

Mod Version: 1.1.3

---

## Other Languages

---

[Japanese \(日本語\)](#)

Thanks to [kudan](#) for the japanese translation!

## Lua Scripting Mod User Manual (EN)

---

### Table Of Contents

---

- [Introduction](#)
- [Useful Links](#)
- [What exactly does mod do?](#)
- [Quick Summary of the Mod](#)
- [Lua Tables](#)
  - [Rectangle](#)
  - [Vector](#)
  - [Key Emulator](#)
  - [Refs Controller](#)
  - [Block Info](#)
  - [Machine Info](#)
  - [Raycast Hit](#)
  - [Collider](#)
  - [Line Renderer \(WIP\)](#)
  - [Player Info](#)
- [Mod libraries](#)
  - [Rectangle](#)
  - [Graphical user interface](#)
  - [Vector](#)
  - [Machine](#)
  - [Input](#)
  - [Cursor](#)
  - [Physics](#)
  - [Players](#)
  - [Lines](#)
  - [Screen](#)
  - [Chat](#)

### 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](#).

### 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 any. Otherwise, it will create a new LuaRoot inside current machine.
- LuaRoot is a directory where all Lua files and folders are stored. You can access this directory by pressing `Open LuaRoot Folder` in mod menu.
- When you run the simulation mod will run the script `main.lua` and if everything started without critical errors mod will save LuaRoot into current machine.
- When you save your machine, LuaRoot 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 callbacks. There are short descriptions for all of them inside the default script file or down below.

### Quick Summary of the Mod

---

- There are 5 main callbacks: `play`, `update`, `late_update`, `fixed_update` and `on_gui`. Each has its own purpose. See [Unity Scripting API](#) about them.
  - `play` called on simulation start. Mostly useless, but can be used for adding chat listeners.
  - `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.
- Key emulation is provided by [key emulator](#), it can be started, stopped or just clicked:

```

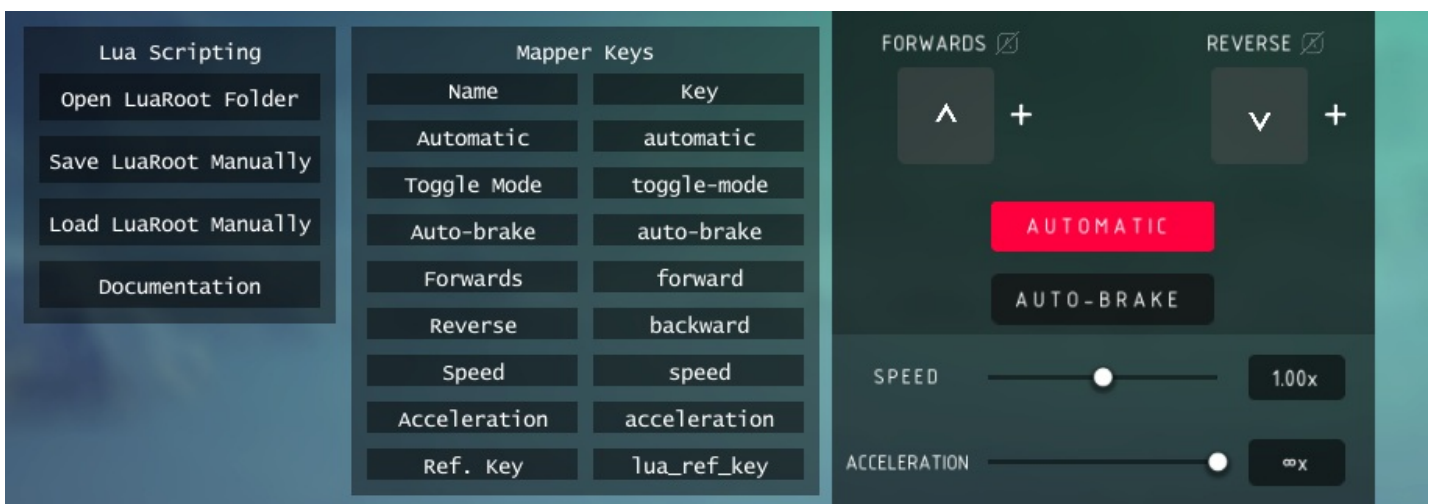
local some_key_emulator = machine.new_key_emulator('c')
some_key_emulator.start()
some_key_emulator.stop()
some_key_emulator.click()

```

- Slider values can be changed by creating a new [reference controller](#). Reference controller controls all blocks with same reference key.
1. Assign reference key to the block. In this case we are using wheels. So, don't forget to enable automatic mode as we are changing wheels speed, not forcing them to spin. Let's call them `rotor`.



2. Find out mapper type key name of the slider you are changing. In our case it is called `speed`.



3. Create a new [reference controller](#) and set the speed.

```

local rotor = machine.get_refs_control('rotor')
rotor.set_slider('speed', 0.5)

```

- Adjusting steering angles can be done by creating a new [reference controller](#) (or using existing) as shown above and setting desired angle.

```

local hinge = machine.get_refs_control('hinge')
hinge.set_steering(45)

```

- Block position, rotation, velocity and other information can be obtained by creating a new [block info](#).

```

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

```

It is also possible to get information about other players blocks and machines.

```

for i = 0, players.count() - 1 do
    local player = players.get(i)
    if not player.is_local_player() and player.is_simulating() then
        local enemy_position = players.get(i).get_machine_info().get_block_info(0).position()
        print('enemy at height ' .. enemy_position.y)
    end
end
end

```

- Raycasting may be done using [physics](#) .

```

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)

```

- Near colliders can be found by using sphere overlapping using [physics](#) .

```

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

```

- Debug lines can be drawn using [lines](#) library.

```

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

```

- Chat messages can be handled by creating a new [chat listener](#) . Don't forget that you must add listener only after declaring callback function.

```

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

chat.add_listener(on_chat)

```

Write your own chat messages with rich text without any nickname prefix.

```

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 very happy to add new 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

Created by [rectangle library](#) using `rect.new(...)`

field	type
<b>x</b>	<code>int</code>
<b>y</b>	<code>int</code>
<b>width</b>	<code>int</code>
<b>height</b>	<code>int</code>

### Vector

Created by [vector library](#) using `vector.new(...)`

field	type
x	number
y	number
z	number
w	number

Key Emulator

Created by [machine](#) library using `machine.new_key_emulator(...)`.

field	type
start	<i>function</i> (no args; void)
stop	<i>function</i> (no args; void)
click	<i>function</i> (no args; void)
active	<i>function</i> (no args; returns boolean)

Refs Controller

Created by [machine](#) library using `machine.new_refs_control(...)`.

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

Block Info

Created by [machine info](#) using `machine_info.get_block_info(...)`.

field	type
position	<i>function</i> (no args; returns <a href="#">vector</a> )
forward	<i>function</i> (no args; returns <a href="#">vector</a> )
right	<i>function</i> (no args; returns <a href="#">vector</a> )
up	<i>function</i> (no args; returns <a href="#">vector</a> )
rotation	<i>function</i> (no args; returns <a href="#">vector</a> )
being_vacuumed	<i>function</i> (no args; returns boolean)
id	<i>function</i> (no args; returns int)
build_index	<i>function</i> (no args; returns int)
health	<i>function</i> (no args; returns number)
burning	<i>function</i> (no args; returns boolean)
flipped	<i>function</i> (no args; returns boolean)

field	type
frozen	<i>function</i> (no args; returns <code>boolean</code> )
in_wind	<i>function</i> (no args; returns <code>boolean</code> )
destroyed	<i>function</i> (no args; returns <code>boolean</code> )
zero_g	<i>function</i> (no args; returns <code>boolean</code> )
original_mass	<i>function</i> (no args; returns <code>number</code> )
scale	<i>function</i> (no args; returns <code>vector</code> )
velocity	<i>function</i> (no args; returns <code>vector</code> )
angular_velocity	<i>function</i> (no args; returns <code>vector</code> )

## Machine Info

Created by [machine](#) using `machine.get_machine_info(...)`.

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

field	type
is_simulating	<i>function</i> (no args; returns <code>boolean</code> )

### Raycast Hit

Created by `physics` using `physics.raycast(...)`.

field	type
distance	<code>number</code>
point	<code>vector</code>
normal	<code>vector</code>
is_block	<code>boolean</code>
get_block_info	<i>function</i> (no args; returns <code>block info</code> )

### Collider

Created by `physics` using `physics.overlap_sphere(...)`.

field	type
is_block	<code>boolean</code>
get_block_info	<i>function</i> (no args; returns <code>block info</code> )

### Line Renderer (WIP)

Created by `lines` library using `lines.new_line_renderer()`.

field	type
set_points	<i>function</i> ( <code>vector</code> start, <code>vector</code> end; <code>void</code> )
set_width	<i>function</i> ( <code>number</code> start_size, <code>number</code> end_size; <code>void</code> )
set_color	<i>function</i> ( <code>vector</code> color; <code>void</code> )

### Player Info

Created by `players` library using `players.get(...)` and other.

field	type
in_local_sim	<i>function</i> (no args; returns <code>boolean</code> )
is_host	<i>function</i> (no args; <code>boolean</code> )
is_local_player	<i>function</i> (no args; returns <code>boolean</code> )
is_spectator	<i>function</i> (no args; returns <code>boolean</code> )
is_simulating	<i>function</i> (no args; returns <code>boolean</code> )
name	<i>function</i> (no args; returns <code>string</code> )
id	<i>function</i> (no args; returns <code>int</code> )

field	type
team	<i>function</i> (no args; returns <code>id</code> )
get_machine_info	<i>function</i> (no args; returns <code>machine info</code> )

## Mod libraries

Warning! Every time when a function returns table it creates a **new** table, not returns a reference to a table. It means that you shouldn't create big tables such as `block info` in `fixed_update` loop. Instead, create `block info` at the top of script or in the `play` callback. However, it isn't necessary, just be careful.

### Rectangle

`rect`

Used by GUI library.

function	arguments	return values
<code>new</code>	<code>int x (optional)</code> , <code>int y (optional)</code> , <code>int width (optional)</code> , <code>int height (optional)</code>	<code>rectangle</code>

### Graphical user interface

`gui`

GUI library 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
<code>world_to_screen_point</code>	<code>vector</code> world_position	<code>vector</code>
<code>label</code>	<code>rectangle</code> position, <code>string</code> text	
<code>button</code>	<code>rectangle</code> position, <code>string</code> text	<code>boolean</code>
<code>toggle</code>	<code>rectangle</code> position, <code>boolean</code> value, <code>string</code> text	<code>boolean</code>
<code>begin_group</code>	<code>rectangle</code> position	
<code>begin_scroll_view</code>	<code>rectangle</code> position, <code>vector</code> scroll_position, <code>rectangle</code> view_rect	
<code>box</code>	<code>rectangle</code> position, <code>string</code> text	
<code>bring_window_to_front</code>	<code>int</code> window_id	
<code>bring_window_to_back</code>	<code>int</code> window_id	
<code>drag_window</code>		
<code>end_group</code>		
<code>end_scroll_view</code>		
<code>focus_control</code>	<code>string</code> name	
<code>focus_window</code>	<code>string</code> name	
<code>get_name_of_focused_control</code>		<code>string</code>
<code>horizontal_scrollbar</code>	<code>rectangle</code> position, <code>number</code> value, <code>number</code> size, <code>number</code> left_value, <code>number</code> right_value	<code>number</code>
<code>horizontal_slider</code>	<code>rectangle</code> position, <code>number</code> value, <code>number</code> left_value, <code>number</code> right_value	<code>number</code>

function	arguments	return values
<b>modal_window</b> ( <i>arguments were changed</i> )	int id, <a href="#">rectangle</a> client_rect, string text, <i>function</i> (args: int window_id; void)	<a href="#">rectangle</a>
<b>password_field</b>	<a href="#">rectangle</a> position, string password, char mask	string
<b>repeat_button</b>	<a href="#">rectangle</a> position, string text	boolean
<b>scroll_to</b>	<a href="#">rectangle</a> position	
<b>selection_grid</b>	<a href="#">rectangle</a> position, int selected, string array texts, int x_count	int
<b>set_next_control_name</b>	string name	
<b>text_area</b>	<a href="#">rectangle</a> position, string text	string
<b>text_field</b>	<a href="#">rectangle</a> position, string text	string
<b>unfocus_window</b>		
<b>vertical_scrollbar</b>	<a href="#">rectangle</a> position, number value, number size, number top_value, number bottom_value	number
<b>vertical_slider</b>	<a href="#">rectangle</a> position, number value, number top_value, number bottom_value	number
<b>window</b> ( <i>arguments are changed</i> )	int window_id, <a href="#">rectangle</a> client_rect, string title, <i>function</i> (args: int window_id; void)	<a href="#">rectangle</a>

## Vector

[vector](#)

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

function	arguments	return values
<b>new</b>	number w ( <i>optional</i> ), number y ( <i>optional</i> ), number z ( <i>optional</i> ), number w ( <i>optional</i> )	<a href="#">vector</a>
<b>distance</b>	<a href="#">vector</a> a, <a href="#">vector</a> b	number
<b>dot</b>	<a href="#">vector</a> a, <a href="#">vector</a> b	number
<b>lerp</b>	<a href="#">vector</a> a, <a href="#">vector</a> b, number t	<a href="#">vector</a>
<b>lerp_unclamped</b>	<a href="#">vector</a> a, <a href="#">vector</a> b, number t	<a href="#">vector</a>
<b>magnitude</b>	<a href="#">vector</a> a	number
<b>max</b>	<a href="#">vector</a> lhs, <a href="#">vector</a> rhs	<a href="#">vector</a>
<b>min</b>	<a href="#">vector</a> lhs, <a href="#">vector</a> rhs	<a href="#">vector</a>
<b>move_towards</b>	<a href="#">vector</a> current, <a href="#">vector</a> target, number max_distance_delta	<a href="#">vector</a>
<b>normalize</b>	<a href="#">vector</a> a	<a href="#">vector</a>
<b>project</b>	<a href="#">vector</a> a, <a href="#">vector</a> b	<a href="#">vector</a>
<b>scale</b>	<a href="#">vector</a> a, <a href="#">vector</a> b	<a href="#">vector</a>
<b>add</b>	<a href="#">vector</a> a, <a href="#">vector</a> b	<a href="#">vector</a>
<b>subtract</b>	<a href="#">vector</a> a, <a href="#">vector</a> b	<a href="#">vector</a>
<b>negative</b>	<a href="#">vector</a> a	<a href="#">vector</a>



function	arguments	return values
multiply	<code>vector</code> a, number b	<code>vector</code>
equals	<code>vector</code> a, <code>vector</code> b	boolean
look_rotation	<code>vector</code> a	<code>vector</code>

Machine

machine

Machine library for key emulation, steering and sliders controlling, machine info.

function	arguments	return values
new_key_emulator	string key_code	<code>key emulator</code> )
get_refs_control	string ref_key	<code>refs controller</code> )
get_machine_info		<code>machine info</code> )

Input

input

Library for handling direct input from keyboard, mouse, joysticks and other. See Unity Scripting API about Input in [Useful Links](#).

function	arguments	return values
mouse_screen_position		<code>vector</code>
mouse_raycast_hit_point		<code>vector</code>
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

Library for controlling mouse cursor.

function	arguments	return values
set_state	boolean state	

Physics

physics

Library for obtaining information using physics. See Unity Scripting API about Physics in [Useful Links](#).

function	arguments	return values
raycast	<code>vector</code> origin, <code>vector</code> direction	<code>raycast hit</code>
overlap_sphere	<code>vector</code> origin, number radius	<code>collider</code> array
gravity		<code>vector</code>

Players

players

Library for getting information about multiplayer session.

function	arguments	return values
count		<code>int</code>
get	<code>int</code> player_index	<code>player info</code>
by_id	<code>int</code> network_id	<code>player info</code>
get_all		<code>player info</code> array

Lines

lines

Library for drawing 3D debug lines (only for local client).

function	arguments	return values
new_line_renderer		<code>line renderer</code>

Screen

screen

Library for getting screen information.

function	arguments	return values
width		<code>number</code>
height		<code>number</code>
fullscreen		<code>boolean</code>
dpi		<code>number</code>

Chat

chat

Library for handling and writing chat messages.

function	arguments	return values
add_listener	<i>function</i> (args: <code>string</code> sender, <code>string</code> text; <code>void</code> )	
set_visible	<code>boolean</code> state	
write_local	<code>string</code> text	

function	arguments	return values
write_team	string text	
write_global	string text	
clear		

Thanks for reading!