LÖVE for Newbies

Sol Bekic <s0lll0s@blinkenshell.org> Ricardo Gomes <desk467@gmail.com>

Table of Contents

1. The Lua Programming Language 1
   1.1. Introduction 2
   1.2. What is Lua? 2
   1.3. Basic Flow Concept 3
   1.4. Storing Data in Variables 4
   1.5. Strings 5
   1.6. True or False? 6
   1.7. Conditional Statements 7
   1.8. Which way should I go? 8
   1.9. Functions 9
   1.10. Tables 10
   1.11. Metatables 11
2. The LÖVE Framwork 12
   2.1. Structure of a LÖVE Project 13
   2.2. Interacting with LÖVE 17
   2.3. Object Oriented Programming 18
   2.4. Drawing! 19
   2.5. Audio 20
   2.6. Fonts 21
   2.7. Advanced input with `love.keyboard` and `love.mouse` 22
   2.8. Game math 23
3. License 24
   3.1. Libraries & Tools: 24

1. The Lua Programming Language

1.1. Introduction

So you want to make games, right?
Basically, every computer program (yes, games are also included) are made using a **programming language**. But what is it? Similarly when we want to talk to someone from another country and we choose a common language between the two, a programming language is a language in which both humans and computers understand. This is how we tell the computer what it should do.
Computer programs are basically a bunch of instructions that a programmer wrote and distributed. Thus, the computer will follow the instructions' guide and begin to perform them.

In this first world, we will study the basics of Lua programming language, which we use to make games using the LÖVE framework.

1.2. What is Lua?

Lua is a simple, light and extremely powerful language, designed to be used within complex software (such as games). It was created by a team of developers of PUC-Rio, Brazil. And because of its efficiency, extreme clarity and **ease of learning**, came to be used in various applications.

1.2.1. Installation

In this world, we will teach the basics of Lua programming language, and therefore, you can test the codes online.
Just visit the site: http://www.lua.org/cgi-bin/demo and write your code there.

1.3. Basic Flow Concept

1.3.1. Flow

When we write some code in a programming language is required us to write very important things first and then we write things depending on what has already been said. The computer will read line by line from top to bottom.

1.3.2. Conditions

We can tell the computer that it chooses other paths and finish lines jumping. Think that you are lost in a field and reaches a point in front there are two possible paths. You will choose one of them by some condition (situation), as there is water and food. In computer programming, this concept is called **conditional structure**.

1.3.3. Loops

It is our usual to repeat the same idea to perform a larger task. If I ask you to do 8 cookies, you will follow a bunch of steps for each cookie to complete 8 cookies. This concept is called **loop**.

1.4. Storing Data in Variables

And now we begin the fun part ..

We have spoken that the computer can make various tasks using some instructions. Let's know the concept of variable.

When you make the cookies, you use a few **ingredients** and some **tools**. Let's start with the ingredients, which for computing, are called variables.

The computer is a mathematical machine that performs tasks from data. Variables are a space that we allocate int the computer memory to store data that will be useful for us.

For example:

```
name = "Joseph"
age = 18
print (name)
```

WARNING	When we want to store textual information (strings, in technical term) into a variable, we put it in quotes.

In this small piece of code we are assigning the given "Joseph" to the variable **name** and 18 to the variable **age**. We use this syntax in Lua to allocate space in memory. In the left side of the equal symbol specify the variable name and the right hand, the data. The last line tells the computer that it should show the data contained in the variable name. *print* is a **tool** from Lua. The language have a lot of useful tools. Test and see the result.

TIP	How about trying to make the computer show the buddy's age?

1.5. Strings

IMPORTANT | Write Level

1.6. True or False?

IMPORTANT | Write Level

1.7. Conditional Statements

IMPORTANT | Write Level

1.8. Which way should I go?

IMPORTANT | Write Level

1.9. Functions

IMPORTANT | Write Level

1.10. Tables

IMPORTANT	Write Level

1.11. Metatables

IMPORTANT | Write Level

2. The LÖVE Framwork

2.1. Structure of a LÖVE Project

So, what makes a LÖVE Project?
At core a LÖVE Project is just a folder containing everything needed to make the game; Code, Images, Sound and Video files and everything else you might need.

2.1.1. Code

Obviously, a game contains code. LÖVE Games are programmed in Lua, which you should have learned about in World 1. Generally Lua Source Files (`.lua`) can lay around anywhere in the project directory and have arbitrary names… except for two special ones: * main.lua * conf.lua

These two are the only files that the LÖVE Framework runs; they are the starting points of every game or project you build.

main.lua

As the name implies this file will contain all your **main code** - what exactly that will be and how you organize your code is up to you. Usually this file contains the used *Callback Routines*, which will be covered in World 2-2. For smaller projects and the next levels in this book this will be the only file (except for `conf.lua`) you will need.

conf.lua

conf is short for **Conf**iguration, and thats what `conf.lua` is all about. You can fill this file with a function called `love.conf( t )` that accepts a table as it's only parameter. In that function you can then modify certain fields of the table and thereby change the configuration the LÖVE Framework uses when it first creates your window.

Here is a function that sets every possible value to it's default value - and thereby does nothing:

Full `conf.lua` *example*

```
function love.conf(t)
    t.identity = nil                    -- The name of the save directory (string)
    t.version = "0.9.1"                 -- The L  VE version this game was made for
(string)
    t.console = false                   -- Attach a console (boolean, Windows only)

    t.window.title = "Untitled"         -- The window title (string)
    t.window.icon = nil                 -- Filepath to an image to use as the window's
icon (string)
    t.window.width = 800                -- The window width (number)
    t.window.height = 600               -- The window height (number)
    t.window.borderless = false         -- Remove all border visuals from the window
(boolean)
    t.window.resizable = false          -- Let the window be user-resizable (boolean)
    t.window.minwidth = 1               -- Minimum window width if the window is resizable
(number)
    t.window.minheight = 1              -- Minimum window height if the window is
resizable (number)
    t.window.fullscreen = false         -- Enable fullscreen (boolean)
    t.window.fullscreentype = "normal" -- Standard fullscreen or desktop fullscreen mode
(string)
    t.window.vsync = true               -- Enable vertical sync (boolean)
    t.window.fsaa = 0                   -- The number of samples to use with multi-sampled
antialiasing (number)
    t.window.display = 1                -- Index of the monitor to show the window in
(number)
    t.window.highdpi = false            -- Enable high-dpi mode for the window on a Retina
display (boolean). Added in 0.9.1
    t.window.srgb = false               -- Enable sRGB gamma correction when drawing to
the screen (boolean). Added in 0.9.1

    t.modules.audio = true              -- Enable the audio module (boolean)
    t.modules.event = true              -- Enable the event module (boolean)
    t.modules.graphics = true           -- Enable the graphics module (boolean)
    t.modules.image = true              -- Enable the image module (boolean)
    t.modules.joystick = true           -- Enable the joystick module (boolean)
    t.modules.keyboard = true           -- Enable the keyboard module (boolean)
    t.modules.math = true               -- Enable the math module (boolean)
    t.modules.mouse = true              -- Enable the mouse module (boolean)
    t.modules.physics = true            -- Enable the physics module (boolean)
    t.modules.sound = true              -- Enable the sound module (boolean)
    t.modules.system = true             -- Enable the system module (boolean)
    t.modules.timer = true              -- Enable the timer module (boolean)
    t.modules.window = true             -- Enable the window module (boolean)
    t.modules.thread = true             -- Enable the thread module (boolean)
end
```

NOTE	You don't need to use a `conf.lua` or specify every key in the conf table; everything you leave out will remain at it's default value.

You will mostly be using this to set a different resolution for your game and set the game title.

Usual `conf.lua`

```
function love.conf( t )
    t.identity      = "GtGLG"
    t.version       = "0.9.1"

    t.window.title  = "Gary, the green-legged Giraffe"
    t.window.width  = 1200
    t.window.height = 720

    t.window.fsaa   = 4
    t.window.vsync  = true
end
```

Other files

Everything else will need to be `require`d by `main.lua` in some way (direct or indirect).

2.1.2. Images, Videos, Sounds and other Assets

All of these files need to be somewhere in the project directory aswell. You will learn to load and draw or play these files throughout this World.

Even though you can just have all the files in one directory, it is advised that you structure your files in a logical hierarchy, for example like this:

```
- mygame/
    + main.lua
    + conf.lua
    + lib/
        + library1.lua
        + library2.lua
        + sometool.lua
    + assets/
        + images/
            + player.png
            + rock.png
        + sounds/
            + impact.wav
            + menumusic.mp3
        + videos/
            + intro.mp4
```

2.2. Interacting with LÖVE

IMPORTANT | Write Level

2.3. Object Oriented Programming

IMPORTANT | Write Level

2.4. Drawing!

IMPORTANT | Write Level

2.5. Audio

IMPORTANT | Write Level

2.6. Fonts

IMPORTANT | Write Level

2.7. Advanced input with `love.keyboard` and `love.mouse`

IMPORTANT | Write Level

2.8. Game math

IMPORTANT | Write Level

3. License

IMPORTANT | Find consensus on a license

3.1. Libraries & Tools:

- AsciiDoctor renders this book
- Moonshine, licensed under the GNU GPL License, and
- punchdrunk by Tanner Rogalsky make LÖVE run in **your** browser
- ...as does of course LÖVE, which this book is all about

wherever not listed above, these are licensed under the MIT License:

The MIT License

```
Copyright (C) held by the respective authors, see the LICENSEs in the links provided
above

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```