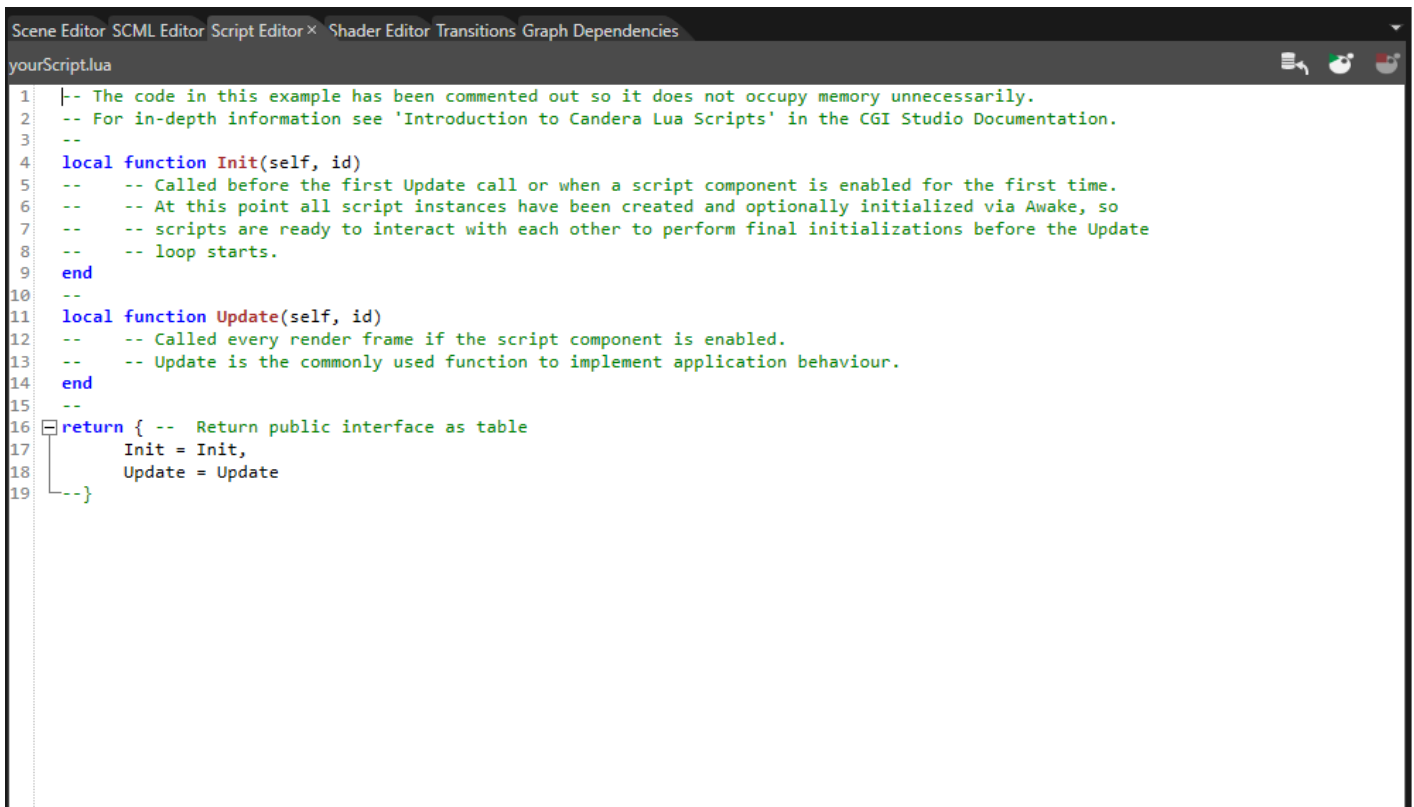


Lua Introduction

Callback Functions

If you open your script in the Script Editor you can see that there are already the most important lines of code. These are the so called callback functions which are called by [Candera](#).

A screenshot of a software interface with a dark theme. At the top, there is a tab bar with several tabs: 'Scene Editor', 'SCML Editor', 'Script Editor' (which is active and has a close button), 'Shader Editor', 'Transitions', and 'Graph Dependencies'. Below the tabs, the title bar of the active window reads 'yourScript.lua'. The main area is a code editor showing a Lua script. The code is as follows:

```
1  |-- The code in this example has been commented out so it does not occupy memory unnecessarily.
2  |-- For in-depth information see 'Introduction to Candera Lua Scripts' in the CGI Studio Documentation.
3  |--
4  local function Init(self, id)
5      |-- -- Called before the first Update call or when a script component is enabled for the first time.
6      |-- -- At this point all script instances have been created and optionally initialized via Awake, so
7      |-- -- scripts are ready to interact with each other to perform final initializations before the Update
8      |-- -- loop starts.
9  end
10 |--
11 local function Update(self, id)
12     |-- -- Called every render frame if the script component is enabled.
13     |-- -- Update is the commonly used function to implement application behaviour.
14 end
15 |--
16 return { -- Return public interface as table
17     Init = Init,
18     Update = Update
19 }
```

Init(self, id)

The Init function gets called before the first Update call or when a script component is enabled for the first time. At this point all script instances have been created and optionally initialized via Awake, so scripts are ready to interact with each other to perform final initializations before the Update loop starts.

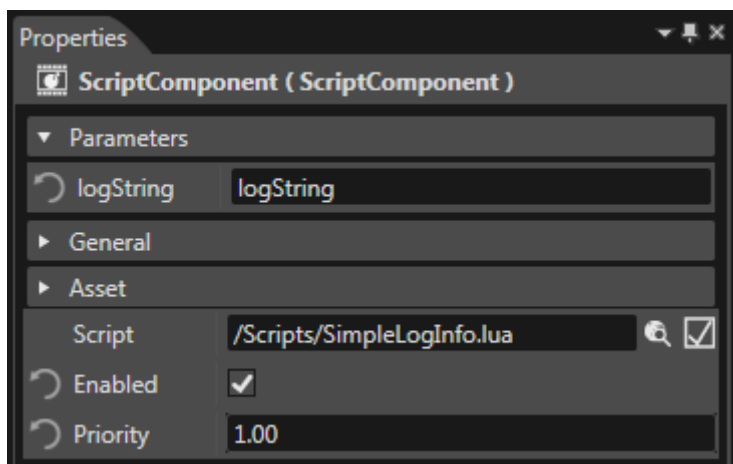
Update(self, id)

Called every render frame if the script component is enabled. Update is the commonly used function to implement application behavior.

Note: if you use one of the callback functions or if you want to make an own function public then don't forget to add them to the public table! (see next paragraph)

return{}

It returns the public interface as table. Variables that have been exposed via the public interface table, and are either of type integer, float, string, or boolean, are automatically displayed in SceneComposer's GUI. In the SceneComposer this variables can be edited and are saved/loaded with the scene. Functions in the return section are not displayed in SceneComposer's GUI but they are public and can get called by another script or by [Candera](#) (see "Callback Functions").



To provide access to variables of the public table of the instance, the table is passed in the function parameter 'self' in all functions that are being called by Candera's script system. So if you define a variable in the public table you can use it in any functions of this script via "self.yourVariable".

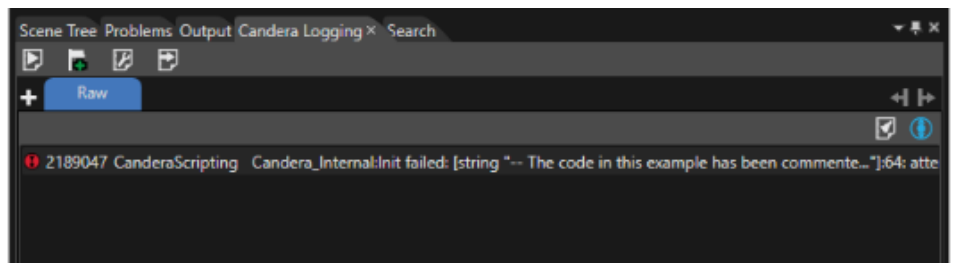
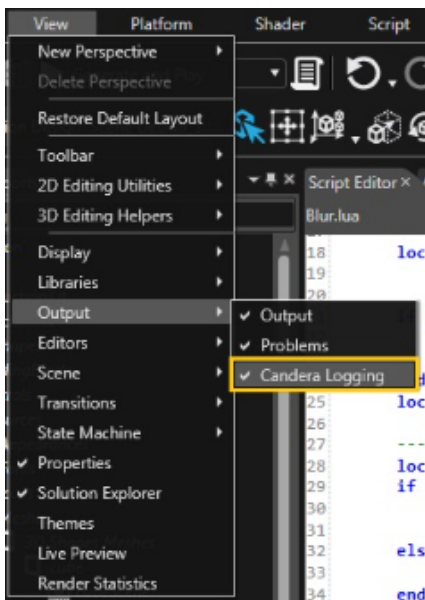
Save and Start a Script

To validate your script press %<F8%> or choose "Script" - "Validate Script" from the menu.

When the script has been validated and there are no errors, you can press the start button to start the execution of the script system and stop it with the second button.



Candera Logging



To check whether your script crashes during runtime or not you can see errors or further information in the [Candera](#) Logging panel. If this panel is not activated go to View -> Output -> and check [Candera](#) Logging

Tip: [Candera](#) Logging is also very useful to test individual parts of your code like if-cases or loops via:

```
Candera.LogInfo(message)
```

Revision #3

Created 2023-02-21 05:15:25 UTC by Tsuyoshi.Kato

Updated 2023-06-13 07:29:30 UTC by Tsuyoshi.Kato