cisc3665
game design
fall 2011
lecture # IV.3
scripting

topics:

- scripting
- lua

references:

- notes from:
  - *Programming Game AI by Example*, by Mat Buckland. Worldware Publishing, 2005, chapter 6.
  - *AI for Game Developers*, by David M. Bourg and Glenn Seemann. O'Reilly Media, 2004, chapter 8.
- tutorial notes from: `http://lua-users.org/wiki/LuaTutorial`
- demo here: `http://www.lua.org/demo.html`

scripts

- a *script* resides in a file separate from the game engine source code
- the simplest scripts contain things like parameter settings
- the program (game engine) needs to be able to read and parse the script file, and interpret it
- the parser/interpreter component of the game engine is a type of *virtual machine*
- more advanced scripts can contain game logic and/or game objects

interpreted vs compiled

- *interpreted* scripts are read and interpreted by the game engine (virtual machine) in the same format as they are written
- *compiled* scripts are converted into a "compiled" format (e.g., byte code) after they are written; and they are read/interpreted by the game engine (virtual machine) in the compiled format
- the reason to use a compiled format for game scripts is for privacy/secrecy
- if the game script contains logic/strategies for how to win the game and/or secret information about the game structure, then it should be hidden from users—here is where a compiled script makes sense

reasons to have a script

- a script is an easy way to initialize variables (game parameters, game settings)
- using a script can save development time and increase productivity because you don't have to recompile the game engine each time you want to test a different parameter setting
- using a script can also increase creativity, because game designers who are not programmers could write the script
- scripts can improve the extensibility of games, by allowing for customized *mods*

contents of scripts

- scripts can contain:
  - variable / parameter settings
  - dialogue between characters in the game
  - "stage" directions, dictating how characters move around in the game environment in response to changes in the game state and/or dialogue elements
  - AI logic

using scripts

- you can write your own scripting "language"
- you need to define a format for what you want to store in the script
- and you need to write code in your game engine that will read the script and parse its contents, and then interpret the parsed information inside the game
- you can also use scripting languages that are already written by other people and that can interface with the language that your game engine is written in

scripting languages

- there are many scripting languages
- such as unix shell scripts (e.g., sh, bash, ksh, tsh, etc.)
- and languages such as perl, php, ruby, javascript, etc.
- one popular scripting language for games is called **Lua**
- Lua was written to interface with C/C++
- but a Java interface also exists

lua

- `http://www.lua.org` — download here
  `http://lua-users.org/wiki/LuaTutorial` — first tutorial here
  `http://lua-users.org/wiki/TutorialDirectory` — many more tutorials here
- what Lua is used for:
  - configuring applications
  - stand-alone scripting
  - modifying run-time behavior in applications, as an embedded language
- first program: "hello world"

```
-- hello.lua
-- the first program in every language

io.write("Hello world, from ",_VERSION,"!\n")
```

to do

- work on homework assignment for unit IV, which is due November 17
- look at the sample "Rock Paper Scissors" Lua script on the class web page