Lua and game development

What is Lua?

Lua is a dynamically typed, garbage collected, multi-paradigm, scripting language.

Designed to be easily:

- Learn
- Port
- Embed

The lore

1993

Roberto Ierusalimschy
Luiz Henrique de Figueiredo
Waldemar Celes

Computer Graphics Technology Group (Tecgraf)
at the Pontifical Catholic University of Rio de Janeiro,
in Brazil.

Circumstances and goals

1977 - 1992 Trade barrier against Brazil led to creation of **SOL** (Simple Object Language) and **DEL** (data-entry language)

The need raised for a full-featured, embedded, scripting language to replace SOL and DEL. Mostly to allow users to generate custom reportings, with little to no programming knowledge, and without the need of re-building those larger systems (where this extension language is embedded to).

Goals:

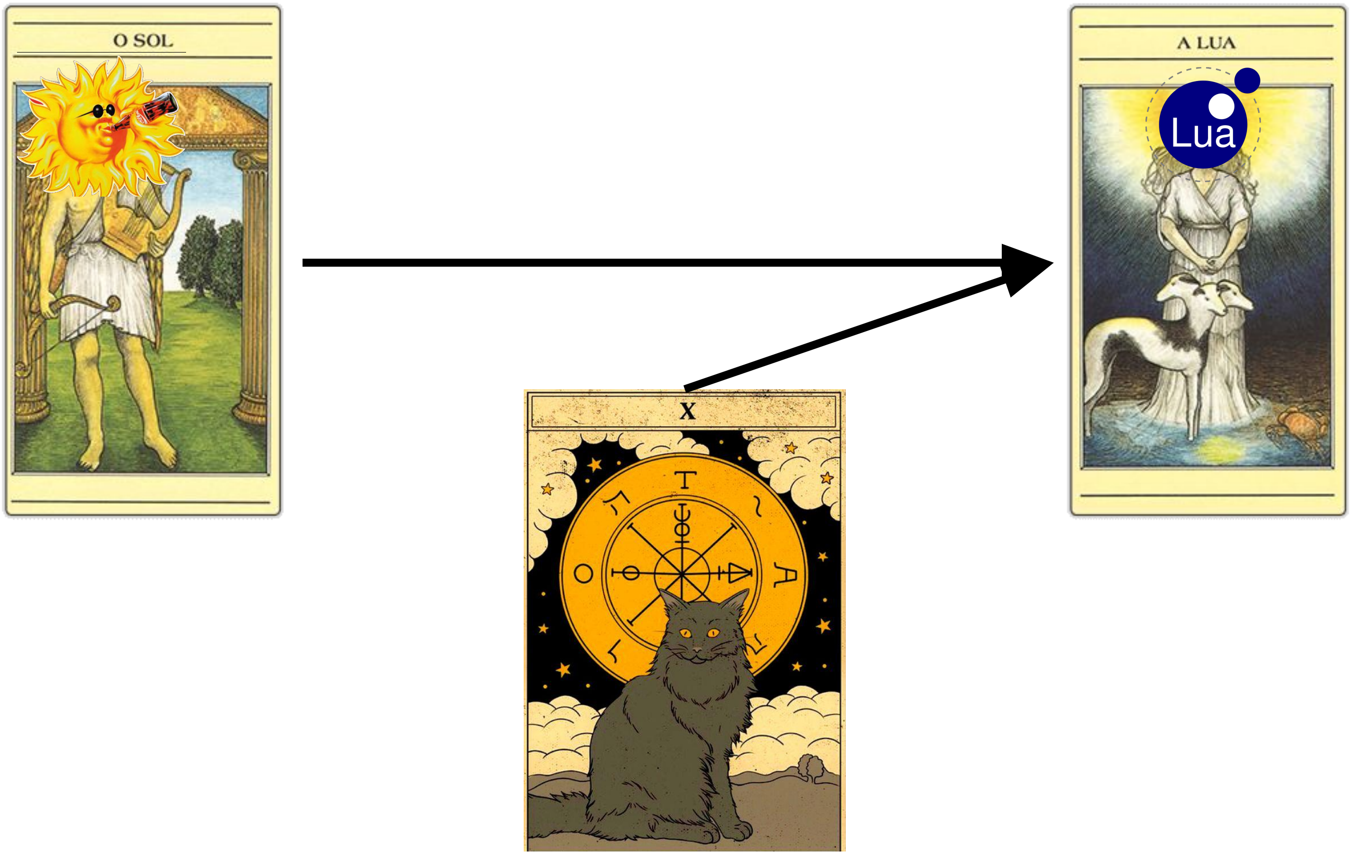
- Small
- Easy non-cryptic syntax (suitable for non-programmers)
- High portability (diverse platforms)
- Embeddability (into existing and new projects)

Alternative(s):

- TCL (ugly syntax, no good data description, Unix only)
- LISP and Scheme (ugly syntax)
- Python (too young)

SOL (Simple Object Language)

Lua means moon in Portuguese



DEL (data-entry language)

Execution

A ~24.000 line ANSI C Library which includes the interpreter, the Virtual Machine, and the C API. Also a reference implementation, which is basically a CL I (like Python, Node.js) ~120 KByte compiled.

A ~100 Page manual (for the language itself, on how to embed it, and also the C API reference)

Features (beside the original goals to be small, simple, and portable)

- Fast
- Multiple programming paradigm
- Secure
- Highly extensible

What happens to your lua script?

- **Lexical Analysis (Tokenization)**
  The Lua script is read and broken down into tokens, which are the smallest units of the script, by the lexer.

- **Syntax Analysis (Parsing)**
  The parser analyzes the tokens to create an abstract syntax tree (AST), representing the grammatical structure of the script.

- **Bytecode Compilation**
  The AST is then translated into Lua bytecode by the compiler. This bytecode is a low-level representation of the script that can be executed by the Lua virtual machine.

- **Execution**
  The Lua virtual machine interprets and executes the bytecode sequentially, line by line. Each instruction is executed by the VM, and the program's behavior is manifested.

Let’s look at the syntax and some code examples!

What is LuaJIT, and why it is important?

LuaJIT is a Just-In-Time Compiler for Lua. It is developed by Mike Pall from 2005 to 2015. Some say it is the fastest implementation of a dynamic programming language. Prized among programmers and widely considered as a **technological marvel**.

From version 2.0 it has its own implementation of the Lua VM. It uses the darkest magic to compile our code on multiple stages, making various optimizations from stage to stage, essentially our program becomes **optimized and compiled,** from being just interpreted, during runtime (but it always can fall back to interpret “unvisited” pieces of the script). So it provides the flexibility of a scripting language with the performance of a compiled language.

Some application:

- CERN, for their Methodical Accelerator Design 'next-generation' software for describing and simulating particle accelerators
- OpenResty, a fork of Nginx with Lua scripting
- Cloudflare, for web application firewall service

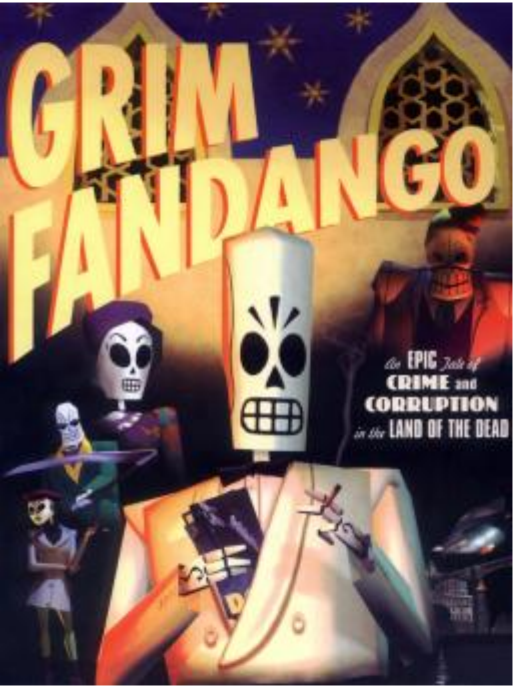
Where Lua is used?

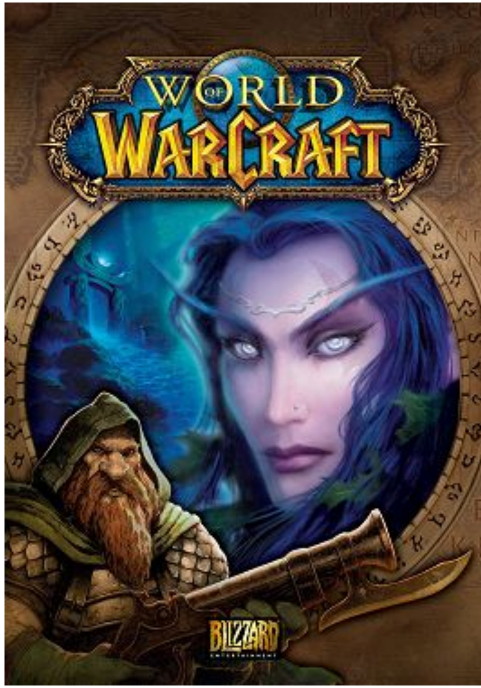
(including but not limited to)

- Operating System kernels: (BSD uses Lua as bootloader)
- Web servers: Apache HTTP Server, OpenResty, (GitHub Pages infra)
- Networking: Cisco Systems, Wireshark
- Photography: Adobe Photoshop Lightroom, The Canon Hack Development Kit
- Database: Redis, MySQL Workbench
- Text Editor: Vim, Neovim
- Media Player: VLC, mpv (an mplayer fork)
- Embedded: various set-top-box, switches and routers

And of course in….

Game Development!


GRIM FANDANGO
An EPIC Tale of CRIME and CORRUPTION in the LAND OF THE DEAD



Game Engines

(including but not limited to)

- Solar2D (former Corona SDK)	2D Lua; Windows, Mac, Android, iOS
- LÖVE 2d	2D Lua; Windows, Linux, Mac, Android, iOS
- Defold	2D Lua; Windows, Linux, Mac, iOS, Android, Web, Switch
- Cocos2d-x	2D C++/Lua/JS; Windows, Linux, Mac, Android, iOS, BlackBerry
- CRYENGINE	3D C++/Lua; Windows, Mac
- Raylib (binding)	2D&3D C++/Lua/Others; Windows, Linux, Mac, Android, Web, Other Ports
- SDL2 (binding)	2D&3D C++/Lua/Others; Windows, Linux, Mac, Android, Console Ports
- Godot (plugin?)	