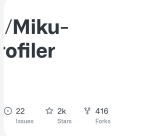


LuaProfiler

由zday项目常达提供，感谢！

Package地址: <http://svn.ifunplus.cn/svn/common/LuaProfiler>
<<http://svn.ifunplus.cn/svn/common/LuaProfiler>>

该工具由下面这个开源工具改写而来，支持LuaJit、Lua5.3, xLua、toLua框架。



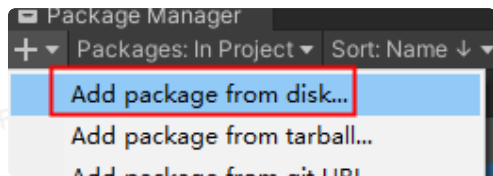
GitHub - leinlin/Miku-LuaProfiler

Contribute to leinlin/Miku-LuaProfiler development by creating an account on GitHub.

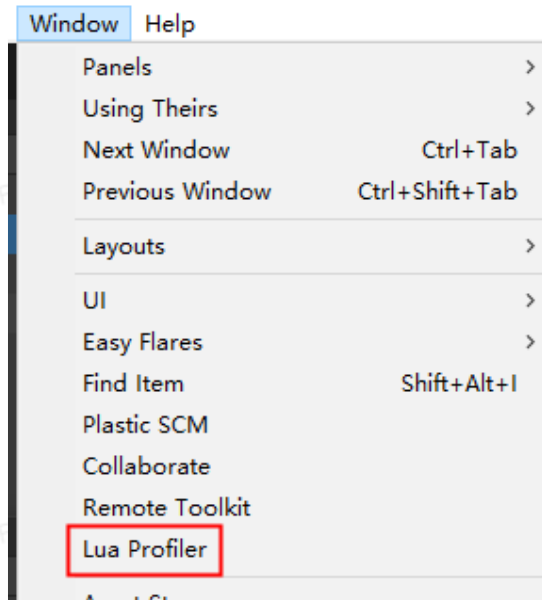
GitHub

安装使用

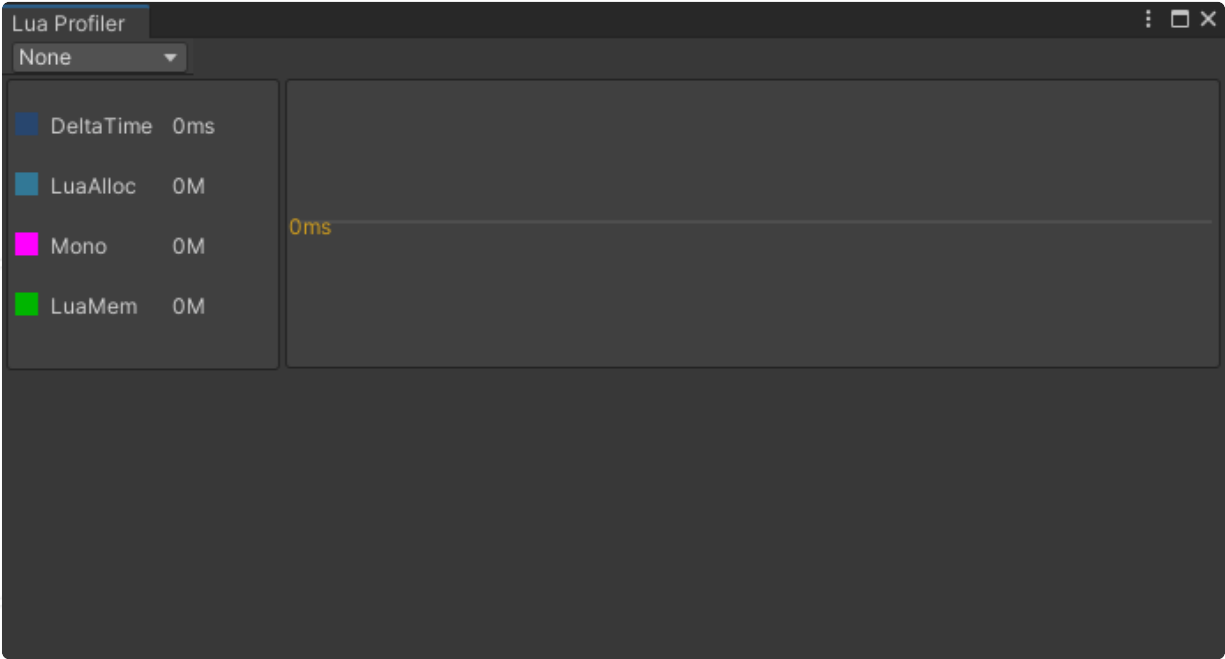
工具以Unity Package形式提供，拷贝至Packages目录或直接通过磁盘安装Package都可以。



点击Window菜单Lua Profiler即可打开工具：

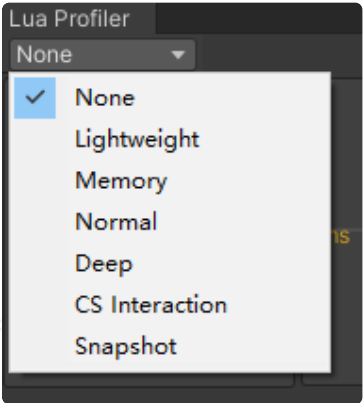


接入工具不需要项目额外进行修改，但目前**仅支持Windows Editor环境**。每次打开Unity Editor **首次启动**游戏时工具是无法工作的（因为Native库加载逻辑的原因），第二次启动游戏后即可正常使用。



主要功能

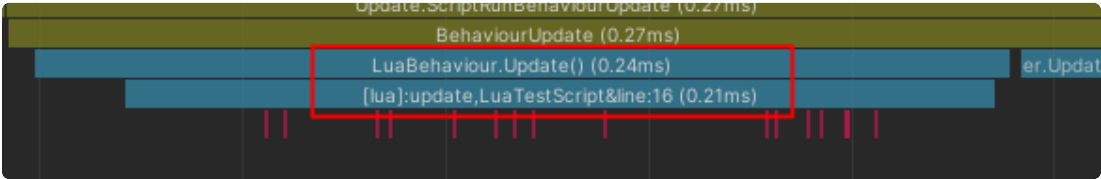
打开后默认模式为None，即什么也不做，可以在启动游戏前设置好模式，**游戏运行期间无法修改模式。**



下面分别介绍几个模式的功能。

Lightweight

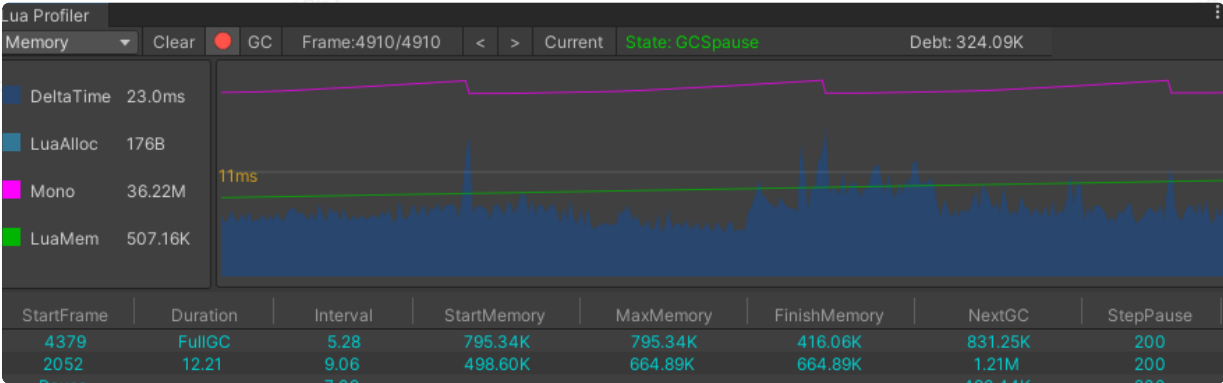
主要的功能是把所有Lua函数用UnityProfiler的BeginSample与EndSample包起来，这样可以在UnityProifler下如同C#函数那样查看Lua的函数执行，工具本身没有任何显示逻辑。



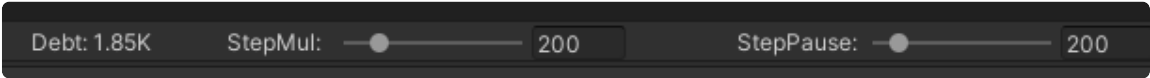
该模式主要用于与C#侧相结合分析逻辑执行与性能。

Memory

该功能也比较简单，就是统计游戏Lua gc的执行情况：每次gc的起始时间、结束时间、内存峰值等。



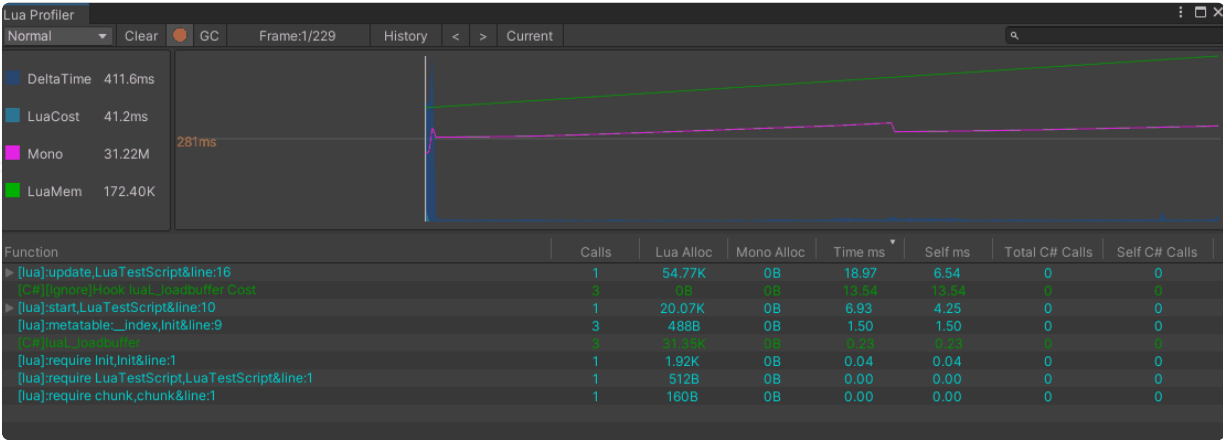
运行时还可以实时调整gc执行的相关参数：



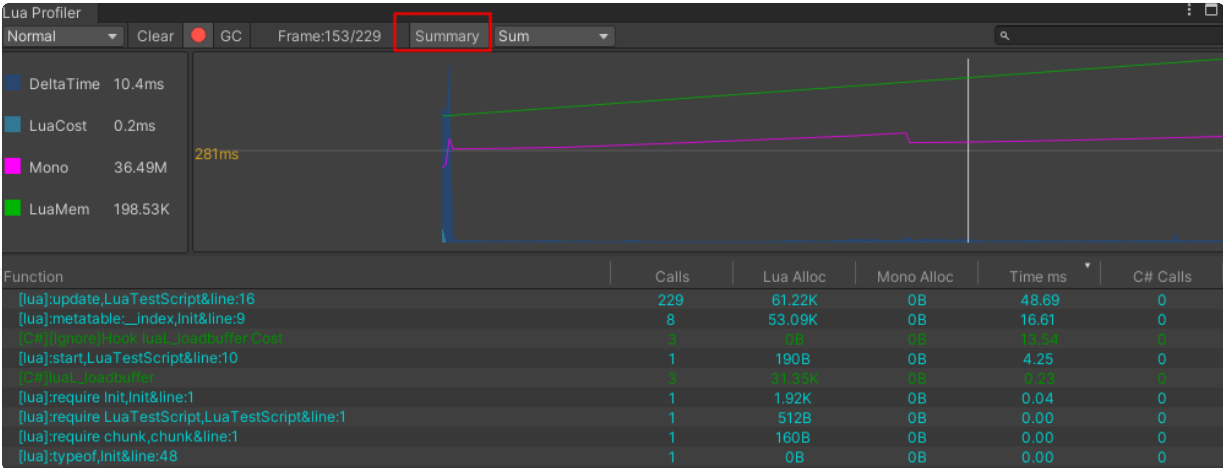
该模式主要是用于测试Lua内存峰值以及相关gc参数的调整测试。

Normal

与Unity Profiler的CPU面板类似，该模式展示了Lua函数的执行行为：每帧都有哪些函数执行、执行堆栈、每个函数执行分配的Lua内存、Mono内存（不准确）以及执行时间等（不准确）。如下图所示：



除了每帧的执行逻辑外，还可以查看累计统计：哪些函数执行次数最多、耗时最多、分配内存最多等。




DelegateBridge等对象，在创建时都会调用luaL_ref，在这些C#对象Dispose时调用luaL_unref。统计面板中列出了被引用的实际Lua对象，以及这些对象当前被引用的个数，通常个数不会特别多（一般来说，个数越多意味着C#侧创建的对象越多），如果持续增长的话，说明C#侧对象一直没有被释放。

通常userdata的问题在于Lua内存泄漏，ref的问题在于C#内存泄漏。

Snapshot

内存快照功能，与Unity的Memory Profiler类似，点击Take Snapshot按钮会创建一个内存快照，统计所有对象的信息（创建快照前会做2次Lua GC和1次Mono GC）：

Lua Profiler		Snapshot		Take Snapshot	Clear	Summary	<	>
	Frame: 4340	Memory: 143.08K GCOjects: 1.49K		Diff				
	Frame: 4666							
					Type	Count	Memory	
					Table	136	48.50K	
					Proto	21	8.52K	
					String	693	23.99K	
					Function	637	50.30K	
					UserData	24	1.07K	
					Thread	1	1.02K	
					global_state	--	9.30K	
					upvalues	--	384B	

如右侧所示，可以看到所有类型的对象个数以及其内存占用情况，可以进一步点开查看细节：

• table

Type	Addr/Key	Value	Attribute	Ref Count	Size	Metatable/Proto	Alias
Table	000000007D189A10		Empty	1	56B	table(000000007D189890)	_R.light(00007FF9F6F72C50)
Table	000000007D188A90	_R	32+32	0	1.55K		
Table	000000007D1889D0	_G	0+64	16	2.05K		_G._G._R[2]._R[7]
Table	000000007D189950	package.loaded	0+16	2	568B		_R._LOADED
Table	000000007D189890		0+1	1	88B		
Table	000000007D189F10	require('package')	0+1	7	312B		_G.package
Table	000000007D189F50	require('package').searchers	8+1	1	184B		
Table	000000007D18AE90	require('package').preload	Empty	2	56B		_R._PRELOAD
Table	000000007D18AAD0	require('coroutine')	0+8	2	312B		_G.coroutine
Table	000000007D18A790	require('table')	0+8	2	312B		_G.table
Table	000000007D18A7D0	require('io')	0+16	2	568B		_G.io
Table	000000007D18ADD0		0+16	5	568B		_R.FILE*
Table	000000007D18A450	require('os')	0+16	2	568B		_G.os
Table	000000007D18A890	require('string')	0+32	3	1.05K		_G.string
Table	000000007D18AB10		0+1	0	88B		
Table	000000007D18A990	require('math')	0+32	2	1.05K		_G.math
Table	000000007D18A1D0	require('utf8')	0+8	2	312B		_G.utf8
Table	000000007D18A250	require('debug')	0+16	2	568B		_G.debug
Table	000000007D18ABD0		0+32	1	1.05K		_G.xlua
Table	000000007D18A550		0+8	1	312B		_G.unix4
Table	000000007D18A510		32+2	1	632B	table(000000007D18AFD0)	_R[3]
Table	000000007D18AFD0		0+1	1	88B		
Table	000000007D18A590		1+2	1	136B		_R[4]
Table	000000007D18A810		0+8	1	312B		_R[5]
Table	000000007D18AD90		0+8	1	312B		_R[6]
Table	000000007D18AB90		0+2	1	120B		_G.template
Table	000000007D18AC10		0+4	5	184B		
Table	000000007D18A210	CS	0+16	2	568B	table(000000007D18AC10)	_G.CS._R.lua_csharp_namespace
Table	000000007D18A910		0+1	1	88B		
Table	000000007D18A950		0+1	4	88B		
Table	000000007D18AA10		0+16	15	568B	table(000000007D18A950)	_R.LuaIndexs
Table	000000007D18B010		0+16	15	568B	table(000000007D18A950)	_R.LuaNewIndexs
Table	000000007D18A290		0+16	13	568B	table(000000007D18A950)	_R.LuaClassIndexs

• string

Type	Addr/Key	Value	Attribute	Ref Count	Size
Short String	0000000220D32D20	_LOADED	7	1	32B
Short String	0000000220D32840	_G	2	2	27B
Short String	0000000220D329F0	assert	6	1	31B
Short String	0000000220D32B40	collectgarbage	14	1	39B
Short String	0000000220D32270	dofile	6	1	31B
Short String	0000000220D322A0	error	5	1	30B
Short String	0000000220D32C60	getmetatable	12	3	37B
Short String	0000000220D324E0	ipairs	6	1	31B
Short String	0000000220D32A50	loadfile	8	1	33B
Short String	0000000220D325D0	load	4	1	29B
Short String	0000000220D32CC0	next	4	1	29B
Short String	0000000220D32B70	pairs	5	1	30B
Short String	0000000220D32AE0	pcall	5	1	30B
Short String	0000000220D32C30	print	5	2	30B
Short String	0000000220D32300	rawequal	8	1	33B
Short String	0000000220D32870	rawlen	6	1	31B
Short String	0000000220D32750	rawget	6	1	31B
Short String	0000000220D32BA0	rawset	6	1	31B

每个对象除了一些值的信息外，还可以看到被引用的情况，点击某对象的Ref Count，展示所有引用它的对象信息，比如：

String 693: 23.99K

Type	Addr/Key	Value	Attribute	Ref Count	Size	Metatable/Proto	Alias
Short String	0000000220D32D20	_LOADED	7	1	32B		
Short String	0000000220D32840	_G	2	2	27B		
Short String	0000000220D329F0	assert	6	1	31B		
Short String	0000000220D32B40	collectgarbage	14	1	39B		
Short String	0000000220D32270	dofile	6	1	31B		
Short String	0000000220D322A0	error	5	1	30B		
Short String	0000000220D32C60	getmetatable	12	3	37B		
Short String	0000000220D324E0	ipairs	6	1	31B		
Short String	0000000220D32A50	loadfile	8	1	33B		
Short String	0000000220D325D0	load	4	1	29B		
Short String	0000000220D32CC0	next	4	1	29B		
Short String	0000000220D32B70	pairs	5	1	30B		
Short String	0000000220D32AE0	pcall	5	1	30B		
Short String	0000000220D32C30	print	5	2	30B		
Short String	0000000220D32300	rawequal	8	1	33B		
Short String	0000000220D32870	rawlen	6	1	31B		
Short String	0000000220D32750	rawget	6	1	31B		
Short String	0000000220D32BA0	rawset	6	1	31B		
Short String	0000000220D32A80	select	6	1	31B		
Short String	0000000220D32AB0	setmetatable	12	3	37B		
Short String	0000000220D32360	tonumber	8	1	33B		
Short String	0000000220D32600	tonstring	8	2	33B		
Short String	0000000220D32330	type	4	5	29B		
Short String	0000000220D323F0	xpcall	6	1	31B		
Short String	0000000220D32C90	_VERSION	8	1	33B		
Short String	0000000220D32420	Lua 5.3	7	1	32B		_G_VERSION
Short String	0000000220D32BD0	package	7	2	32B		
Short String	0000000220D32C00	loadlib	7	1	32B		

还可以对两个快照进行对比，只列出新增的对象：

Lua Profiler

Snapshot Take Snapshot Clear Summary

Frame: 4340
Memory: 143.08K
GCOBJECTS: 1.49K

Diff

Type	Count	Memory
Table	136	48.50K
Proto	21	8.52K
String	693	23.99K
Function	637	50.30K
UserData	24	1.07K
Thread	1	1.02K
global_state	--	9.30K
upvalues	--	384B

样例两次快照没有变化，之后补充更好的例子