

# Lua Scripting Mod ユーザーマニュアル (JP)

## Table Of Contents

- [導入](#)
- [便利なリンク](#)
- [このModでできること](#)
- [このModの概要](#)
- [Luaのテーブル](#)
  - [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](#)

## 導入

このmodは、コンパイルの手順を踏まずにゲーム内でLuaスクリプトを作成して実行する機能を提供し、マルチプレイにも対応することで、より複雑なマシンを作成することを補助します。スクリプトはローカルプレイヤーのマシンのみ制御でき、レベルオブジェクトやほかのプレイヤーのマシンには一切影響を与えません。

UniLuaの著者さんありがとう！

## 便利なリンク

- [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.](#)

## このModでできること

- マシンをロードしたり、新規作成したりすると、MODはそのマシンのLuaRootファイルを探し、あれば自動的にロードします。無い場合は、現在のマシンに新しいLuaRootを作成します。
- LuaRootは全てのLuaファイルやフォルダが格納されているディレクトリです。Ctrl+Lを押すと表示されるModメニューの Open LuaRoot ボタンでこのフォルダを開けます。
- マシンを再生すると、Modはmain.lua というスクリプトを実行し、致命的なエラーが無ければ、ModはLuaRootをマシンに保存します。
- マシンを保存すると、Luaのセーブも行われます。 Save LuaRoot manually ボタンでも保存されます。
- Machineのスクリプトには全てのunityのコールバックが含まれます。デフォルトのLuaスクリプトファイルには全てのコールバックについて簡単な説明書きが書いてあります。

## このModの概要

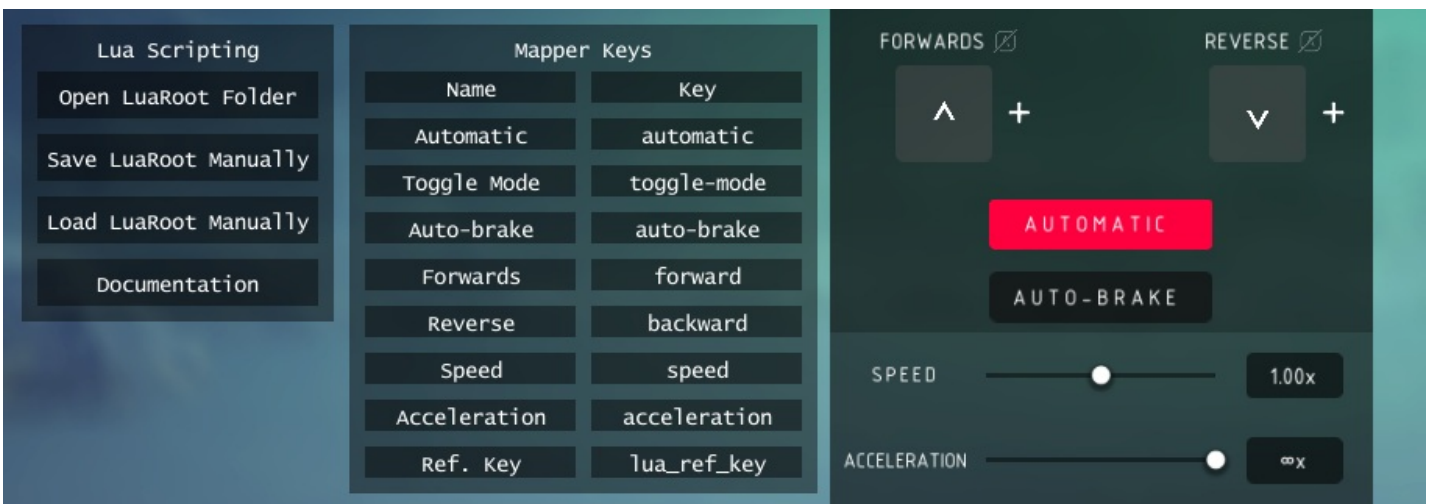
- 主な機能として、play、update、late\_update、fixed\_update、on\_guiがあります。これらに関しては[Unity Scripting API](#) を参照してください。
  - play はシミュレーション開始時に読み込まれます。ほとんど役に立ちません。
  - update はフレーム更新のたびに読み込まれます。 [プレイヤーの入力の検出](#) に用います。
  - late\_update はフレーム更新後に呼び出されます。
  - fixed\_update は1秒ごとに一定回数 ( 100Hz ) 呼び出されます。物理演算などに用います。
  - on\_gui は、画面上にGUIを描画するためだけに呼び出されます。
- キーのエミュレートは [key emulator](#)によって行います。起動、停止、クリックを行うことができます。

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

- パーツ設定のそれぞれのスライダーの値は、新しい **reference controller** を作成することで変更できます。reference controllerは、同じ reference key が与えられたパーツには同じ制御を行います。
1. ブロックに reference key を割り当てます。ここでは車輪を用います。車輪を強制的に回転させるのではなく、速度を変えているので、自動をオンにするのを忘れないようにしましょう。この画像の場合、reference key は rotor と名付けました。



2. 変更したいスライダーのmapper typeを調べます。ctrl+L を押した状態でパーツ設定を開くと確認できます。ここでは speed のスライダーの値を変更することにします。



3. 新しい **reference controller** を作り、速度を設定します。

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

- ステアリングヒンジ等の操舵角の調整は、**reference controller** を作成し、**set\_steering** を用い、以下のように行えます。

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

- ブロックの位置や回転、速度などの情報は、新たに **block info** を作成することで取得できます。

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

- また、他のプレイヤーのブロックやマシンの情報を得ることも可能です。

```

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

```

- レイキャスト（センサーみたいなこと）は [physics](#) を用いて行えます。

```

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)

```

- 指定された半径内のコライダー情報の取得なんかも [physics](#) でできます。

```

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

```

- 線の描画は [lines](#) を使って行います。

```

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

```

- チャットメッセージは、新しい [chat listener](#) を作成することで処理できます。コールバック関数を宣言した後にリスナーを追加しなければならないことを忘れずに。

```

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

chat.add_listener(on_chat)

```

頭にニックネームをつけずにリッチテキストでチャットメッセージを書く事ができます。

```

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>')

```

- 新しい機能もドシドシ募集中！

## Luaのテーブル

このModで使われているLuaインタプリタにはオブジェクト指向プログラミングのようなものではありませんが、テーブル型はあります！テーブル型は値を格納したり、関数を格納したりできます。但し参照は無く、毎回新しいテーブルが生成されるだけなので注意してください（参照はいずれ導入するかも）。

### Rectangle

矩形

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

ベクトル

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

field	type
x	number
y	number
z	number
w	number

Key Emulator

キーのエミュレーター

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

リファレンスコントローラー

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

ブロックの情報

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 )

field	type
build_index	<i>function</i> (no args; returns <code>int</code> )
health	<i>function</i> (no args; returns <code>number</code> )
burning	<i>function</i> (no args; returns <code>boolean</code> )
flipped	<i>function</i> (no args; returns <code>boolean</code> )
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

マシンの情報

Created by `machine` using `machine.get_machine_info(...)`.

field	type
<code>get_block_info</code> ( <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> )
<code>get_block_info</code> ( <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> )

field	type
-------	------

size	<i>function</i> (no args; returns <a href="#">vector</a> )
unbreakable	<i>function</i> (no args; returns <a href="#">boolean</a> )
infinite_ammo	<i>function</i> (no args; returns <a href="#">boolean</a> )
is_dragging_blocks	<i>function</i> (no args; returns <a href="#">boolean</a> )
team	<i>function</i> (no args; returns <a href="#">int</a> )
is_simulating	<i>function</i> (no args; returns <a href="#">boolean</a> )

### Raycast Hit

レイキャストのヒット判定

Created by [physics](#) using `physics.raycast(...)` .

field	type
distance	<a href="#">number</a>
point	<a href="#">vector</a>
normal	<a href="#">vector</a>
is_block	<a href="#">boolean</a>
get_block_info	<i>function</i> (no args; returns <a href="#">block info</a> )

### Collider

コライダー

Created by [physics](#) using `physics.overlap_sphere(...)` .

field	type
is_block	<a href="#">boolean</a>
get_block_info	<i>function</i> (no args; returns <a href="#">block info</a> )

### Line Renderer (WIP)

線の描画

Created by [lines](#) library using `lines.new_line_renderer()` .

field	type
set_points	<i>function</i> ( <a href="#">vector</a> start, <a href="#">vector</a> end; <a href="#">void</a> )
set_width	<i>function</i> ( <a href="#">number</a> start_size, <a href="#">number</a> end_size; <a href="#">void</a> )
set_color	<i>function</i> ( <a href="#">vector</a> color; <a href="#">void</a> )

### 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> )
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

注意！関数がテーブルを返すたびに、そのテーブルに参照を返すのではなく、**新しい**テーブルを作成します。つまり、`fixed_update` ループの中では、`block_info` のような大きなテーブルは作ってはいけません。代わりに、スクリプトの先頭や `play` コールバックで `block info` を作成するようにしてください。

### Rectangle

矩形

`rect`

GUIで使います。

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

### Graphical user interface

`gui`

GUIライブラリは `UnityEngine.GUI` クラスをベースにしていますが、引数が変更されたクラスもあります。詳しくは[Useful Links](#)のUnity Scripting APIを参照してください。

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	

function	arguments	return values
drag_window		
end_group		
end_scroll_view		
focus_control	string name	
focus_window	string name	
get_name_of_focused_control		string
horizontal_scrollbar	rectangle position, number value, number size, number left_value, number right_value	number
horizontal_slider	rectangle position, number value, number left_value, number right_value	number
modal_window) ( <i>arguments are changed</i> )	int id, rectangle client_rect, string text, <i>function</i> (args: int window_id; void)	rectangle
password_field	rectangle position, string password, char mask	string
repeat_button	rectangle position, string text	boolean
scroll_to	rectangle position	
selection_grid	rectangle position, int selected, string array texts, int x_count	int
set_next_control_name	string name	
text_area	rectangle position, string text	string
text_field	rectangle position, string text	string
unfocus_window		
vertical_scrollbar	rectangle position, number value, number size, number top_value, number bottom_value	number
vertical_slider	rectangle position, number value, number top_value, number bottom_value	number
window ( <i>arguments are changed</i> )	int window_id, rectangle client_rect, string title, <i>function</i> (args: int window_id; void)	rectangle

## Vector

ベクトル

vector

Vector4ライブラリは `UnityEngine.Vector4` をベースにしています。詳しくは[Useful Links](#)のUnity Scripting APIを参照してください。

function	arguments	return values
new	number x ( <i>optional</i> ), number y ( <i>optional</i> ), number z ( <i>optional</i> ), number w ( <i>optional</i> )	vector
distance	vector a, vector b	number
dot	vector a, vector b	number
lerp	vector a, vector b, number t	vector
lerp_unclamped	vector a, vector b, number t	vector
magnitude	vector a	number



function	arguments	return values
max	<code>vector</code> lhs, <code>vector</code> rhs	<code>vector</code>
min	<code>vector</code> lhs, <code>vector</code> rhs	<code>vector</code>
move_towards	<code>vector</code> current, <code>vector</code> target, <code>number</code> max_distance_delta	<code>vector</code>
normalize	<code>vector</code> a	<code>vector</code>
project	<code>vector</code> a, <code>vector</code> b	<code>vector</code>
scale	<code>vector</code> a, <code>vector</code> b	<code>vector</code>
add	<code>vector</code> a, <code>vector</code> b	<code>vector</code>
subtract	<code>vector</code> a, <code>vector</code> b	<code>vector</code>
negative	<code>vector</code> a	<code>vector</code>
multiply	<code>vector</code> a, <code>number</code> b	<code>vector</code>
equals	<code>vector</code> a, <code>vector</code> b	boolean
look_rotation	<code>vector</code> a	<code>vector</code>

Machine

マシン

machine

キーのエミュレート、スライダとステアリングの制御、マシン情報のためのライブラリ。

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

Input

入力

input

キーボード、マウス、ジョイスティックなどから直接入力を行うためのライブラリ。詳しくは[Useful Links](#)のUnity Scripting APIを参照してください。

function	arguments	return values
mouse_screen_position		<code>vector</code>
mouse_raycast_hit_point		<code>vector</code>
get_axis	<code>string</code> axis	<code>number</code>
get_axis_raw	<code>string</code> axis	<code>number</code>
get_key	<code>string</code> key_code	boolean
get_key_down	<code>string</code> key_code	boolean
get_mouse_button	<code>int</code> mouse_button	boolean

function	arguments	return values
get_mouse_button_down	int mouse_button	boolean
get_mouse_button_up	int mouse_button	boolean
any_key		boolean
any_key_down		boolean

## Cursor

カーソル

cursor

マウスカーソルのライブラリ。

function	arguments	return values
set_state	boolean state	

## Physics

物理

physics

物理関係の情報を得るためのライブラリ。詳しくは[Useful Links](#)のUnity Scripting APIを参照してください。

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

## Players

プレイヤー

players

マルチプレイのセッションの情報を得るためのライブラリ。

function	arguments	return values
count		int
get	int player_index	player info
by_id	int network_id	player info
get_all		player info array

## Lines

lines

線を描画するライブラリ（ローカルのみで動きます）。

function	arguments	return values
new_line_renderer		line renderer

## Screen

画面

screen

ゲーム画面の大きさなどの情報を得るためのライブラリ。

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

Chat

チャット

chat

チャットメッセージを操作したり、書いたりするためのライブラリ。

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

読んでくれてありがとう！