CSCE 693 Software Evolution

Homework 2 – Enhancing a Simple SDL-based Game Loop (100 pts)

Directions:

- No written report is required, but it is expected that software code should be liberally commented to clearly indicate the program logic that was implemented.
- Use Linux, GCC, and the provided Git repository to complete this homework.
- Clone the csce693-2020 repo: "git clone https://github.com/doughodson/csce693-2020"
- Complete the two skeleton projects: "game_config" which should be an enhanced version of the "game_loop" example that reads the Lua script named "config.lua", and "game_logic" which should be an enhanced version of the "game_loop" example that reads the Lua script "logic.lua" which defines an "update()" function.
- Skeleton projects for both "game_config" and "game_logic" have been created with the initial C++ source code, Makefile and Lua files.
- Consider *forking* a copy of the public repository so that it can be jointly viewed and edited by your team members.
- Several references to complete this homework are available: the provided books (on the L: drive) and numerous Internet-based websites present the Lua C-API with examples. For sol2, use the extensive online documentation as a guide too properly use the templates it defines – the online tutorial is a rich source of information. Also, consider reviewing the provided "sol_test" example.
- To better understand the "game_loop" example, it's recommended to watch the YouTube posted videos: "How to Make A Game in C++ & SDL2 From Scratch!" starting here: https://www.youtube.com/playlist?list=PLhfAbcv9cehhkG7ZQK0nfIGJC_C-wSLrx

Tasks:

1. (25) Complete the skeleton "game_config" project so that it enhances the provided game_loop example by reading the provided Lua script "config.lua" which defines initial SDL2 window dimensions (x, y, width, and height). Use the Lua API to access this file – do not use the sol2 templates.
2. (75) Complete the skeleton "game_logic" project so that it enhances the provided "game_loop" example by reading the Lua script "logic.lua" which defines a function called "update()". The C++-based update() function should call the Lua update() function which increments a counter variable and returns it. Print this value (in the C++ code) to the console, just like the original "game_loop" example. Expected output should be the current value of the counter that the Lua interpreter is incrementing (e.g., 1, 2, 3, 4…).

Submit homework files (in a zip archive named "team0<x>.zip") to my personal email at: doug@sidechannel.net Submit to me ONLY the two projects - not the entire repo! I would

expect to see a zip archive that includes two top-level directories “game_config” and “game_logic” with a few files in each (e.g., “config.lua”, main.cpp, Makefile, etc.).

I know the composition of the teams for grading purposes, but cc’ing your team mates on submission is always a nice thing in the case there is some confusion. ONLY submit original source files - do NOT include miscellaneous compiler-generated files (e.g., .o, final executables, etc.).