Embedding Lua

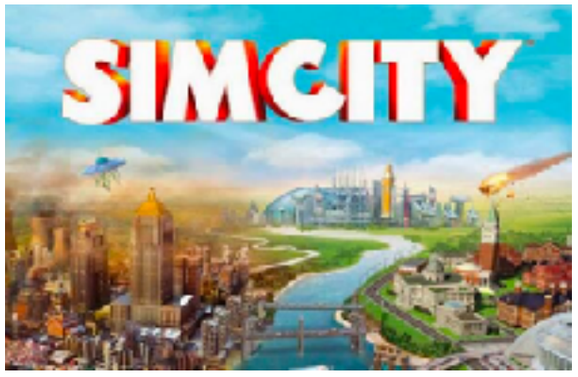
CS 242
October 2, 2017

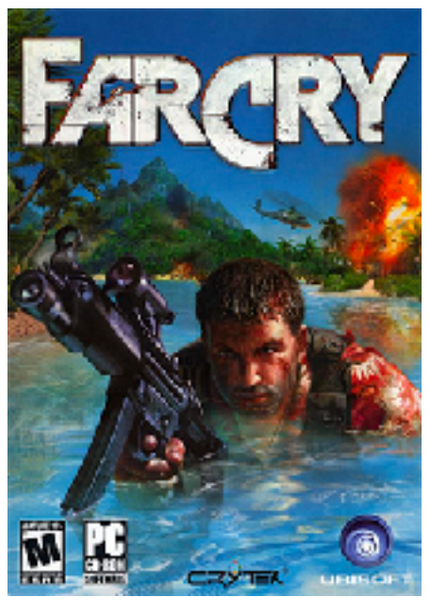
Today's goals

- Understand the motivation for embedding a PL
- Explore the issues of interop between high/low level PLs
- Understand/evaluate Lua's interop solution

Low level code needs high level scripts

Games





Tools

Lua is designed for embedding

- Interpreter is small
  - Lua: 35 files, 14K LOC, 700 KB, ~1s compile time on Will's laptop
  - Python: 4000 files, 900K LOC, 300 MB, ~1.5m compile time

- Language is small
  - Not much syntax
  - Few core language concepts (e.g. no classes)
  - Small standard library

- Relatively simple C API
  - Small language = small API surface
  - Easily sandboxed

C embedded in Lua?

- Easy access to many libraries
  - C ABI is the lowest common denominator for many PLs
- Improve performance of bottlenecks
  - In Python, numpy and stdlib
- Extend language semantics
  - Async I/O, threading, …

Let’s see it in action!

Today's summary

- Two modes of use: Lua in C vs. C in Lua
- Lua uses a stack as an API
  - Prevent user from having to manage memory
  - Provide easy way to manage variadic inputs/outputs
  - Contrasts with Python manual refcounting
- Userdata provides a means of giving C values to Lua

Unit summary

- Lua is a scripting language
  - Dynamic typing, reflection, small language surface, everything is tables
  - Easy to express many programming patterns
  - Easy to shoot yourself in the foot

- Class systems don’t have to be fundamental
  - We built one in lecture, you’ll build one in the assignment
  - Many ways to achieve the same goals, different syntax/perf

- Lua is designed to be embedded
  - Many purposeful choices behind a well-designed C API

Next up...

OCaml!