Lua Language Write up

Language name, origin/history, and design goals; and development tool availability:

**Name:** Lua (/ˈluːə/ LOO-ə; from Portuguese: lua [ˈlu.(w)ɐ] meaning moon)

**Origin / History:** Created in 1993 by Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes, members of the Computer Graphics Technology Group (Tecgraf) at the Pontifical Catholic University of Rio de Janeiro, in Brazil. It was built as a language for extending software applications to meet the increasing demand for customization at the time.

**Design Goals:** Designed to be a lightweight, high-level, multi-paradigm programming language designed primarily for embedded use in applications. It’s also cross-platform.

**Development Tool Availability: (Source:** <http://lua-users.org/wiki/LuaIntegratedDevelopmentEnvironments>)

These are integrated development environments ([IDE]) and related tools for Lua, sorted in the alphabetical order.

* **Eclipse** **(**Features:Code Assistance**,** Debugger, Code Template**,** Syntax Coloring**,** Error Markers**,** Outline**,** Variable Highlight**,** Code Formatter**,** Code Folding**,** Goto Definition**,** andCross Platform)
  + [Lua Development Tools] (5.0/5.1/5.2) - Lua IDE for the Eclipse Platform, includes a visual debugger. Official Eclipse project.
  + [LuaEclipse] (5.0/5.1) - Lua IDE for the Eclipse Platform. Integrates with LuaProfiler.
  + [akdebugger] Debugger and editor for Lua. Plugin for Eclipse.
* **Emacs**
  + [Lua-mode] (4.0/5.0/5.1) - syntax highlighting, auto-indent, interactive lua shell, paren and brace matching, auto-move to function or block begin/end, online documentation look-ups, postprocess output buffer (jump to error lines using traceback).
* **Howl**
  + [howl] - An emerging editor(2017) implemented in lua and moonscript. supports split views, custom lua/moonscript extensions, luacheck, plug-able syntax highlight and autocomplete
* **IntelliJ IDEA**
  + [lua-for-idea] (5.1) - a Lua plugin for IDEA. Comes with an embedded Lua compiler written in Java. [1]
* **Visual Studio**
  + [BabeLua] (5.1) - Visual Studio 2012/2013 extension. Auto completion, Syntax highlighting, Syntax error check, Formatting code, Lightweight project management, Quick search, Preview file outline and jump, List token references quickly, Object-oriented model support, Visual Studio project template, Debugger (Watch, Globals, Locals, Call Stack, Lua Stack, etc)
  + [vslua] (5.1) - Lua language service for Visual Studio 2008/2010.
  + [Lua Language Service] (5.1) - Lua syntax highlighting, code outlining, custom code completion (via editable XML files) into Visual Studio 2010/2013/2015 (open sourced).
  + See also CompilingLuaScriptsInVisualStudio.
* **Standalone (IDEs)**
  + [Decoda] (5.0/5.1) - An open source IDE and debugger for Lua that allows debugging of scripts in your own application, without any code changes - Win32.
  + [Glider] (5.1/5.2) - A commercial IDE featuring one click debugging, profiling, class aware code completion, declaration/occurrences finder, semantic highlighting and more. Plugins are available for integration with Corona, Gideros, Love2D, Cocos-2dx, and MOAI.
  + [ZeroBrane Studio] (5.1/5.2/5.3/LuaJIT) - A lightweight Lua IDE with code completion, syntax highlighting, remote debugger, code analyzer, live coding, and integration with several Lua engines (Löve 2D, Moai, and others) - Win32, Mac OSX, Linux.
  + [ZeroBrane Studio for Vera] - a package based on [ZeroBrane Studio] to provide development and debugging support for [Vera/MiCasaVerde] home automation devices.
  + [LuaEdit 2010] (5.1) - Complete professional looking Lua IDE - Windows 98/2000/XP/W7 (Very buggy)
  + [wxLua] (5.1) - a blend of Lua and wxWidgets. Provides its own IDE (written in wxLua) with a GUI debugger, a binding generator and wxWidgets bindings usable as a module. See also GraphicalUserInterfaceToolkits.
  + [Estrela editor] (5.1) - built upon wxLua's editor, features project view and rudimentary autocomplete and tooltip support based on api files.
  + [Gideros Studio] (5.1) - A cross-platform Lua IDE used to create mobile applications for Android and iOS easily
  + [Game Kitchen (formerly Löve Studio)] (5.1) An IDE for löve2d which features a debugger, an optional type system, and type-driven auto completion. - Windows vista or newer
  + [Comet] (1.2) - Lua development environment for numerical computing. Available for Android, Linux and Windows.
* **Lua-oriented scriptable editor**s (see also LuaEditorSupport)
  + [CodeMAX] (5.1) - simple and fast open source text editor with the possibility to add custom features using Lua. Supports syntax highlighting, code folding, completion proposal, hints for function parameters, and more. It is possible to build a complete IDE for different programming languages. (No debugging)
  + [SciTE] (5.2) - an extensible text editor. See also SciteScripts.
  + [Textadept] (5.1) - an extensible open source cross-platform text editor. Consists of minimalistic C++ barebone fleshed with Lua scripts. Nearly everything is controlled by Lua, making the extensibility almost limitless. Features project manager, dynamic lexers, multibuffer, split views. (No debugging "from the box")
  + [Lua file type plug-in for the Vim text editor] (5.1) - Automatic syntax checking, code completion (supports standard library identifiers, follows dofile/require calls and supports used defined modules by scanning the $LUA\_PATH and loading all defined modules), looking up documentation with one key press, keyboard shortcuts for quick navigation, checking for (undefined) globals, etc.
  + [Zeus IDE] (5.3) - IDE with Lua syntax highlighting, code folding, project management, debugging, integrated version control. The IDE is fully scriptable using Lua. (shareware)
* **IDEs/Editors** (Not updated for more than one year)
  + [B:Lua] - Open source project to create a full featured standalone Lua IDE. (development is on ice since April 2005)
  + [DForD LuaCoding] (5.1) - A full graphical Lua IDE. Features syntax highlighting, symbol browsing/searching, auto-completion, code snippet, debugging, building and more. Supported by a fully Lua script written plugin framework. (shareware) (Debugging is dead)
  + [Lua Studio] (5.1) - Simple, open source Lua IDE: editor + complete debugger. (Dead download Link to Kepler Project)
  + [QDE] (5.?) - Quotix Development Environment for Lua. It supports project management, autocomplete functionality, a multi-document interface, debugging and much more. (shareware, MS Windows only) (No debugging "from the box")
  + [Vortex Lua IDE] (5.0) - Another free Lua IDE, but in Brazilian Portuguese! (Outra IDE para Lua gratuita, mas em Portugues!) (link broken)
  + [VisualWx] (5.?) - IDE for Lua & wxLua. (freeware)
  + [LuaLite] (5.0) - syntax highlighting and auto-completion plug-in for Visual Studio .NET 2003.
  + [VSLua] (5.1) - A commercial solution to Lua language including an editor and debugger integrated within Visual Studio .NET (2002, 2003, 2005).
  + [VS Lua Language Pack] (5.1) - Open Source integration of Lua files into Visual Studio 2005.
  + [Lua Eclipse IDE] (5.1) - a set of Eclipse plug-ins that provide an editor and debugger for developing Lua scripts.
  + [LunarEclipse] (5.0/5.1) - Lua IDE for the Eclipse Platform. Implements a proper Eclipse perspective.
  + [LuaPatternView] (5.0) - an Eclipse plugin for testing [Lua patterns]. Supplied with the LuaJava Plugin for Eclipse, as well as (MIT-licensed) source code (binaries are available for MacOSX only for now).

Programming paradigm(s) you used in your project:

Is the language statically typed or dynamically typed?

It’s Dynamically Typed!

What data representations are available?

The Lua C API is stack based so Lua can provide functions to push and pop most simple C data types (integers, floats, etc.) to and from the stack, as well as functions for manipulating tables through the stack.

Is the language compiled or interpreted?

Lua is commonly described as a "multi-paradigm" language, providing a small set of general features that can be extended to fit different problem types.

Lua does not contain explicit support for inheritance but allows it to be implemented with meta-tables.

Lua grants programmers toe ability to implement namespaces, classes, and other related features using its single table implementation.

Discussion of language design criteria based on the material in Chapter 2 of the text.

It was designed from the beginning to be a software that can be integrated with the code written in C and other conventional languages. This integration brings many benefits. It does not try to do what C can already do but aims at offering what C is not good at: a good distance from the hardware, dynamic structures, no redundancies, ease of testing and debugging. For this, Lua has a safe environment, automatic memory management, and good facilities for handling strings and other kinds of data with dynamic size.

The use of this language can be simple if it does only a few things such as a script. Lua really shines when its used for simplicity and lightweight. If it is used as its own application, it can be hard if it is a big project. Things can get confusing since there is no typed variables and creating classes is also hard. If the project is small, it can be easy to use.

Lessons learned while using the language on your project

Spaces are very important, even after a comma, it can cause the game to throwing errors or exceptions