Lua Language Write up

**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)
* **Emacs**
* **Howl**
* **IntelliJ IDEA**
* **Visual Studio**
* **Standalone (IDEs)**
* **Lua-oriented scriptable editor**s (see also LuaEditorSupport)
* **IDEs/Editors** (Not updated for more than one year)

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