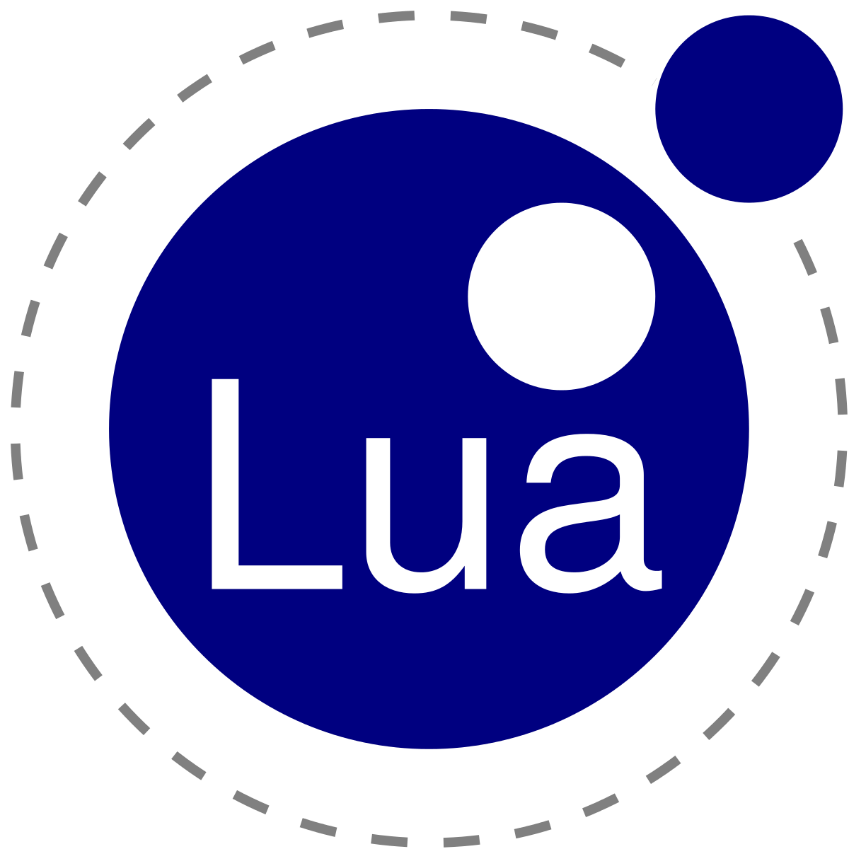
LuaEngine API

# Developer Documentation

Rodrigo Sanchez 436703

Saxion University of Applied Sciences

Preamade Lua Scripts

# Game.lua

The point of entry for the game. It contains Unity-like functions that are called by the Engine on their respective times.

|  |  |  |
| --- | --- | --- |
| Function | Parameters | Use |
| **start()** |  | start is called once at the start of the game, after all the initialization has finished. |
| **update(deltaTime)** | **deltaTime** **:** the timestep of the game update. | update is called every frame. The function call happens at the end of the engine’s core update function. |

As the main lua file that will be run by the engine, other scrips that you wish to reference directly will need to be included in this file by the [require “ ”] keyword.



# WindowSettings.lua

This file holds changeable settings for setting up the sfml window on initialization. The read variables are passed through to the Engine by the windowTable.

|  |  |
| --- | --- |
| Variable | Use |
| **windowTable** | Holds the variables to be passed to the Engine. |
| **windowTable.width** | The width of the sfml window. |
| **windowTable.height** | The height of the sfml window. If forceAspectRation is set to true, the value will be automatically set by aspectRatio relative to the width. |
| **windowTable.fullscreen** | Should the window start on fullscreen or windowed mode. |
| **windowTable.title** | The title name of the sfml window. |

# Cards.lua

This script holds all information and functions to create and modify cards.

Libraries

# LuaObject

A powerful collection of functions that let you create, modify and destroy game objects. All functions can be used from an instance of a LuaObject. First, create a new LuaObject by calling its constructor.

|  |  |  |
| --- | --- | --- |
| Function | Parameter | Use |
| **LuaObject(name, sprite, x, y)** | **name :** string name of the object. **sprite :** string path + name of the file to load as a sprite. **x :** float horizontal position.  **y :** float vertical position. | This is the constructor, to use the rest of the functions we first need to initialize a new object.  Create a new LuaObject like this: local obj = LuaObject(“newObject”, “card.png”, 400.0, 300.0) |
| **:getName()** |  | Returns the assigned name of the object in a string. |
| **:setName(name)** | **name :** string new text to give as a name for the object. | Sets the name of the object to the parameter value. |
| **:translate(x, y)** | **x :** float number of units to move horizontally.  **y :** float number of units to move vertically. | Translate moves an object X and Y amount of units relative to the current position. |
| **:rotate(angle)** | **angle :** float number of angles in degrees to turn clockwise. | Rotate turns an object clockwise by the specified angle in degrees, relative to its current rotation. |
| **:scale(factor)** | **factor :** float number to resize the object by multiplication. | Scale changes the size of an object by multiplying it by the given factor, relative to the current scale. |
| **:getPosition()** |  | Gets the absolute position of the object in the screen. Returns two values (x, y). |
| **:setPosition(x ,y)** | **x :** float number of units to move horizontally.  **y :** float number of units to move vertically. | Sets the position of the object absolute to the screen. |
| **:getRotation()** |  | Gets the absolute rotation of the object in degrees. Returns one value (angle). |
| **:setRotation(angle)** | **angle :** float number of angles in degrees of the rotation. | Sets the absolute rotation of the object in degrees. Clockwise is positive. |
| **:getScale()** |  | Gets the width and height of an object’s sprite. Returns two values (width, height). |
| **:setScale(width, height)** | **width :** float number of the new horizontal size of the sprite.  **height :** float number of the new vertical size of the sprite. | Sets the width and height of a sprite by the given amount in pixels. |
| **:setSprite(filename)** | **filename :** string path and name of the texture file to load. | Loads a new texture by the given filename and sets a new sprite using that texture. |
| **:setSpriteOrigin(x, y)** | **x :** float number of units to move horizontally.  **y :** float number of units to move vertically. | Sets the origin (pivot) of the sprite to the given coordinates. |
| **:setSpriteColor(r, g, b, a)** | **r :** int number from 0 to 255 that represents the red channel.  **g :** int number from 0 to 255 that represents the green channel. **b :** int number from 0 to 255 that represents the blue channel.  **a :** int number from 0 to 255 that represents the alpha channel (transparency). | Changes the color of the sprite by RGBA. Values go from 0 to 255. |
| **:getSpriteCenter()** |  | Gets the relative position of the center of the sprite. Returns two values (x, y). |
| **:isMouseHovering()** |  | Returns if the mouse is currently over the sprite’s bounding box. This can be used in combination with Input.getMouseButton(n) to detect a click on a sprite.  Returns one value (boolean). |
| **:setAsBackground()** |  | Sets the object to be the first in the world order, effectively making it render before all other objects. |
| **:destroy()** |  | Destroys the object at the end of the frame. |

# Input

A collection of functions that register input from the sfml Window events in the Engine and sends them to Lua.

|  |  |  |
| --- | --- | --- |
| Function | Parameters | Use |
| **getMousePos()** |  | Returns the current position of the mouse pointer on the window. Returns two values (x, y). |
| **getMouseButton(n)** | **n** **:** int number of the mouse button to check for. 0 = LMB, 1 = RMB, 2 = MMB | Constantly checks if the given mouse button is pressed or not. Returns one value (isPressed). |

When checking for a mouse button pressed, the engine constantly sends true every frame that a button is held down. You might want to check for a single mouse press, this is possible from lua and can be done and added to the Input library like this:



# Sound

This library holds all basic functions to play sounds and music.

|  |  |  |
| --- | --- | --- |
| Function | Parameters | Use |
| **loadSound(filename)** | **filename :** string of the sound file to load. | Sounds need to be loaded before calling *playSound(filename)*. This function will load the sound file into memory. Returns one value (filename). |
| **playSound( filename )** | **filename :** string of the sound file to play. | Use this function to play a loaded soundfile once. |
| **playMusic( filename )** | **filename :** string of the music file to play. | Use this function to load and start playing a music file. The music will loop by default. Only one music file can be active at a time, calling this function again will replace the current music playing. |
| **stopMusic()** |  | Stops playing the current music file. |
| **setMusicVolume(value)** | **value :** int number to set the music volume to. | Sets the volume of the playing music file. |
| **setMusicLoop(bool)** | **bool :** Boolean value true or false to set if the music loops after end or not. | Sets if the music file will loop after ending. The default for this option is set to true. |