200 lines (162 sloc) | 5.98 KB

```
/*
        Embedding Lua in C++ Part #1
        "Stupid 3D printer, I didn't want a nest..." - javidx9

        License (OLC-3)
        ~~~~~~~~~~~~~~~

        Copyright 2018-2019 OneLoneCoder.com

        Redistribution and use in source and binary forms, with or without
        modification, are permitted provided that the following conditions
        are met:

        1. Redistributions or derivations of source code must retain the above
        copyright notice, this list of conditions and the following disclaimer.

        2. Redistributions or derivative works in binary form must reproduce
        the above copyright notice. This list of conditions and the following
        disclaimer must be reproduced in the documentation and/or other
        materials provided with the distribution.

        3. Neither the name of the copyright holder nor the names of its
        contributors may be used to endorse or promote products derived
        from this software without specific prior written permission.

        THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
        "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
        LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
        A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
```

```
30          HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
31          SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
32          LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
33          DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
34          THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
35          (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
36          OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
37
38          Relevant Video: https://youtu.be/4l5HdmPoynw
39
40          Links
41          ~~~~~
42          YouTube:        https://www.youtube.com/javidx9
43                                  https://www.youtube.com/javidx9extra
44          Discord:        https://discord.gg/WhwHUMV
45          Twitter:        https://www.twitter.com/javidx9
46          Twitch:         https://www.twitch.tv/javidx9
47          GitHub:         https://www.github.com/onelonecoder
48          Patreon:        https://www.patreon.com/javidx9
49          Homepage:       https://www.onelonecoder.com
50
51          Author
52          ~~~~~~
53          David Barr, aka javidx9, ©OneLoneCoder 2019
54  */
55
56  #include <iostream>
57  #include <string>
58
59  // include Lua, assumes it is local to this file
60  extern "C"
61  {
62  #include "lua535/include/lua.h"
63  #include "lua535/include/lauxlib.h"
64  #include "lua535/include/lualib.h"
65  }
66
67  // Link to lua library
68  #ifdef _WIN32
69  #pragma comment(lib, "lua535/liblua53.a")
70  #endif
71
72
73  // Little error checking utility function
74  bool CheckLua(lua_State *L, int r)
75  {
76          if (r != LUA_OK)
77          {
78                  std::string errormsg = lua_tostring(L, -1);
```

```
79                  std::cout << errormsg << std::endl;
80                  return false;
81          }
82          return true;
83  }
84
85
86  int lua_HostFunction(lua_State *L)
87  {
88          float a = (float)lua_tonumber(L, 1);
89          float b = (float)lua_tonumber(L, 2);
90          std::cout << "[CPP S4] HostFunction(" << a << ", " << b << ") called from Lua" << std::end
91          float c = a * b;
92          lua_pushnumber(L, c);
93          return 1;
94  }
95
96
97  // NOTE !!!
98  // YOU WILL NEED THE "EmbeddingLua1.lua" FILE FROM GITHUB REPO
99
100
101 int main()
102 {
103
104         struct Player
105         {
106                 std::string title;
107                 std::string name;
108                 std::string family;
109                 int level;
110         } player;
111
112         // Create Lua State
113         lua_State *L = luaL_newstate();
114
115         // Add standard libraries to Lua Virtual Machine
116         luaL_openlibs(L);
117
118         // Register our C++ Function in the global Lua space
119         lua_register(L, "HostFunction", lua_HostFunction);
120
121
122         // Load and parse the Lua File
123         if(CheckLua(L, luaL_dofile(L, "EmbeddingLua1.lua")))
124         {
125                 // Stage 1: Just read simple variables
126                 std::cout << "[CPP] Stage 1 - Read Simple Variables" << std::endl;
127                 lua_getglobal(L, "a");
```

```
128                 if (lua_isnumber(L, -1)) std::cout << "[CPP S1] a = " << (int)lua_tointeger(L, -1)
129                 lua_getglobal(L, "b");
130                 if (lua_isnumber(L, -1)) std::cout << "[CPP S1] b = " << (int)lua_tointeger(L, -1)
131                 lua_getglobal(L, "c");
132                 if (lua_isnumber(L, -1)) std::cout << "[CPP S1] c = " << (int)lua_tointeger(L, -1)
133                 lua_getglobal(L, "d");
134                 if (lua_isstring(L, -1)) std::cout << "[CPP S1] d = " << lua_tostring(L, -1) << st
135
136                 // Stage 2: Read Table Object
137                 std::cout << "[CPP] Stage 2 - Read Table (Key/Value pairs)" << std::endl;
138                 lua_getglobal(L, "player");
139                 if (lua_istable(L, -1))
140                 {
141                         lua_pushstring(L, "Name");
142                         lua_gettable(L, -2);
143                         player.name = lua_tostring(L, -1);
144                         lua_pop(L, 1);
145
146                         lua_pushstring(L, "Family");
147                         lua_gettable(L, -2);
148                         player.family = lua_tostring(L, -1);
149                         lua_pop(L, 1);
150
151                         lua_pushstring(L, "Title");
152                         lua_gettable(L, -2);
153                         player.title = lua_tostring(L, -1);
154                         lua_pop(L, 1);
155
156                         lua_pushstring(L, "Level");
157                         lua_gettable(L, -2);
158                         player.level = (int)lua_tointeger(L, -1);
159                         lua_pop(L, 1);
160                 }
161                 std::cout << "[CPP S2] " << player.title << " " << player.name << " of " << player
162
163                 // Stage 3: Call Lua Function
164                 std::cout << "[CPP] Stage 3 - Call Lua Function" << std::endl;
165                 lua_getglobal(L, "CalledFromCPP1");
166                 if (lua_isfunction(L, -1))
167                 {
168                         lua_pushnumber(L, 5.0f);
169                         lua_pushnumber(L, 6.0f);
170                         lua_pushstring(L, "Bwa ha ha!");
171                         std::cout << "[CPP S3] Calling 'CalledFromCPP1' in lua script" << std::end
172                         if (CheckLua(L, lua_pcall(L, 3, 1, 0)))
173                         {
174                                 std::cout << "[CPP S3] 'CalledFromCPP1' returned " << (float)lua_t
175                         }
176                 }
```

```
177
178                 // Stage 4: Call Lua Function, which calls C++ Function
179                 std::cout << "[CPP] Stage 4 - Call Lua Function, whcih in turn calls C++ Function"
180                 lua_getglobal(L, "CalledFromCPP2");
181                 if (lua_isfunction(L, -1))
182                 {
183                         lua_pushnumber(L, 5.0f);
184                         lua_pushnumber(L, 6.0f);
185                         std::cout << "[CPP S4] Calling 'CalledFromCPP2' in lua script" << std::end
186                         if (CheckLua(L, lua_pcall(L, 2, 1, 0)))
187                         {
188                                 std::cout << "[CPP S4] 'CalledFromCPP2' returned " << (float)lua_t
189                         }
190                 }
191         }
192
193
194
195
196
197         system("pause");
198         lua_close(L);
199         return 0;
200 }
```