SLywnow Mini (SLM) Manual

Оглавление

SLM Commands

This script'll help you to create the scenario of events or dialogue. This is main script in this asset, all another can connect to him (but you can also use them separately).

Variable:

Textsc – SLM_RPGText

Imagesc - SLM_NovelImages

Audiosc – SLM_AudioMix

CharAI – SLM_CharAI

Stats – SLM_Stats script

Qte – SLM_QTEEvents

ALSLBridge – SLM_ALSLBridge

AddonManager - SLM_AddonManager

stopIfError – Stop current BOK if error

Script use Blocks of Commands (BOK) to work

Inside Block you has:

(hiden) Autorunnextcommands– automatically starts the command after the previous one is completed, unless the command contradicts this (if you want use SLM_Commands set it to true, use false only for debug) (disabling or changing some commands, like wait)

usePoints – use point system on code, WARNING, the block will take longer to load

useTextPoints – use point system on text, WARNING, the block will take longer to load

Commands (as List<string>) – commands of BOK

Texts (as List<string>) – texts or ALSL keys for BOK

Events (as List<UnityEvent>) – UnityEvents for BOK

Files – options to load files from the outside

CustomCommands – create a custom command

Some explanations:

- Commands run from 0 to end or special commands.
- Commands can read Texts and send it to SLM RPG Text
- Commands can Invoke Events
- Commands are executed in a single frame until execution is interrupted
- You must activate set BOK to run command list, use RunBlock(id)
- Script can read only one BOK at same time

Command format:

command:: attribute1:: attribute2::etc

Some commands have sub attributes, they are separated ||, example:

runpoll::text::var1||var2||etc::commandid1||commandid2||etc::time;;

attribute can be string, bool, float, int. It parse automatically

;; is optional, it displays the end of the line, recommended when loading commands from a file

if some attribute has (vw), then you can use ValueWork on this attribute.

You can add your commands if you want, see SLM Addons Manager.

Commands:

Commands that don't interrupt next command:

- skip – skip this command line and go next
- setpoint::name(string) – add some point, works like skip (work only if usePoints is true). Same rules for text (work only if useTextPoints is true).
- endpoint – set end of some point, if runpointasfunction is not running then works like skip
- endrepeat– set end of loop, if loop not running, then works like skip
- debug::text(string(vw)) - outputs a message to the console
- wait::secs(float(vw)) – waiting before run next command
- runevent::id(int) – run some event from Events
- addstat::statName(string)::value(only float or int(vw)) – add some value to stat in Stats
- subtractstat::statName(string)::value(only float or int(vw)) – subtract some value from stat in Stats
- setstat::statname(string)::value(any(vw))::type(string(options – int/float/bool/string)) – force set some value to stat in Stats (creates a new variable if such a name does not exist)
- setstatmin::statname(string)::value(any(vw)) - set stat min
- setstatmax::statname(string)::value(any(vw)) - set stat max
- setstatminmax::statname(string)::min(any(vw))::max(any(vw)) - set stat min and max
- setssa::ssakey(string)::statname(string)::type(string(options – int/float/bool/string)) – save some value in SSA from Stat in order not to lose it when restarting. You can get it back using Value Work. WARNING! The types of variables must be the same
- setssavalue::ssakey::value(any(vw)):: type(string(options – int/float/bool/string)) - save some value in SSA, you can use Value Work.
- showimage::layer(string)::id(int(vw)/string(vw)) – set image in layer in NovelImages, you can use sprite id or file name (without format) (you can use <next> for next image and <prev> for previous image)
- disablelayer::layer(string) – disable layer in NovelImages

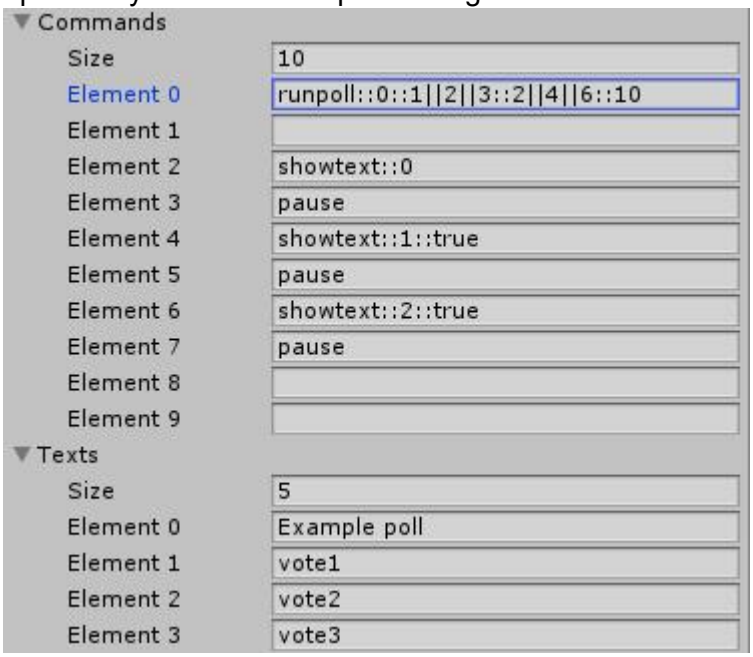
- showlayer::layer(string):: id(int=-1(vw)/string=””(vw)) – show layer in NovelImages, id = -1/”” current image, you can use sprite id or file name (without format) (you can use <next> for next image and <prev> for previous image)
- imagecolor::layer(string)::color(string(vw) - change color of image (reset when change image)
- imageanim::layer(string)::anim(int(vw)::duration(float (vw))(optional) - set anim type on layer (0 - none, 1 - replace, 2 - pingpong, 3+ - change only duration), duration = 0 - don’t change duration
- selectchar::name(string(vw)) – select char in charAI
- setcharemotion::emotion(string name) – changing emotion on your char (and show it) variables: nml = normal, agr = aggressive, emb = embarrassed, sad = sad, hap = happy. To select custom emotion use cus||id.
- setcharemotionbyid(int id) - changing emotion on your char by id. 0 = normal, 1 = aggressive, 2 = embarrassed, 3 = sad, 4 = happy, 5 = zero emotion in custom emotions
- deselectchars - deselect all chars in charAI
- setpollpreset::id(int) – select poll preset in RPGText
- settextpreset::id(int) – select text preset in RPGText
- audio::command::name::state(any) – many commands for SLM_AudioMix
    - audio::clip::name(string(vw))::id(int(vw)) – set clip in some audio block
    - audio::play::name(string(vw)) – start playing some audio block
    - audio::setplay::name(string(vw))::id(int(vw)) - set clip and start playing it in some audio block
    - audio::stop::name(string(vw)) - stop playing some audio block
    - audio::volume::name(string(vw))::state(float (0-1) (vw)) – set volume in some audio block
    - audio::mute::name(string(vw))::state(bool(vw)) – mude some audio block
    - audio::loop::name(string(vw))::state(bool(vw)) – loop some audio block
    - audio::pan::name(string(vw))::state(float(-1-1) (vw)) – set StereoPan in some audio block
    - audio::priority::name(string(vw))::state(int(0-255) (vw)) – set priority of some audio block
    - audio::pitch::name(string(vw))::state(float(-3-3)(vw)) – set pitch of some audio block
    - audio::mixer::name(string(vw))::id(int(vw) - set mixer in some audio block

- cc::customCommand(string) – run custom command
- ccc::text(string) – run your listener (see setup listeners)

Commands that interrupt next command:

- pause – stop running commands (you can continue by calling RunNextCommand()
- debugerror::text - debug to console some error text (with info about last command line) and invoke error listener
- exit – stop current BOK
- runblock::id(int) – stop current BOK and run another
- restart – restart current BOK
- runcommand::id(int(vw(command))) – run command on this BOK by id (and continues running from a new id)
- runpoint::name(string(vw)) – run some point (work only if usePoints is true, if false send error)

- runpointasfunction::name(string(vw)) — run some point as funtion (to endpoint) and then continue run commands from this line (you can use nested functions)
- repeat::count(int(vw)) – loop come part of code (to endloop) (you can use nested loops)
- showtext::id(int(vw))::offtext(bool=true) – show text using RPGText, sends text from Texts by id, use offtext (default false) to disable text panel after displaying the text. You can use <next> for next text line and <prev> for previous text line, example: showtext::<next>;;
- qte::id(int(vw))::win(int(vw(command)))::lose(int(vw(command))) – run QTE Event using Qte. Id – QTE preset id, win - command id if win, lose – command id if lose;
- runpoll::polltext(int(vw))::texts(int(id1||id2)(vw)::commands(int(id1||id2)(vw(command))::timer(float=-1(vw))::choice(int=-2(vw)) – run poll using RPGText, warning, count of texts and commands must be the same, all texts are taken from Texts by id. Set timer to -1 to unlimited poll. Set choice to -2 to default a standard choice in the case of the end of time, and -1 to random. You can skip timer and choice options if you want. Example of usage:



- rundpoll::polltext(int(vw))::texts(int(id1||id2...)(vw)::commands(int(id1||id2...)(vw(command))::available(bool(id1||id2...)(vw))::timer(float=-1(vw))::choice(int=-2(vw)) – like runpoll, but you can select what will be shown by bools
- runbpoll::polltext(int(vw))::texts(int(id1||id2...)(vw)::commands(int(id1||id2...)(vw(command))::available(bool(id1||id2...)(vw))::timer(float=-1(vw))::choice(int=-2(vw)) – like rundpoll, but disabled buttons will be shown as uninteractable
- isvalue::value(any(vw))::operation(string)::value(any(vw))::type(string)::true(int(vw(command)))::false(int(vw(command))) - checks the some value (special for ValueWork), in true/false insert command id, type has options int float bool string, operation has options == != > < >= <=, special options <> <=> - checks whether the value is within the values. Format: min(float/int(vw))||max(float/int(vw))
- isstat::statname(string)::operation(string)::value(any(vw))::type(string)::true(int(vw(command)))::false(int(vw(command))) – checks the some stat, in true/false insert command id, type has options any(auto get type) int float bool string, operation has options == != > < >= <=, special options <> <=> - checks whether the value is within the values. Format: min(float/int(vw))||max(float/int(vw))
- isstattostat::statname1(string)::operation(string)::statname2(string):: true(int(vw(command)))::false(int(vw(command))) – compares 2 stats, the types

of stats must be the same (except int and float), in true/false insert command id, operation has options == != > < >= <=

- switch::count(int)::statname(name)::type(string)::values(any(v1||v2...)(vw))::output (int(command(id1||id2...)(vw)))::dafault(int(command(vw)))(optional) — make a stat check==something for a lot of options, default in case nothing is true, if omitted, the first value is taken, type has options any(auto get type) int float bool string. If count = -1, then the number of values will be taken as the count
- random::commands(command(id1||id2||id3...)(vw))– call a random command from the list
- randomchance::(int(chance1||chance2…)(vw))::commands(command(id1||id2||id3...)(vw)) - call a random command from the list with chances, for example, if you set 1||1||1 then all commands has same chances, if you set 1||1||2, then last command will be called more often than others, you can use that with ValueWorks to manipulate chances for some events.

Create custom command:

You can add custom command if you need (for custom editor inside game for example)

Name – name of custom command. You will use it to run command, for example, if name of command is myFirstCommand, to run add in BOK cc::myFirstCommand

`RunNextCommand – running next command if true (works only if autorunnextcommand true)`

Type – what will Invoke when calling the command

EventId – id from Events (only if type is `useId or both`)

`CustomEvent – UnityEvent` (only if type is `useEvent or both`)

You can find it in Example folder.

ValueWork:

Value Work is system to work with input value, you can ignore it and input just value, but if you want use it, then replace value one of this commands (Warning, ValueWork use // not ||):

- vw//random//min//max – get random value in some range (only int, float and command)
- vw//randomvalues//count//value1//value2//… - return values from list, you must specify the number of values and then list them, for example vw//randomvalues//4//1//3//4//0 (any)
- vw//randomint//min//max – get random int value in some range (only float)
- vw//getstat//valuename//default(optional) – return some value from stat (any)
- vw//getstatwithinvalues// valuename//min//max – return some value from stat within min/max values (only int and float)
- vw//getstatround//valuename//round – retun some value from stat and round it (float only)

- vw//getstatmath//valuename//mathstring – calculate math string with value from stats using DataTable().Compute, for example you has Stat “ex” with value 2, you call vw//getstatmath//ex//(2*ex)+ex-3, in output you’ll have 3 (only float and int)
- vw//getssa//valuename//defaultvalue – return some value from SaveSystemAlt (any)
- vw//getssawithinvalues// valuename//min//max – return some value from SaveSystemAlt within min/max values (only int and float)
- vw//getssaround//valuename//round – retun some value from SaveSystemAlt and round it (float only)
- vw//getssamath//valuename//mathstring – calculate math string with value from SaveSystemAlt using DataTable().Compute, for example you has SaveSystemAlt key “ex” with value 2, you call vw//getssamath//ex//(2*ex)+ex-3, in output you’ll have 3 (only float and int)
- vw//getalsl//key – get key from ALSL or ALSLBridge (replaced words will be taken from the ALSL in any case) (only string)
- vw//getralsl//key – get replaced words from ALSL (only string)
- vw//getpoint//name//offset – return id of point with offset, if not found return -1, works only if usePoints is true (command, int)
- vw//randompoints//count//point1//point2//… - return id of random point from list, works like randomvalues but with points. Example: vw//randompoints//3//pointA//pointB//pointC (command)
- vw//gettextpoint//name//offset – return id of line with point int text with offset, if not found return -1, works only if useTextPoints is true (int). Return text line, if not found return “” (string). Use this to make the text editable as you work, without rewriting the values in the entire code. (int and string)
- vw//next//offset=0 – just return next command with some offset (default next command, like skip) (command, int)
- vw//getssainvert//valuename – invert bool value from SaveSystemAlt (bool)
- vw//getstatinvert//valuename – invert bool value from stat (bool)
- vw//checkstatint//statname//value - check value in stat (bool)
- vw//checkstatfloat//statname//value - check value in stat (bool)

Load file from the outside:

useLoadFromFiles – activate these options (without that you can only set up commands inside editor)

useTextObject – load commands from Object file, that disable useSLMContainer and useStreamingAssets (use it, if you want add scene to bundle or you want to add a file to the compilation) (only .txt or .json files)

useSLMContainer – activate SLM_Container for these options, use that if you want make custom levels in your game

useStreamingAssets – works only if useSLMContainer is false. Load file from streamingAssets (using BetterStreamingAssets) by path.

CommandObject – works only if useTextObject. Add here a .txt or .json file with commands

pathToCommands – works only if useSLMContainer is false. Load command file from some directory. You can use <docs> - for docs (using SLywnow), <dp> - for dataPath

pathToTexts – works only if useSLMContainer is false. Load text file from some directory. You can use <docs> - for docs (using SLywnow), <dp> - for dataPath

useALSL - works only if useSLMContainer is false. Activate ALSL for texts. Loads files based on the current language in ALSL. Format: Folder/**prefixALSLFile**language Name.lang Convenient for quickly creating translations. If custom translations are enabled, it automatically creates an Example.lang file at the path specified in the ALSL settings. If you add ALSLBridge, than it will be loaded through SLM_ALSLBridge.

prefixALSLFile - The prefix for the translation file, if not add, then the file must have the same name as the current language.

ALSLFolder - Required if useALSL is true! Folder with language files. The folder must be located in the ALSL folder in Streaming Assets and will automatically be created in the folder with custom translations.

container - SLM_Container, works only if useSLMContainer is true. Can read only single file and text type. container.blocks[0] must contain commands and container.blocks[1] must contain texts. If it has some error, then script use commands or/and texts from editor

You can load only commands or only texts if you want, just type inside pathToCommands or pathToTexts off (even if useSLMContainer true). If you use ALSL I recommended to you don't load texts from outside.

Commands is loading when BOK activating.

Voids:

RunBlock(int) – run BOK by id

StopBlock() – stop current BOK

RunNextCommand() - Run next command

RunCommand(int) – run command by id

RunPoint(string) - run point by name

Wait(float) – wait before run next command in sec

Error(string) - debug to console some error text (with info about last command line) and invoke error listener. Add it in Awake.

Setup listeners:

SLM_Commands has 2 listeners: onError and CustomCC

onError running when code has some error, CustomCC running when you your code has ccc (old system, use SLM Addon Manager)

Setup:

1. Create a function with one string argument, for example: public void error(string text)
2. On Awake (if you run block in Start) add this: commands.onError = error;
   where commands is SLM_Commands.
3. Configure CustomCC in the same way

4. Your function will called when the event occurred. You can change functions using customCommands for example. Warning! The line that you write after ccc will be passed to CustomCC (but don't use :: or ;; inside it)

Tips:

- Sometimes, when using external files for commands, the last part of the command is not read. This is because unity adds an unknown character to the end of strings (this is especially common with the `cc` command). I recommend simply adding `;;` to the end of the line, this will not affect the command's performance and will completely remove this bug.
- Although I added the ability to load texts with ALSL without using ALSLBridge, I still recommend using this script for more stable operation.
- Custom commands are very convenient for creating effects, just add them to event animation with the Play function
- Don't forget to check the ALSLDebugMode toggle in ALSLBridge when you test the code. If it is not enabled, there is a high chance that the code will not run in the editor. This is due to the peculiarities of ALSL, everything is fine if the game starts with a menu or some mini-scene, but if you try to call functions from Start in 0 scene, you will get an array error. (ALSLDebugMode toggle forces the ALSL to load a frame earlier, which solves this problem). However, it is then loaded again, which can cause severe fps drops, so this feature is automatically excluded when compiling

SLM Container

This script'll help you to load many files with different solutions in folders. For example, you can use it for Level editor, where you have many levels with many files in any folder.

To add SLM Container to your script just add `public SLM_Container container;`

After this you can see options on editor.

Options:

Mainpath – path to projects folder. Don't add `/` at the end. `<dp>` - Application.DataPath, `<docs>` - /Documents folder on PC and /sdcard on android. Example: `<docs>/SLywnow/levels`

C urrentproject – name of folder with current project, you must setup it by yourself.

Blocks – blocks with files.

Inside block you has:

Path – path to file inside project folder. Example: `texts/char1/dialogue1` Don't add `/` at the end.

Singlefile - Only need to load one file?

Single – single file options (if Singlefile true)

Multi – many files options (if Singlefile false)

Single options:

Filename – name of file (with format), Example: `file1.txt`

Type – type of file

Containers (loaded data is saved to these variables)

Containertext – array with any line of text (only if Type is text)

Containertextoneline – string with all text (only if Type is textoneline)

Containertexture – Texture2D (only if Type is texture)

Containersprite – Sprite (only if Type is sprite)

Multi options:

Filehasname – what to look for in files names. Set `.` for all. You can specify part of the name or format.

Type – type of files

Containers (loaded data is saved to these variables)

Containertext – List of array with any line of text (only if Type is text) (you can’t see it in editor)

Containertextoneline – List of string with all text (only if Type is textoneline)

Containertexture – List of Texture2D (only if Type is texture)

Containersprite – List of Sprite (only if Type is sprite)

Voids:

LoadBlock(int) – load containers of some block by id

LoadAllBlocks() – load containers in all blocks

ClearBlock(int) – clear containers in some block by id

ClearAllBlocks() – clear all containers in a

SLM QTE

This script'll help you to create the QTE Event. You can start it from your code or from SLM Commands.

Variables:

QteObject – object with qte gui

timerText – Text script to show how many time player has

timerSlider - Slider script to show how many time player has

spawnobj – button that spawn in find mode, all onClick Events will add automatically (only for find mode)

spawnArea – RectTransform, that parrent for spawnobj and min/max position for it (calculated by position and size) (WARNING! Use only both Center Anchors for this object)

progressSlider – slider to show progress (optional)

buttons – setup for buttons

presets – setup for presets

Buttons:

name – name of this option or name of option in InputManager or for mobileInput (see below)

useInput – use InputManager

key – if useInput is false, then specify the button

obj – activated some object when player need to press button

Presets:

Name - name og that qte

QteType – type of qte

- Basic – press buttons on screen to win
- Find – click on random spawned buttons to win, you need spawnobj and spawnArea setup
- Fast – fast press positive then negative buttons to win. See the different types on example scene.
- Concentration – you need to force the header (that change direction) to stay in the safe zone
- Reaction – You need to press button when header on safe zone

disableTimerVis - disable timer bar and timer text (if they exist)

disableProgressrVis – disable progress bar (if it exist)

buttonIds - Which buttons will be used in this stage of QTE. Format: int|int|int etc. Example: 1|0 or 3|1|10 (only for basic mode), you also can use random|count(int) to get random buttons, example: random|2

mode - Button reading mode. If savePress, the player must press the button once, without holding it down. If withhold, the player must hold down all the necessary buttons. If pressTogether, the player must simultaneously press all the necessary buttons. (only for basic mode)

time - How much time does the player have to go through all the stages of this QTE

useColors – use custom colors for buttons (basic and fast mode)

ColorPressed – color when button pressed (basic and fast mode)

NonPressed - color when button not pressed (basic and fast mode)

Missing - What to do if the player missed (only reserved buttons are read). None - nothing, fromstart - start QTE from scratch without resetting the time, oneback - go back one stage, fail - end QTE with a loss. (only for basic mode)

clickCount – how much buttons will spawn (only for find mode)

buttonPossitive – positive buttonId, id from buttons that used for fast and concentration mode (only for fast modeand concentration)

buttonNegative – negative buttonId, id from buttons that used for fast and concentration mode (only for fast and concentration mode)

addCount – how much score will be add for one press (1 = win) (only for fast mode)

removePerSec – how much score will be removed per sec (only for fast mode)

headerSpeed - speed that header will have by itself (concentration and reaction mode)

pressSpeed - speed that will be added to header when player press button (concentration mode)

safeSize - zone where you need to stay header (concentration mode)

rndTimeMin/rndTimeMax - min and max time to change direction (concentration mode)

ChanceToChangeDir - chance that direction of header will be changes when random time will end (concentration mode)

safeAreaColorConc - safe zone color (concentration mode)

failAreaColorConc - other zone color (concentration mode)

headerColorConc - color of header (concentration mode)

buttonReaction - button that you need to press in reaction mode (reaction mode)

reactionMinPos/reactionMaxPos - min/max position of safe zone in reaction (reaction mode)

reactionSafeZone - size of safe zone in reaction mode (reaction mode)

reactionFailMode - what happend if player press button outside of safe zone? (reaction mode)

- Nothing - header will move next
- Fail - player lose 1 life and zone will be regenerated

gamesCount - count of games before qte will win (reaction mode)

lifes - count of lifes that player has before lose qte (reaction mode)

safeAreaColorReact - safe zone color (reaction mode)

failAreaColorReact- other zone color (reaction mode)

headerColorReact - color of header (reaction mode)

Voids:

RunQTE(int id, SLM_Commands com, int win, int lose) – void for SLM_Commands, please, don't use it in your scripts, this may cause errors

RunQTE(int id, UnityEvent win, UnityEvent lose) – run QTE Event by id, you can specify which UnityEvents will be launched if player win or lose

RunQTE(string name, SLM_Commands com, int win, int lose) – void for SLM_Commands, please, don't use it in your scripts, this may cause errors

RunQTE(string name, UnityEvent win, UnityEvent lose) – run QTE Event by name, you can specify which UnityEvents will be launched if player win or lose

EndQTE(bool lose) – stop current QTE Event with a certain result

Press() - say script that button pressed,

Mobile input:

In order to reduce the required libraries, we abandoned Cross Platform Input and wrote our own system. Add the SLM_QTE_Button script to any object with Image so that it starts working as input for phones.

Qte - script of QTE

Key - name of button that you set in Buttons

onlyMobile - disable that input in Standalone

canHold - is player can hold that button?

frameSaveClick - how much frames QTE will see that button pressed (after click if canHold = false and when player remove finger from button if canHold = true)

SLM Stats

This script'll help you create the stats of a characters or game.

Variables:

GUIUpdate - when to update the GUI (off – whtn value changed by SLM_Stats's voids, everysecond - once in a certain period of time, onUpdate – every frame (totally not recommended))

Timer - period of time, if GUIUpdate is everysecond

Stats – List of stats blocks.

useALSL – add keys of stats to ALSL

Stat Block:

Name – name of stats (you can must call stat by name)

ALSLkey – key for ALSL and key for SLM_RPGText

Type – type of value in this stat

Value – value of this stat

Min – Min value. Only if Type if float or int

Max- Max value. Only if Type if float or int

Gui -gui options.

GUI options:

Type – off – disable visual element, slider – use slider to show stat (only int or float type), image – use fillAmount in Image script to show stat (only int or float type)

Slider - slider for show stat

Image – image to show stat

forBool – Toggle to show bool stat (select slider or image on Type to activate, work if Type of stat is bool)

TextType- off – disable text element, if you want to show bool or string just select any of other, curMaxHp – show current value and max with separator, curHp – show only current value, percent – show % of value

Separator – separator for curMaxHp TextType.

Prefix – any text before value

Suffix – any text after value

typeOfValue – how we need to show value (intType - only integers, floatType - with numbers after the decimal point)

round - to what sign should script round value (only of typeOfValue is floatType)

Text - Text object to show stat

Voids:

GetStatID(string) – return id of stat by name

GetStat(string) return SLM_Stats_Block by name

AddValue(string, value) – add some value to stat by name (only float and int)

SubtractValue(string, value) – subtract some value in stat by name (only float and int)

SetValue(name, value) – set some value to stat

GetValue(name, default value) – return value from stat, if stat not found or type error, than return default value.

SetValueMin(name, value) - setup value min

SetValueMax(name, value) - setup value max

SetValueMinMax(name, min, max) - setup value min and max

SLM RPG Text

This script'll help you to create text and poll interface.

Variables:

useALSL – use input text like ALSL key (and auto replace all replaces)

checkStats – check replaces for ALSL in stats without use ALSL, if useALSL false

stats - SLM_Stats (if checkStats true)

useHideButton – use button to hide text window

keyCode – use KeyCode to hide text window (if useHideButton true)

key – keyCode to hide text window (if keyCode true)

keyString – string in InputManager (if keyCode false)

audioSource – audio output for voices (if you don't need it, just don't choose anything)

textopt – options for text

autoSetUpZeroTextPreset – set textPresets[0] when start

textPresets – options of objects in textopt to change it ingame (optional)

poll – options for poll

autoSetUpZeroPollPreset – set pollpresets[0] when start

pollpresets – options like in poll to change fast change it ingame (optional)

Text options:

mainTextObj- GameObject which will be enabled when the text is called. It must contain Button with EndShowText() in UnityEvent. See PollExample scene for an example.

bg – background of text (optional)

text – Text object that where the input text will be shown

charlogo – Text object, that where the char name will be shown (only if CharAI script use it) (optional)

charIcon – Image object, that where the char icon will be shown (only if CharAI script use it) (optional)

charEmotionLayer – Image object, which is used for the layer of emotions in charIcon (optional)

auto – auto mode for dialogues

timer - how long should the text be displayed in auto mode (-1 without limits)

showMode – mode to show text (fullshow - show all text at once, animBySymbols - to display text character by character (does not work well with rich text), animByWords - show text word by word)

speedShow – show objects in sec (symbols or words) (only if animBySymbols or animByWords).

activestory – ativate story list

story – list with story of all input texts

runafterclick – what to do after showing text (only if text showing by your script, if text showing by SLM_Commands it will be run next command)

Poll opions:

mainPollObj - GameObject which will be enabled when the poll is called

text - The text on which the poll question will be displayed

timershow - if poll has timer, shown it in this slider (if you don’t need it, just don't choose anything)

timershowText – if poll has timer, displays how much time is left (if you don’t need it, just don't choose anything)

maxchoice - how many possible answers can there be (anything more than this number will not be counted)

RandomChoiceAfterEndOfTimmer – select random answer when the time is up

nonRandomChoice - what answer to choose if the time is over (if RandomChoiceAfterEndOfTimmer false)

blocks – answer objects to shown (if you want to place answer objects in advance) (the number of objects must be at least the number in maxchoice)

autogen – generate answer objects automatically

parent – parent of autogen objects

spawnobj – object to clone in autogen

SLM_RPGTextBlock:

To make answer object you need to add in button SLM_RPGTextBlock and add in UnityEvent Clicked() function

Options:

id - set automatically at poll startup

text – Text object to shown answer text

mainsc - SLM_RPGText, set automatically at poll startup

Voids:

ShowText(string text, SLM_Commands commsc, bool offafternext) - void for SLM_Commands, please, don’t use it in your scripts, this may cause errors

ShowText(string text, bool offafternext) – run showing text. Text- text or ALSL key to shown, offafternext – disable text object after displaying a text

EndShowText() – stop displaying a text. The script will call this automatically, we do not recommend using it

RunPoll(string text, string[] texts, int[] commands, SLM_Commands commsc, float time = -1,int defaultChoice = -2) - void for SLM_Commands, please, don't use it in your scripts, this may cause errors

RunPoll(string text, string[] texts, UnityEvent[] events, float time = -1, int defaultChoice=-2) - run showing poll, text - poll question, texts – answers on poll's buttons, events – events which will be launched after clicking the button (the number must be the same as texts), time- the time limit for answer (to -1 without any restrictions), defaultChoice - default selection, if the time ends (-2 as in the script options, -1 random)

SelectPoll(int id) – void for SLM_RPGTextBlock, please, don't use it.

ForseStopPoll() – forse disable current poll. Warning! If this void is called, no answer will be counted!

SetPreset(int id) – select poll preset, works only if poll is not currently active

SLM RPG Text Auto Key

This script'll help you to connect SaveSystemAlt to SLM_RPGText

Variables:

autoKey – bool type key, which configures the Auto parameter

autoKeyDefault – default Auto parameter

autoTimeoutKey – int type key, which configures the timer parameter

autoTimeoutKeyDefault default timer parameter

showModeKey – int type key (0-2 only), which configures the showMode parameter

showModeKeyDefault - default showMode parameter

showSpeedKey – float type key, which configures the speedShow parameter

showSpeedKeyDefault – default speedShow parameter

SLM Char AI

This script'll help to create chars for SLM_RPGText

Variables:

useALSL – use ALSL to replace Chars names

ALSLBridge – use strings from SLM_ALSLBridge to get Chars names (if useALSL only)

textsc - SLM_RPGText to connect

chars – list of char's options

Char options:

Name – name to call char and display name if useALSL false

ALSLkey – key to replace name if useALSL true

textColor – text color of this character

charIcon – Sprite with character to show it while the text is being displayed (optional)

voice – voice loop of this character (if you don't need it, just don't choose anything)

useEmotions – use emotions int this char. The emotions of a every character are saved. If no emotion was assigned, then the normal emotion is taken.

emotionType – type of blending emotion and main sprite. oneSprite – use single sprite for Icon and emotions; layer – use additional layer for emotions (you need a separate sprite with only the face)

normal/aggressive/embarrassed/sad/happy – default emotions

custom – custom emotions

currentEmotion – current id of emotion

Voids:

int GetCharId(string name) – return id of char by name

SelectChar(string name) – select some char

DeselectChars() - deselect all chars and back SLM_RPGText to default

SetEmotion(string name) – set emotion by name (normal / aggressive / embarrassed / sad / happy)

SetCustomEmotion(int id) – set emotion from custom

SetEmotionId(int id) – set currentEmotion itself;

SLM Novel Images

This script'll help you to create the layers of iamges/UI for noverls, dialogue or something else. You can start it from your code or from SLM Commands.

Variables:

Layers - list of layers

Layer options:

name – name of this layer

showAtStart – show this layer when script starts

group – Group to show (optional). It is intended for displaying UI elements that are not part of images, for example, Slider for SLM_Stats. You must use at least one display tool (group or imageobj).

imageobj – Image object to show layer (optional). You must use at least one display tool (group or imageobj)

animation – Type of animation. None - the sharp change of the image, Replace - Smooth image replacement, Pingpong - Smooth fading of the original image, then smooth appearance of a new one.

imageMask – Image object to work with animaton, you can use 1 mask to many layers, must be in the upper plan relative to imageobj (optional)

durationInSec - the duration of the animation in secs

sprites – List of sprites, which can be used in the layer

Voids:

SetImage(string layerName, int id) – select sprite in layer

SetImage(string layerName, string spriteName) – select sprite in layer using the file name instead of the id (without format). You can use <next> for next image and <prev> for previous image

DisableLayer(string layerName) – disable some layer

ShowLayer(string layerName, int id=-1) – show some layer. You can choose which image to show, or leave an already placed one.

ShowLayer(string layerName, string spriteName="") – show some layer. You can choose which image to show using the file name instead of the id (without format), or leave an already placed one. You can use <next> for next image and <prev> for previous image

GetLayer(string layerName) - get some block by name (return null if not exist)

Tips:

- Script can play many animations in same time
- Animation will be playing when a layer is disabled or enabled by voids

SLM AudioMix

This script'll help you manage your audio

Variables:

Blocks - list of audio blocks

Block options:

Name – name of this block

Audio – AudioSource of this block

Mixers - AudioMixerGroups that you can easily change

Clips – clips you can use in this block

useAudioVolume – use SLM_AudioVolumeSetter for volume (you must add AudioVolumeSetter to an object in Audio.

Voids:

SLM_AudioMix_Block GetBlock(string name) – get some block by name

GetID(string name) – get id of some block by name

StartPlay(string name) – start playing some block by name

StopPlay(string name) – stop playing some block by name

SelectClip(string name, int clipId) – select clip by id in some block by name

AddClip(string name,AudioClip clip)- add clip to some block by name

AddClipAndPlay(string name, AudioClip clip) – add clip and play it in some block by name

SetVolume(string name, float volume) – set volume in some block by name

SetPitch(string name, float pitch) – set pitch in some block by name

SetPriority(string name, int priority) – set priority in some block by name

SetStereoPan(string name, float stereopan) – set Stereo Pan in some block by name

SetLoop(string name, bool loop) – set loop option in some block by name

SetMute(string name, bool mute) – set mute option in some block by name

SetMixer(string name, int id) - set AudioMixerGroup in some block by name and id

DisableMixer(string name) - disable AudioMixerGroup in some block by name

SLM Audio Volume Setter

This script'll help you work with volume of AudioSource by combine 3 options

Variables:

SSAKey – key for SaveSystemAlt (I recomded use it instead of PlayerPerfs)

mainVolume – global volume of audio

optionVolume- volume from player's options (getting from SSAKey with default 1)

additionVolume – volume for animations or some effects, for example smooth disappearing

useUpdate – update AusioSource volume in Update (without this option only during the start of a scene)

updateOptionVolumeInUpdate – update optionVolume from SSAKey, only if useUpdate is true

Voids:

UpdateSSA() – update optionVolume from SSAKey (WARNING! It don't update volume in AudioSource)

UpdateVolume() - forcibly recalculate the volume

Tips:

- Script autodetect AudioSource and require it object with script
- Volume calculate by multiplying all parameters

SLM ALSL Bridge

This script'll help you load translated string without having to add them to ALSL. It has 2 modes of operation. The first is loading data via the ALSL system, i.e. via the StreamingAssets folder and creating Example.lang in Output files. The second option is to load all translation files from an external folder (for example, if you want to add translation capabilities to user levels). You must have a translation file with the default language.

Buttons:

Add all exist langs – auto set ups Custom Sets. Works only if UseCustomSetUp is true. Add all languages from ALSL and set TextFiles count to blocks count in Commands

Create new files – generate translation files from Commands' blocks and setupStrings. This won't replace any files. (if useCustomSetUp is true force create files with .txt format)

Update all files – like Create new files, but replace all files. (can't create files if useCustomSetUp is true, but still can update them)

Variables:

Commands - SLM_Commands to get blocks.

Setup – settings

setupStrings – keys and words which will be created when the buttons are clicked

Setup:

useCustomSetUp – use Object files for setUp (use it if you want add lang files to bundle) (only .txt or.json files (if you use ALSLFolders, than example file will create whth your format))

useALSLFolders – get files from ALSL Folders

GlobalFolder – global folder (<docs> - to get User's docs folder, <dp> - get Application.dataPath) (only if useALSLFolders false)

SceneProject – Folder for this scene. (recommended)

newLineSymbol - the string that indicates a new line in the file

commandsTextPrefix – prefix for commands' blocks files

stringsPrefix – prefix for strings file

format – format of files

dontUpdateStrings – dont update Strings files (only if you use Custom Sets)

dontUpdateTexts – dont update Text files (only if you use Custom Sets)

Custom Sets: - workds only if UseCustomSetUp is true. List with custom settings for langs. You must create setup with Default language

langName – name of language. Use Language name, not Display name

StringsFile – .txt or .json file with strings (you can use Update all files button to update it)

TextFiles – list with .txt or .json files for any blocks (you can use Update all files button to update it) (count must be same as blocks count in Commands)

Voids:

GetString(string key) – get string from strings by key

LoadStrings() – reload strings file

LoadBlock(int id) – used by SLM_Commands

SLM Addon Manager

This script'll help you to add some addons for SLM Commands.

You can easy migrate your ingame systems to SLM by addons.

All addons will be displayed in names

You need add to your script or create new script with it in Awake

```
AddAddon(string addonName, Addon add, List<string> commands, List<bool> runnextcommand)
```

addonName – name of your addon

add – your command processing function (more details below)

commands – commands you use (only if you need to stop block) (optional) (by default runnextcommand is true)

runnextcommand – bools that say SLM Addon Manager stop or not the code (optional) (WARNING, size of commands and runnextcommand must be match!)

example:

```
AM.AddAddon("test", Command, new List<string>() {"command"}, new List<bool>() { false});
```

*AM – SLM_AddonManager;

Now about function.

You need to create public bool Command(string command, string[] args) structure. You need return true if the command worked and false if none of the commands matched or had syntax errors.

Example of function that debug first argument:

```
public bool Command(string comm, string[] args)
        {
                if (comm == "mytestcommand")
                {
                        Debug.Log(args[0]);
                        return true;
                }

                return false;
        }
```

You can run it by mytestcommand::hello world!;; in commands

Also, I recommended use switch for more performance

You can use ValueWork if you add reference to SLM_Commands

Full example script looks like that:

```
public class MyAddon : MonoBehaviour
{
	public SLM_AddonManager AM;
	public SLM_Commands C;

	public void Awake()
	{
		//register our addon
		AM.AddAddon("My first Addon", Command, new List<string>()
{ "mytestcommand", "mysecondcommand" }, new List<bool>() { false, true });
	}

	public bool Command(string comm, string[] args)
	{
		//processing
		switch (comm)
		{
			case "mytestcommand":
				{
					Debug.Log(args[0]);
					return true;
				}
			case "mysecondcommand":
				{
					Debug.Log((C.ValueWorkInt(args[0]) +
C.ValueWorkInt(args[1])).ToString());
					return true;
				}
		}
```

```
                //if none of the commands matched
                return false;
        }
}
```

Voids:

AddAddon(string addonName, Addon add, List<string> commands, List<bool> runnextcommand) – add addon to SLM

Additional script:

You can use SLM_AddonBase as script class to easily create addons.

SLM_AddonBase:

commandScript - SLM_Command to connect

Void AddCommand (string command, bool runNextCommand) - add new command to addon

Void Initialize(string name, Command) - auto Initialize addon in Addon Manager, use it in Awake