Introduction to Lua

Fabio Mascarenhas

http://www.dcc.ufrj.br/~fabiom/lua
Lua is...

• ...an scripting language:
  • Robust, fast, portable, extensible, small, and open

• Lua is similar to other scripting languages such as Perl, Pyton, Ruby, and JavaScript

• We can also use Lua as a data description language, such as XML and JSON

• Finally, Lua is an *extensible extension language*, focusing on multi-language development
Lua in Games

• “It is easy to see why Lua is rapidly becoming the de facto standard for game scripting” - Artificial Intelligence for Games, Morgan Kaufmann, 2006.

• “It’s quite possible that game developers will look back at the 2000s as the decade of Lua” - Game Programming Gems 5, Charles River Media, 2005.

• “A TREMENDOUS amount of this game is written in Lua. The engine, including the Lua interpreter, is really just a small part of the finished product.” - Bret Mogilefsky, programador-chefe do jogo Grim Fandango.

• Lua is used by games in all platforms and genres: mobile, consoles, PCs, FPS, strategy, casual, MMORPGs…
Lua in Games
But not just games…

- Scripting and template language for *Wikipedia*

- Interactive applications on the Brazilian Digital TV standard (Ginga)

- Embedded software: printers (Olivetti, Océ), routers (Cisco), telephones and smartphones (several, including Huawei), smart tvs (Samsung), Logitech keyboards, Lego Mindstorms...

- Security: scripting vulnerability scanners (nmap, Wireshark, Snort)

- A million lines of Lua code makes the bulk of Adobe Photoshop Lightroom, and several other applications have Lua as a scripting language: VLC, Tex, vim, lighttpd, Apache, nginx…
Why use Lua?

• Portability

• Simplicity

• Small size

• Embeddability

• Efficiency
Portability

- Lua runs in practically all known platforms
  - Not just “famous” ones such as Windows, Linux, *BSD, OS X, Android, iOS, Windows Mobile, …
  - … but lots of embedded platforms that do not have even operating systems and run Lua on the “bare metal”
  - If it has a C cross-compiler and about 64Kb of free RAM, it can run Lua
- Lua is written in a common subset of C and C++, and the core of the language has very few dependencies on libc
Simplicity and small size

• Just a small set of powerful primitives

• The reference manual, documenting the language, the C interface, and the standard library, has about 100 pages

• Mechanisms instead of policies for higher-level features such as object orientation and concurrency

• Less than 200Kb of compiler code, of which less then 100Kb is the core, the rest is the optional standard library
Embeddability

- The Lua interpreter is a library for C programs.

- The API for communication with C is simple and well-defined.

- C programs have bi-directional communication, with Lua values going from the application to Lua and back with ease, and no marshalling.

- Programs in other languages can easily consume the API, as long as the language can interface with C code: C++, Java, FORTRAN, C#, Pascal, Perl, Python...

- Yes, even other scripting languages; a large application that embeds Lua for scripting is a version control system written in Python.
Efficiency

- Independent benchmarks show Lua as the fastest language in the class of interpreted scripting languages.

- An alternative implementation, LuaJIT, provides performance similar to compiled languages such as Java.
How Lua started

• Lua was born in 1993 inside PUC-Rio, at the Tecgraf, PUC-Rio’s Computer Graphics Laboratory

• Tecgraf needed an structured language that non-programmers could use for data description tasks

• The language needed to be portable, as Tecgraf had heterogeneous hardware, and needed to interface easily with C, as the applications were written in C

• Not many options at the time that fulfilled all prerequisites, so they decided to create their own language
Lua 1

- Lua 1.0 was implemented as a library, in less then 6000 lines of C

- "The simplest thing that could possibly work": compiler used lex and yacc, simple stack based virtual machine, linked lists for associative arrays

- Some of the syntax still lives in the current version:

  ```lua
  function track(t)
    if type(t.x) ~= "number" then
      print("invalid 'x' value")
    end
    if type(t.y) ~= "number" then
      print("invalid 'y' value")
    end
  end
  t1 = @track{ x = 10.3, y = 25.9, title = "depth" }
  ```

- Lua 1.1 just added a reference manual, and a cleaned-up C API
Lua 2

- From Lua 2.1 (February 1995) to Lua 2.5 (November 1996)

- Object oriented programming via delegation

- Pattern matching in the standard library

- Hooks for writing debuggers

- First users outside Tecgraf, with papers in *Software: Practice and Experience* and *Dr. Dobb's Journal*

- LucasArts begins using Lua in games
Lua 3

- From Lua 3.0 (September 1997) to Lua 3.2 (September de 1999)

- Anonymous functions and a restricted form of *closures* give better support for functional programming, which would mature in Lua 5

- Major refactoring in the source code

- The next version brings big changes to the C API, so some applications from this time still embed this version of Lua
Lua 4

• A single version, Lua 4.0, released on November 2000

• C API completely redone, using the stack model that we will see in this course

• An application can now have several independent instances of the Lua interpreter

• The standard library has been rewritten to use just the public C API, reinforcing the separation between the core and the standard libraries
Lua 5

- From Lua 5.0 (April 2003) to Lua 5.2, the current version, released December 2011

- Maturity of the language, and the release of the “Programming in Lua” book

- Several big changes: metatables, true lexical scope for anonymous functions, the module system, coroutines, lexical environments…

- Changes in the implementation: more efficient register-based virtual machine, replacing the stack-based one, an incremental garbage collector for shorter pauses

- The implementation now has around 20,000 lines of code, 3x Lua 1.0
Lua today

• Current license is the MIT license, free for both non-commercial and commercial use

• Open language, but closed development: new releases are still the responsibility of the three original authors

• Big community participation in the lua-l mailing list and the lua-users wiki

• A package manager, LuaRocks, and alternative Lua implementations: LuaJIT, JVM, .NET, JavaScript...

• Several frameworks for developing mobile games: Corona, Gideros, Codea, MOAI...