Speare Code Editor
The Small IDE for Game and Lua Development

Version 0.0.2
Update: 4 Mar 2020

Free, Lightweight, Open Source, Extendable Flexibility

Speare (http://www.sevenuc.com/en/speare.html) is an ultra lightweight code editor and a small IDE that has an efficient code navigation and call routines tracing ability, and has integrated debugging environment for C, C++ and Lua. It was originally developed to providing a native scripting language debugging environment that seamlessly integrated with C and C++.
Speare well supports all kinds of game development that using Lua as its scripting language. It not only has very concise user interface but also has a very flexible architecture to easily add a new programming language in it.

For general information of debugging C and C++ projects with Speare code editor, please refer this document:
http://www.sevenuc.com/download/Small_IDE_for_C_and_C++_development.pdf

Debug C, C++ and Lua Projects

1. Show the debug toolbar

Click main menu "Navigate" → "Toggle Output".

2. Debug toolbar

From left to right: Start, Stop, Step Into, Step Out, Run To/Continue, Step Over/Step Next, Show Watches. The command

"Run To" tell the debugger run to meet a breakpoint or an exception occurred or the program exited. On the rightmost there are three other icons, they are, search items in the stackview, siding stackview, and clean debug output.

Search in the debug output

Click in the output area to let it got focus and use the shortcut key "Command + F" to do the searching.

3. Socket Port

You can set the socket communication port number both used by Debug Server and the editor. Open the Preferences of Speare and select the "Extra" panel and then input your number.

Note: Please remember to empty the port number when you switched to debugging with the default built-in programming languages with default port number.

4. Watches

Watch List: click debug toolbar "Show Watches" to manage watched variables and expressions, batches of watched items can be removed by multiple selection.

a. Whenever values of watched variables changed, debug session will pause at that point.
b. The values of evaluated expressions will be sent as a *"virtual stack node"* that paired the expression and its value together and representing as JSON key and value.

When watched variables and expressions display in stackview, their nodes normally has a different icon and text colour, and always placed on the top of stackview.

Add Watches:

a. Click debug toolbar "Show Watches".
b. Right click stackview and select menu item "Watch Variable".

Remove Watches:

a. Click debug toolbar "Show Watches".
b. Right click stackview and select menu item "Remove Watch".

5. Run Extra Commands
a. Click main menu "Debug" → "Send Debug Command".
b. Right click stackview and select menu item "Send Command".
When send an extra debug command, the input box will prompt in stackview, so please click stackview to let it got focus before select menu item.

6. Show Stackview Item Values
Right click stackview and select menu item "View Value AS", variables can be configured to display as "Hex", "Decimal", "Octal", "UTF-8" and "Unicode" sequences.

Caution:
Please ensure all source files have been dragged in the workspace before start a debug session, because macOS app can't be allowed to access files outside of its sandbox.

Startup Debugging Session

1. Launch a Terminal.app window or tab, and start Debug Server.
Note: *The content directly printed by the debugging program will be displayed in the terminal instead of in the debug "output view".*

2. Click ▷ start button on the "Debug toolbar" to start debugging session.

3. Continually click "Step Over" button several times on the "Debug toolbar" to ignore some initialising steps in debugging session until it reached the main entry point.

The Lua Debugger

The Lua debugger of Speare code editor implemented as a module of Lua (https://www.lua.org), and support all common versions of Lua. You can conveniently enjoy debugging with any kinds of customised Lua interpreter and LuaJIT.

Tested Lua version includes: 5.1.4, 5.1.5, 5.2.4, 5.3.5, 5.3.6, and 5.4.2.

Start Debugging Steps

1. Download Speare Debug Server:
http://sevenuc.com/download/lua_debugger.tar.gz (518KB)

2. Uncompress the tarball
Uncompress it to your local directory. e.g ~/Desktop and take a look at the readme.txt in it.

3. Start the debug server
```
$ cd ~/Desktop/debugger/5.1
$ ./lua_514 server.lua
```

4. Debug session start
Click "Start" button on the debug toolbar of Speare code editor.
Add breakpoint, step in, step out, step next, watch stack trace ...

Replace Lua Interpreter

You can directly replace the Lua interpreter with your own customised version under the debugger directory.

Extend Speare Code Editor for Game engines and Lua development

Speare Code Editor can be easily extended to support any type of game engine and framework that using Lua as scripting language, such as LÖVE 2D, Moai open source game engine, the Gideros mobile game platform, the Corona 2D game engine, the Marmalade game engine, the Cocos2d and cocos2d-x game engine, the OpenResty Nginx web server and the Lapis web framework, the high performance Prosody instant message server, the Torch machine learning framework, the Redis open source database etc, and other countless game engines.

To add scripts to support better debugging game and Lua in Speare code editor, please download guide document and following the description in it.