Jeroen P. Broks

A beginner's guide to:

# Table of Contents

# 1. Introduction to Lua and this book

## *1.1 What is Lua?*

Lua is a scripting language, which works completely in an interpreted environment. It has been coded in C and has been set up to be easily implemented with any program written in C. It's been used for many kinds of application. For example for creating addons for utilities, and most of all for games.

Lua was designed by Roberto Ierusalimschy, Waldemar Celes and Luiz Henrique de Figueiredo in a university in Rio de Janeiro in Brazil, and is completely free. It was named after the moon as “lua” means “moon” in Portuguese. Despite Lua itself being set up in C, you don't need any knowledge of C in order to use it, due to language quite often being implemented in tools in which you may need it.

Lua is very extremely symplistic in its syntax and general setup, making it very easy to learn and understand, even when you never touched a programming language before. Still Lua has a kind of low-level approach and is despite being interpreted quite fast. Since it's specifically designed to work in a C environment, it's very popular in the professional industry and knowing the basics of Lua can get you some extra tickets into the professional gaming industry.

I will go into the deep of how Lua works later in this book, but the famous “Hello World” program is only one line: “print('Hello World!')”.

Lua has also been designed to not bother about platform specific issues. So basically a Lua script should work in Windows, MacOS and Linux and basically any platform you can think of.

## *1.2 What do you need to install for this course.*

When it comes to Game programming one of the most complete Lua engines to make anything is LÖVE (http://love2d.org) but I advise you not to start with LÖVE right away. Before we get onto a complete engine like that it's essential that you first understand how Lua itself works. For that you need a Lua program you can run from the CLI. This program can only perform the core features, but it's good enough for testing what you've learned. If CLI tools are too difficult for you to understand, you can also use the Lua playground (https://www.lua.org/cgi-bin/demo). You can just enter the Lua code and the result will appear as soon as you click “run”. For training this will do.

To Test the playground just type “print('Hello World!')” in the textarea and then click “Run”, and “Hello World!” should appear below. When you use the cli tool type “cat > hello.lua” in the terminal when you use Linux, Mac or BSD and “copy con hello.lua” on Windows and type

“print('Hello World!')” and press enter and press ctrl-D in Linux/Mac/BSD and ctrl-z on Windows then type “lua hello.lua” and if you see “Hello World!”, you can see you installed the CLI tool correctly.

When you really get serious into Lua scripting you will need an editor that can handle Lua Scripting. I use the Lua Devolopment Tools for Eclipse, but that can be a rather complex to understand. Atom, Geany, SublimeText, NotePad++ should all be able to allow you to work with Lua scripts.

## *1.3 How to use this book*

I will try to take you step by step into the secrets of Lua. I will show you many scripts. It could be wise to copy these and try what they do.

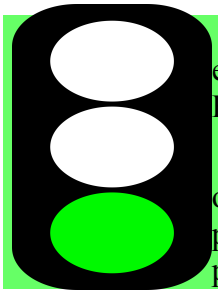
Code will look like this:

```
-- Test
a="Hello World"
print(a)
```

The colors you see is called “Syntax Highlight”. This will help you a lot keep some overview over your code.

When you see this icon and the text beside it with a blue background, it means that I got something very important to tell. Like all programming languages Lua can be full of trapdoors you can easily fall through, and you should therefore be aware of those. I make these warnings for a reason.

When you see this icon on an amber background, it means that I am about to tell you some typical nerdy stuff, that is just nice to know, but most of all aimed to people who can look beyond the regular stuff. If you have no understanding of what I tell here, then don't you worry one bit, as it's not really important to move on. Perhaps if you got more understanding of Lua and you want to read this over, hey, why not ;)



And lastly when you see this icon on a green background, I'll be giving you some exercise. The best way to learn to code, no matter if you do that in Lua, Pascal, BASIC or even in C is by doing it. And thus a few assignments never hurt.

These assignments may sometimes have some code snippets, but mostly you are on your own, but you area allowed to look things up in the previous chapter or paragraphs or sections to see the solution. Sometimes I may even give you some pointers of the expected output, so you can check a few things for yourself.

And with this all discussed, I wish you good luck with this book.

## 2. Hello World! Your first real program.

Most traditionally the first program you'll be writing in whatever language you're gonna learn is Hello World! It's been said that the first time Hello World was written was to demonstrate a prototype of the C programming language.

In C Hello World looks like this:

```
#include <stdio.h>
int main(void){
     printf("Hello World\n");
}
```

Not really much to it, even in C, although in C you need 4 lines and in Lua, just one.

```
print("Hello World")
```

Now “print” is a function. I'll go into the deep about what functions are later in this book, but for now it's important to know that a function can be used to make Lua do something. In the case of “print” that is to put something on the screen. “Hello World” is was we call a string. A string is a series of characters, usually used to form text. So in English we said to Lua *put the text “Hello World” on the screen.*

Let's test this out. Copy that line into a program or onto the playground and run it and see what happens. It should show “Hello World”.

When it comes to strings in nearly any language you will need quotes. In the example I used double quotes. Lua will also understand single quotes. It doesn't really matter if you use single or double quotes, as long as you are consequent in using the same quotes to end the quotes as you started them. I however advice double quotes. If you will ever consider moving on to C, Go, C# and some other languages in this family single quotes have a different meaning, and thus you can confuse yourself when you use single quotes (big exception is Pascal that requires single quotes).

Now I'm gonna take you into something important. Especially when you coded in BASIC or Pascal before you can really suffer here. Lua is case sensitive.

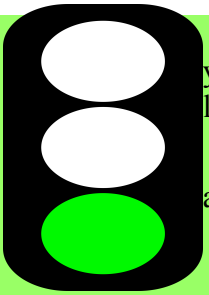
```
PRINT("Hello World")
```

This code will therefore not work. Try it and you will see Lua complain about a trying to call a “nil value” or something like that. I shall go into the deep about what a “nil value” is later, but for now let's suffice to say that a nil value means that you are trying to do something with something that does not exist, and therefore Lua can't handle that. Lua does make a difference between upper and lower case letters. So “print” and “PRINT” are therefore not the same. Especially when you move on to more complex Lua scripting you should be very strict on yourself on how you use upper and lower case or your scripts can become a big mess.

Now with this knowledge you can take this a little bit further. “print” will not only show text on screen. It will also move to the next line. And thus we can also do this:

So without trying this out in the playground or a cli tool, you must be able to tell what this program does:

```
print("USA stands for United States of America")
print("Its first president was George Washington")
print("It borders to Canada and Mexico")
```



Now write a program that shows your name and your age, your date of birth and your city of birth onto the screen. All data fields I asked for should have their own line in the output.

If you can do this, you have understood the basic of the Hello World sequence and then we can get on the move for the next chapter.

s

d