Lua Scripting Mod ユーザーマニュアル (JP)

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 manualy` 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

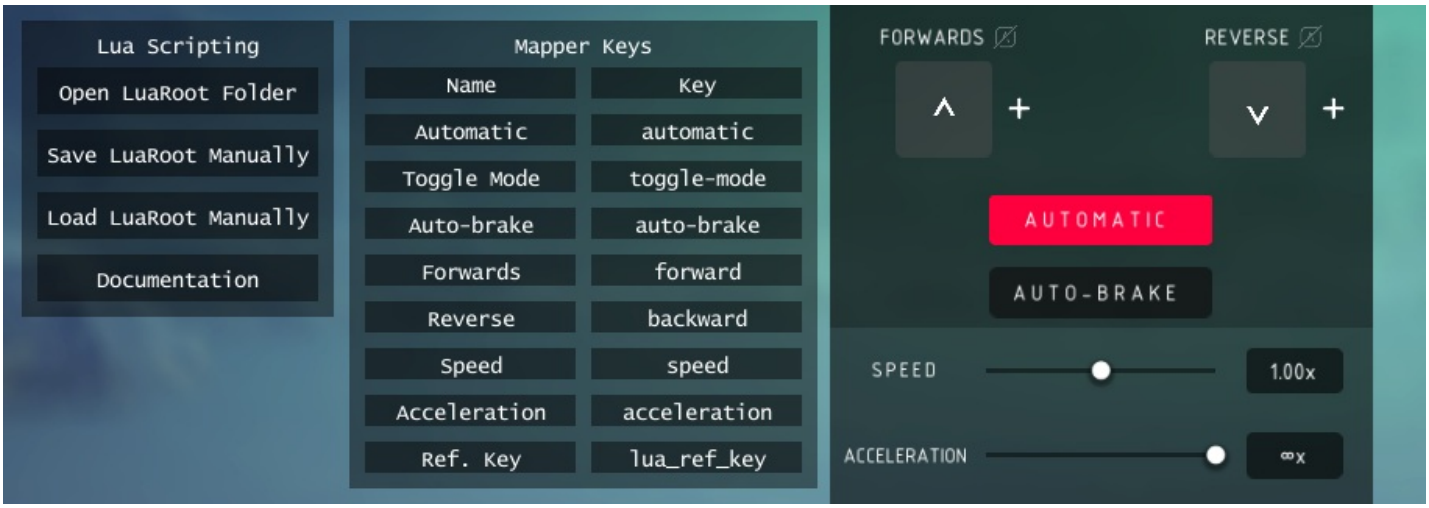
- 主な機能として、play、update、late_update、fixed_update、on_guiがあります。これらに関してはUnity Scripting API を参照してください。
  - `play` はシミュレーション開始時に読み込まれます。ほとんど役に立ちません。
  - `update` はフレーム更新のたびに読み込まれます。 プレイヤーの入力の検出に用います。
  - `late_update` はフレーム更新後に呼び出されます。
  - `fixed_update` は1秒ごとに一定回数（100Hz）呼び出されます。物理演算などに用います。
  - `on_gui` 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() do
    local other_player_mi = machine.get_machine_info(i)
    launch_rocket_at(other_player_mi.get_block_info(0).position())
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インタプリンタにはオブジェクト指向プログラミングのようなものはありませんが、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
x	`int`
y	`int`
width	`int`
height	`int`

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	*function* (no args; `void`)
stop	*function* (no args; `void`)
click	*function* (no args; `void`)

field	type
active	*function* (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	*function* (args: `string` mapper_key, `number` value; `void`)
set_steering	*function* (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	*function* (no args; returns `vector`)
forward	*function* (no args; returns `vector`)
right	*function* (no args; returns `vector`)
up	*function* (no args; returns `vector`)
rotation	*function* (no args; returns `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 `vector`)
velocity	*function* (no args; returns `vector`)

field	type
angular_velocity	*function* (no args; returns `vector`)

Machine Info

マシンの情報

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

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

レイキャストのヒット判定

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

field	type
distance	`number`
point	`vector`
normal	`vector`
is_block	`boolean`

field	type
get_block_info	*function* (no args; returns `block info`)

Collider

コライダー

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

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

Line Renderer (WIP)

線の描画

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

field	type
set_points	*function* (`vector` start, `vector` end; `void`)
set_width	*function* (`number` start_size, `number` end_size; `void`)
set_color	*function* (`vector` color; `void`)

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
new	`int` x *(optional)*, `int` y *(optional)*, `int` width *(optional)*, `int` height *(optional)*	`rectangle`

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
label	`rectangle` position, `string` text	
button	`rectangle` position, `string` text	`boolean`
begin_group	`rectangle` position	
begin_scroll_view	`rectangle` position, `vector` scroll_position, `rectangle` view_rect	

function	arguments	return values
box	`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	`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) *(arguments are changed)*	`int` id, `rectangle` client_rect, `string` text, *function* (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` buttom_value	`number`
vertical_slider	`rectangle` position, `number` value, `number` top_value, `number` buttom_value	`number`
window *(arguments are changed)*	`int` window_id, `rectangle` client_rect, `string` title, *function* (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 *(optional)*, `number` y *(optional)*, `number` z *(optional)*, `number` w *(optional)*	`vector`
distance	`vector` a, `vector` b	`number`
dot	`vector` a, `vector` b	`number`

function	arguments	return values
lerp	`vector` a, `vector` b, `number` t	`vector`
lerp_unclamped	`vector` a, `vector` b, `number` t	`vector`
magnitude	`vector` a	`number`
max	`vector` lhs, `vector` rhs	`vector`
min	`vector` lhs, `vector` rhs	`vector`
move_towards	`vector` current, `vector` target, `number` max_distance_delta	`vector`
normalize	`vector` a	`vector`
project	`vector` a, `vector` b	`vector`
scale	`vector` a, `vector` b	`vector`
add	`vector` a, `vector` b	`vector`
subtract	`vector` a, `vector` b	`vector`
negative	`vector` a	`vector`
multiply	`vector` a, `number` b	`vector`
equals	`vector` a, `vector` b	`boolean`
look_rotation	`vector` a	`vector`

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	`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

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		`vector`
mouse_raycast_hit_point		`vector`

function	arguments	return values
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_mmouse_button_down	`int` mouse_button	`boolean`
get_mmouse_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`

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	*function* (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！