ETGG3802	Lab1: **Lua + ssuge = awesome!**	**100 points** (148 possible)
Assigned: **1/10/2017**	Due: **Before class on 1/19/2017**	

Objectives:

- Re-familiarize everyone with the existing ssuge codebase
- Add lua dependencies (something most of us still can use some practice on)
- Explore the foundations of our C++ / Lua integration, which will pervade all future development

Tasks:

1. Start early and ask lots of questions – some areas are intentionally left vague-ish
2. Get a working copy of our last ssuge lab (Lab9) from last semester
    a. Mine is available at xxx
    b. If you use yours, at least look at mine to make sure we have similar layouts.
3. Create a ScriptManager class
    a. (**5 points**) General structure, make it a singleton
    b. (**8 points**) Create and set up and destroy a lua_State object (load the default libs, and some custom functions [below])
    c. (**8 points**) Create a **runFile** method that loads a file from disk, requests compilation from and runs a file. Dump any errors (be specific) to the ssuge log.
    d. (**10 points**) an executeScriptMethod (see 4.c)
4. In GameObject.cpp:
    a. (**8 points**) create a **getField** and **setField** method.
        i. For getField:
            1. The sole parameter should be a string (the value they want)
            2. For now, always return a string (we'll make this better later)
            3. For now, support at least these value names: "name", "tag" or a user-defined key
        ii. For setField you'll need to maintain (in C++) a map of key-value pairs
    b. (**3 points**) Create a **setScriptName** method. Very simple, but this makes the game object "script-aware"
        i. This is the module name (see 6.b) to use when invoking script methods.
    c. (**4 points**) Create an **callScriptMethod** method
        i. For now just take a name for the method (e.g. "onCreate"). Later, we'll add the ability to pass parameters.
        ii. Don't do any Lua calls here – I'd prefer you do them in ScriptManager. You can, however, pick out relevant info (the group name, game object name, script name, method name), only the latter of which needs to be passed to this function.
5. Create a GameObject Lua "class" (similar to the Image class in Lab11)
    a. (**10 points**) new method
    b. (**8 points**) __index and __newindex methods
    c. (**5 points**) Properly registering this with the lua interpreter (probably in ScriptManager constructor)
    d. (**10 points**) "Wrapper" methods. These are pretty simple once you see the pattern (see sample script for the list)
6. (**20 points**) Properly organizing / managing the lua global state:
    a. Create a global lua table for each GOM group
    b. Create a global variable (with the file name, minus the ".lua") for each script referenced by a GameObject.
7. (**17 points**) Create these "top-level" global functions (see the sample script for the list)
    a. I'd strongly recommend you get the log function working ASAP – it was an invaluable tool for me.
8. (**4 points**) Call init.lua from Application's create scene.
9. (**20 bonus points**) for good error detection
    a. Some points: check for errors in lua-c calls
    b. More points: when an error is triggered in script, generate a full stack trace / dump (and by stack I mean lua's "normal" stack *and* the invocation stack [sometimes called the call stack])
10. (**10 points**) Proper submission (a working bin/Release folder plus your /src and /include in one zip / 7z file)