommands.messageShow.PositionCenter, etc. useConnect: bool connectId: int adjustX: int adjustY: int topBottomMargin: int leftRightMargin: int horzAlign: int *Agtk.constants.actionCommands.messageShow.HorzAlignLeft, etc. vertAlign: int *Agtk.constants.actionCommands.messageShow.VertAlignTop, etc. actionChangeHide: bool closeByKey: bool keyId: int objectStop: bool gameStop: bool priorityMostFront: bool (To be removed) priority: bool priorityType: int *For priorityType, designate Agtk.constants.actionCommands.priorityType.PriorityBackground~PriorityMostFrontWithMenu.	✓
	execCommandScrollMessageShow	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Show Scrolling Text".	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, etc. templateId: int *Agtk.constants.actionCommands.scrollMessageShow.TemplateBlack, etc. imageId: int backgroundTransparency: int topBottomMargin: int leftRightMargin: int horzAlign: int *Agtk.constants.actionCommands.scrollMessageShow.HorzAlignLeft, etc. positionType: int *Agtk.constants.actionCommands.scrollMessageShow.PositionCenter, etc. useConnect: bool connectId: int adjustX: int adjustY: int actionChangeHide: bool speedUpByKey: bool keyId: int objectStop: bool gameStop: bool priorityMostFront: bool (To be removed) priority: bool priorityType: int *For priorityType, designate Agtk.constants.actionCommands.priorityType.PriorityBackground~PriorityMostFrontWithMenu.	✓
	execCommandEffectShow	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Show Effect".	effectId: int positionType: int *Agtk.constants.actionCommands.effectShow.PositionCenter, etc. useConnect: bool connectId: int adjustX, adjustY: int duration300: int durationUnlimited: bool	✓
	execCommandEffectRemove	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Hide Effects".	effectId: int targettingType: int *Agtk.constants.actionCommands.effectRemove.TargettingByGroup, etc. targetObjectGroup: int targetObjectId: int	✓

Action Commands [4]

Namespace	Name	Parameter	Description	Notes (Properties that can be specified)	Player Implementation
Agtk.objectInstance	execCommandParticleShow	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Show Particles".	particleId: int positionType: int *Agtk.constants.actionCommands.particleShow.PositionCenter, etc. useConnect: bool connectId: int adjustX, adjustY: int duration300: int durationUnlimited: bool	✓
	execCommandParticleRemove	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Hide Particles".	particleId: int targettingType: int *Agtk.constants.actionCommands.particleRemove.TargettingByGroup, etc. targetObjectGroup: int targetObjectId: int	✓
	execCommandMovieShow	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Play Video".	movieId: int loop: bool volume: int defaultSize: bool width, height: int positionType: int *Agtk.constants.actionCommands.movieShow.PositionCenter, etc. useConnect: bool connectId: int vertAlign: int *Agtk.constants.actionCommands.movieShow.VertAlignCenter, etc. horzAlign: int *Agtk.constants.actionCommands.movieShow.HorzAlignCenter, etc. adjustX, adjustY: int hideOnObjectActionChange: bool stopObject: bool stopGame: bool fillBlack: bool priority: bool priorityMostFront: bool (To be removed) priorityType: int *For priorityType, designate Agtk.constants.actionCommands.priorityType.PriorityBackground~PriorityMostFrontWithMenu.	✓
	execCommandImageShow	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Display Image".	imageId: int duration300: int durationUnlimited: bool defaultSize: bool width: int height: int positionType: int *Agtk.constants.actionCommands.imageShow.PositionCenter, etc. useConnect: bool connectId: int vertAlign: int *Agtk.constants.actionCommands.imageShow.VertAlignCenter, etc. horzAlign: int *Agtk.constants.actionCommands.imageShow.HorzAlignCenter, etc. adjustX: int adjustY: int hideOnObjectActionChange: bool closeByOk: bool stopObject: bool stopGame: bool priority: bool priorityMostFront: bool (To be removed) priorityType: int *For priorityType, designate Agtk.constants.actionCommands.priorityType.PriorityBackground~PriorityMostFrontWithMenu.	✓
	execCommandSwitchVariableChange	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Change Switch/Variable".	swtch: bool switchObjectId: int switchQualifierId: int switchId: int switchValue: int *Agtk.constants.assignments.SwitchAssignOn, etc. variableObjectId: int variableQualifierId: int variableId: int variableAssignOperator: int *Agtk.constants.assignments.VariableAssignOperatorSet, etc. variableAssignValueType: int *Agtk.constants.assignments.VariableAssignValue, etc. assignValue: double assignVariableObjectId: int assignVariableQualifierId: int assignVariableId: int randomMin: int randomMax: int assignScript: string See: Agtk.constants.assignments * Do NOT specify Agtk.constants.assignments.VariableAssignScript in variableAssignValueType.	✓
	execCommandSwitchVariableReset	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Reset Switch/Variable".	Specified by array. The following properties can be configured with each element object. swtch: bool objectId: int itemId: int * If objectId == 0, that means it's common between projects.	✓
	execCommandGameSpeedChange	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.Co mmandBehaviorNext	Executes "Change Game Speed".	gameSpeed: double duration300: int durationUnlimited: bool targetObject: bool targetEffect: bool targetTile: bool targetMenu: bool targettingType: int *Agtk.constants.actionCommands.gameSpeedChange.TargettingByGroup, etc. targetObjectGroup: int targetObjectId: int targetQualifierId: int excludeObjectGroupBit: int	✓
	execCommandTimer	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Timer Function".	start: bool timerVariableObjectId: int timerVariableQualifierId: int timerVariableId: int countUp: bool secondType: int *Agtk.constants.actionCommands.timer.SecondByValue, etc. second300: int secondVariableObjectId: int secondVariableQualifierId: int secondVariableId: int	✓
	execCommandSceneTerminate	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "End Scene".	No parameters * arg1 is optional.	✓

Action Commands [5]

Namespace	Name	Parameter	Description	Notes (Properties that can be specified)	Player Implementation
Agtk.objectInstance	execCommandDirectionMove	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Set Move Direction and Move".	direction: double directionId: int *Agtk.constants.actionCommands.directionMove: Designate direction of movement and move. (Match Move Direction) moveDistance: int moveDistanceEnabled: bool	✓
	execCommandForthBackMoveTurn	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Back and Forth Moving and Turning".	moveType: int *Agtk.constants.actionCommands.forthBackMoveTurn.MoveNone, etc. turnType: int *Agtk.constants.actionCommands.forthBackMoveTurn.TurnNone, etc. directionId: int	✓
	execCommandActionExec	arg1: <Parameter Object> ret: <Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext or Agtk.constants.actionCommands.commandBehavior.CommandBehaviorBreak>	Executes "Execute Object Action". When action of this object self is executed, Agtk.constants.actionCommands.commandBehavior.CommandBehaviorBreak is returned.	objectId: int qualifierId: int actionId: int	✓
	execCommandSceneShake	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Shake Scene".	duration300: int fadeIn: bool fadeOut: bool interval300: int height: int heightDispersion: int width: int widthDispersion: int	✓
	execCommandLayerHide	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Execute "Disable Layer Display".	layerIndex: int exceptFlag: bool	✓
	execCommandLayerShow	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Enable Layer Display".	layerIndex: int exceptFlag: bool	✓
	execCommandLayerDisable	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Enable Layer Motion".	layerIndex: int exceptFlag: bool	✓
	execCommandLayerEnable	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Disable Layer Motion".	layerIndex: int exceptFlag: bool	✓
	execCommandMenuShow	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Show Menu Screen".	layerId: int useEffect: bool effectType: int *Agtk.constants.actionCommands.menuShow.None, etc. fadein: bool duration300: int	✓
	execCommandMenuHide	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Hide Menu Screen".	hideExceptInitial: bool useEffect: bool effectType: int *Agtk.constants.actionCommands.menuHide.None, etc. fadeout: bool duration300: int disableObjects: bool	✓
	execCommandDisplayDirectionMove	arg1: <Parameter object> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "Move Towards Display Direction".	moveDistance: int reverse: bool distanceOverride: bool	✓
	execCommandFileLoad	arg1: <Parameterobject> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Executes "File load".	projectCommonVariables: bool projectCommonSwitches: bool sceneAtTimeOfSave: bool objectsStatesInSceneAtTimeOfSave: bool effectType: int *Agtk.constants.actionCommands.fileLoad.None, etc. duration300: int *For effectType, designate constant for Agtk.constants.actionCommands.fileLoad.	✓
	execCommandSoundPositionRemember	arg1: <Parameterobject> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Execute "Save Sound Playback Location".	soundType: int *Agtk.constants.actionCommands.soundPositionRemember.SoundSe, etc. variableObjectId: int variableQualifierId: int variableId: int	✓
	execCommandObjectUnlock	arg1: <Parameterobject> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Execute "Unlock".	objectType: int *Agtk.constants.actionCommands.objectUnlock.SetByObjectGroup, etc. objectGroup: int objectId: int *Specify the index for the object group configured in Object Group Administration for objectGroup. Object groups prepared by the system can be specified by Agtk.constants.objectGroup.ObjectGroupPlayer, Agtk.constants.objectGroup.ObjectGroupEnemy.	✓
	execCommandResourceSetChange	arg1: <Parameterobject> ret: Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Execute "Change Animation Resource Set".	*Values that can be specified for resourceSetId can be acquired with Agtk.animation.getResourceSetIdList().	✓
	execCommandDatabaseReflect	arg1: <Object Parameter> ret:Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext	Run the runtime action "Object ParameterApply to Database"	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 etc. reflectVariableId: int, rowIndexFromName: bool, rowNumberFromValue: bool, rowIndex: int, rowVariableObjectId: int, rowVariableQualifierId: int, rowVariableId: int	✓

Action Box "Other Runtime Actions" Processing Flow

* "Other Runtime Actions" will be referred to as "commands".

When an action box is switched, the command list of the action box being switched to is processed.

Basically, all commands in the list are processed within the same frame, but depending on the return value, the command processing flow can be changed.

a. Proceed with processing the next command. (Specify Agtk.constants.actionCommands.commandBehavior.CommandBehaviorNext as return value.)
b. Proceed with processing the next command, but re-execute this command on the next frame. (Specify Agtk.constants.actionCommands.commandBehavior.CommandBehaviorLoop as return value.)
   * Make sure the return values of c and d are not made meaningless when re-executed.
c. Delay the next command and re-execute this command on the next frame. (Specify Agtk.constants.actionCommands.commandBehavior.CommandBehaviorBlock as return value.)
d. Do not process next command onward. (Specify Agtk.constants.actionCommands.commandBehavior.CommandBehaviorBreak as return value.)

Most commands will fall under type A.
Type B is for "Push/Pull Object" commands. The push/pull action will continue until the action box switches. (If "Apply Only Once to Target" is unchecked.)
Type C commands insert a wait. The next command will be processed when the wait is over.
Type D commands are cases where object self is specified in "Execute Object Action". Subsequent commands will not be processed.

Execute Script is called during command list processing, so the processing flow can be controlled depending on the "Execute Script" return value.

Link Condition Determination

Namespace	Name	Parameter	Description	Notes (Properties that can be specified)	Player Implementation
Agtk.objectInstance	isWallTouched	arg1: <Parameter object> ret: <bool: Value>	Detects "Contact with Tile"s Wall Detection"	wallBit: int useTileGroup: bool tileGroup: int Specifies boolean value for direction of contact to be detected from the following. Agtk.constants.tile.WallTopBit Agtk.constants.tile.WallLeftBit Agtk.constants.tile.WallRightBit Agtk.constants.tile.WallBottomBit ＊tileGroup is used to specify a group assigned via the Tile Group Management. The default system tile group is Agtk.constants.tileGoup.Tile GroupDefault.	✓
	isWallAhead	arg1: <Parameter object> ret: <bool: Value>	Detects "Contact with Tile's Wall Detection When Moving One Tile"	wallBit: int useTileGroup: bool tileGroup: int	✓
	isObjectWallTouched	arg1: <Parameter object> ret: <bool: Value>	Detects Contact with Wall Detection of Other Objects	wallBit: int objectType: int *Agtk.constants.linkCondition.objectWallTouched.SetByObjectGroup, etc. objectGroup: int objectId: int *Specify either Agtk.constants.actionCommands.ObjectByType or Agtk.constants.actionCommands.ObjectById in objectType. *Specify one of Agtk.constants.objectType.ObjectTypeAll/ObjectTypePlayer/ ObjectTypeEnemy in objectTypeByType.	✓
	isObjectHit	arg1: <Parameter object> ret: <bool: Value>	Detects "Contact with Collision Detection of Other Objects"	wallBit: int objectType: int *Agtk.constants.linkCondition.objectHit.SetByObjectGroup, etc. objectGroup: int objectId: int	✓
	isAttackAreaTouched	arg1: <Parameter object> ret: <bool: Value>	Detects "Hit an Attack Detection"	wallBit: int objectType: int *Agtk.constants.linkCondition.attackAreaTouched.SetByObjectGroup, etc. objectGroup: int objectId: int attributeType: int *Agtk.constants.linkCondition.attackAreaTouched.AttributeNone, etc. attributePresetId: int attributeEqual: bool attributeValue: int	✓
	isAttackAreaNear	arg1: <Parameter object> ret: <bool: Value>	Detects "Distance with Attack Detection"	otherDirections: bool objectDirection: bool directionBit: int distanceType: int *Agtk.constants.linkCondition.attackAreaNear.DistanceNone, etc. distance: int objectType: int *Agtk.constants.linkCondition.attackAreaNear.SetByObjectGroup, etc. objectGroup: int objectId: int attributeType: int *Agtk.constants.linkCondition.attackAreaNear.AttributeNone, etc. attributePresetId: int *Agtk.constants.attackAttributes.Fire,etc. attributeEqual: bool attributeValue: int	✓
	isObjectNear	arg1: <Parameter object> ret: <bool: Value>	Detects "Distance with Other Objects"	otherDirections: bool objectDirection: bool directionBit: int distanceType: int *Agtk.constants.linkCondition.objectNear.DistanceNone, etc. distance: int objectType: int *Agtk.constants.linkCondition.objectNear.SetByObjectGroup, etc. objectGroup: int objectId: int	✓
	isObjectFacingEachOther	arg1: <Parameter object> ret: <bool: Value>	Detects "Face-to-Face with Other Objects"	objectType: int *Agtk.constants.linkCondition.objectFacingEachOther.SetByObjectGroup, etc. objectGroup: int objectId: int	✓
	isObjectFacing	arg1: <Parameter object> ret: <bool: Value>	Detects "Facing the Direction of Other Objects"	objectType: int *Agtk.constants.linkCondition.objectFacing.SetByObjectGroup, etc. objectGroup: int objectId: int	✓
	isObjectFound	arg1: <Parameter object> ret: <bool: Value>	Detects "Discovered Other Objects"	viewportId: int objectType: int *Agtk.constants.linkCondition.objectFound.SetByObjectGroup, etc. objectGroup: int objectId: int	✓
	isObjectFacingDirection	arg1: <Parameter object> ret: <bool: Value>	Detects "Other Objects Facing Specified Direction"	otherDirections: bool objectDirection: bool directionBit: int objectType: int *Agtk.constants.linkCondition.objectFacingDirection.SetByObjectGroup, etc. objectGroup: int objectId: int	✓
	isHpZero	arg1: <Parameter object> ret: <bool: Value>	Detects "HP is 0"	objectId: int	✓
	isCameraOutOfRange	arg1: <Parameter object> ret: <bool: Value>	Detects "Going Off Camera"	objectId: int distanceFlag: bool distance: int	✓
	isLocked	arg1: <Parameter object> ret: <bool: Value>	Detects "Locked"	lockingObjectId: int lockedObjectType: int *Agtk.constants.linkCondition.locked.SetByObjectGroup, etc. lockedObjectGroup: int lockedObjectId: int	✓
	isProbability	arg1: <Parameter object> ret: <bool: Value>	Detects "Use Probability"	probability: double	✓
	isSwitchVariableChanged	arg1: <Parameter object> ret: <bool: Value>	Detects "Switch/Variable Changes"	swtch: bool switchObjectId: int switchQualifierId: int switchId: int switchCondition: int *Agtk.constants.conditions.SwitchConditionOn, etc. variableObjectId: int variableQualifierId: int variableId: int compareVariableOperator: int *Agtk.constants.conditions.OperatorLess, etc. compareValueType: int *Agtk.constants.conditions.CompareValue, etc. compareValue: double compareVariableObjectId: int compareVariableQualifierId: int compareVariableId: int	✓
	isAnimationFinished	arg1: <Parameter object> ret: <bool: Value>	Detects "Finished Showing All Motion"	No parameters	✓
	isObjectActionChanged	arg1: <Parameter object> ret: <bool: Value>	Detects "Specified Object's Action Changes" *Action comparisons are to be done by action name.	objectId: int ＊ Agtk.constants.actionCommands.SelfObject, ChildObject, LockedObject, ParentObject can also be designated. actionObjectId: int ＊ objectId designates which object is being addressed for Agtk.constants.actionCommands.ChildObject, LockedObject, or ParentObject. actionId: int otherActions: bool actionId: int otherActions: bool	✓
	isJumpTop	arg1: <Parameter object> ret: <bool: Value>	Detects "Jump Peak Reached"	No parameters	✓
	isSlopeTouched	arg1: <Parameter object> ret: <bool: Value>	Detects "Contact with Slope"	directionType: int *Agtk.constants.linkCondition.slopeTouched.DirectionUpper, etc. downwardType: int *Agtk.constants.linkCondition.slopeTouched.DownwardLeft, etc.	✓
	isBuriedInWall	arg1: <Parameter object> ret: <bool: Value>	Detects "Embedded in Wall Detection"	objectId: int	✓

Script Constants

Namespace	Notes	Editor Implementation	Player Implementation
Agtk.constants.actionCommands	Refer to following file for details. <Player Directory>/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(Deprecated)			✓
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			✓

Plug-in info managed by editor

Project data is stored in plugins[ ].

Stored info:

id	Plug-in ID	
filename	Plug-in file name	Looks at file extension and determines whether the file is JavaScript or CoffeeScript. If CoffeeScript, a .js JavaScript file will be automatically generated.
enabled	Enabled Y/N	
parameters	Parameter info	[{id: <ID1>, name: <Name 1>, value: <Value 1>}, ⋯]

UI Parameters

Plug-ins can add numerical and string parameter input to the Editor UI.

Parameter Descriptions

Documented with JSON	[{id: <Parameter ID 1>, name: <Parameter Name 1>, type: String/Number/Json/ImageId/TextId/…/Custom, decimals: <Only used when type is Number. Number of Decimal Places>, minimumValue: <Only used when type is Number. Minimum Value>, maximumValue: <Only used when type is Number. Maximum Value>, customParam: [{id: <Custom ID1>, name: <Custom Name 1>}, ...], defaultValue: <Default Value 1>, …}]

id	Parameter ID	Integer that does not overlap with another within more than one plug-in
name	Parameter name	
type	Parameter classification	String/Number/Json/ImageId/TextId/…/Custom
defaultValue	Default value	

Specified for each classification

type=String　Text string can be input

type=MultiLineString　Text string including line breaks can be input

type=Number　Numerical value can be input. (Numerical values are stored as double-precision floating-point numbers.)

decimals	Number of decimal place	Default value: 0
minimumValue	Minimum value	Default: No limit
maximumValue	Maximum value	Default: No limit

type=Json　JSON text string can be input.

type=ImageId　Image ID can be selected.

type=TextId　Text ID can be selected.

type=SceneId　Scene ID can be selected.

type=TilesetId　Tileset ID can be selected.

type=AnimationId　Animation ID can be selected.

type=ObjectId　Object ID can be selected.

type=FontId　Font ID can be selected.

type=MovieId　Video ID can be selected.

type=BgmId　BGM ID can be selected.

type=SeId　SE ID can be selected.

type=VoiceId　Voice ID can be selected.

type=VariableId　Object variable ID can be selected.

referenceId: <Specify the Variable Id here>
withNewButton: true/false
※if the referenceId of the project is selected then IDs from the common project variables can be used.

type=SwitchId　Object switch ID can be selected.

referenceId: <Specify the Switch Id here>
withNewButton: true/false
※if the referenceId of the project is selected then IDs from the common project switches can be used.

type=AnimOnlyId　Animation-only ID can be selected.

type=PortalId　Portal ID can be selected.

type=CustomId　Can selected from a combo box.

customParam	Combo box definition	Required Documentation example: [{id: <Custom ID 1>, name: <Custom Name 1>}, ...]

type=Embedded　Embedded display of other parameters.

type=EmbeddedEditable　Enables editing of embedded display of other parameters.

Requires additional specification of sourceId: <Other resource parameter ID>, reference: <Other resource reference location>.

Can specify additional width: <Display width>, height: <Display height>.

sourceId: <Other resource parameter ID>	Presently only the following parameter IDs can be specified in <Other resource parameter ID>. If Embedded: Parameter ID where type is "ImageId"/"TextId"　If EmbeddedEditable: Parameter ID where type is "TextId"
reference: <Other resource reference location>	Not specified in "ImageId". Can specify fontId/text in "TextId".
width: <Display width>	Specifies display width for Embedded items
height: <Display height>	Specifies display height for Embedded items

type=SwitchVariableObjectId　Switch or variable Object ID can be selected.

Option	Add special Object ID designation. Designate strings including text string elements. (Only enabled for other runtime actions, other condition settings.)	
	“SelfObject”	Object self
	“LockedObject”	Locked Object
	“ParentObject”	Parent Object

type=DatabaseId　Database ID can be selected.

JSB API

cocos2d-x JSB API can be used.

You can add functions not provided by Pixel Game Maker MV by using JSB API.

Pixel Game Maker MV manages scenes and layers uniquely.

To display objects alongside unique Pixel Game Maker MV scenes and layers, retrieve and use the unique scene layer.

* See: Agtk.sceneInstances.getCurrent().getLayerByIndex()

JSB API call test

Open the appropriate project in the Editor.

In the Scenes tab, check if there is a scene. (If not, create a new Scene.)
Switch to the Objects tab, and select an Object from the Objects list. (If there are none, create a new one.)
Switch to the Action Program tab if you're not already on it. Select an Action Box. (If there are none, create a new one.) Click the + button for Other Runtime Actions.

Select "Execute Script" on the third page.

Paste the following into the "Execute Script" section.

```
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);
}
```

Click Quick Run, and a light green square with a white border will be displayed on the lower left of the preview screen.

* Make sure there are no tag values conflicts between plug-ins.

tag value　　(pluginId << 16) | <A 16 bit numerical value that can be freely used>

Scripts within Game Data

Can specify a script within the game data and execute it when the game is played.

Using Script Within a Tile

It is possible to run script from a tile via the use of a Gimmick tile (Gimmick Settings tab).

- Gimmick Tiles ＞ Gimmick Settings ＞ Change Switch/Variable Change variable: script

Tileset ID, tile X location, tile Y location, and index can be referenced. (tilesetId, tileX, tileY, index)

Ensure that the script evaluation produces a number.

Single equation example:

```
Math.random()
```

Example of using a function to return a value:

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

Using Script Within an Object

It is possible to run script within an Object in either an Action node or Link condition.

- Action ＞ Other runtime actions ＞ Change switch/variable Change variable: script

Can refer to object's own object ID or instance ID. (objectId, instanceId)

Can refer to its own action ID or command ID. (actionId, commandId)

*If isCommonAction is 1, then Common Action ID is entered into actionId.

Additional parameters:	isCommonAction: 0: Not Common Action, 1: Common Action
	sceneId: Placement scene ID if changeable after placement, −1 if not

Ensure that the script evaluation produces a number.

Single equation example:

```
Math.random()
```

Example of using a function to return a value:

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

- Action ＞ Other runtime actions ＞ Execute script

Can refer to object's own object ID or instance ID. (objectId, instanceId)

Can refer to its own action ID or command ID. (actionId, commandId)

*If isCommonAction is 1, then Common Action ID is entered into actionId.

Additional parameters:	isCommonAction: 0: Not Common Action, 1: Common Action
	sceneId: Placement scene ID if changeable after placement, -1 if not

* By explicitly specifying a return value, the behavior after "Execute Script" has been processed can be controlled. (See "Action Command" tab.) Example:

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

－ Action link ＞ Other Condition Settings ＞ Execute script

Can refer to object's own object ID or instance ID. (objectId, instanceId)

Can refer to its own action ID or command ID. (actionId, commandId)

*If commonActionLinkIndex is 0 or 1, then the linkId must contain the common action ID.

Additional parameters:

commonActionLinkIndex: 0 if incoming link of common action, 1 if outgoing link of common action, −1 if not common action.

As with "Change Variable/Switch", ensure that script evaluation results in a boolean value. Single equation example:

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

Example of using a function to return a value:

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

Using Script Within a Scene

- Others ＞ Course (Straight) ,Course (Curve),Course(Circle) ＞ Change Switch/Variable

Change variable: script

Can refer to its own sceneID, partsID, pointID or index.(sceneId, scenePartId, pointIndex, index)

Ensure that the script evaluation produces a number.

Using Script Within a Transitions

－ Screen Flow ＞ Transition link ＞ Sequence After Changeover ＞ Change Switch/Variable

Change variable: script

Can refer to its own transition link ID or index.(transitionLinkId, index)

Ensure that the script evaluation produces a number.

－ Portal Transfer ＞ Portal ＞ A→B Settings,B→A Settings

■Before Transfer *Settings When Touched by Player　＞　Change Switch/Variable

Change variable: script

Can refer to its own portalID, a/b ID or index.(portalId, abId, index)

Ensure that the script evaluation produces a number.

■After Transfer ＞　Change Switch/Variable

Change variable: script

Can refer to its own portalID, a/b ID or index.(portalId, abId, index)

Ensure that the script evaluation produces a number.