

TABLE OF CONTENTS

Introduction	3
Quick summary of all available libraries	3
Base libraries	3
Mod libraries	3
Useful links	5
What exactly does mod do?	6
Quick Summary of the Mod	7
Lua tables	10
Rectangle	10
Vector	10
Key Emulator	10
Refs Controller	10
Block Info	10
Machine Info	11
Raycast Hit	11
Collider	11
Line Renderer (WIP)	11
Mod libraries	13
Rectangle	13
Graphical user interface	13
Vector	14
Machine	15
Input	15
Cursor	16
Physics	16
Players	16

Lines 16

Screen 16

Chat..... 16

LUA SCRIPTING MOD USER MANUAL

NOTE: ENGLISH IS NOT MY PRIMARY LANGUAGE. THIS MANUAL CAN CONTAIN A LOT OF SPELLING MISTAKES. I ALSO DON'T KNOW HOW TO WRITE USER MANUALS. THIS IS A VERY BAD MANUAL TO BE HONEST. BETTER SEE EXAMPLES PROVIDED IN STEAM WORKSHOP PAGE. ENJOY!

INTRODUCTION

This mod provides the ability to create and run Lua scripts in game without any compilation process with multiplayer support allowing you to create much more complex machines. Scripts can only control local player machine; they do not affect the level or other players machines in any way.

Big thanks to the authors of [UniLua](#).

Quick summary of all available libraries

BASE LIBRARIES

<code>_G</code>	Base module.
<code>package</code>	Loading other libraries, creating and using modules.
<code>coroutine</code>	Performing async tasks.
<code>table</code>	Functions to manipulate tables as arrays.
<code>io</code>	Removed.
<code>time</code>	Time (Game time, not OS time).
<code>string</code>	Functions to manipulate strings.
<code>bit32</code>	Bitwise operations.
<code>math</code>	Common math functions.
<code>debug</code>	Traceback.
<code>enc</code>	Encoding and decoding.

MOD LIBRARIES

<code>gui</code>	Creating in-game GUI based on <code>UnityEngine.GUI</code> .
------------------	--

rect	Creating rectangles for GUI based on UnityEngine.Rect.
vector	Functions to manipulate vectors up to 4 dimensions based on UnityEngine.Vector4.
machine	Functions to manipulate player local machine.
input	Reading player direct input: keyboard, mouse, joysticks.
cursor	Enabling and disabling cursor.
physics	Ray casting and sphere overlapping based on UnityEngine.Physics.
players	Info about current multiplayer session players.
lines	Drawing 3D lines.
screen	Game screen info.
chat	Handling chat messages. Writing chat messages.

USEFUL LINKS

- [Lua 5.2 Reference Manual.](#)
- [Unity – Manual: Order of execution for event functions.](#)
- [UnityEngine.GUI.](#)
- [Unity – Scripting API: Vector4.](#)
- [Unity – Scripting API: Rect.](#)
- [Unity – Scripting API: Input.](#)
- [Unity – Scripting API: Physics.](#)

WHAT EXACTLY DOES MOD DO?

- When you load a machine or create a new one mod will automatically load it's LuaRoot files if it has them. Otherwise, it will create a new LuaRoot inside this machine.
- LuaRoot is a folder where all Lua files and directories are stored. You can access this folder by pressing "Open LuaRoot Folder" in mod menu.
- When you run the simulation of your machine mod will run the script called "main.lua" and if everything started without critical errors then mod will save LuaRoot into machine.
- When you save your machine, this saved data will also be included into a save file. You also can save LuaRoot manually by pressing "Save LuaRoot manually".
- Machine script contains all base unity calls. There are short descriptions for all of them inside default script file.

QUICK SUMMARY OF THE MOD

- There are main functions: play, update, late_update, fixed_update and on_gui. Each has its own purpose. See [Unity Scripting API about it](#).
 - Play called on simulation start. Most ly useless.
 - Update called each frame update. Used to get [player input](#).
 - Late Update called after frame update.
 - Fixed Update called fixed times per second. Used for physics and so. Put all your cool code here.
 - On GUI called only to draw [GUI](#) on screen.

- You can emulate keys by creating a new [key emulator](#):
`local some_key_emulator = machine.new_key_emulator('c')`

Starting and stopping emulation by:

```
some_key_emulator.start()
```

```
some_key_emulator.stop()
```

Or just run it one time:

```
some_key_emulator.click()
```

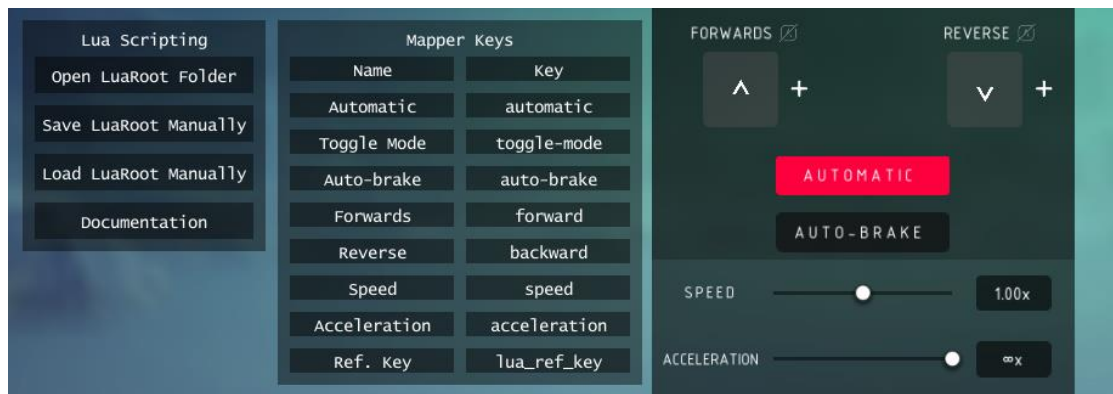
- You can change slider values by creating new [reference controller](#). Firstly, you need to assign reference keys to blocks in block mapper. Don't forget to enable automatic mode because you are just changing speed, not forcing wheel to spin.



Create [reference controller](#):

```
local rotor = machine.get_refs_control('rotor')
```

Find how your slider is called, you can find its name by both enabling mod menu (Ctrl+L) and opening block mapper:



Here we have “Speed” slider called “speed”. It is necessary because “Power” may be called “strength”.

So begin changing slider values somewhere:

```
rotor.set_slider('speed', 0.5)
```

- You can [set angle on steering wheels or hinges](#) like slider values:

```
local hinge = machine.get_refs_control('hinge')
hinge.set_steering(45)
```
- You can get block position by getting [block info](#) by reference key. [Block info](#) is given by [machine info](#), not just by [machine](#) because you can get machine info of other players machines.

```
local machine_info = machine.get_machine_info()
local starting_block = machine_info.get_block_info(0)
local position = starting_block.position()
```

Other player machine info:

```
for i = 0, players.count() do
    local other_player_mi = machine.get_machine_info(i)
    launch_rocket_at(other_player_mi.get_block_info(0).position())
end
```

- You can [cast rays](#):

```
local raycast_start = vector.add(starting_block.position(),
vector.multiply(starting_block.up(), -0.5))
local raycast_direction = vector.multiply(starting_block.up(), -1)
local raycast_hit = physics.raycast(raycast_start, raycast_direction)
```
- You can get all [colliders](#) in specified radius:

```
local colliders = physics.overlap_sphere(starting_block.position(), 5)
for i in pairs(colliders) do
    print(colliders[i].is_block)
end
```


- Draw lines:

```
local line = lines.new_line_renderer()  
line.set_points(vector.new(0, 0, 0), vector.new(10, 10, 10))
```

- Handle chat messages:

```
local function on_chat(sender, text)  
    print(sender .. ' just said ' .. text)  
end  
  
chat.add_listener(on_chat)
```

Write your own colored messages:

```
chat.set_visible(true)  
  
chat.write_local('<color=\"red\">Only you can see this!</color>')  
chat.write_team('<color=\"red\">Only your team can see this!</color>')  
chat.write_global('<color=\"green\">Hello, everyone!</color>')
```

- I'll be happy to add suggested features!

LUA TABLES

There is no such thing in this Lua interpreter used in mod as object-oriented programming, but there are tables! Tables allows to store values or even functions, so they are used as objects. However, remember: there are no references, only new tables every time (reference support WIP).

Rectangle

field	type
x	int
y	int
width	int
height	int

Vector

field	type
x	number
y	number
z	number
w	number

Key Emulator

field	type
start	function (no args; void)
stop	function (no args; void)
click	function (no args; void)
active	function (no args; returns boolean)

Refs Controller

field	type
set_slider	function (args: string mapper_key, number value; void)
set_steering	function (args: number angle; void)

Block Info

field	type
position	function (no args; returns table (vector))
forward	function (no args; returns table (vector))
right	function (no args; returns table (vector))
up	function (no args; returns table (vector))
rotation	function (no args; returns table (vector))
being_vacuumed	function (no args; returns boolean)
id	function (no args; returns int)
build_index	function (no args; returns int)
health	function (no args; returns number)
burning	function (no args; returns boolean)
flipped	function (no args; returns boolean)
frozen	function (no args; returns boolean)
in_wind	function (no args; returns boolean)
destroyed	function (no args; returns boolean)
zero_g	function (no args; returns boolean)
original_mass	function (no args; returns number)

scale	function (no args; returns table (vector))
velocity	function (no args; returns table (vector))
angular_velocity	function (no args; returns table (vector))

Machine Info

field	type
get_block_info (local machine only)	function (args: string ref_key, int index_of_all (optional); returns table (block info))
get_block_info (both local and another player`s machine)	function (args: int build_index (optional); returns table (block info))
block_count	function (no args; returns int)
cluster_count	function (no args; returns int)
center	function (no args; returns table (vector))
mass	function (no args; returns number)
middle	function (no args; returns table (vector))
name	function (no args; returns string)
player_id	function (no args; returns int)
position	function (no args; returns table (vector))
rotation	function (no args; returns table (vector))
velocity	function (no args; returns table (vector))
angular_velocity	function (no args; returns table (vector))
size	function (no args; returns table (vector))
unbreakable	function (no args; returns boolean)
infinite_ammo	function (no args; returns boolean)
is_dragging_blocks	function (no args; returns boolean)
team	function (no args; returns int)

Raycast Hit

field	type
distance	number
point	table (vector)
normal	table (vector)
is_block	boolean
get_block_info	function (no args; returns table (block info))

Collider

field	type
is_block	boolean
get_block_info	function (no args; returns table (block info))

Line Renderer (WIP)

field	type
set_points (multiple arguments possible, still don't figured out how to pass vector array as argument)	function (table (vector) start, table (vector) end; void)
set_width	function (number start_size, number end_size; void)
set_color	function (table (vector) color; void)

MOD LIBRARIES

Functions from UnityEngine were renamed in order to follow Lua style and make code cleaner and prettier. Basically: “GUI.DragWindow()” changes to “gui.drag_window()”.

Warning! Every time when a function returns table it creates a **new table, not returns a reference to a table, so don't get big tables such as machine info in update loop as these tables uses functions to get values, not stores values itself.**

Rectangle

rect

Used by GUI library.

function	arguments	return values
new	int x (optional), int y (optional), int width (optional), int height (optional)	table (rectangle)

Graphical user interface

gui

Mod GUI library is based on UnityEngine.GUI class except for some functions where arguments were changed. See Unity Scripting API about GUI in [Useful Links](#).

function	arguments	return values
label	table (rectangle) position, string text	
button	table (rectangle) position, string text	boolean
begin_group	table (rectangle) position	
begin_scroll_view	table (rectangle) position, table (vector) scroll_position, table (rectangle) view_rect	
box	table (rectangle) position, string text	
bring_window_to_front	int window_id	
bring_window_to_back	int window_id	
drag_window		
end_group		
end_scroll_view		
focus_control	string name	
focus_window	string name	
get_name_of_focused_control		string
horizontal_scrollbar	table (rectangle) position, number value, number size, number left_value, number right_value	number

horizontal_slider	table (rectangle) position, number value, number left_value, number right_value	number
modal_window (arguments are changed)	int id, table (rectangle) client_rect, string text, function (args: int window_id; void)	table (rectangle)
password_field	table (rectangle) position, string password, char mask	string
repeat_button	table (rectangle) position, string text	boolean
scroll_to	table (rectangle) position	
selection_grid	table (rectangle) position, int selected, table (string array) texts, int x_count	integer
set_next_control_name	string name	
text_area	table (rectangle) position, string text	string
text_field	table (rectangle) position, string text	string
unfocus_window		
vertical_scrollbar	table (rectangle) position, number value, number size, number top_value, number bottom_value	number
vertical_slider	table (rectangle) position, number value, number top_value, number bottom_value	number
window (arguments are changed)	int window_id, table (rectangle) client_rect, string title, function (args: int window_id; void)	table (rectangle)

Vector

vector

Basic Vector4 library based on UnityEngine.Vector4. See Unity Scripting API about Vector4 in [Useful Links](#).

function	arguments	return values
new	int x (optional), int y (optional), int z (optional), int w (optional)	table (vector)
distance	table (vector) a, table (vector) b	number
dot	table (vector) a, table (vector) b	number
lerp	table (vector) a, table (vector) b, number t	table (vector)
lerp_unclamped	table (vector) a, table (vector) b, number t	table (vector)
magnitude	table (vector) a	number
max	table (vector) lhs, table (vector) rhs	table (vector)
min	table (vector) lhs, table (vector) rhs	table (vector)

move_towards	table (vector) current, table (vector) target, number max_distance_delta	table (vector)
normalize	table (vector) a	table (vector)
project	table (vector) a, table (vector) b	table (vector) table (vector)
scale	table (vector) a, table (vector) b	table (vector)
add	table (vector) a, table (vector) b	table (vector)
subtract	table (vector) a, table (vector) b	table (vector)
negative	table (vector) a	table (vector)
multiply	table (vector) a, number b	table (vector)
equals	table (vector) a, table (vector) b	boolean
look_rotation	table (vector) a	table (vector)

Machine

machine

Full control of local active machine.

function	arguments	return values
new_key_emulator	string key_code	table (key emulator)
get_refs_control	string ref_key	table (refs controller)
get_machine_info		table (machine info)
get_machine_info	string nickname	table (machine info)
get_machine_info	int player_id	table (machine info)

Input

input

Direct input from player. See Unity Scripting API about Input in [Useful Links](#).

function	arguments	return values
mouse_screen_position		table (vector)
mouse_raycast_hit_point		table (vector)
get_axis	string axis	number
get_axis_raw	string axis	number
get_key	string key_code	boolean
get_key_down	string key_code	boolean
get_mouse_button	int mouse_button	boolean
get_mouse_button_down	int mouse_button	boolean
get_mouse_button_up	int mouse_button	boolean
any_key		boolean
any_key_down		boolean

Cursor

cursor

Sets cursor visible to false and cursor lock state to locked and back.

function	arguments	return values
set_state	boolean sate	

Physics

physics

Physics helper functions. See Unity Scripting API about Physics in [Useful Links](#).

function	arguments	return values
raycast	table (vector) origin, table (vector) direction	table (raycast hit)
overlap_sphere	table (vector) origin, number radius	array (collider)
gravity		table (vector)

Players

players

Players info.

function	arguments	return values
count		integer

Lines

lines

Drawing 3D lines (only for local client).

function	arguments	return values
new_line_renderer		table (line renderer)

Screen

screen

Screen info.

function	arguments	return values
width		number
height		number
fullscreen		boolean
dpi		number

Chat

chat

Handling chat messages.

function	arguments	return values
add_listener	function (args: string sender, string text; void)	
set_visible	boolean state	
write_local	string text	
write_team	string text	
write_global	string text	
clear		