

OficinaFramework

2.0.11

Generated by Doxygen 1.8.13

Mon Apr 17 2017 22:32:21

Contents

1	Oficina Framework	1
1.1	About	1
1.2	License	1
1.3	Dependencies	1
1.4	Building Your Own Projects With Oficina2	2
1.5	Building (Without GNU Guile or Lua)	2
1.5.1	Without Guile	2
1.5.2	Without Lua	2
1.5.3	Without Both	2
1.6	Learn	2
2	IronScheme API Reference v0.3.5b	3
2.1	General Scheme Syntax	3
2.2	IronScheme Specific Syntax	3
2.2.1	Global symbols	3
2.2.2	Oficina Common API	4
2.2.2.1	Output	4
2.2.2.2	Input	6
2.2.2.3	Display	8
2.2.3	Oficina Entity API	9
2.2.3.1	Referencing objects	9
2.2.3.2	Retrieving components	10
2.2.3.3	Object transformation	10
2.2.4	Oficina Rendering API	13
2.2.4.1	General Rendering API	14
2.2.4.2	Animation API	14
2.3	Usage Guide	17
2.3.1	Basic Example	17
2.3.2	A More Complex Example	17
2.4	IronScheme Example	18

3	IronLua API Reference v0.1.5b	21
3.1	General Lua Syntax	21
3.2	IronLua Specific Syntax	21
3.2.1	Global symbols	21
3.2.2	Oficina Common API	22
3.2.2.1	Output	22
3.2.2.2	Input	24
3.2.2.3	Display	26
3.2.3	Oficina Entity API	27
3.2.3.1	Referencing objects	27
3.2.3.2	Retrieving components	27
3.2.3.3	Object transformation	28
3.2.4	Oficina Rendering API	31
3.2.4.1	General Rendering API	31
3.2.4.2	Animation API	32
3.3	Usage Guide	34
3.3.1	Basic Example	35
3.3.2	A More Complex Example	35
3.4	IronLua Example	35
4	GameArgs Reference	39
4.1	What Are GameArgs?	39
4.1.1	General GameArgs	39
4.1.1.1	datad	39
4.1.2	Display GameArgs	40
4.1.2.1	wname	40
4.1.2.2	wicon	40
4.1.2.3	winsz	40
4.1.2.4	frmrt	41
4.1.3	Context GameArgs	41
4.1.3.1	vsync	41

5 Hierarchical Index	43
5.1 Class Hierarchy	43
6 Class Index	45
6.1 Class List	45
7 File Index	47
7.1 File List	47
8 Class Documentation	49
8.1 oficina::ofAnimator Class Reference	49
8.1.1 Detailed Description	50
8.1.2 Member Function Documentation	50
8.1.2.1 draw()	50
8.1.2.2 GetAnimationRunning()	51
8.1.2.3 GetAnimationSpeed()	51
8.1.2.4 GetCurrentAnimationName()	51
8.1.2.5 GetDefaultAnimationSpeed()	52
8.1.2.6 getPosition()	52
8.1.2.7 init()	52
8.1.2.8 isInit()	53
8.1.2.9 reg()	53
8.1.2.10 SetAnimation()	54
8.1.2.11 SetAnimationRunning()	54
8.1.2.12 SetAnimationSpeed()	54
8.1.2.13 SetAnimationTexture()	55
8.1.2.14 setPosition()	55
8.1.2.15 SetRenderer()	55
8.1.2.16 SyncToFrameRate()	56
8.1.2.17 unreg()	56
8.1.2.18 update()	56
8.2 oficina::ofBuffer Class Reference	56

8.2.1	Detailed Description	57
8.2.2	Member Function Documentation	57
8.2.2.1	getName()	58
8.2.2.2	isInit()	58
8.2.2.3	operator=()	58
8.2.2.4	setData()	58
8.3	oficina::ofCanvas Class Reference	59
8.3.1	Detailed Description	59
8.3.2	Member Function Documentation	59
8.3.2.1	init()	60
8.3.2.2	load()	60
8.3.2.3	remove()	60
8.3.2.4	update()	60
8.4	oficina::ofCanvasManager Class Reference	61
8.4.1	Detailed Description	62
8.4.2	Member Enumeration Documentation	62
8.4.2.1	ofDebuggerState	62
8.4.3	Member Function Documentation	62
8.4.3.1	add()	62
8.4.3.2	dbg_callEval()	63
8.4.3.3	dbg_ChangeState()	63
8.4.3.4	dbg_getState()	63
8.4.3.5	dbg_ReplLineNumber()	64
8.4.3.6	dbg_ReplOutputStream()	64
8.4.3.7	dbg_setFont()	64
8.4.3.8	draw()	64
8.4.3.9	getCanvasList()	65
8.4.3.10	remove()	65
8.4.3.11	unload()	65
8.4.3.12	update()	65

8.5	ofContext Class Reference	66
8.5.1	Detailed Description	66
8.5.2	Member Function Documentation	66
8.5.2.1	viewportSize()	67
8.5.2.2	isInit()	67
8.5.2.3	open()	67
8.5.2.4	pushArg()	67
8.5.2.5	setClearColor()	68
8.5.2.6	setViewportSize()	68
8.6	ofDisplay Class Reference	68
8.6.1	Detailed Description	69
8.6.2	Member Function Documentation	69
8.6.2.1	close()	69
8.6.2.2	getDeltaTime()	70
8.6.2.3	getHandle()	70
8.6.2.4	getSize()	70
8.6.2.5	isFullscreen()	70
8.6.2.6	isOpen()	71
8.6.2.7	open()	71
8.6.2.8	pushArg()	71
8.6.2.9	setFullscreen()	71
8.6.2.10	setSize()	72
8.6.2.11	setSwapInterval()	72
8.6.2.12	swap()	72
8.7	ofElementBuffer Class Reference	73
8.7.1	Detailed Description	73
8.7.2	Member Function Documentation	73
8.7.2.1	draw()	73
8.7.2.2	getCount()	74
8.7.2.3	getType()	74

8.7.2.4	setCount()	74
8.7.2.5	setProps()	74
8.7.2.6	setType()	75
8.8	ofEntity Class Reference	75
8.8.1	Detailed Description	77
8.8.2	Member Function Documentation	77
8.8.2.1	AddComponent()	77
8.8.2.2	draw()	78
8.8.2.3	DrawComponents()	78
8.8.2.4	GetComponent()	78
8.8.2.5	getEulerAngles()	78
8.8.2.6	getModelMatrix()	79
8.8.2.7	getName()	79
8.8.2.8	getPosition()	79
8.8.2.9	getProperty()	79
8.8.2.10	getPropertyMask()	80
8.8.2.11	getScale()	80
8.8.2.12	init()	80
8.8.2.13	load()	81
8.8.2.14	RemoveComponent()	81
8.8.2.15	rotate()	82
8.8.2.16	scale()	82
8.8.2.17	setName()	82
8.8.2.18	setProperty()	83
8.8.2.19	toggleProperty()	83
8.8.2.20	translate()	83
8.8.2.21	update()	84
8.8.2.22	UpdateComponents()	84
8.8.3	Member Data Documentation	84
8.8.3.1	rotation	84

8.8.3.2	scaling	85
8.8.3.3	translation	85
8.9	oficina::ofFont Class Reference	85
8.9.1	Detailed Description	86
8.9.2	Member Function Documentation	86
8.9.2.1	getGlyphSize()	86
8.9.2.2	init()	86
8.9.2.3	isInit()	87
8.9.2.4	measure()	87
8.9.2.5	operator=()	87
8.9.2.6	write()	88
8.10	oficina::ofFrameSpan Class Reference	88
8.10.1	Detailed Description	89
8.10.2	Member Function Documentation	89
8.10.2.1	isRunning()	89
8.10.2.2	resetSpan()	89
8.10.2.3	stop()	89
8.10.2.4	yieldSpan()	90
8.11	oficina::ofIComponent Class Reference	90
8.11.1	Detailed Description	91
8.11.2	Member Function Documentation	91
8.11.2.1	draw()	91
8.11.2.2	getType()	91
8.12	oficina::ofInputState Struct Reference	92
8.12.1	Detailed Description	92
8.13	oficina::ofLua Class Reference	92
8.13.1	Detailed Description	93
8.13.2	Member Function Documentation	94
8.13.2.1	getBoolean()	94
8.13.2.2	getComponent()	94

8.13.2.3	getEntity()	94
8.13.2.4	getInteger()	95
8.13.2.5	getNumber()	95
8.13.2.6	getString()	95
8.13.2.7	isInit()	96
8.13.2.8	loadfile()	96
8.13.2.9	regFunc()	96
8.13.2.10	regSym() [1/6]	97
8.13.2.11	regSym() [2/6]	97
8.13.2.12	regSym() [3/6]	97
8.13.2.13	regSym() [4/6]	98
8.13.2.14	regSym() [5/6]	98
8.13.2.15	regSym() [6/6]	98
8.13.2.16	update()	99
8.14	ofPrimitive Struct Reference	99
8.14.1	Detailed Description	100
8.15	ofPrimitiveRenderer Class Reference	100
8.15.1	Detailed Description	100
8.15.2	Member Function Documentation	100
8.15.2.1	draw()	100
8.15.2.2	makePrimitive()	101
8.16	ofScheme Class Reference	101
8.16.1	Detailed Description	102
8.16.2	Member Function Documentation	102
8.16.2.1	getSymRef()	102
8.16.2.2	isInit()	103
8.16.2.3	loadfile()	103
8.16.2.4	regFunc()	103
8.16.2.5	regSym()	104
8.16.2.6	update()	104

8.17 oficina::ofShader Class Reference	105
8.17.1 Detailed Description	106
8.17.2 Member Function Documentation	106
8.17.2.1 compile()	106
8.17.2.2 getName()	106
8.17.2.3 init()	106
8.17.2.4 isCompiled()	107
8.17.2.5 isInit()	107
8.17.2.6 operator=()	107
8.17.2.7 setSource()	107
8.18 oficina::ofShaderAttribute Class Reference	108
8.18.1 Detailed Description	109
8.18.2 Member Function Documentation	109
8.18.2.1 bindVertexArrayData()	109
8.18.2.2 getSize()	109
8.18.2.3 getStride()	109
8.18.2.4 getType()	110
8.18.2.5 isAutoNormalizing()	110
8.18.2.6 isValid()	110
8.18.2.7 operator=()	110
8.18.2.8 setAutoNormalize()	111
8.18.2.9 setProps()	111
8.18.2.10 setSize()	111
8.18.2.11 setStride()	112
8.18.2.12 setType()	112
8.19 oficina::ofShaderProgram Class Reference	112
8.19.1 Detailed Description	113
8.19.2 Member Function Documentation	113
8.19.2.1 attach()	113
8.19.2.2 attachUnload()	114

8.19.2.3	bindFragmentDataLocation()	114
8.19.2.4	getAttributeLocation()	114
8.19.2.5	getName()	115
8.19.2.6	getUniformLocation()	115
8.19.2.7	isInit()	115
8.19.2.8	isLinked()	116
8.19.2.9	link()	116
8.19.2.10	operator=()	116
8.19.2.11	use()	116
8.20	oficina::ofShaderUniform Class Reference	117
8.20.1	Detailed Description	118
8.20.2	Member Function Documentation	118
8.20.2.1	isValid()	118
8.20.2.2	operator=()	118
8.20.2.3	set() [1/21]	119
8.20.2.4	set() [2/21]	119
8.20.2.5	set() [3/21]	119
8.20.2.6	set() [4/21]	120
8.20.2.7	set() [5/21]	120
8.20.2.8	set() [6/21]	120
8.20.2.9	set() [7/21]	120
8.20.2.10	set() [8/21]	121
8.20.2.11	set() [9/21]	121
8.20.2.12	set() [10/21]	121
8.20.2.13	set() [11/21]	122
8.20.2.14	set() [12/21]	122
8.20.2.15	set() [13/21]	122
8.20.2.16	set() [14/21]	122
8.20.2.17	set() [15/21]	123
8.20.2.18	set() [16/21]	123

8.20.2.19 set() [17/21]	123
8.20.2.20 set() [18/21]	124
8.20.2.21 set() [19/21]	124
8.20.2.22 set() [20/21]	124
8.20.2.23 set() [21/21]	125
8.21 oficina::ofTexture Class Reference	125
8.21.1 Detailed Description	125
8.21.2 Member Function Documentation	126
8.21.2.1 bind()	126
8.21.2.2 getFileName()	126
8.21.2.3 getSize()	126
8.21.2.4 isLoaded()	126
8.21.2.5 operator()()	127
8.21.2.6 operator=()	127
8.21.2.7 unbind()	127
8.22 oficina::ofTexturePool Class Reference	127
8.22.1 Detailed Description	128
8.22.2 Member Function Documentation	128
8.22.2.1 load() [1/2]	128
8.22.2.2 load() [2/2]	129
8.22.2.3 loadDefaultFont()	129
8.22.2.4 unload()	129
8.23 oficina::ofTextureRenderer Class Reference	130
8.23.1 Detailed Description	130
8.23.2 Member Function Documentation	130
8.23.2.1 init()	130
8.23.2.2 isInit()	131
8.23.2.3 operator=()	131
8.23.2.4 render()	131
8.23.2.5 SetTexture()	132

8.23.2.6	unload()	132
8.24	oficina::ofTimeSpan Class Reference	132
8.24.1	Detailed Description	133
8.24.2	Member Function Documentation	133
8.24.2.1	isRunning()	133
8.24.2.2	resetSpan()	133
8.24.2.3	stop()	134
8.24.2.4	yieldSpan()	134
8.25	oficina::ofVertexArray Class Reference	134
8.25.1	Detailed Description	135
8.25.2	Member Function Documentation	135
8.25.2.1	draw()	135
8.25.2.2	operator=()	135
8.26	oficina::ofVertexBuffer Class Reference	136
8.26.1	Detailed Description	136
9	File Documentation	137
9.1	benchmark.hpp File Reference	137
9.1.1	Detailed Description	137
9.1.2	Function Documentation	137
9.1.2.1	ofBenchmarkIsRunning()	138
9.1.2.2	ofBenchmarkStart()	138
9.2	benchmark.hpp	138
9.3	canvas.hpp File Reference	138
9.3.1	Detailed Description	139
9.4	canvas.hpp	139
9.5	display.hpp File Reference	140
9.5.1	Detailed Description	141
9.5.2	Enumeration Type Documentation	141
9.5.2.1	ofFrameRateConfig	141
9.6	display.hpp	141

9.7	entity.hpp File Reference	142
9.7.1	Detailed Description	143
9.8	entity.hpp	143
9.9	input.hpp File Reference	144
9.9.1	Detailed Description	146
9.9.2	Enumeration Type Documentation	146
9.9.2.1	ofMouseButton	146
9.9.2.2	ofPadButton	147
9.9.2.3	ofPlayer	148
9.9.2.4	ofStick	148
9.9.2.5	ofStickAxis	148
9.9.2.6	ofStickSignal	149
9.9.3	Function Documentation	149
9.9.3.1	ofButtonPress()	149
9.9.3.2	ofButtonTap()	150
9.9.3.3	ofGetInputState()	150
9.9.3.4	ofGetLeftStick()	151
9.9.3.5	ofGetLeftTrigger()	151
9.9.3.6	ofGetMousePos()	152
9.9.3.7	ofGetRightStick()	152
9.9.3.8	ofGetRightTrigger()	152
9.9.3.9	ofGetTextInput()	153
9.9.3.10	ofIsGamepadConnected()	153
9.9.3.11	ofIsInputtingText()	153
9.9.3.12	ofMapButtonRemove()	153
9.9.3.13	ofMapDefaultsP1()	154
9.9.3.14	ofMapKeyToButton()	154
9.9.3.15	ofMapKeyToStick()	155
9.9.3.16	ofMappingClear()	155
9.9.3.17	ofMapStickRemove()	155

9.9.3.18	ofMouseButtonPress()	156
9.9.3.19	ofMouseButtonTap()	156
9.9.3.20	ofSetTextInput()	157
9.9.3.21	ofStartTextInput()	157
9.9.3.22	ofStickMovedTowards()	157
9.9.3.23	ofStopTextInput()	158
9.9.3.24	ofTextInputSetPadding()	158
9.9.3.25	ofUpdateEventDispatch()	159
9.10	input.hpp	159
9.11	io.hpp File Reference	161
9.11.1	Detailed Description	162
9.11.2	Enumeration Type Documentation	162
9.11.2.1	ofLogLevel	162
9.11.2.2	ofLogType	163
9.11.3	Function Documentation	163
9.11.3.1	ofFindAsset()	163
9.11.3.2	ofLoadImage()	163
9.11.3.3	ofLoadText()	164
9.11.3.4	ofLog()	164
9.11.3.5	ofLogGetType()	165
9.11.3.6	ofLogSetLevel()	165
9.11.3.7	ofLogUseFile()	165
9.11.3.8	ofSetDataDirectoryName()	165
9.12	io.hpp	166
9.13	oficina.hpp File Reference	167
9.13.1	Detailed Description	168
9.13.2	Enumeration Type Documentation	168
9.13.2.1	ofReplType	168
9.13.3	Function Documentation	169
9.13.3.1	ofGameLoop()	169

9.13.3.2	ofGetReplType()	169
9.13.3.3	ofGetWindowSize()	169
9.13.3.4	ofInit() [1/2]	169
9.13.3.5	ofInit() [2/2]	170
9.13.3.6	ofIsFullscreen()	170
9.13.3.7	ofQuit()	170
9.13.3.8	ofQuitFlagRaised()	170
9.13.3.9	ofReplEval()	170
9.13.3.10	ofSetClearColor()	171
9.13.3.11	ofSetFullscreen()	171
9.13.3.12	ofSetReplType()	171
9.13.3.13	ofSetSwapInterval()	172
9.13.3.14	ofSetWindowSize()	172
9.13.3.15	ofSoftStop()	172
9.14	oficina.hpp	173
9.15	oflua.hpp File Reference	173
9.15.1	Detailed Description	174
9.15.2	Function Documentation	175
9.15.2.1	ofLuaDefineFunc()	175
9.15.2.2	ofLuaDefineSymbol() [1/6]	175
9.15.2.3	ofLuaDefineSymbol() [2/6]	175
9.15.2.4	ofLuaDefineSymbol() [3/6]	176
9.15.2.5	ofLuaDefineSymbol() [4/6]	176
9.15.2.6	ofLuaDefineSymbol() [5/6]	176
9.15.2.7	ofLuaDefineSymbol() [6/6]	177
9.15.2.8	ofLuaEval()	177
9.15.2.9	ofLuaGetBoolean()	177
9.15.2.10	ofLuaGetComponent()	178
9.15.2.11	ofLuaGetEntity()	178
9.15.2.12	ofLuaGetInteger()	179

9.15.2.13 ofLuaGetNumber()	179
9.15.2.14 ofLuaGetString()	180
9.15.2.15 ofLuaIsInit()	180
9.15.2.16 ofLuaUndefine()	180
9.16 oflua.hpp	181
9.17 ofscheme.hpp File Reference	182
9.17.1 Detailed Description	183
9.17.2 Function Documentation	183
9.17.2.1 ofScmDefineFunc()	183
9.17.2.2 ofScmDefineSymbol() [1/3]	184
9.17.2.3 ofScmDefineSymbol() [2/3]	184
9.17.2.4 ofScmDefineSymbol() [3/3]	184
9.17.2.5 ofScmEval()	185
9.17.2.6 ofScmGetReference()	185
9.17.2.7 ofScmIsInit()	185
9.17.2.8 ofScmUndefine()	186
9.18 ofscheme.hpp	187
9.19 platform.hpp File Reference	187
9.19.1 Detailed Description	188
9.20 platform.hpp	189
9.21 render.hpp File Reference	190
9.21.1 Detailed Description	192
9.21.2 Enumeration Type Documentation	192
9.21.2.1 ofBufferUsage	192
9.21.2.2 ofContextType	193
9.21.2.3 ofDataType	193
9.21.2.4 ofFontFaces	194
9.21.2.5 ofPrimitiveType	194
9.21.2.6 ofShaderType	194
9.21.3 Function Documentation	195

9.21.3.1	ofLoadDefaultFragShader()	195
9.21.3.2	ofLoadDefaultShaderProgram()	195
9.21.3.3	ofLoadDefaultVertexShader()	195
9.21.3.4	ofSetVSync()	195
9.21.4	Variable Documentation	196
9.21.4.1	ofDefaultShaderSrc_FS	196
9.21.4.2	ofDefaultShaderSrc_VS	196
9.22	render.hpp	197
9.23	timer.hpp File Reference	202
9.23.1	Detailed Description	203
9.24	timer.hpp	203
9.25	types.hpp File Reference	204
9.25.1	Detailed Description	204
9.25.2	Function Documentation	205
9.25.2.1	ofClamp()	205
9.26	types.hpp	205
Index		207

Chapter 1

Oficina Framework

Copyright (c) 2017 Lucas Vieira.

1.1 About

OficinaFramework is a multiplatform framework for game development, created by Lucas Vieira. It is focused on bringing a layer of accessibility for modern OpenGL games, using C++ as language. While it makes a game developer's life easier, it still brings about a lot of support for advanced system features which other languages and engines insist on hiding. This way, the programmer can tweak the game's performance without a heavyweight system.

1.2 License

This engine is distributed under the LGPL v3.0 license. You can read more about it [here](#).

1.3 Dependencies

- SDL2 \geq 2.0.5
- SDL2_Image \geq 2.0.0
- OpenGL 3.3 support or higher
- GLEW \geq 2.0.0
- GL Mathematics (GLM) \geq 0.9.8
- (Optional) GNU Guile \geq 2.0

1.4 Building Your Own Projects With Oficina2

If you're using a Linux environment, building projects with Oficina2 is very easy, as you can use the `oficina2-config` tool for that.

Suppose you created a file called `game.cpp` which holds the code for your game. To build your oficina2 app with GCC, simply use:

```
{g++}

This will also work with LLVM Clang, and will respect the libraries which you build Oficina2 with, so it's
very important to build
the 'oficina2-config' target on CMake along with Oficina2's static library and test app.

@section build Building
Just 'cd' to the repo's folder and use CMAKE. This will create a static library. You'll then be able to
install it to your path.

```bash
mkdir build
cd build
cmake ..
make
sudo make install
```

## 1.5 Building (Without GNU Guile or Lua)

### 1.5.1 Without Guile

There's a possibility that you don't want to use Scheme, or you don't have GNU Guile installed in your system (specially on Windows, which, by the time of this writing, has no proper port of GNU Guile - some of the ports will make your application crash on startup, due to lack of stable multithread support).

Given that, you might want to disable the IronScheme support. When running CMake, use this command:

Notice that, on Windows systems, IronScheme support is DISABLED BY DEFAULT. This is due to the current GNU Guile version (2.0.14, by the time of this writing)

### 1.5.2 Without Lua

There's also a possibility that you don't want to use Lua, with no scripting nor REPL language at all. In that case, you should run CMake, when building, as the following command:

That will disable building and linking of Lua libraries for Oficina.

### 1.5.3 Without Both

If you want to disable both Scheme and Lua support, just combine the two commands into one:

## 1.6 Learn

You can access Oficina2 documentation online on <https://luksamuk.github.io/oficina> or explore the `examples` folder, which is didatically commented so you can learn the basics.

Be mindful that these libraries are still in development, so some intended features are still lacking. The same applies for tutorials and documentations, which hopefully will be delivered soon.

## Chapter 2

# IronScheme API Reference v0.3.5b

Quick API reference for IronScheme.

### 2.1 General Scheme Syntax

IronScheme is a custom Scheme, powered by GNU Guile. IronScheme uses Guile's features and speed to deliver an in-game REPL, and custom functions and variables organized in modules so you can write the behaviour of your game's entities in Scheme language.

### 2.2 IronScheme Specific Syntax

#### 2.2.1 Global symbols

These symbols are available for use on all functions, and should be used when necessary. All sequential symbols are just aliases for integers; the first of each "collection" always represent the value 0.

#### Players

```
:player-one
:player-two
:player-three
:player-four
```

#### Gamepad Triggers

```
:left-trigger
:right-trigger
```

## Gamepad Buttons

```
:pad-start
:pad-back
:pad-a
:pad-b
:pad-x
:pad-y
:pad-ls
:pad-rs
:pad-d-up
:pad-d-down
:pad-d-left
:pad-d-right
:pad-lb
:pad-lt
:pad-rb
:pad-rt
```

## Mouse Buttons

```
:mouse-left
:mouse-mid
:mouse-right
```

## Coordinate Components

```
:x
:y
:z
:w
```

## Hardcoded Font Faces

```
:typeface-fixedsys
:typeface-gohu
:typeface-fantasque
:typeface-terminus
```

### 2.2.2 Oficina Common API

All functions described here are available to all instantiated Schemes, be it the global Scheme REPL (controlled by `oficina::ofScmXXX` C++ functions) or the object-based Scheme ([oficina::ofScheme](#) class).

To access these functions, one must use the module (`oficina common`).

#### 2.2.2.1 Output

These functions will write or affect directly the debugger's REPL output.

##### 2.2.2.1.1 hex

```
(hex number)
```

Returns a string with a hexadecimal format of 0x00000000 for the given number. Ideal for representing memory pointers.



**Parameters**

<i>number</i>	Number to be formatted.
---------------	-------------------------

**Returns**

A string containing the formatted number.

**2.2.2.1.2 print-hex**

```
(print-hex number)
```

Prints an integer on REPL's output with hexadecimal format 0x00000000.

**Parameters**

<i>number</i>	Number to be printed.
---------------	-----------------------

**Returns**

Undefined.

**2.2.2.1.3 newline**

```
(newline)
```

Inputs new line on REPL's output.

**Returns**

Undefined.

**2.2.2.1.4 clear**

```
(clear)
```

Clears REPL's output.

**Returns**

Undefined.

**2.2.2.1.5 canvas-list**

```
(canvas-list)
```

Shows information on currently loaded canvases.

**Returns**

Undefined.

#### 2.2.2.1.6 quit

```
(quit)
```

Soft stops the entire engine and quits game.

##### Returns

Undefined.

#### 2.2.2.1.7 set-face!

```
(set-face! typename-enum)
```

Changes the font typeface used on the debugger in general.

##### Parameters

<i>typename-enum</i>	Enumeration specifying which typeface should be used. Possible values are :typeface-fixedsys, :typeface-gohu, :typeface-fantasque and :typeface-terminus.
----------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------

##### Returns

Undefined.

#### 2.2.2.2 Input

These functions will get player-related input from game controllers and such.

##### 2.2.2.2.1 lstick?

```
(lstick? #:optional player)
```

Gets player's left stick.

##### Parameters

<i>player</i>	(Optional) Player to be compared. Defaults to :player-one if ignored.
---------------	-----------------------------------------------------------------------

##### Returns

A Scheme vector with two real coordinates ranging from -1.0 to 1.0.

##### 2.2.2.2.2 rstick?

```
(rstick? #:optional player)
```

Gets player's right stick.

**Parameters**

<i>player</i>	(Optional) Player to be compared. Defaults to :player-one if ignored.
---------------	-----------------------------------------------------------------------

**Returns**

A Scheme vector with two real coordinates ranging from -1.0 to 1.0.

**2.2.2.2.3 trigger?**

```
(trigger? which #:optional player)
```

Gets a controller's trigger pressing ratio value, for a specific player's controller.

**Parameters**

<i>which</i>	Specification for which trigger should be compared. You may use :left-trigger or :right-trigger.
<i>player</i>	(Optional) Player to be compared. Defaults to :player-one if ignored.

**Returns**

A real value ranging from 0.0 to 1.0, depending on how much the trigger is being pressed.

**2.2.2.2.4 btnpress?**

```
(btnpress? which #:optional player)
```

Gets whether a button is being held at a specific player's controller.

**Parameters**

<i>which</i>	Button to be compared. All atoms similar to :pad-start, :pad-back and etcetera may be used.
<i>player</i>	(Optional) Player to be compared. Defaults to :player-one if ignored.

**Returns**

Whether the button is being held by the player or not.

**2.2.2.2.5 btntap?**

```
(btntap? which #:optional player)
```

Gets whether a button was pressed on the current frame. Different from btnpress?, a btntap? only lasts for a single frame.

**Parameters**

<i>which</i>	Button to be compared. All atoms similar to :pad-start, :pad-back and etcetera may be used.
<i>player</i>	(Optional) Player to be compared. Defaults to :player-one if ignored.

**Returns**

Whether the button was tapped by the player or not.

**2.2.2.2.6 mousepos?**

```
(mousepos?)
```

Gets the current mouse position.

**Returns**

A vector with two real values representing screen coordinates.

**2.2.2.2.7 mousepress?**

```
(mousepress? which)
```

Gets whether a mouse button is being held.

**Parameters**

<i>which</i>	Mouse button to be compared. May be :mouse-left, :mouse-mid or :mouse-right.
--------------	------------------------------------------------------------------------------

**Returns**

Whether the mouse button is being pressed or not.

**2.2.2.2.8 mousetap?**

```
(mousetap? which)
```

Gets whether a mouse button was tapped on the current frame.

**Parameters**

<i>which</i>	Mouse button to be compared. May be :mouse-left, :mouse-mid or :mouse-right.
--------------	------------------------------------------------------------------------------

**Returns**

Whether the mouse button was tapped or not.

**2.2.2.3 Display**

Display-related functions to get useful information regarding screen size, etc.

#### 2.2.2.3.1 vwprt?

```
(vwprt?)
```

Gets the viewport size.

##### Returns

A vector of two integers containing the current viewport size.

#### 2.2.2.3.2 set-fullscr!

```
(set-fullscr! state)
```

Sets the fullscreen state of the global display.

##### Parameters

<i>state</i>	The new fullscreen state; Active (#t) or not (#f).
--------------	----------------------------------------------------

##### Returns

Undefined.

#### 2.2.2.3.3 fullscr?

```
(fullscr?)
```

Gets the fullscreen state of the global display.

##### Returns

Whether the screen is in fullscreen state or not.

### 2.2.3 Oficina Entity API

These functions serve the purpose of object manipulation.

To access these functions, one must use the module (oficina entity).

#### 2.2.3.1 Referencing objects

Most of these functions will use some of these resources or functions to refer to other objects. Each one holds/returns a handle to an object, which can be searched on the parent object collection.

#### 2.2.3.1.1 `+this+`

`+this+`

Value referencing the current object, the one which loaded the current script. Use this value to save searching time. Each object has a different value.

#### 2.2.3.2 Retrieving components

These functions provide an interface to deal directly with an Entity's components. Said components, from the Scheme interface, will also retain the derived class' type saved on a string. Said type is deduced once the component is attached to an Entity, unless directly specified. For more info, see [oficina::oflComponent](#).

See also

[oficina::oflComponent](#)

##### 2.2.3.2.1 `get-component`

`(get-component name #:optional objref)`

Retrieves a component from a specified entity or from `+this+`.

Parameters

<i>name</i>	Name of the component, as registered on the Entity.
<i>objref</i>	(Optional) Reference to object. Defaults to <code>+this+</code> .

Returns

A SMOB containing a reference to the component, or NIL otherwise.

##### 2.2.3.2.2 `component-type?`

`(component-type? compref)`

Retrieves a string specifying the component's type.

Parameters

<i>compref</i>	Reference to the component.
----------------	-----------------------------

Returns

A string containing the component's type, as deduced when registering said component.

#### 2.2.3.3 Object transformation

Use this to change overall object's properties and matrices.

**2.2.3.3.1 trl!**

```
(trl! coord #:optional load-identity objref)
```

Translates object to/by a coordinate.

**Parameters**

<i>coord</i>	List with translation coordinates in X, Y and Z axis.
<i>load-identity</i>	(Optional) Whether the positioning matrix must be reset before positioning. Defaults to #f.
<i>objref</i>	(Optional) Reference to object. Defaults to +this+.

**2.2.3.3.2 rot!**

```
(rot! theta axis #:optional load-identity objref)
```

Rotates object by an angle around a specified axis.

**Parameters**

<i>theta</i>	Angle of rotation, in radians.
<i>axis</i>	List describing the rotation axis in X, Y and Z axis.
<i>load-identity</i>	(Optional) Whether the rotation matrix must be reset before rotating. Defaults to #f.
<i>objref</i>	(Optional) Reference to object. Default to +this+.

**2.2.3.3.3 scl!**

```
(scl! scl #:optional load-identity objref)
```

Scales object to/by an amount.

**Note**

Scaling defaults to 1.0 on all three axis, so if you feel like resetting the scaling, simply scale all axis by 1.0 and set load-identity to #t.

**Parameters**

<i>scl</i>	List describing scaling factors on X, Y and Z axis. Each axis defaults to 1.0.
<i>load-identity</i>	(Optional) Whether the scaling matrix must be reset before scaling. Different from other functions, defaults to #t.
<i>objref</i>	(Optional) Reference to object. Defaults to +this+.

**2.2.3.3.4 pos?**

```
(pos? #:optional objref)
```

Gets an object's position.

**Parameters**

<i>objref</i>	(Optional) Reference to object. Defaults to +this+.
---------------	-----------------------------------------------------

**Returns**

A VECTOR containing two real values, representing the position of an object.

**2.2.3.3.5 eulerangle?**

```
(eulerangle? axis #:optional objref)
```

Gets the Euler angle related to a specific rotated axis.

**Parameters**

<i>axis</i>	Desired axis coordinate (:x, :y or :z) of rotation to reference.
<i>objref</i>	(Optional) Reference to object. Defaults to +this+.

**Returns**

A real value containing the euler value of the desired axis.

**2.2.3.3.6 mag?**

```
(mag? axis #:optional objref)
```

Gets ratio of magnification (scaling) related to a specific coordinate axis.

**Parameters**

<i>axis</i>	Desired axis coordinate (:x, :y or :z) of magnification to reference.
<i>objref</i>	(Optional) Reference to object. Defaults to +this+.

**Returns**

A real value containing the magnitude of the object on the desired axis.

**2.2.3.3.7 propset!**

```
(propset! which state #:optional objref)
```

Sets a specific property to true or false.

**Parameters**

<i>which</i>	Property index, ranging from 0 to 31.
<i>state</i>	Active (#t) or inactive (#f).
<i>objref</i>	(Optional) Reference to object. Defaults to +this+.



#### 2.2.3.3.8 proptog!

```
(proptog! which #:optional objref)
```

Toggles a specific property's state.

##### Parameters

<i>which</i>	Property index, ranging from 0 to 31.
<i>objref</i>	(Optional) Reference to object. Defaults to +this+.

#### 2.2.3.3.9 propget?

```
(propget? which #:optional objref)
```

Gets whether a property is active or inactive.

##### Parameters

<i>which</i>	Property index, ranging from 0 to 31.
<i>objref</i>	(Optional) Reference to object. Defaults to +this+.

##### Returns

True (#t) or False (#f), depending on the state of the property.

#### 2.2.3.3.10 propmask?

```
(propmask? #:optional objref)
```

Gets the properties mask as an integer. Can be printed with print-hex and formatted with hex.

##### Parameters

<i>objref</i>	(Optional) Reference to object. Defaults to +this+.
---------------	-----------------------------------------------------

##### Returns

An integer value containing the properties mask of an object.

## 2.2.4 Oficina Rendering API

These functions and methods provide interfaces for dealing directly with rendering. Some of them will require components of certain types as parameters, as they're basically clones of those components' methods. You can access these functions using the module (oficina render).

### 2.2.4.1 General Rendering API

These methods should be used to modify the rendering canvas in a live fashion.

#### 2.2.4.1.1 set-clear-color!

```
(set-clear-color! color)
```

Sets the background color when clearing the context on an update.

##### Parameters

<i>color</i>	List containing four real values, expressing normalized values for RGBA, respectively.
--------------	----------------------------------------------------------------------------------------

##### Returns

Undefined.

### 2.2.4.2 Animation API

These methods are clones of most methods containing in [ofAnimator](#) and, therefore, require a reference to an animator component as first parameter to work.

#### 2.2.4.2.1 set-anim!

```
(set-anim! animator name)
```

Sets the current animation on the animator.

##### Parameters

<i>animator</i>	Reference to an <a href="#">ofAnimator</a> component.
<i>name</i>	String containing the name of the new animation to be played.

##### Returns

Undefined.

#### 2.2.4.2.2 set-anim-spd!

```
(set-anim-spd! animator speed)
```

Sets the current animation speed on the animator.

##### Parameters

<i>animator</i>	Reference to an <a href="#">ofAnimator</a> component.
<i>speed</i>	Real value for the animation speed.

**Returns**

Undefined.

**2.2.4.2.3 set-anim-running!**

```
(set-anim-running! animator state)
```

Sets the current playing state of the animation.

**Parameters**

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
<i>state</i>	New playing state of the animation. Can be #t or #f.

**Returns**

Undefined.

**2.2.4.2.4 set-anim-pos!**

```
(set-anim-pos! animator position)
```

Sets the animation position related to the parent entity's matrix.

**Parameters**

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
<i>position</i>	List containing two real values, with the new relative position for the animation.

**Returns**

Undefined.

**2.2.4.2.5 anim-spd?**

```
(anim-spd? animator)
```

Gets the current animation speed.

**Parameters**

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

**Returns**

A real value representing the current animation speed.

**2.2.4.2.6 anim-def-spd?**

```
(anim-def-spd? animator)
```

Gets the default animation speed, defined when the animation was registered.

#### Parameters

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

#### Returns

A real value representing the default animation speed.

#### 2.2.4.2.7 anim-pos?

```
(anim-pos? animator)
```

Gets the current position of the animation related to the parent entity's matrix.

#### Parameters

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

#### Returns

A vector with two real values, representing X and Y coordinates of the animation related to the matrix.

#### 2.2.4.2.8 anim-running?

```
(anim-running? animator)
```

Gets whether the animation is currently running.

#### Parameters

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

#### Returns

Boolean #t or #f representing the current playing state of the animation.

#### 2.2.4.2.9 anim-name?

```
(anim-name? animator)
```

Gets the name of the currently playing animation.

#### Parameters

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

### Returns

A string containing the name of the animation which is currently being played.

## 2.3 Usage Guide

### 2.3.1 Basic Example

Every script needs a module, for the object which the behaviour should be defined, creating two functions that will be exported, to work properly: (init) and (update dt). Below is an example of an empty script with those requirements:

```
(use-modules ((oficina common))
 ((oficina entity)))

(define init
 (lambda ()
 #f))

(define update
 (lambda (dt)
 #f))
```

If you wish to use a more compact form, you can omit the lambda:

```
(use-modules ((oficina common))
 ((oficina entity)))

(define (init)
 #t)

(define (update dt)
 #t)
```

The reason for those functions is that, any time your script is loaded, everything is evaluated. This is why you must encapsulate your code inside functions (or lambdas), so the whole code is not executed at once.

### 2.3.2 A More Complex Example

You can, though, predefine some variables outside of functions for later use. The following example will rotate a specific object by 0.5rad per second in the Z axis:

```
(use-modules ((oficina common))
 ((oficina entity)))

(define *rotation-speed* 0.5)

(define init
 (lambda ()
 #t))

(define update
 (lambda (dt)
 (rot! (* *rotation-speed* dt)
 (list 0.0 0.0 1.0)
 #f)))
```

Notice that, in the first line of code, we define a global object variable called *rotation-speed*. Despite the use of the "define" keyword, it is just a variable.

By multiplying *rotation-speed* by dt, we ensure that the current frame's rotation is corrected so each second spins our object by 0.5rad. dt represents the Delta-Time, which is the amount of time, in seconds (as a real number) the game has taken to get from the last frame to the current frame. If we did not correct our rotation speed on a per-frame basis, the object would spin 0.5rad PER FRAME. That could be dangerous if you're not purposely limiting your frame rate; your game could run at less than 30 or at much more than 1000 frames per second! To better understand that, you can remove the speed correction and try disabling and enabling VSync on Oficina to spot the difference.

## 2.4 IronScheme Example

The example below defines a behaviour for an object called MyObject.

```

;;; MyObject.scm
;;; Oficina Scheme Script file for class MyObject

;;; Imports oficina's general modules so we can
;;; manipulate our object
(use-modules ((oficina common))
 ((oficina entity)))

;;; Default movement speed for object
(define *spd* 300.0)
;;; Default scaling speed for object
(define *scale-speed* 10.0)

;;; Keeps track of object's magnification on X
;;; and Y axis
(define *x-mag* 1.0)
(define *y-mag* 1.0)

;;; This variable will hold a reference to the
;;; component which executes this script
(define *this-component* #f)

(define clamp
 (lambda (value min max)
 "Helper function to clamp a value to a minimum and a maximum value"
 (if (< value min)
 min
 (if (> value max)
 max
 value))))

(define print-object-properties
 (lambda ()
 "Helper function to display stuff on REPL"
 (clear)
 ;; My reference
 (format #t "Entity: ~a\n" +this+)
 (format #t "Component: ~a\n" *this-component*)
 (format #t "Component Type: ~a\n"
 (component-type? *this-component*))
 (format #t "Is a Scheme script? ~a\n"
 (string=? (component-type? *this-component*)
 "oficina::ofScheme"))
 (newline)

 ;; Show player stuff
 (format #t "Object Position: ~a\n" (pos?))
 (format #t "Object Rotation: ~a\n" (eulerangle? :z))
 (format #t "Object Magnification: ~a\n"
 (list (mag? :x) (mag? :y) (mag? :z)))
 (format #t "Object Property Mask: ~a\n" (hex (propmask?))))

(define init
 (lambda ()
 "Initialization function"
 ;; Define three properties for example
 (propset! 1 #t)
 (propset! 31 #t)
 (propset! 30 #t)

 ;; Set initial angle
 (rot! 0.0 (list 0.0 0.0 1.0) #t)

 ;; Set initial position to 200, 200
 (trl! (list 600.0 400.0 0.0) #t)

 ;; Retrieve reference for component
 (set! *this-component* (get-component "Scheme"))))

(define update
 (lambda (dt)
 "Update function"
 ;; Rotate object
 (rot! (* 0.5 dt) (list 0.0 0.0 1.0))

 ;; Magnify/minify object according to
 ;; right stick input

```

```
(let ((rstk (rstick?)))
 (set! *x-mag* (+ *x-mag*
 (* (vector-ref rstk :x) dt)))
 (set! *y-mag* (+ *y-mag*
 (* (vector-ref rstk :y) dt)))
 (set! *x-mag* (clamp *x-mag* 0.5 2.0))
 (set! *y-mag* (clamp *y-mag* 0.5 2.0))
 (scl! (list *x-mag* *y-mag* 1.0) #t))

;; Toggle property #10 when pressing B button
(if (btntap? :pad-b)
 (proptog! 10))

;; Translate object according to stick
(let ((lstk (lstick?)))
 (trl! (list (* *spd* (vector-ref lstk :x) dt)
 (* *spd* (vector-ref lstk :y) dt)
 0.0)))

;; Print object info
(print-object-properties))
```





## Chapter 3

# IronLua API Reference v0.1.5b

Quick API reference for IronLua.

### 3.1 General Lua Syntax

IronLua is an implementation of Lua 5.3 for OficinaFramework. Its goal is to fulfill the need for an object scripting language for Oficina, as well as a language for Oficina's REPL. While it's not always possible to use IronScheme on some platforms, Lua is a wide-support scripting language, featuring minimalist implementation, and can fulfill the need for a scripting language where GNU Guile may fall short.

### 3.2 IronLua Specific Syntax

#### 3.2.1 Global symbols

These symbols are available for use on all scripts, and should be used when necessary. All sequential symbols are just aliases for integers; the first of each "collection" always represent the value 0, EXCEPT for the coordinate components, which start at 1, in conformance to Lua's 1-based approach to numbering.

#### Players

```
PLAYER_ONE
PLAYER_TWO
PLAYER_THREE
PLAYER_FOUR
```

#### Gamepad Triggers

```
LEFT_TRIGGER
RIGHT_TRIGGER
```

## Gamepad Buttons

```
PAD_START
PAD_BACK
PAD_A
PAD_B
PAD_X
PAD_Y
PAD_LS
PAD_RS
PAD_D_UP
PAD_D_DOWN
PAD_D_LEFT
PAD_D_RIGHT
PAD_LB
PAD_LT
PAD_RB
PAD_RT
```

## Mouse Buttons

```
MOUSE_LEFT
MOUSE_MID
MOUSE_RIGHT
```

## Coordinate Components

```
X
Y
Z
W
```

## Hardcoded Font Faces

```
TYPEFACE_FIXEDSYS
TYPEFACE_GOHU
TYPEFACE_FANTASQUE
TYPEFACE_TERMINUS
```

### 3.2.2 Oficina Common API

All functions described here are available to all instantiated Lua scripts, be it the global Lua REPL (controlled by the `oficina::ofLuaXXX` C++ functions) or the object-based Lua script ([oficina::ofLua](#) class).

To access these functions, one must use the prefix `common`.

#### 3.2.2.1 Output

These functions will write or affect directly the debugger's REPL output.

##### 3.2.2.1.1 `format`

```
common.format(format, ...)
```

Prints a formatted string to the REPL. Formatting works with an underlying call to `string.format`, in Lua's string library.

#### Note

The format should follow the same rules as C's `printf`-like functions.

**Parameters**

<i>format</i>	String format to be used.
...	Any variables to be formatted, as long as they fit the input format string.

**Returns**

None (void).

**3.2.2.1.2 hex**

```
common.hex(number)
```

Returns a string with a hexadecimal format of 0x00000000 for the given number. Ideal for representing memory pointers.

**Parameters**

<i>number</i>	Number to be formatted.
---------------	-------------------------

**Returns**

A string containing the formatted number.

**3.2.2.1.3 clear**

```
common.clear()
```

Clears REPL's output.

**Returns**

None (void).

**3.2.2.1.4 canvasList**

```
common.canvasList()
```

Shows information on currently loaded canvases.

**Returns**

None (void).

**3.2.2.1.5 quit**

```
common.quit()
```

Soft stops the entire engine and quits game.

**Returns**

None (void).

**3.2.2.1.6 setFace**

```
common.setFace(typeface)
```

Changes the font typeface used on the debugger in general.

**Parameters**

<i>typename</i>	Enumeration specifying which typeface should be used. Possible values are TYPEFACE_FIXEDSYS, TYPEFACE_GOHU, TYPEFACE_FANTASQUE and TYPEFACE_TERMINUS.
-----------------	-------------------------------------------------------------------------------------------------------------------------------------------------------

**Returns**

None (void).

**3.2.2.2 Input**

These functions will get player-related input from game controllers and such.

**3.2.2.2.1 lstick**

```
common.lstick(player)
```

Gets player's left stick.

**Parameters**

<i>player</i>	(Optional) Player to be compared. Defaults to PLAYER_ONE if ignored.
---------------	----------------------------------------------------------------------

**Returns**

A Lua table with two numeric coordinates ranging from -1.0 to 1.0.

**3.2.2.2.2 rstick**

```
common.rstick(player)
```

Gets player's right stick.

**Parameters**

<i>player</i>	(Optional) Player to be compared. Defaults to PLAYER_ONE if ignored.
---------------	----------------------------------------------------------------------

**Returns**

A Lua table with two numeric coordinates ranging from -1.0 to 1.0.

**3.2.2.2.3 trigger**

```
common.trigger(which, player)
```

Gets a controller's trigger pressing ratio value, for a specific player's controller.

**Parameters**

<i>which</i>	Specification for which trigger should be compared. You may use LEFT_TRIGGER or RIGHT_TRIGGER.
<i>player</i>	(Optional) Player to be compared. Defaults to PLAYER_ONE if ignored.

**Returns**

A real value ranging from 0.0 to 1.0, depending on how much the trigger is being pressed.

**3.2.2.2.4 btnpress**

```
common.btnpress(which, player)
```

Gets whether a button is being held at a specific player's controller.

**Parameters**

<i>which</i>	Button to be compared. All values similar to PAD_START, PAD_BACK and etcetera may be used.
<i>player</i>	(Optional) Player to be compared. Defaults to PLAYER_ONE if ignored.

**Returns**

Whether the button is being held by the player or not.

**3.2.2.2.5 btntap**

```
common.btntap(which, player)
```

Gets whether a button was pressed on the current frame. Different from common.btnpress, a common.btntap only lasts for a single frame.

**Parameters**

<i>which</i>	Button to be compared. All values similar to PAD_START, PAD_BACK and etcetera may be used.
<i>player</i>	(Optional) Player to be compared. Defaults to PLAYER_ONE if ignored.

**Returns**

Whether the button is being held by the player or not.

**3.2.2.2.6 mousepos**

```
common.mousepos()
```

Gets the current mouse position.

**Returns**

A table with two numeric values representing screen coordinates.

### 3.2.2.2.7 mousepress

```
common.mousepress(which)
```

Gets whether a mouse button is being held.

#### Parameters

<i>which</i>	Mouse button to be compared. May be MOUSE_LEFT, MOUSE_MID or MOUSE_RIGHT.
--------------	---------------------------------------------------------------------------

#### Returns

Whether the mouse button is being pressed or not.

### 3.2.2.2.8 mousetap

```
common.mousetap(which)
```

Gets whether a mouse button was tapped on the current frame.

#### Parameters

<i>which</i>	Mouse button to be compared. May be MOUSE_LEFT, MOUSE_MID or MOUSE_RIGHT.
--------------	---------------------------------------------------------------------------

#### Returns

Whether the mouse button was tapped or not.

### 3.2.2.3 Display

Display-related functions to get useful information regarding screen size, etc.

#### 3.2.2.3.1 vwprt

```
common.vwprt()
```

Gets the viewport size.

#### Returns

A table of two numeric values containing the current viewport size.

#### 3.2.2.3.2 setFullscr

```
common.setFullscr(state)
```

Sets the fullscreen state of the global display.

**Parameters**

<code>state</code>	The new fullscreen state; Active (true) or not (false).
--------------------	---------------------------------------------------------

**Returns**

None (void).

**3.2.2.3.3 isFullscr**

```
common.isFullscr()
```

Gets the fullscreen state of the global display.

**Returns**

Whether the screen is in fullscreen state or not.

**3.2.3 Oficina Entity API**

These functions serve the purpose of object manipulation.

To access these functions, one must use the prefix entity.

**3.2.3.1 Referencing objects**

Most of these functions will use some of these resources or functions to refer to other objects. Each one holds/returns a handle to an object, which can be searched on the parent object collection.

**3.2.3.1.1 this**

```
this
```

Lua userdata referencing the current object, the one which loaded the current script. Use this value to save searching time. Each object has a different value.

**3.2.3.2 Retrieving components**

These functions provide an interface to deal directly with an Entity's components. Said components, from the Scheme interface, will also retain the derived class' type saved on a string. Said type is deduced once the component is attached to an Entity, unless directly specified. For more info, see [oficina::oflComponent](#).

**See also**

[oficina::oflComponent](#)

**3.2.3.2.1 getComponent**

```
entity.getComponent(name, objref)
```

Retrieves a component from a specified entity or from this.

**Parameters**

<i>name</i>	Name of the component, as registered on the Entity.
<i>objref</i>	(Optional) Reference to object. Defaults to this.

**Returns**

A Lua userdata containing a reference to the component.

**3.2.3.2.2 componentType**

```
entity.componentType(compref)
```

Retrieves a string specifying the component's type.

**Parameters**

<i>compref</i>	Reference to the component.
----------------	-----------------------------

**Returns**

A string containing the component's type, as deduced when registering said component.

**3.2.3.3 Object transformation**

Use this to change overall object's properties and matrices.

**3.2.3.3.1 translate**

```
entity.translate(coord, loadIdentity, objref)
```

Translates object to/by a coordinate.

**Parameters**

<i>coord</i>	Table with three numeric values, each representing a translation coordinate (X, Y and Z axis).
<i>loadIdentity</i>	(Optional) Whether the positioning matrix must be reset before positioning. Defaults to false.
<i>objref</i>	(Optional) Reference to object. Defaults to this.

**3.2.3.3.2 rotate**

```
entity.rotate(theta, axis, loadIdentity, objref)
```

Rotates object by an angle around a specified axis.



## Parameters

<i>theta</i>	Angle of rotation, in radians.
<i>axis</i>	Table with three numeric values, each representing a rotation axis coordinate (X, Y and Z axis).
<i>loadIdentity</i>	(Optional) Whether the rotation matrix must be reset before positioning. Defaults to false.
<i>objref</i>	(Optional) Reference to object. Defaults to this.

## 3.2.3.3.3 scale

```
entity.scale(scl, loadIdentity, objref)
```

Scales object to/by an amount.

## Note

Scaling defaults to 1.0 on all three axis, so if you feel like resetting the scaling, simply scale all axis by 1.0 and set loadIdentity to true.

## Parameters

<i>scl</i>	Table with three numeric values describing scaling factors on X, Y and Z axis. Each axis defaults to 1.0.
<i>loadIdentity</i>	(Optional) Whether the scaling matrix must be reset before scaling. Different from other functions, defaults to true.
<i>objref</i>	(Optional) Reference to object. Defaults to this.

## 3.2.3.3.4 getPosition

```
entity.getPosition(objref)
```

Gets an object's position.

## Parameters

<i>objref</i>	(Optional) Reference to object. Defaults to this.
---------------	---------------------------------------------------

## Returns

A table containing two numeric values, representing the position of an object.

## 3.2.3.3.5 getEulerAngle

```
entity.getEulerAngle(axis, objref)
```

Gets the Euler angle related to a specific rotated axis.

**Parameters**

<i>axis</i>	Desired axis coordinate (X, Y or Z) of rotation to reference.
<i>objref</i>	(Optional) Reference to object. Defaults to this.

**Returns**

A numeric value containing the euler value of the desired axis.

**3.2.3.3.6 getMagnification**

```
entity.getMagnification(axis, objref)
```

Gets ratio of magnification (scaling) related to a specific coordinate axis.

**Parameters**

<i>axis</i>	Desired axis coordinate (X, Y or Z) of magnification to reference.
<i>objref</i>	(Optional) Reference to object. Defaults to this.

**Returns**

A real value containing the magnitude of the object on the desired axis.

**3.2.3.3.7 setProperty**

```
entity.setProperty(which, state, objref)
```

Sets a specific property to true or false.

**Parameters**

<i>which</i>	Property index, ranging from 1 to 32, as per Lua's 1-based numbering.
<i>state</i>	Active (true) or inactive (false).
<i>objref</i>	(Optional) Reference to object. Defaults to this.

**3.2.3.3.8 toggleProperty**

```
entity.toggleProperty(which, objref)
```

Toggles a specific property's state. Gets whether a property is active or inactive.

**Parameters**

<i>which</i>	Property index, ranging from 1 to 32, as per Lua's 1-based numbering.
<i>objref</i>	(Optional) Reference to object. Defaults to this.

### 3.2.3.3.9 getProperty

```
entity.getProperty(which, objref)
```

Gets whether a property is active or inactive.

#### Parameters

<i>which</i>	Property index, ranging from 1 to 32, as per Lua's 1-based numbering.
<i>objref</i>	(Optional) Reference to object. Defaults to this.

#### Returns

Boolean true or false, depending on the state of the property.

### 3.2.3.3.10 getPropertyMask

```
entity.getPropertyMask(objref)
```

Gets the properties mask as an integer. Can be formatted with hex.

#### Parameters

<i>objref</i>	(Optional) Reference to object. Defaults to this.
---------------	---------------------------------------------------

#### Returns

An integer value containing the properties mask of an object.

## 3.2.4 Oficina Rendering API

These functions and methods provide interfaces for dealing directly with rendering. Some of them will require components of certain types as parameters, as they're basically clones of those components' methods. You can access these functions using the prefix render.

### 3.2.4.1 General Rendering API

These methods should be used to modify the rendering canvas in a live fashion.

#### 3.2.4.1.1 setClearColor

```
render.setClearColor(color)
```

Sets the background color when clearing the context on an update.

**Parameters**

<i>color</i>	Table containing four numeric values, expressing normalized values for RGBA, respectively.
--------------	--------------------------------------------------------------------------------------------

**Returns**

None (void).

**3.2.4.2 Animation API**

These methods are clones of most methods containing in [oficina::ofAnimator](#) and, therefore, require a reference to an animator component as first parameter to work. All of those methods also require a prefix animator.

**3.2.4.2.1 setAnimation**

```
render.animator.setAnimation(animator, name)
```

Sets the current animation on the animator.

**Parameters**

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
<i>name</i>	String containing the name of the new animation to be played.

**Returns**

None (void).

**3.2.4.2.2 setSpeed**

```
render.animator.setSpeed(animator, speed)
```

Sets the current animation speed on the animator.

**Parameters**

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
<i>speed</i>	Numeric value for the animation speed.

**Returns**

None (void).

**3.2.4.2.3 setRunning**

```
render.animator.setRunning(animator, state)
```

Sets the current playing state of the animation.

**Parameters**

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
<i>state</i>	New playing state of the animation. Can be true or false.

**Returns**

None (void).

**3.2.4.2.4 setPosition**

```
render.animator.setPosition(animator, position)
```

Sets the animation position related to the parent entity's matrix.

**Parameters**

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
<i>position</i>	Table containing two numeric values, with the new relative position for the animation.

**Returns**

None (void).

**3.2.4.2.5 getSpeed**

```
render.animator.getSpeed(animator)
```

Gets the current animation speed.

**Parameters**

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

**Returns**

A numeric value representing the current animation speed.

**3.2.4.2.6 getDefaultSpeed**

```
render.animator.getDefaultSpeed(animator)
```

Gets the default animation speed, defined when the animation was registered.

**Parameters**

<i>animator</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

**Returns**

A numeric value representing the default animation speed.

**3.2.4.2.7 getPosition**

```
render.animator.getPosition(animador)
```

Gets the current position of the animation related to the parent entity's matrix.

**Parameters**

<i>animador</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

**Returns**

A table with two numeric values, representing X and Y coordinates of the animation related to the matrix.

**3.2.4.2.8 isRunning**

```
render.animator.isRunning(animador)
```

Gets whether the animation is currently running.

**Parameters**

<i>animador</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

**Returns**

Boolean true or false representing the current playing state of the animation.

**3.2.4.2.9 getName**

```
render.animator.getName(animador)
```

Gets the name of the currently playing animation.

**Parameters**

<i>animador</i>	Reference to an <a href="#">oficina::ofAnimator</a> component.
-----------------	----------------------------------------------------------------

**Returns**

A string containing the name of the animation which is currently being played.

## 3.3 Usage Guide

### 3.3.1 Basic Example

Every script needs two global functions, which will be exported to the Lua object: `init()` and `update(dt)`. Below is an example of an empty script with those requirements:

```
function init()
 -- Your code here
end

function update(dt)
 -- Your code here
end
```

The reason for those functions is that, any time your script is loaded, everything is evaluated. This is why you must encapsulate your code inside functions, so the whole code is not executed at once. Also, there is no requirement for importing Oficina's libraries: those are already defined and will only need their prefixes for accessing its fields and values.

### 3.3.2 A More Complex Example

You can, though, predefine some (local or global) variables or functions outside of the predefined ones for later use. The following example will rotate a specific object by 0.5rad per second in the Z axis:

```
local rotationSpeed = 0.5

function init()
 -- Your code here
end

function update(dt)
 entity.rotate((rotationSpeed * dt),
 {0.0, 0.0, 1.0},
 false)
end
```

Notice that, in the first line of code, we define a global object variable called `rotationSpeed`. The "local" keyword ensures that this variable is not accessible from outside the script object, so it cannot be fetched by C++ code.

By multiplying `rotationSpeed` by `dt`, we ensure that the current frame's rotation is corrected so each second spins our object by 0.5rad. `dt` represents the Delta-Time, which is the amount of time, in seconds (as a numeric real value) the game has taken to get from the last frame to the current frame. If we did not correct our rotation speed on a per-frame basis, the object would spin 0.5rad PER FRAME. That could be dangerous if you're not purposely limiting your frame rate; your game could run at less than 30 or at much more than 1000 frames per second! To better understand that, you can remove the speed correction and try disabling and enabling VSync on Oficina to spot the difference.

## 3.4 IronLua Example

The example below defines a behaviour for an object called `MyObject`.

```

-- MyObject.lua
-- Oficina Lua Script file for class MyObject

-- Default movement speed for object
local spd = 300.0
-- Default scaling speed for object
local scaleSpeed = 10.0

-- Keeps track of object's magnification on X
-- and Y axis
local xMag = 1.0
local yMag = 1.0

-- This variable will hold a reference to the
-- component which executes this script
local thisComponent = nil

-- Helper function to clamp a value to a minimum and a maximum value
local function clamp(value, min, max)
 if (value < min) then return min
 elseif (value > max) then return max end
 return value
end

-- Helper function to display stuff on REPL
local function printObjectProperties()
 common.clear()
 common.format("Entity: %s\n", this)
 common.format("Component: %s\n", thisComponent)
 common.format("Component Type: %s\n", entity.componentType(thisComponent))
 common.format("Is a Lua script? %s\n",
 (entity.componentType(thisComponent) == "oficina:ofLua"))
 common.format("\n")

 -- Show player stuff
 common.format("Object Position: %f, %f\n",
 entity.getPosition()[X], entity.getPosition()[Y])
 common.format("Object Rotation: %f\n", entity.getEulerAngle(Z))
 common.format("Object Magnification: %0.2fx%0.2fx%0.2f\n",
 entity.getMagnification(X), entity.getMagnification(Y),
 entity.getMagnification(Z))
 common.format("Object Property Mask: %s\n", common.hex(entity.getPropertyMask()))
end

-- Initialization function
function init()
 -- Define three properties for example
 entity.setProperty(2, true)
 entity.setProperty(32, true)
 entity.setProperty(31, true)

 -- Set initial angle
 entity.rotate(0.0, {0.0, 0.0, 1.0}, true)

 -- Set initial position
 entity.translate({600.0, 400.0, 0.0}, true)

 -- Retrieve reference to component
 thisComponent = entity.getComponent("Lua")
end

-- Update function
function update(dt)
 -- Rotate object
 entity.rotate((0.5 * dt), {0.0, 0.0, 1.0})

 -- Magnify/minify object according to
 -- right stick input
 local rstk = common.rstick()
 xMag = xMag + (rstk[X] * dt)
 yMag = yMag + (rstk[Y] * dt)
 xMag = clamp(xMag, 0.5, 2.0)
 yMag = clamp(yMag, 0.5, 2.0)
 entity.scale({xMag, yMag, 1.0}, true)

 -- Toggle property #10 when pressing B button
 if common.btntap(PAD_B) then
 entity.toggleProperty(11)
 end

 -- Translate object according to stick
 local lstk = common.lstick()
 entity.translate({lstk[X] * spd * dt,
 lstk[Y] * spd * dt,
 0.0})

```



```
-- Print object info
printObjectProperties()
end
```



## Chapter 4

# GameArgs Reference

GameArgs general reference.

### 4.1 What Are GameArgs?

GameArgs are the one simple way for you to initialize your game's Displays and Contexts. You can directly push a GameArg to an ofDisplay or an ofContext by using these classes' pushArg() method, or to the general display and context of Oficina's default initialization functions by using the general function ofInit.

Here is an example of ofInit, which optionally receives an std::vector of std::strings, each containing your GameArg.

```
oficina::ofInit({
 "wname=My Awesome Game",
 "wicon=res/gameicon.png",
 "winsz=1920x1080",
 "frmt=60c20m",
 "vsync=on"
});
```

#### 4.1.1 General GameArgs

The arguments below can ONLY be used to manipulate the general engine, and will only have effect when called from ofInit.

##### 4.1.1.1 datad

```
datad=DirName
```

Specifies the directory name in which the game's assets are installed.

**Parameters**

<i>DirName</i>	Name of the directory in which the assets are installed. Game assets are searched once on the same executable's folder but, after game installation, if not on the execution folder, game assets will be looked on folders such as /usr/local/share on Linux and C:\Program Files on Windows, and this is where your DirName goes; DirName tells the game exactly on which subfolder these assets are located. For example, if "MyGame" were passed, then the game would look for assets on /usr/local/share/MyGame (Linux) and on C:\Program Files\MyGame on Windows.
----------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**See also**

oficina::ofSetDataDirectoryName

**4.1.2 Display GameArgs**

The arguments below may be used to manipulate the display's behaviour. You can either push those arguments to an ofDisplay or use them on ofInit.

**4.1.2.1 wname**

wname=Name

Defines the name for your display (window's title).

**Parameters**

<i>Name</i>	Name to be given to the window. Everything after the equals sign counts, whitespaces included, so do not enclose it on quotes.
-------------	--------------------------------------------------------------------------------------------------------------------------------

**4.1.2.2 wicon**

wicon=path

Defines the path to the icon file (a supported image file).

**Parameters**

<i>path</i>	Path to the icon file, which the game will load as an in-game image asset. Do not enclose it on quotes.
-------------	---------------------------------------------------------------------------------------------------------

**4.1.2.3 winsz**

winsz=size

Defines the size of the window to be used, once the window is created. This differs from ofSetWindowSize function since said function can only act after the window is created.

## Parameters

<i>size</i>	Size to be assumed by window. The syntax should be WIDTHxHEIGHT (e.g. 800x600), or you can use the literals 720p, 768p, 900p, 1080p, 1440p, 2160p. Other accepted literals are HD (works like 720p), FHD (works like 1080p), FULLHD (works like 1080p), 4K (works like 2160p). Those literals are case-insensitive.
-------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 4.1.2.4 frmt

```
frmt=config
```

Defines how the window should handle framerate when swapping each frame. You can use up to three different configurations, or mix two of them.

## Parameters

<i>config</i>	<p>Configuration string for the framerate. This argument is case-insensitive.</p> <ul style="list-style-type: none"> <li>• If you wish to let the window swap freely, just use the literal <code>VARIABLE</code>;</li> <li>• If you wish to never let the framerate go BEYOND a certain FPS value (that is, you wish to use an FPS capped to a certain value), input the value (floating point or integer) with the suffix <code>C</code>;</li> <li>• If you wish to never let the framerate go BELOW a certain FPS value (purposely let your game slowdown so it doesn't become unplayable; useful when interpolating physics with the <code>deltaTime</code>), input the value (floating point or integer) with the suffix <code>M</code>.</li> <li>• You can also fix a capped FPS and a minimum FPS by inputting two values, each one with a suffix <code>C</code> or <code>M</code>.</li> </ul>
---------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 4.1.3 Context GameArgs

The arguments below may be used to manipulate the context's behaviour. You can either push those arguments to an `ofContext` or use them on `ofInit`. These arguments are case-insensitive.

## 4.1.3.1 vsync

```
vsync=state
```

Defines whether vertical synchro should be on or off.

## Parameters

<i>state</i>	Use either on or off.
--------------	-----------------------



## Chapter 5

# Hierarchical Index

### 5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

oficina::ofBuffer . . . . .	56
oficina::ofElementBuffer . . . . .	73
oficina::ofVertexBuffer . . . . .	136
oficina::ofCanvas . . . . .	59
oficina::ofCanvasManager . . . . .	61
oficina::ofContext . . . . .	66
oficina::ofDisplay . . . . .	68
oficina::ofEntity . . . . .	75
oficina::ofFont . . . . .	85
oficina::ofFrameSpan . . . . .	88
oficina::ofIComponent . . . . .	90
oficina::ofAnimator . . . . .	49
oficina::ofLua . . . . .	92
oficina::ofScheme . . . . .	101
oficina::ofInputState . . . . .	92
oficina::ofPrimitive . . . . .	99
oficina::ofPrimitiveRenderer . . . . .	100
oficina::ofShader . . . . .	105
oficina::ofShaderAttribute . . . . .	108
oficina::ofShaderProgram . . . . .	112
oficina::ofShaderUniform . . . . .	117
oficina::ofTexture . . . . .	125
oficina::ofTexturePool . . . . .	127
oficina::ofTextureRenderer . . . . .	130
oficina::ofTimeSpan . . . . .	132
oficina::ofVertexArray . . . . .	134





## Chapter 6

# Class Index

### 6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">oficina::ofAnimator</a>	Tool for controlling a texture renderer to generate animations . . . . .	49
<a href="#">oficina::ofBuffer</a>	Specifies a generic buffer. Override this class to create your own buffers . . . . .	56
<a href="#">oficina::ofCanvas</a>	Default interface for creating and managing canvases . . . . .	59
<a href="#">oficina::ofCanvasManager</a>	Static class for handling canvases in general . . . . .	61
<a href="#">oficina::ofContext</a>	Describes a context for your display . . . . .	66
<a href="#">oficina::ofDisplay</a>	Represents a single window prepared for receiving a context . . . . .	68
<a href="#">oficina::ofElementBuffer</a>	Represents an Element Buffer object (EBO), useful for holding sequences of vertices for drawing on screen . . . . .	73
<a href="#">oficina::ofEntity</a>	Abstract class representing one ingame entity . . . . .	75
<a href="#">oficina::ofFont</a>	Represents a font . . . . .	85
<a href="#">oficina::ofFrameSpan</a>	Tool for counting and comparing frames, depending of the game's time variation . . . . .	88
<a href="#">oficina::ofIComponent</a>	Defines a single component to be attached to an entity . . . . .	90
<a href="#">oficina::ofInputState</a>	Holds an input state every frame . . . . .	92
<a href="#">oficina::ofLua</a>	Defines one Lua environment to be used inside an entity . . . . .	92
<a href="#">oficina::ofPrimitive</a>	A structure representing a primitive. Can be used for rendering . . . . .	99
<a href="#">oficina::ofPrimitiveRenderer</a>	A static class containing methods for creating and drawing simple primitives onscreen . . . . .	100
<a href="#">oficina::ofScheme</a>	Defines one Scheme environment to be used inside an entity . . . . .	101
<a href="#">oficina::ofShader</a>	Describes a shader . . . . .	105

<a href="#">oficina::ofShaderAttribute</a>	Represents the location of an attribute for the program shader . . . . .	108
<a href="#">oficina::ofShaderProgram</a>	Represents a shader program . . . . .	112
<a href="#">oficina::ofShaderUniform</a>	Represents and handles a shader's uniform . . . . .	117
<a href="#">oficina::ofTexture</a>	Represents a texture on the GPU . . . . .	125
<a href="#">oficina::ofTexturePool</a>	Static object for managing textures. Most (if not all) textures should be loaded using this tool . . . . .	127
<a href="#">oficina::ofTextureRenderer</a>	Tool for easily rendering 2D textures or texture atlases . . . . .	130
<a href="#">oficina::ofTimeSpan</a>	Tool for counting and compare fixed amounts of time, independent from the game's time variation	132
<a href="#">oficina::ofVertexArray</a>	Represents a vertex array for binding shader and vertex data . . . . .	134
<a href="#">oficina::ofVertexBuffer</a>	Represents a Vertex Buffer object (VBO). Use this to hold data related to drawing . . . . .	136

## Chapter 7

# File Index

### 7.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">benchmark.hpp</a>	Oficina's default benchmarking utilities . . . . .	137
<a href="#">canvas.hpp</a>	Tools for creating game scenes and manage such scenes . . . . .	138
<a href="#">display.hpp</a>	Tools for configuring windows for video output . . . . .	140
<a href="#">entity.hpp</a>	Interfaces and tools for managing objects ingame . . . . .	142
<a href="#">input.hpp</a>	Special tools for handling player input . . . . .	144
<a href="#">io.hpp</a>	Tools for handling non-player-related input and output . . . . .	161
<a href="#">oficina.hpp</a>	Default tools for easily initializing Oficina . . . . .	167
<a href="#">oflua.hpp</a>	Tools for object scripting and for the Repl, in Lua language . . . . .	173
<a href="#">ofscheme.hpp</a>	Tools for object scripting and for the Repl, in Scheme language . . . . .	182
<a href="#">platform.hpp</a>	Definitions for the platform currently executing the game . . . . .	187
<a href="#">render.hpp</a>	Tools and classes for rendering inside a context . . . . .	190
<a href="#">timer.hpp</a>	Tools for counting and processing time-related events . . . . .	202
<a href="#">types.hpp</a>	Tools for predefining default types and math tools used by OficinaFramework . . . . .	204



## Chapter 8

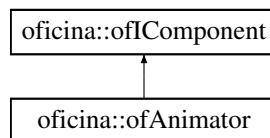
# Class Documentation

### 8.1 oficina::ofAnimator Class Reference

Tool for controlling a texture renderer to generate animations.

```
#include <render.hpp>
```

Inheritance diagram for oficina::ofAnimator:



#### Public Member Functions

- void **init** ()  
*Initializes the animator.*
- void **unload** ()  
*Unloads the animator, and unloads the texture if texture is being managed by this tool.*
- void **update** (float dt)  
*Updates the animation step.*
- void **draw** (glm::mat4 ViewProjection)  
*Draws the animation.*
- void **reg** (std::string animName, ofdword nFrames, const ofdword \*animFrames, float speed, bool loops=false, ofdword loopBackTo=0u)  
*Registers an animation by name.*
- void **unreg** (std::string animName)  
*Unregisters an animation.*
- void **SetAnimation** (std::string animName)  
*Sets the current animation to another one.*
- void **SyncToFrameRate** (bool state)  
*Sets whether the animation should remain synced to frame rate (frame-dependent) or not (frame-independent).*
- void **SetAnimationSpeed** (float spd)  
*Dynamically change the animation speed. This speed is never stored.*

- float [GetAnimationSpeed](#) () const  
*Yields the animation speed.*
- float [GetDefaultAnimationSpeed](#) () const  
*Yields the default animation speed.*
- void [SetAnimationRunning](#) (bool state)  
*Sets whether the animation should be played or not. Defaults to true.*
- std::string [GetCurrentAnimationName](#) () const  
*Retrieves which is the current animation.*
- void [SetRenderer](#) (ofTextureRenderer renderer, bool manage=false)  
*Defines a texture renderer for the animation, which should hold the texture atlas.*
- void [SetAnimationTexture](#) (ofTexture t)  
*Dynamically changes the internal texture atlas. Particularly useful for handling skins and such.*
- bool [isInit](#) () const  
*Checks if the animator was initialized.*
- glm::vec2 [getPosition](#) ()  
*Yields the position of the animation on the matrix.*
- void [setPosition](#) (glm::vec2 pos)  
*Sets the position of the animation on the matrix.*
- bool [GetAnimationRunning](#) () const  
*Checks if the animation is currently running.*

## Additional Inherited Members

### 8.1.1 Detailed Description

Tool for controlling a texture renderer to generate animations.

Definition at line 851 of file [render.hpp](#).

### 8.1.2 Member Function Documentation

#### 8.1.2.1 draw()

```
void oficina::ofAnimator::draw (
 glm::mat4 ViewProjection) [virtual]
```

Draws the animation.

#### Parameters

<i>ViewProjection</i>	ViewProjection matrix containing information on the viewport and the projection frustum.
-----------------------	------------------------------------------------------------------------------------------

**Note**

If you're looking for a way to define the animation's position, rotation and scale, you should define both on the parent [ofEntity](#).

Reimplemented from [oficina::ofComponent](#).

**8.1.2.2 GetAnimationRunning()**

```
bool oficina::ofAnimator::GetAnimationRunning () const
```

Checks if the animation is currently running.

**Returns**

Whether the animation is running or not.

**8.1.2.3 GetAnimationSpeed()**

```
float oficina::ofAnimator::GetAnimationSpeed () const
```

Yields the animation speed.

**Returns**

Current speed of the current animation.

**Warning**

To understand animation speed behaviour, see the `reg` method.

**See also**

[ofAnimator::reg](#)

**8.1.2.4 GetCurrentAnimationName()**

```
std::string oficina::ofAnimator::GetCurrentAnimationName () const
```

Retrieves which is the current animation.

**Returns**

Name of the animation which is currently being played.

#### 8.1.2.5 GetDefaultAnimationSpeed()

```
float oficina::ofAnimator::GetDefaultAnimationSpeed () const
```

Yields the default animation speed.

##### Returns

Animation speed which the animation was registered with.

##### Warning

To understand animation speed behaviour, see the `reg` method.

##### See also

[ofAnimator::reg](#)

#### 8.1.2.6 getPosition()

```
glm::vec2 oficina::ofAnimator::getPosition ()
```

Yields the position of the animation on the matrix.

##### Returns

A 2D vector containing the animation position.

#### 8.1.2.7 init()

```
void oficina::ofAnimator::init () [virtual]
```

Initializes the animator.

##### Warning

Be wary that this function should only be called after specifying the renderer with [ofAnimator::SetRenderer](#).

Implements [oficina::ofComponent](#).



### 8.1.2.8 isInit()

```
bool oficina::ofAnimator::isInit () const
```

Checks if the animator was initialized.

#### Returns

Whether the animator was initialized or not.

### 8.1.2.9 reg()

```
void oficina::ofAnimator::reg (
 std::string animName,
 ofdword nFrames,
 const ofdword * animFrames,
 float speed,
 bool loops = false,
 ofdword loopBackTo = 0u)
```

Registers an animation by name.

#### Parameters

<i>animName</i>	Desired animation name.
<i>nFrames</i>	amount of frames on the animation.
<i>animFrames</i>	Pointer to an array containing all animation frames, numbered.
<i>speed</i>	<p>Speed of the animation.</p> <p>Animation speed handling changes depending on the syncing type.</p> <ul style="list-style-type: none"><li>• When animation is synced to frame rate (default), speed relates to how many GAME FRAMES each ANIMATION FRAME lasts; therefore the value will always be converted to an integer, and the minimum value will be 1. Also, by this logic, the lower this number is, the faster the animation plays.</li><li>• When animation is NOT synced to frame rate, speed relates on how many SECONDS each ANIMATION FRAME lasts; therefore the value can be an actual float, as you can set the animation to less than a second of duration.</li></ul>
<i>loops</i>	Optionally set the animation to loop, jumping to the looping frame.
<i>loopBackTo</i>	Optionally set the index of the frame, on the frames array, which the animator will jump to when looping the animation. Defaults to the first frame of the animation (0).

#### See also

[ofAnimator::SyncToFrameRate](#)

#### 8.1.2.10 SetAnimation()

```
void oficina::ofAnimator::SetAnimation (
 std::string animName)
```

Sets the current animation to another one.

##### Warning

If the set animation is already being played, then nothing happens.

##### Parameters

<i>animName</i>	Name of the animation to be played.
-----------------	-------------------------------------

#### 8.1.2.11 SetAnimationRunning()

```
void oficina::ofAnimator::SetAnimationRunning (
 bool state)
```

Sets whether the animation should be played or not. Defaults to true.

##### Parameters

<i>state</i>	State of the animation: whether it should play or not.
--------------	--------------------------------------------------------

#### 8.1.2.12 SetAnimationSpeed()

```
void oficina::ofAnimator::SetAnimationSpeed (
 float spd)
```

Dynamically change the animation speed. This speed is never stored.

##### Warning

To understand animation speed behaviour, see the reg method.

##### See also

[ofAnimator::reg](#)

##### Parameters

<i>spd</i>	Speed value to be given to the currently played animation.
------------	------------------------------------------------------------

### 8.1.2.13 SetAnimationTexture()

```
void oficina::ofAnimator::SetAnimationTexture (
 ofTexture t)
```

Dynamically changes the internal texture atlas. Particularly useful for handling skins and such.

#### Warning

This operation will not be performed if the animator is automatically handling the stored texture.

#### Parameters

<i>t</i>	Texture to be now associated with the animation.
----------	--------------------------------------------------

### 8.1.2.14 setPosition()

```
void oficina::ofAnimator::setPosition (
 glm::vec2 pos)
```

Sets the position of the animation on the matrix.

#### Parameters

<i>pos</i>	The new position of the animation.
------------	------------------------------------

### 8.1.2.15 SetRenderer()

```
void oficina::ofAnimator::SetRenderer (
 ofTextureRenderer renderer,
 bool manage = false)
```

Defines a texture renderer for the animation, which should hold the texture atlas.

#### Parameters

<i>renderer</i>	Instantiated Texture Renderer to be used.
<i>manage</i>	Whether the given renderer should be managed by this tool (disposed when the animator is disposed).

### 8.1.2.16 SyncToFrameRate()

```
void oficina::ofAnimator::SyncToFrameRate (
 bool state)
```

Sets whether the animation should remain synced to frame rate (frame-dependent) or not (frame-independent).

#### Parameters

<i>state</i>	State of syncing. Defaults to true.
--------------	-------------------------------------

### 8.1.2.17 unreg()

```
void oficina::ofAnimator::unreg (
 std::string animName)
```

Unregisters an animation.

#### Parameters

<i>animName</i>	Desired animation to unregister.
-----------------	----------------------------------

### 8.1.2.18 update()

```
void oficina::ofAnimator::update (
 float dt) [virtual]
```

Updates the animation step.

#### Parameters

<i>dt</i>	Time difference, in seconds, from the last drawn frame to the currently drawn frame (delta time).
-----------	---------------------------------------------------------------------------------------------------

Implements [oficina::ofComponent](#).

The documentation for this class was generated from the following file:

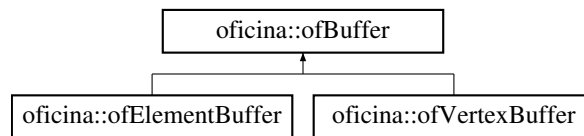
- [render.hpp](#)

## 8.2 oficina::ofBuffer Class Reference

Specifies a generic buffer. Override this class to create your own buffers.

```
#include <render.hpp>
```

Inheritance diagram for oficina::ofBuffer:



## Public Member Functions

- virtual void `init` () final  
*Initializes (generates) the buffer.*
- virtual void `unload` () final  
*Unloads (deletes) the buffer.*
- virtual void `bind` () final  
*Binds the buffer.*
- virtual void `unbind` () final  
*Unbinds all buffers of this type.*
- virtual void `setData` (size\_t dataSize, void \*data, ofBufferUsage usage)  
*Sets the data present on this buffer.*
- ofBuffer & `operator=` (const ofBuffer &other)  
*"Equals" operator for cloning the buffer.*
- virtual bool `isInit` () const final  
*Checks for buffer's initialization.*
- virtual GLuint `getName` () const final  
*Gets the buffer's real name on the GPU.*

## Protected Attributes

- GLenum `m_type` = GL\_ARRAY\_BUFFER  
*Type of this buffer. Redefine this on the constructor if you need a different type of buffer.*
- GLuint `m_name` = 0u  
*Buffer's real name (on the GPU).*

### 8.2.1 Detailed Description

Specifies a generic buffer. Override this class to create your own buffers.

#### Note

Buffer type should be defined directly on constructor.

Definition at line 246 of file [render.hpp](#).

### 8.2.2 Member Function Documentation

### 8.2.2.1 getName()

```
virtual GLuint oficina::ofBuffer::getName () const [final], [virtual]
```

Gets the buffer's real name on the GPU.

#### Returns

Unsigned integer containing the buffer's GPU index.

### 8.2.2.2 isInit()

```
virtual bool oficina::ofBuffer::isInit () const [final], [virtual]
```

Checks for buffer's initialization.

#### Returns

Whether the buffer was initialized or not.

### 8.2.2.3 operator=()

```
ofBuffer& oficina::ofBuffer::operator= (
 const ofBuffer & other)
```

"Equals" operator for cloning the buffer.

#### Parameters

<i>other</i>	Buffer to be cloned.
--------------	----------------------

#### Returns

A reference to this buffer.

### 8.2.2.4 setData()

```
virtual void oficina::ofBuffer::setData (
 size_t dataSize,
 void * data,
 ofBufferUsage usage) [virtual]
```

Sets the data present on this buffer.

## Parameters

<i>dataSize</i>	Size of the data to be fed, in bytes.
<i>data</i>	Pointer to the beginning of data.
<i>usage</i>	Type of usage of the buffer.

The documentation for this class was generated from the following file:

- [render.hpp](#)

## 8.3 oficina::ofCanvas Class Reference

Default interface for creating and managing canvases.

```
#include <canvas.hpp>
```

### Public Member Functions

- virtual [~ofCanvas](#) ()  
*Default destructor.*
- virtual void [init](#) ()=0  
*Initializes the current canvas.*
- virtual void [load](#) ()=0  
*Loads assets and processor/memory/GPU-intensive data for the canvas.*
- virtual void [unload](#) ()=0  
*Unloads the current canvas' assets.*
- virtual void [update](#) (float dt)=0  
*Updates logic for the current canvas on each of the game's frame.*
- virtual void [draw](#) ()=0  
*Drawing logic for the current canvas on each of the game's frame.*
- virtual void [remove](#) () final  
*Schedules this canvas for removal, if attached to canvas manager.*

### Friends

- class **ofCanvasManager**

#### 8.3.1 Detailed Description

Default interface for creating and managing canvases.

Definition at line 39 of file [canvas.hpp](#).

#### 8.3.2 Member Function Documentation

### 8.3.2.1 init()

```
virtual void oficina::ofCanvas::init () [pure virtual]
```

Initializes the current canvas.

#### Note

This method is always called by the manager before the "load" method.

### 8.3.2.2 load()

```
virtual void oficina::ofCanvas::load () [pure virtual]
```

Loads assets and processor/memory/GPU-intensive data for the canvas.

#### Note

This method is always called by the manager after the "init" method.

### 8.3.2.3 remove()

```
virtual void oficina::ofCanvas::remove () [final], [virtual]
```

Schedules this canvas for removal, if attached to canvas manager.

#### See also

[ofCanvasManager](#)

### 8.3.2.4 update()

```
virtual void oficina::ofCanvas::update (
 float dt) [pure virtual]
```

Updates logic for the current canvas on each of the game's frame.

#### Parameters

<i>dt</i>	Time difference, in seconds, from the last drawn frame to the currently drawn frame (delta time). Use this to interpolate your logic.
-----------	---------------------------------------------------------------------------------------------------------------------------------------



The documentation for this class was generated from the following file:

- [canvas.hpp](#)

## 8.4 oficina::ofCanvasManager Class Reference

Static class for handling canvases in general.

```
#include <canvas.hpp>
```

### Public Types

- enum [ofDebuggerState](#) { [ofDebuggerOff](#) = 0u, [ofDebuggerVars](#) = 1u, [ofDebuggerRepl](#) = 2u }
- State of the Debugger.*

### Static Public Member Functions

- static void [init](#) ()  
*Initializes the manager.*
- static void [add](#) ([ofCanvas](#) \*c, int depth=0, std::string name="")  
*Adds a canvas to the manager.*
- static void [remove](#) ([ofCanvas](#) \*c)  
*Removes a canvas from the manager.*
- static void [unload](#) ()  
*Unloads the manager.*
- static void [update](#) (float dt)  
*Updates the manager.*
- static void [draw](#) ()  
*Draws all canvases registered within the manager.*
- static std::vector< std::pair< std::string, std::string > > [getCanvasList](#) ()  
*Yields text information regarding the canvas list.*
- static std::ostream & [dbg\\_ReplOutputStream](#) ()  
*References the Repl output stream.*
- static [ofDebuggerState](#) [dbg\\_getState](#) ()  
*Current state of the debugger.*
- static void [dbg\\_callEval](#) ()  
*Forces the debugger to evaluate the text input.*
- static void [dbg\\_ChangeState](#) ()  
*Cycles through the debugger's state orderly.*
- static void [dbg\\_ReplHistoryPrev](#) ()  
*Walks backwards on the Repl's history.*
- static void [dbg\\_ReplHistoryNext](#) ()  
*Walks forward on the Repl's history.*
- static [ofdword](#) [dbg\\_ReplLineNumber](#) ()  
*Retrieves current command number on Repl.*
- static void [dbg\\_setFont](#) ([oficina::ofFontFaces](#) fontFace)  
*Sets a new hardcoded font for the debugger.*

### 8.4.1 Detailed Description

Static class for handling canvases in general.

General manager for canvases and the debugger. Can add, remove and reorder canvases. Will also load and unload canvases accordingly.

Includes a set of methods beginning with `dbg_` to handle the debugger, namely the Variable Watcher and the REPL.

#### Note

You should never have to actually instantiate this class, since its methods are all static.

Definition at line 86 of file [canvas.hpp](#).

### 8.4.2 Member Enumeration Documentation

#### 8.4.2.1 ofDebuggerState

enum [oficina::ofCanvasManager::ofDebuggerState](#)

State of the Debugger.

#### Enumerator

<code>ofDebuggerVars</code>	Disabled.
<code>ofDebuggerRepl</code>	Variable Watcher Mode.

Definition at line 90 of file [canvas.hpp](#).

### 8.4.3 Member Function Documentation

#### 8.4.3.1 add()

```
static void oficina::ofCanvasManager::add (
 ofCanvas * c,
 int depth = 0,
 std::string name = "") [static]
```

Adds a canvas to the manager.

#### Parameters

<i>c</i>	Pointer to the newly-initialized canvas.
<i>depth</i>	Optional canvas depth.
<i>name</i>	Optional canvas name for identification.

**Note**

Adding references to canvases instantiated on the memory stack is not recommended; since the manager tries to delete the canvas pointer when unloading it.

**8.4.3.2 dbg\_callEval()**

```
static void oficina::ofCanvasManager::dbg_callEval () [static]
```

Forces the debugger to evaluate the text input.

**Note**

You should not have to actually call this at any time.

**See also**

ofStartTextInput  
ofStopTextInput  
ofGetTextInput  
ofClearTextInput

**8.4.3.3 dbg\_ChangeState()**

```
static void oficina::ofCanvasManager::dbg_ChangeState () [static]
```

Cycles through the debugger's state orderly.

**See also**

[ofDebuggerState](#)

**8.4.3.4 dbg\_getState()**

```
static ofDebuggerState oficina::ofCanvasManager::dbg_getState () [static]
```

Current state of the debugger.

**See also**

[ofDebuggerState](#)

#### 8.4.3.5 dbg\_ReplLineNumber()

```
static ofdword oficina::ofCanvasManager::dbg_ReplLineNumber () [static]
```

Retrieves current command number on Repl.

##### Returns

An unsigned integer representing the repl command number since the last input.

#### 8.4.3.6 dbg\_ReplOutputStream()

```
static std::ostream& oficina::ofCanvasManager::dbg_ReplOutputStream () [static]
```

References the Repl output stream.

References the Repl's output stream. You can use this to output your own text to the Repl output.

##### Returns

A reference to the Repl output.

#### 8.4.3.7 dbg\_setFont()

```
static void oficina::ofCanvasManager::dbg_setFont (
 oficina::ofFontFaces fontFace) [static]
```

Sets a new hardcoded font for the debugger.

##### Parameters

<i>fontFace</i>	An enumeration specifying which font face (from the hardcoded fonts) should be used.
-----------------	--------------------------------------------------------------------------------------

#### 8.4.3.8 draw()

```
static void oficina::ofCanvasManager::draw () [static]
```

Draws all canvases registered within the manager.

##### Note

This method should always be called after "update".

#### 8.4.3.9 getCanvasList()

```
static std::vector<std::pair<std::string, std::string> > oficina::ofCanvasManager::getCanvasList () [static]
```

Yields text information regarding the canvas list.

##### Returns

A vector of string pairs. Each pair contains the canvas name and its pointer in hex form.

#### 8.4.3.10 remove()

```
static void oficina::ofCanvasManager::remove (ofCanvas * c) [static]
```

Removes a canvas from the manager.

##### Parameters

<i>c</i>	Pointer to the already initialized canvas.
----------	--------------------------------------------

##### Note

This procedure will also attempt to unload and dispose said canvas.

#### 8.4.3.11 unload()

```
static void oficina::ofCanvasManager::unload () [static]
```

Unloads the manager.

Unloads all canvases currently loaded, plus resets the manager's internal values.

#### 8.4.3.12 update()

```
static void oficina::ofCanvasManager::update (float dt) [static]
```

Updates the manager.

Updates the manager by removing any canvases that are scheduled for removal, or by calling their respective "update" method.

## Parameters

<i>dt</i>	Time difference, in seconds, from the last drawn frame to the currently drawn frame (delta time). Use this to interpolate your logic.
-----------	---------------------------------------------------------------------------------------------------------------------------------------

## Note

This method should always be called before "draw".

The documentation for this class was generated from the following file:

- [canvas.hpp](#)

## 8.5 oficina::ofContext Class Reference

Describes a context for your display.

```
#include <render.hpp>
```

### Public Member Functions

- void [pushArg](#) (std::string arg)  
*Handles context arguments.*
- void [open](#) (ofContextType type, const ofDisplay &hwnd)  
*Effectively opens the context.*
- void [close](#) ()  
*Closes the context.*
- bool [isInit](#) () const  
*Checks for context initialization.*
- void [setViewportSize](#) (glm::uvec2 sz)  
*Defines a new size for the viewport. Useful for whenever the window is resized.*
- glm::uvec2 [getViewportSize](#) ()  
*Yields the current viewport size.*
- void [setClearColor](#) (glm::vec4 color)  
*Sets the background color for the renderer.*

### 8.5.1 Detailed Description

Describes a context for your display.

Definition at line 194 of file [render.hpp](#).

### 8.5.2 Member Function Documentation

### 8.5.2.1 getViewportSize()

```
glm::uvec2 oficina::ofContext::getViewportSize ()
```

Yields the current viewport size.

#### Returns

A 2D vector of unsigned integers with the viewport size.

### 8.5.2.2 isInit()

```
bool oficina::ofContext::isInit () const
```

Checks for context initialization.

#### Returns

Whether the context was opened or not.

### 8.5.2.3 open()

```
void oficina::ofContext::open (
 ofContextType type,
 const ofDisplay & hwnd)
```

Effectively opens the context.

#### Parameters

<i>type</i>	Type of context. Currently, only OpenGL is supported.
<i>hwnd</i>	Reference to the display on which the context will be opened.

### 8.5.2.4 pushArg()

```
void oficina::ofContext::pushArg (
 std::string arg)
```

Handles context arguments.

Handles context arguments for context configuration. See documentation for details.

**Parameters**

<i>arg</i>	Argument to be treated and added to the configuration.
------------	--------------------------------------------------------

**8.5.2.5 setClearColor()**

```
void oficina::ofContext::setClearColor (
 glm::vec4 color)
```

Sets the background color for the renderer.

**Parameters**

<i>color</i>	Four-dimensional vector containing RGBA colors, normalized.
--------------	-------------------------------------------------------------

**8.5.2.6 setViewportSize()**

```
void oficina::ofContext::setViewportSize (
 glm::uvec2 sz)
```

Defines a new size for the viewport. Useful for whenever the window is resized.

**Parameters**

<i>sz</i>	2D vector of unsigned integers specifying the new viewport size.
-----------	------------------------------------------------------------------

The documentation for this class was generated from the following file:

- [render.hpp](#)

**8.6 oficina::ofDisplay Class Reference**

Represents a single window prepared for receiving a context.

```
#include <display.hpp>
```

**Public Member Functions**

- void [pushArg](#) (std::string arg)  
*Handles display arguments.*
- void [open](#) ()



- Opens the display.*
- void [close](#) ()
- Closes the display.*
- void [swap](#) ()
- Swaps display.*
- SDL\_Window \* [getHandle](#) () const
- Retrieves a low-level handle for the display.*
- glm::uvec2 [getSize](#) () const
- Retrieves the window's real size.*
- bool [isOpen](#) () const
- Display open state.*
- void [setSize](#) (glm::uvec2 NewSize)
- Sets size of the window.*
- void [setSwapInterval](#) (ofFrameRateConfig cfg, float max=0.0f, float min=0.0f)
- Changes the way that the display deals with swap interval.*
- bool [isFullscreen](#) () const
- Gets the state of the window (fullscreen/windowed).*
- void [setFullscreen](#) (bool state)
- Sets the state of the window on screen.*
- float [getDeltaTime](#) () const
- Retrieves the DeltaTime for swapping this display.*

### 8.6.1 Detailed Description

Represents a single window prepared for receiving a context.

See also

[ofContext](#)

Definition at line 59 of file [display.hpp](#).

### 8.6.2 Member Function Documentation

#### 8.6.2.1 close()

```
void oficina::ofDisplay::close ()
```

Closes the display.

Closes the display, effectively closing the window.

#### 8.6.2.2 getDeltaTime()

```
float oficina::ofDisplay::getDeltaTime () const
```

Retrieves the DeltaTime for swapping this display.

##### Returns

Display time variation for the latest swap.

#### 8.6.2.3 getHandle()

```
SDL_Window* oficina::ofDisplay::getHandle () const
```

Retrieves a low-level handle for the display.

##### Returns

an SDL2 window pointer.

#### 8.6.2.4 getSize()

```
glm::uvec2 oficina::ofDisplay::getSize () const
```

Retrieves the window's real size.

##### Returns

a 2D vector containing unsigned integers with the width (x) and the height (y) of the window.

#### 8.6.2.5 isFullscreen()

```
bool oficina::ofDisplay::isFullscreen () const
```

Gets the state of the window (fullscreen/windowed).

Checks whether the display is windowed or fullscreen.

##### Returns

Whether the display is fullscreen.

#### 8.6.2.6 isOpen()

```
bool oficina::ofDisplay::isOpen () const
```

Display open state.

Checks for the openness of the current state (i.e. if [open\(\)](#) was called).

##### Returns

Whether the display is open.

#### 8.6.2.7 open()

```
void oficina::ofDisplay::open ()
```

Opens the display.

Opens the display, effectively initializing the window.

#### 8.6.2.8 pushArg()

```
void oficina::ofDisplay::pushArg (
 std::string arg)
```

Handles display arguments.

Handles display arguments for display configuration. See documentation for details.

##### Parameters

<i>arg</i>	Argument to be treated and added to the configuration.
------------	--------------------------------------------------------

#### 8.6.2.9 setFullscreen()

```
void oficina::ofDisplay::setFullscreen (
 bool state)
```

Sets the state of the window on screen.

Sets the window to fullscreen or windowed.

##### Parameters

<i>state</i>	Window state to be assumed.
--------------	-----------------------------

#### 8.6.2.10 setSize()

```
void oficina::ofDisplay::setSize (
 glm::uvec2 NewSize)
```

Sets size of the window.

Changes size of the window. Resized windows will always be centered on screen.

##### Warning

Size must not be below 120x90 for width and height respectively.

#### 8.6.2.11 setSwapInterval()

```
void oficina::ofDisplay::setSwapInterval (
 ofFrameRateConfig cfg,
 float max = 0.0f,
 float min = 0.0f)
```

Changes the way that the display deals with swap interval.

##### Parameters

<i>cfg</i>	Desired swap interval configuration.
<i>max</i>	Maximum swap interval, in frames per second (FPS). If applicable.
<i>min</i>	Minimum swap interval, in frames per second (FPS). If applicable.

#### 8.6.2.12 swap()

```
void oficina::ofDisplay::swap ()
```

Swaps display.

Swaps the display by swapping buffers and clearing the window.

The documentation for this class was generated from the following file:

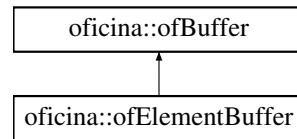
- [display.hpp](#)

## 8.7 oficina::ofElementBuffer Class Reference

Represents an Element Buffer object (EBO), useful for holding sequences of vertices for drawing on screen.

```
#include <render.hpp>
```

Inheritance diagram for oficina::ofElementBuffer:



### Public Member Functions

- [ofElementBuffer](#) ()  
*Buffer constructor.*
- void [setCount](#) (GLsizei count)  
*Defines the amount of elements fed to the object.*
- void [setType](#) (ofDataType type)  
*Defines the type of data fed to the object.*
- void [setProps](#) (GLsizei count, ofDataType type)  
*Defines both amount of elements and type of data fed to the object.*
- GLsizei [getCount](#) () const  
*Yields the amount of elements stored.*
- ofDataType [getType](#) () const  
*Yields the type of data stored.*
- void [draw](#) (ofPrimitiveType mode)  
*Draws a primitive respecting the elements fed to this buffer.*

### Additional Inherited Members

#### 8.7.1 Detailed Description

Represents an Element Buffer object (EBO), useful for holding sequences of vertices for drawing on screen.

Definition at line 299 of file [render.hpp](#).

#### 8.7.2 Member Function Documentation

##### 8.7.2.1 draw()

```
void oficina::ofElementBuffer::draw (
 ofPrimitiveType mode)
```

Draws a primitive respecting the elements fed to this buffer.

#### Warning

You must also have a vertex buffer and a shader program bound with the vertex attributes correctly set up.

**Parameters**

<i>mode</i>	Type of primitive to be drawn.
-------------	--------------------------------

**8.7.2.2 getCount()**

```
GLsizei oficina::ofElementBuffer::getCount () const
```

Yields the amount of elements stored.

**Returns**

Amount of buffer elements.

**8.7.2.3 getType()**

```
ofDataType oficina::ofElementBuffer::getType () const
```

Yields the type of data stored.

**Returns**

Type of data used by the elements.

**8.7.2.4 setCount()**

```
void oficina::ofElementBuffer::setCount (
 GLsizei count)
```

Defines the amount of elements fed to the object.

**Parameters**

<i>count</i>	Amount of elements.
--------------	---------------------

**8.7.2.5 setProps()**

```
void oficina::ofElementBuffer::setProps (
 GLsizei count,
 ofDataType type)
```

Defines both amount of elements and type of data fed to the object.

#### Parameters

<i>count</i>	Amount of elements.
<i>type</i>	Type of data.

#### 8.7.2.6 setType()

```
void oficina::ofElementBuffer::setType (
 ofDataType type)
```

Defines the type of data fed to the object.

#### Parameters

<i>type</i>	Type of data.
-------------	---------------

The documentation for this class was generated from the following file:

- [render.hpp](#)

## 8.8 oficina::ofEntity Class Reference

Abstract class representing one ingame entity.

```
#include <entity.hpp>
```

### Public Member Functions

- virtual [~ofEntity](#) ()  
*Default destructor.*
- virtual void [init](#) ()=0  
*Initializes logic for this entity.*
- virtual void [load](#) ()=0  
*Loads CPU/memory/GPU-heavy assets for this entity.*
- virtual void [unload](#) ()=0  
*Unloads assets for this entity.*
- virtual void [update](#) (float dt)=0  
*Updates logic for this entity.*
- virtual void [draw](#) (glm::mat4 ViewProjection)=0  
*Draws this entity.*
- void [translate](#) (glm::vec3 coord, bool loadIdentity=false)  
*Translates this entity.*
- void [rotate](#) (float theta, glm::vec3 axis, bool loadIdentity=false)

- Rotates this entity using Euler angles.
  - void [scale](#) (glm::vec3 amount, bool loadIdentity)
- Scales this entity.
  - void [setProperty](#) (ofbyte which, bool state)
- Changes a single property of this entity.
  - void [toggleProperty](#) (ofbyte which)
- Toggles the state of a single property of this entity.
  - void [setName](#) (std::string name)
- Defines the name of this entity.
  - glm::mat4 [getModelMatrix](#) ()
- Yields a copy of the entity's own internal Model matrix.
  - glm::vec3 [getPosition](#) () const
- Yields the entity's position.
  - glm::vec3 [getEulerAngles](#) () const
- Yields the entity's euler angles.
  - glm::vec3 [getScale](#) () const
- Yields the entity's scale.
  - bool [getProperty](#) (ofbyte which)
- Yields the state of a single property of this entity.
  - ofdword [getPropertyMask](#) () const
- Yields the entire mask of property states of this entity.
  - std::string [getName](#) () const
- Yields the name of this entity.
  - void [AddComponent](#) (std::string name, ofIComponent \*component)
- Adds a component to this entity.
  - ofIComponent \*const [GetComponent](#) (std::string name)
- Retrieves a component registered to this entity.
  - void [RemoveComponent](#) (std::string name)
- Removes and disposes a specific component on this entity.
  - void [ClearComponents](#) ()
- Removes and disposes all components on this entity.
  - void [UpdateComponents](#) (float dt)
- Updates all components of this entity.
  - void [DrawComponents](#) (glm::mat4 ViewProjection)
- Draws all components of this entity (when the draw method of such component is overridden).

## Protected Attributes

- glm::mat4 [translation](#)
- The translation matrix.
- glm::mat4 [rotation](#)
- The rotation matrix.
- glm::mat4 [scaling](#)
- The scale matrix.
- glm::vec3 [position](#)
- 3D vector containing the entity's actual position. Defaults to (0, 0, 0).
- glm::vec3 [eulerAngles](#)
- 3D vector containing the entity's euler angles. Defaults to (0, 0, 0).
- glm::vec3 [magnification](#) = glm::vec3(1.0f)
- 3D vector containing the entity's actual scale. Defaults to (1, 1, 1).



- `ofdword propertymask = 0x00000000u`  
*The entity's actual properties mask.*
- `std::map< std::string, ofIComponent * > components`  
*Holds all components associated with this entity.*
- `std::string name = "[unnamed]"`  
*String holding the entity's actual name.*

### 8.8.1 Detailed Description

Abstract class representing one ingame entity.

#### Note

When handling entities and, specially, components, be wary to use the component handling methods when necessary.

Definition at line 88 of file [entity.hpp](#).

### 8.8.2 Member Function Documentation

#### 8.8.2.1 AddComponent()

```
void oficina::ofEntity::AddComponent (
 std::string name,
 ofIComponent * component)
```

Adds a component to this entity.

#### Warning

You will not be able to add two components with the same name.

#### Parameters

<i>name</i>	Name of the component to be added.
<i>component</i>	Pointer to object compatible with the component interface.

#### Warning

The pointer will be managed by the entity itself.

### 8.8.2.2 draw()

```
virtual void oficina::ofEntity::draw (
 glm::mat4 ViewProjection) [pure virtual]
```

Draws this entity.

#### Parameters

<i>ViewProjection</i>	View * Projection matrix. Notice that the lack of a Model matrix is on purpose, since you should manipulate the object's model using the translation, rotation and scale methods. But you can also ignore them and pass the MVP to this method at once.
-----------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 8.8.2.3 DrawComponents()

```
void oficina::ofEntity::DrawComponents (
 glm::mat4 ViewProjection)
```

Draws all components of this entity (when the draw method of such component is overridden).

#### Parameters

<i>ViewProjection</i>	ViewProjection matrix containing information on the viewport and the projection frustum.
-----------------------	------------------------------------------------------------------------------------------

### 8.8.2.4 GetComponent()

```
ofIComponent* const oficina::ofEntity::GetComponent (
 std::string name)
```

Retrieves a component registered to this entity.

#### Parameters

<i>name</i>	Name of the component to be retrieved.
-------------	----------------------------------------

#### Returns

Const pointer to the component, or null if not registered.

### 8.8.2.5 getEulerAngles()

```
glm::vec3 oficina::ofEntity::getEulerAngles () const
```

Yields the entity's euler angles.

**Returns**

This entity's euler rotation for each axis on a 3D vector.

**8.8.2.6 getModelMatrix()**

```
glm::mat4 oficina::ofEntity::getModelMatrix ()
```

Yields a copy of the entity's own internal Model matrix.

**Returns**

This entity's model matrix.

**8.8.2.7 getName()**

```
std::string oficina::ofEntity::getName () const
```

Yields the name of this entity.

**Returns**

A string containing this entity's name.

**8.8.2.8 getPosition()**

```
glm::vec3 oficina::ofEntity::getPosition () const
```

Yields the entity's position.

**Returns**

This entity's position in a 3D vector.

**8.8.2.9 getProperty()**

```
bool oficina::ofEntity::getProperty (
 ofbyte which)
```

Yields the state of a single property of this entity.

**Parameters**

<i>which</i>	A property, ranging from 0 to 31.
--------------	-----------------------------------

**Returns**

Whether the property is on or off.

**8.8.2.10 getPropertyMask()**

```
ofdword oficina::ofEntity::getPropertyMask () const
```

Yields the entire mask of property states of this entity.

**Returns**

A 32-bit unsigned integer containing all the 31 properties, encoded in binary.

**8.8.2.11 getScale()**

```
glm::vec3 oficina::ofEntity::getScale () const
```

Yields the entity's scale.

**Returns**

A 3D vector containing the scale for each axis of the space.

**8.8.2.12 init()**

```
virtual void oficina::ofEntity::init () [pure virtual]
```

Initializes logic for this entity.

**Note**

This method should be called before "load".

#### 8.8.2.13 load()

```
virtual void oficina::ofEntity::load () [pure virtual]
```

Loads CPU/memory/GPU-heavy assets for this entity.

##### Note

This method should be called after "init".

#### 8.8.2.14 RemoveComponent()

```
void oficina::ofEntity::RemoveComponent (
 std::string name)
```

Removes and disposes a specific component on this entity.

## Parameters

<i>name</i>	Name of the component to be disposed.
-------------	---------------------------------------

## 8.8.2.15 rotate()

```
void oficina::ofEntity::rotate (
 float theta,
 glm::vec3 axis,
 bool loadIdentity = false)
```

Rotates this entity using Euler angles.

## Parameters

<i>theta</i>	Angle to rotate the entity, in radians.
<i>axis</i>	Axis of the Euler rotation.
<i>loadIdentity</i>	Whether the object should have a new rotation, or the rotation should build from the previous one.

## 8.8.2.16 scale()

```
void oficina::ofEntity::scale (
 glm::vec3 amount,
 bool loadIdentity)
```

Scales this entity.

## Parameters

<i>amount</i>	3D Vector containing how much should the object be scaled. Use positive numbers to scale up, and negative to scale down.
<i>loadIdentity</i>	Whether the object should have a new scale, or the scale should build from the previous one.

## 8.8.2.17 setName()

```
void oficina::ofEntity::setName (
 std::string name)
```

Defines the name of this entity.

## Parameters

<i>name</i>	Desired name for the entity to assume.
-------------	----------------------------------------

## Warning

The name should be defined before initializing the internal scripting system.

## 8.8.2.18 setProperty()

```
void oficina::ofEntity::setProperty (
 ofbyte which,
 bool state)
```

Changes a single property of this entity.

## Parameters

<i>which</i>	A property, ranging from 0 to 31.
<i>state</i>	State for the property to assume.

## 8.8.2.19 toggleProperty()

```
void oficina::ofEntity::toggleProperty (
 ofbyte which)
```

Toggles the state of a single property of this entity.

## Parameters

<i>which</i>	A property, ranging from 0 to 31.
--------------	-----------------------------------

## 8.8.2.20 translate()

```
void oficina::ofEntity::translate (
 glm::vec3 coord,
 bool loadIdentity = false)
```

Translates this entity.

## Parameters

<i>coord</i>	3D Vector containing the coordinates for the object.
<i>loadIdentity</i>	Whether the object should have a new position, or the translation should build from the previous one.

## 8.8.2.21 update()

```
virtual void oficina::ofEntity::update (
 float dt) [pure virtual]
```

Updates logic for this entity.

## Parameters

<i>dt</i>	Time difference, in seconds, from the last drawn frame to the currently drawn frame (delta time). Use this to interpolate your logic.
-----------	---------------------------------------------------------------------------------------------------------------------------------------

## 8.8.2.22 UpdateComponents()

```
void oficina::ofEntity::UpdateComponents (
 float dt)
```

Updates all components of this entity.

## Parameters

<i>dt</i>	Time difference, in seconds, from the last drawn frame to the currently drawn frame (delta time). Use this to interpolate your logic.
-----------	---------------------------------------------------------------------------------------------------------------------------------------

## 8.8.3 Member Data Documentation

## 8.8.3.1 rotation

```
glm::mat4 oficina::ofEntity::rotation [protected]
```

The rotation matrix.

## Note

This is automatically included when retrieving/generating the Model matrix.

Definition at line 214 of file [entity.hpp](#).



### 8.8.3.2 scaling

```
glm::mat4 oficina::ofEntity::scaling [protected]
```

The scale matrix.

#### Note

This is automatically included when retrieving/generating the Model matrix.A

Definition at line 218 of file [entity.hpp](#).

### 8.8.3.3 translation

```
glm::mat4 oficina::ofEntity::translation [protected]
```

The translation matrix.

#### Note

This is automatically included when retrieving/generating the Model matrix.

Definition at line 210 of file [entity.hpp](#).

The documentation for this class was generated from the following file:

- [entity.hpp](#)

## 8.9 oficina::ofFont Class Reference

Represents a font.

```
#include <render.hpp>
```

### Public Member Functions

- void [init](#) ([ofTexture](#) fontTexture, glm::uvec2 glyphSize, bool manageTexture=false)  
*Initializes the font.*
- void [write](#) (std::string text, glm::vec2 position, glm::mat4 mvp, glm::vec4 color=glm::vec4(1.0f))  
*Renders a text on the screen.*
- void [unload](#) ()  
*Unloads the font, and also unloads the texture if texture is being managed by the structure.*
- glm::vec2 [measure](#) (std::string text)  
*Measures the size of a string based on this font.*
- [ofFont](#) & [operator=](#) (const [ofFont](#) &other)  
*"Equals" operator for cloning fonts.*
- glm::uvec2 [getGlyphSize](#) () const  
*Gets the size of a single glyph on the font.*
- bool [isInit](#) () const  
*Checks if the font was initialized.*

### 8.9.1 Detailed Description

Represents a font.

#### Note

Fonts are texture atlases with each frame being a character in white color. Characters should range from 31 (unit separator) to 126 (tilde - '~'); it is also recommended that the first character (replacing unit separator) should be a block, for it can also be used as cursor on Repl.

Definition at line 802 of file [render.hpp](#).

### 8.9.2 Member Function Documentation

#### 8.9.2.1 getGlyphSize()

```
glm::uvec2 oficina::ofFont::getGlyphSize () const
```

Gets the size of a single glyph on the font.

#### Returns

A 2D vector containing the size of a glyph.

#### 8.9.2.2 init()

```
void oficina::ofFont::init (
 ofTexture fontTexture,
 glm::uvec2 glyphSize,
 bool manageTexture = false)
```

Initializes the font.

#### Parameters

<i>fontTexture</i>	Texture atlas containing the font characters.
<i>glyphSize</i>	2D unsigned integer vector containing the size of each glyph frame on the atlas.
<i>manageTexture</i>	Whether the texture should be managed (disposal when the font is also disposed) Defaults to false.

### 8.9.2.3 isInit()

```
bool oficina::ofFont::isInit () const
```

Checks if the font was initialized.

#### Returns

Whether the font was initialized or not.

### 8.9.2.4 measure()

```
glm::vec2 oficina::ofFont::measure (
 std::string text)
```

Measures the size of a string based on this font.

#### Parameters

<i>text</i>	Text to be used for measurement.
-------------	----------------------------------

#### Returns

A 2D vector containing the width and height of the text to be rendered.

### 8.9.2.5 operator=()

```
ofFont& oficina::ofFont::operator= (
 const ofFont & other)
```

"Equals" operator for cloning fonts.

#### Parameters

<i>other</i>	Font to be cloned.
--------------	--------------------

#### Returns

A reference to this font.

### 8.9.2.6 write()

```
void oficina::ofFont::write (
 std::string text,
 glm::vec2 position,
 glm::mat4 mvp,
 glm::vec4 color = glm::vec4(1.0f))
```

Renders a text on the screen.

#### Parameters

<i>text</i>	Text to be written.
<i>position</i>	Position of the first text glyph (centered) on the matrix.
<i>mvp</i>	Model-View-Projection matrix to be used when drawing the texture.

#### Warning

It is advised to use an ortographic projection if trying to draw readable text.

#### Parameters

<i>color</i>	4D vector specifying which color the text should be tinted with. Corresponds to a format {R, G, B, A}. Default values are {1, 1, 1, 1}.
--------------	--------------------------------------------------------------------------------------------------------------------------------------------

The documentation for this class was generated from the following file:

- [render.hpp](#)

## 8.10 oficina::ofFrameSpan Class Reference

Tool for counting and comparing frames, depending of the game's time variation.

```
#include <timer.hpp>
```

### Public Member Functions

- void [begin](#) ()  
*Begins counting frames.*
- void [update](#) ()  
*Counts current frame.*
- uint32\_t [yieldSpan](#) ()  
*Yields the current amount of frames, counting from the beginning.*
- uint32\_t [resetSpan](#) ()  
*Resets the frame counting.*
- uint32\_t [stop](#) ()  
*Stops the frame counting.*
- bool [isRunning](#) () const  
*Yields the state of the frame count.*

### 8.10.1 Detailed Description

Tool for counting and comparing frames, depending of the game's time variation.

Definition at line 62 of file [timer.hpp](#).

### 8.10.2 Member Function Documentation

#### 8.10.2.1 isRunning()

```
bool oficina::ofFrameSpan::isRunning () const
```

Yields the state of the frame count.

##### Returns

Whether the frame count is running or not.

#### 8.10.2.2 resetSpan()

```
uint32_t oficina::ofFrameSpan::resetSpan ()
```

Resets the frame counting.

##### Returns

Unsigned integer value with amount of frames passed before resetting the counter.

#### 8.10.2.3 stop()

```
uint32_t oficina::ofFrameSpan::stop ()
```

Stops the frame counting.

##### Returns

Unsigned integer value with amount of frames passed before stopping the counter.

#### 8.10.2.4 yieldSpan()

```
uint32_t oficina::ofFrameSpan::yieldSpan ()
```

Yields the current amount of frames, counting from the beginning.

##### Returns

Unsigned integer value with amount of frames passed since the beginning of the counting.

The documentation for this class was generated from the following file:

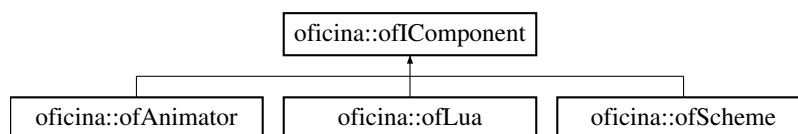
- [timer.hpp](#)

## 8.11 oficina::ofIComponent Class Reference

Defines a single component to be attached to an entity.

```
#include <entity.hpp>
```

Inheritance diagram for oficina::ofIComponent:



### Public Member Functions

- virtual [~ofIComponent](#) ()  
*Default destructor.*
- virtual void [init](#) ()=0  
*Initializes logic for the component. Overriding is obligatory.*
- virtual void [load](#) ()  
*Loads assets and such for the component. Overriding is optional.*
- virtual void [unload](#) ()  
*Unloads assets and such for the component. Overriding is optional.*
- virtual void [update](#) (float dt)=0  
*Updates logic for the component. Overriding is obligatory.*
- virtual void [draw](#) (glm::mat4 ViewProjection)  
*Draws the component. Overriding is optional.*
- std::string [getType](#) () const  
*Provides a getter for a string which informs the type of this component.*

### Protected Member Functions

- void [parseType](#) ()  
*Parses the type of current component. If creating a component without attaching it to an entity, you might want to call this anywhere on your constructor, init or load methods.*

## Protected Attributes

- [ofEntity](#) \* [parent](#)

*Direct pointer to this component's parent entity. It is advised not to change this pointer.*

## Friends

- class **ofEntity**

### 8.11.1 Detailed Description

Defines a single component to be attached to an entity.

See also

[ofEntity](#)

Definition at line 37 of file [entity.hpp](#).

### 8.11.2 Member Function Documentation

#### 8.11.2.1 draw()

```
virtual void oficina::ofIComponent::draw (
 glm::mat4 ViewProjection) [inline], [virtual]
```

Draws the component. Overriding is optional.

Parameters

<i>ViewProjection</i>	ViewProjection matrix containing information on the viewport and the projection frustum.
-----------------------	------------------------------------------------------------------------------------------

Reimplemented in [oficina::ofAnimator](#).

Definition at line 59 of file [entity.hpp](#).

#### 8.11.2.2 getType()

```
std::string oficina::ofIComponent::getType () const
```

Provides a getter for a string which informs the type of this component.

**Returns**

A string specifying the type of this component, if already attached to an entity; otherwise outputs "oficina::ofIComponent".

The documentation for this class was generated from the following file:

- [entity.hpp](#)

## 8.12 oficina::ofInputState Struct Reference

Holds an input state every frame.

```
#include <input.hpp>
```

**Public Attributes**

- [ofword padButtons](#) = 0x0000u  
*Bitmask holding the state of each gamepad button.*
- float [leftStick](#) [2] = {0.0f, 0.0f}  
*Holds the state of each of left stick's axis. Each axis ranges from -1.0f to 1.0f.*
- float [rightStick](#) [2] = {0.0f, 0.0f}  
*Holds the state of each of right stick's axis. Each axis ranges from -1.0f to 1.0f.*
- float [triggers](#) [2] = {0.0f, 0.0f}  
*Holds the state of each (0 = left, 1 = right) trigger. Each trigger ranges from 0.0f to 1.0f.*

### 8.12.1 Detailed Description

Holds an input state every frame.

Definition at line 142 of file [input.hpp](#).

The documentation for this struct was generated from the following file:

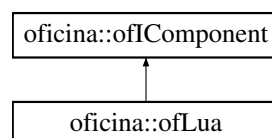
- [input.hpp](#)

## 8.13 oficina::ofLua Class Reference

Defines one Lua environment to be used inside an entity.

```
#include <oflua.hpp>
```

Inheritance diagram for oficina::ofLua:





## Public Member Functions

- void [init](#) ()  
*Initializes the script logic.*
- void [loadfile](#) (std::string filename)  
*Loads and evaluates an actual script file. You can also reload your script at runtime with this function, if needed.*
- void [reload](#) ()  
*Reloads the script at runtime, if already loaded.*
- void [unload](#) ()  
*Disposes the script object.*
- void [update](#) (float dt)  
*Calls the script object's update function, if existing.*
- void [regSym](#) (std::string symbol, std::string value)  
*Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.*
- void [regSym](#) (std::string symbol, double value)  
*Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.*
- void [regSym](#) (std::string symbol, int value)  
*Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.*
- void [regSym](#) (std::string symbol, bool value)  
*Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.*
- void [regSym](#) (std::string symbol, [oficina::ofEntity](#) \*value)  
*Defines/registers a symbol with an entity reference on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.*
- void [regSym](#) (std::string symbol, [oficina::ofComponent](#) \*value)  
*Defines/registers a symbol with a component reference on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.*
- void [regFunc](#) (std::string symbol, lua\_CFunction fun)  
*Defines/registers a foreign function on the script object. The registered function will be available only inside the Lua object instantiated on the script object.*
- std::string [getString](#) (std::string symbol)  
*Retrieves the value to a symbol, defined on the script, which holds a string type.*
- double [getNumber](#) (std::string symbol)  
*Retrieves the value to a symbol, defined on the script, which holds a numeric type.*
- int [getInteger](#) (std::string symbol)  
*Retrieves the value to a symbol, defined on the script, which holds an integer type.*
- bool [getBoolean](#) (std::string symbol)  
*Retrieves the value to a symbol, defined on the script, which holds a boolean type.*
- [oficina::ofEntity](#) \* [getEntity](#) (std::string symbol)  
*Retrieves an entity reference held by a symbol, defined on the script.*
- [oficina::ofComponent](#) \* [getComponent](#) (std::string symbol)  
*Retrieves a component reference held by a symbol, defined on the script.*
- bool [isInit](#) () const  
*Checks for script's initialization.*

## Additional Inherited Members

### 8.13.1 Detailed Description

Defines one Lua environment to be used inside an entity.

Definition at line 150 of file [oflua.hpp](#).

## 8.13.2 Member Function Documentation

### 8.13.2.1 getBoolean()

```
bool oficina::ofLua::getBoolean (
 std::string symbol)
```

Retrieves the value to a symbol, defined on the script, which holds a boolean type.

#### Parameters

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

#### Returns

A boolean value held by the symbol, if valid.

### 8.13.2.2 getComponent()

```
oficina::ofIComponent* oficina::ofLua::getComponent (
 std::string symbol)
```

Retrieves a component reference held by a symbol, defined on the script.

#### Parameters

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

#### Returns

A pointer to a component held by the symbol, if valid.

### 8.13.2.3 getEntity()

```
oficina::ofEntity* oficina::ofLua::getEntity (
 std::string symbol)
```

Retrieves an entity reference held by a symbol, defined on the script.

#### Parameters

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

**Returns**

A pointer to an entity held by the symbol, if valid.

**8.13.2.4 getInteger()**

```
int oficina::ofLua::getInteger (
 std::string symbol)
```

Retrieves the value to a symbol, defined on the script, which holds an integer type.

**Parameters**

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

**Returns**

An integer value held by the symbol, if valid.

**8.13.2.5 getNumber()**

```
double oficina::ofLua::getNumber (
 std::string symbol)
```

Retrieves the value to a symbol, defined on the script, which holds a numeric type.

**Parameters**

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

**Returns**

A double value held by the symbol, if valid.

**8.13.2.6 getString()**

```
std::string oficina::ofLua::getString (
 std::string symbol)
```

Retrieves the value to a symbol, defined on the script, which holds a string type.

#### Parameters

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

#### Returns

A string value held by the symbol, if valid.

#### 8.13.2.7 isInit()

```
bool oficina::ofLua::isInit () const
```

Checks for script's initialization.

#### Returns

Whether the script object was initialized and the script file was loaded.

#### 8.13.2.8 loadfile()

```
void oficina::ofLua::loadfile (
 std::string filename)
```

Loads and evaluates an actual script file. You can also reload your script at runtime with this function, if needed.

#### Parameters

<i>filename</i>	File path to the script file.
-----------------	-------------------------------

#### Note

See the [IronLua API Reference](#) for details.

#### 8.13.2.9 regFunc()

```
void oficina::ofLua::regFunc (
 std::string symbol,
 lua_CFunction fun)
```

Defines/registers a foreign function on the script object. The registered function will be available only inside the Lua object instantiated on the script object.

## Parameters

<i>symbol</i>	Name of the function to be defined.
<i>fun</i>	Function to be used. Notice that this function should follow the specifications of the Lua API: It needs to return an integer, containing the number of return values in the function, and accept a lua_State* as parameter, representing the function stack. For more info, consult the Lua 5.3 Reference.

**8.13.2.10 regSym()** [1/6]

```
void oficina::ofLua::regSym (
 std::string symbol,
 std::string value)
```

Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.

## Parameters

<i>symbol</i>	Name of the variable to be defined.
<i>value</i>	String value to be bound to the symbol.

**8.13.2.11 regSym()** [2/6]

```
void oficina::ofLua::regSym (
 std::string symbol,
 double value)
```

Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.

## Parameters

<i>symbol</i>	Name of the variable to be defined.
<i>value</i>	Double value to be bound to the symbol.

**8.13.2.12 regSym()** [3/6]

```
void oficina::ofLua::regSym (
 std::string symbol,
 int value)
```

Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.

## Parameters

<i>symbol</i>	Name of the variable to be defined.
<i>value</i>	Integer value to be bound to the symbol.

**8.13.2.13** `regSym()` [4/6]

```
void oficina::ofLua::regSym (
 std::string symbol,
 bool value)
```

Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.

## Parameters

<i>symbol</i>	Name of the variable to be defined.
<i>value</i>	Boolean value to be bound to the symbol.

**8.13.2.14** `regSym()` [5/6]

```
void oficina::ofLua::regSym (
 std::string symbol,
 oficina::ofEntity * value)
```

Defines/registers a symbol with an entity reference on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.

## Parameters

<i>symbol</i>	Name of the variable to be defined.
<i>value</i>	Entity reference value to be bound to the symbol.

**8.13.2.15** `regSym()` [6/6]

```
void oficina::ofLua::regSym (
 std::string symbol,
 oficina::ofIComponent * value)
```

Defines/registers a symbol with a component reference on the script object. The registered symbol will be available only inside the Lua object instantiated on the script object.

## Parameters

<i>symbol</i>	Name of the variable to be defined.
<i>value</i>	Component reference value to be bound to the symbol.

## 8.13.2.16 update()

```
void oficina::ofLua::update (
 float dt) [virtual]
```

Calls the script object's update function, if existing.

## Parameters

<i>dt</i>	Time difference, in seconds, from the last drawn frame to the currently drawn frame (delta time). Use this to interpolate your logic.
-----------	---------------------------------------------------------------------------------------------------------------------------------------

Implements [oficina::ofComponent](#).

The documentation for this class was generated from the following file:

- [oflua.hpp](#)

## 8.14 oficina::ofPrimitive Struct Reference

A structure representing a primitive. Can be used for rendering.

```
#include <render.hpp>
```

## Public Member Functions

- [~ofPrimitive](#) ()  
*Destructor of the primitive.*

## Public Attributes

- [ofPrimitiveType](#) type  
*Type of the primitive.*
- [ofVertexBuffer](#) vbo  
*Vertex Buffer on the GPU containing static primitive data for drawing.*
- [ofdword](#) NumberOfVertices  
*Number of vertices on the primitive.*

## Friends

- class **ofPrimitiveRenderer**

### 8.14.1 Detailed Description

A structure representing a primitive. Can be used for rendering.

Definition at line 982 of file [render.hpp](#).

The documentation for this struct was generated from the following file:

- [render.hpp](#)

## 8.15 oficina::ofPrimitiveRenderer Class Reference

A static class containing methods for creating and drawing simple primitives onscreen.

```
#include <render.hpp>
```

### Static Public Member Functions

- static [ofPrimitive](#) \* [makePrimitive](#) ([ofPrimitiveType](#) type, [ofdword](#) verticesAmount, [size\\_t](#) verticesSize, float \*vertices)  
*A method for creating an [ofPrimitive](#).*
- static void [draw](#) ([ofPrimitive](#) \*p, [glm::vec4](#) color, [glm::mat4](#) mvp)  
*A method for rendering a primitive.*

### 8.15.1 Detailed Description

A static class containing methods for creating and drawing simple primitives onscreen.

Definition at line 1002 of file [render.hpp](#).

### 8.15.2 Member Function Documentation

#### 8.15.2.1 draw()

```
static void oficina::ofPrimitiveRenderer::draw (
 ofPrimitive * p,
 glm::vec4 color,
 glm::mat4 mvp) [static]
```

A method for rendering a primitive.



## Parameters

<i>p</i>	Primitive to be rendered.
<i>color</i>	A vector specifying the color of the primitive to be rendered. Should contain RGBA values.
<i>mvp</i>	Model-View-Projection matrix to be fed to the GPU when rendering.

## 8.15.2.2 makePrimitive()

```
static ofPrimitive* oficina::ofPrimitiveRenderer::makePrimitive (
 ofPrimitiveType type,
 ofdword verticesAmount,
 size_t verticesSize,
 float * vertices) [static]
```

A method for creating an [ofPrimitive](#).

Use this method to instantiate a new primitive.

## Parameters

<i>type</i>	Type of the primitive.
<i>verticesAmount</i>	Number of vertices to be used on creation.
<i>verticesSize</i>	Total size, in bytes, of the vector array to be fed on the next argument.
<i>vertices</i>	An array of floating points, describing the primitive's vertices. Use THREE floats to specify a vertex (X, Y, Z coordinates, respectively).

The documentation for this class was generated from the following file:

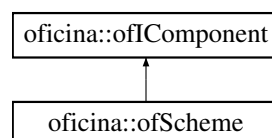
- [render.hpp](#)

## 8.16 oficina::ofScheme Class Reference

Defines one Scheme environment to be used inside an entity.

```
#include <ofscheme.hpp>
```

Inheritance diagram for oficina::ofScheme:



## Public Member Functions

- void `init` ()  
*Initializes the script object.*
- void `loadfile` (std::string module, std::string filename)  
*Loads and evaluates an actual script file.*
- void `reload` ()  
*Reloads the script at runtime, if already loaded.*
- void `unload` ()  
*Disposes the script object.*
- void `update` (float dt)  
*Calls the script object's update function, if existing.*
- void `regSym` (std::string symbol, SCM value)  
*Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the module defined on the script.*
- void `regFunc` (std::string symbol, int n\_params, scm\_t\_subr fun, int n\_optional\_params=0)  
*Defines/registers a foreign function on the script object. The registered function will be available only inside the module defined on the script.*
- SCM `getSymRef` (std::string symbol)  
*Retrieves a reference to a symbol defined on the object's module.*
- bool `isInit` () const  
*Checks for script's initialization.*

## Additional Inherited Members

### 8.16.1 Detailed Description

Defines one Scheme environment to be used inside an entity.

Definition at line 108 of file [ofscheme.hpp](#).

### 8.16.2 Member Function Documentation

#### 8.16.2.1 `getSymRef()`

```
SCM oficina::ofScheme::getSymRef (
 std::string symbol)
```

Retrieves a reference to a symbol defined on the object's module.

#### Parameters

<i>symbol</i>	Symbol which reference should be retrieved.
---------------	---------------------------------------------

**Returns**

A reference to the retrieved symbol.

**8.16.2.2 isInit()**

```
bool oficina::ofScheme::isInit () const
```

Checks for script's initialization.

**Returns**

Whether the Scheme system was initialized and the script file was loaded.

**8.16.2.3 loadfile()**

```
void oficina::ofScheme::loadfile (
 std::string module,
 std::string filename)
```

Loads and evaluates an actual script file.

**Parameters**

<i>module</i>	Name of the module in which the script will be enclosed.
---------------	----------------------------------------------------------

**Note**

In IronScheme, modules are important for enclosing files, since the VM is instantiated at a global level. Therefore, if not for modules, we would probably overlap other objects' behaviours.

**Parameters**

<i>filename</i>	File path to the script file.
-----------------	-------------------------------

**Note**

See the [IronScheme API Reference](#) for details.

**8.16.2.4 regFunc()**

```
void oficina::ofScheme::regFunc (
 std::string symbol,
```

```

int n_params,
scm_t_subr fun,
int n_optional_params = 0)

```

Defines/registers a foreign function on the script object. The registered function will be available only inside the module defined on the script.

#### Parameters

<i>symbol</i>	Name of the function to be defined.
<i>n_params</i>	Number of required/obligatory parameters to be passed to the function.
<i>fun</i>	Function pointer to be used. Pass the function name with the SCHEME_FUNCST macro to cast it appropriately.

#### See also

[SCHEME\\_FUNCST](#)

#### Parameters

<i>n_optional_params</i>	Optionally specify the amount of optional parameters which the function should have. Optional parameters should begin right after obligatory parameters.
--------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------

#### 8.16.2.5 regSym()

```

void oficina::ofScheme::regSym (
 std::string symbol,
 SCM value)

```

Defines/registers a symbol with a value on the script object. The registered symbol will be available only inside the module defined on the script.

#### Parameters

<i>symbol</i>	Name of the variable to be defined.
<i>value</i>	Value to be bound to the symbol.

#### 8.16.2.6 update()

```

void oficina::ofScheme::update (
 float dt) [virtual]

```

Calls the script object's update function, if existing.

## Parameters

<i>dt</i>	Time difference, in seconds, from the last drawn frame to the currently drawn frame (delta time). Use this to interpolate your logic.
-----------	---------------------------------------------------------------------------------------------------------------------------------------

Implements [oficina::ofComponent](#).

The documentation for this class was generated from the following file:

- [ofscheme.hpp](#)

## 8.17 oficina::ofShader Class Reference

Describes a shader.

```
#include <render.hpp>
```

### Public Member Functions

- virtual void [init](#) ([ofShaderType](#) type) final  
*Initializes (generates) the shader.*
- virtual void [unload](#) () final  
*Unloads (deletes) the shader.*
- virtual void [setSource](#) (const char \*src) final  
*Defines a source code for the shader.*
- virtual void [compile](#) () final  
*Compiles the shader.*
- virtual bool [isInit](#) () const final  
*Checks if the shader was initialized.*
- virtual bool [isCompiled](#) () const final  
*Checks if the shader was compiled.*
- virtual GLuint [getName](#) () const final  
*Yields the shader's real name on the GPU.*
- [ofShader](#) & [operator=](#) (const [ofShader](#) &shader)  
*"Equals" operator for cloning the shader.*

### Protected Attributes

- [ofShaderType](#) [m\\_type](#) = ofShaderFragment  
*Type of shader.*
- GLuint [m\\_name](#) = 0u  
*True name of shader on the GPU.*
- bool [m\\_srcassign](#) = false  
*Whether the shader source code was assigned.*
- bool [m\\_compiled](#) = false  
*Whether the shader was compiled.*

### 8.17.1 Detailed Description

Describes a shader.

Definition at line 341 of file [render.hpp](#).

### 8.17.2 Member Function Documentation

#### 8.17.2.1 compile()

```
virtual void oficina::ofShader::compile () [final], [virtual]
```

Compiles the shader.

##### Warning

You must define a source for the shader before.

#### 8.17.2.2 getName()

```
virtual GLuint oficina::ofShader::getName () const [final], [virtual]
```

Yields the shader's real name on the GPU.

##### Returns

Unsigned integer representing the shader's index on the GPU.

#### 8.17.2.3 init()

```
virtual void oficina::ofShader::init (
 ofShaderType type) [final], [virtual]
```

Initializes (generates) the shader.

##### Parameters

<i>type</i>	Type of shader to be used.
-------------	----------------------------

#### 8.17.2.4 isCompiled()

```
virtual bool oficina::ofShader::isCompiled () const [final], [virtual]
```

Checks if the shader was compiled.

##### Returns

Whether the shader was compiled or not.

#### 8.17.2.5 isInit()

```
virtual bool oficina::ofShader::isInit () const [final], [virtual]
```

Checks if the shader was initialized.

##### Returns

Whether the shader was initialized or not.

#### 8.17.2.6 operator=()

```
ofShader& oficina::ofShader::operator= (
 const ofShader & shader)
```

"Equals" operator for cloning the shader.

##### Parameters

<i>shader</i>	Shader to be cloned.
---------------	----------------------

##### Returns

A reference to this shader.

#### 8.17.2.7 setSource()

```
virtual void oficina::ofShader::setSource (
 const char * src) [final], [virtual]
```

Defines a source code for the shader.

## Parameters

src	String containing the source code of the shader.
-----	--------------------------------------------------

The documentation for this class was generated from the following file:

- [render.hpp](#)

## 8.18 oficina::ofShaderAttribute Class Reference

Represents the location of an attribute for the program shader.

```
#include <render.hpp>
```

### Public Member Functions

- void [setSize](#) (GLint s)  
*Defines the size of the attribute.*
- void [setType](#) (ofDataType t)  
*Defines the type of data of the attribute.*
- void [setStride](#) (GLsizei stride)  
*Defines the stride of the attribute on the vertex data.*
- void [setAutoNormalize](#) (bool state)  
*Defines if the attribute should be automatically normalized.*
- void [setProps](#) (GLint size, ofDataType type, GLsizei stride, bool normalize=false)  
*Defines all attribute properties at once.*
- void [enable](#) ()  
*Enables the shader attribute.*
- int [getSize](#) ()  
*Yields the size of the attribute.*
- ofDataType [getType](#) ()  
*Yields the data type of the attribute.*
- size\_t [getStride](#) ()  
*Yields the stride of the attribute.*
- bool [isAutoNormalizing](#) ()  
*Yields the automatic normalization state of the attribute.*
- bool [isValid](#) () const  
*Checks if the attribute is valid.*
- void [bindVertexArrayData](#) (void \*byteOffset=nullptr)  
*Binds the vertex array data to the attribute.*
- ofShaderAttribute & [operator=](#) (const ofShaderAttribute &attr)  
*"Equals" operator for cloning the attribute.*

### Friends

- class **ofShaderProgram**



### 8.18.1 Detailed Description

Represents the location of an attribute for the program shader.

Definition at line 386 of file [render.hpp](#).

### 8.18.2 Member Function Documentation

#### 8.18.2.1 bindVertexArrayData()

```
void oficina::ofShaderAttribute::bindVertexArrayData (
 void * byteOffset = nullptr)
```

Binds the vertex array data to the attribute.

##### Parameters

<i>byteOffset</i>	Byte offset of the attribute on the array data. You can define a position from the beginning and cast it to void*. Defaults to nullptr AKA the beginning of the vertex array data.
-------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### 8.18.2.2 getSize()

```
int oficina::ofShaderAttribute::getSize ()
```

Yields the size of the attribute.

##### Returns

Attribute size.

#### 8.18.2.3 getStride()

```
size_t oficina::ofShaderAttribute::getStride ()
```

Yields the stride of the attribute.

##### Returns

Attribute stride.

#### 8.18.2.4 getType()

```
ofDataType oficina::ofShaderAttribute::getType ()
```

Yields the data type of the attribute.

##### Returns

Attribute data type.

#### 8.18.2.5 isAutoNormalizing()

```
bool oficina::ofShaderAttribute::isAutoNormalizing ()
```

Yields the automatic normalization state of the attribute.

##### Returns

Whether the attribute automatically normalizes or not.

#### 8.18.2.6 isValid()

```
bool oficina::ofShaderAttribute::isValid () const
```

Checks if the attribute is valid.

##### Returns

Whether the attribute is valid or not.

#### 8.18.2.7 operator=()

```
ofShaderAttribute& oficina::ofShaderAttribute::operator= (
 const ofShaderAttribute & attr)
```

"Equals" operator for cloning the attribute.

##### Parameters

<i>attr</i>	Attribute to be cloned.
-------------	-------------------------

**Returns**

Reference to this attribute.

**8.18.2.8 setAutoNormalize()**

```
void oficina::ofShaderAttribute::setAutoNormalize (
 bool state)
```

Defines if the attribute should be automatically normalized.

**Parameters**

<i>state</i>	Whether the attribute should be automatically normalized.
--------------	-----------------------------------------------------------

**8.18.2.9 setProps()**

```
void oficina::ofShaderAttribute::setProps (
 GLint size,
 ofDataType type,
 GLsizei stride,
 bool normalize = false)
```

Defines all attribute properties at once.

**Parameters**

<i>size</i>	Size of the attribute.
<i>type</i>	Type of attribute data.
<i>stride</i>	Stride of the attribute on vertex data.
<i>normalize</i>	Whether the attribute should be normalized automatically or not.

**8.18.2.10 setSize()**

```
void oficina::ofShaderAttribute::setSize (
 GLint s)
```

Defines the size of the attribute.

**Parameters**

<i>s</i>	Size to be given to the attribute.
----------	------------------------------------

### 8.18.2.11 setStride()

```
void oficina::ofShaderAttribute::setStride (
 GLsizei stride)
```

Defines the stride of the attribute on the vertex data.

#### Parameters

<i>stride</i>	Stride of the attribute.
---------------	--------------------------

### 8.18.2.12 setType()

```
void oficina::ofShaderAttribute::setType (
 ofDataType t)
```

Defines the type of data of the attribute.

#### Parameters

<i>t</i>	Type of attribute data.
----------	-------------------------

The documentation for this class was generated from the following file:

- [render.hpp](#)

## 8.19 oficina::ofShaderProgram Class Reference

Represents a shader program.

```
#include <render.hpp>
```

### Public Member Functions

- void [init](#) ()  
*Initializes (generates) the shader program.*
- void [unload](#) ()  
*Unloads (deletes) the shader program.*
- void [attach](#) (const [ofShader](#) &shader)  
*Attaches a shader to the shader program.*
- void [attachUnload](#) ([ofShader](#) &shader)  
*Attaches a shader to the shader program and unloads the shader if attachment was successful.*

- void `bindFragmentDataLocation` (std::string name, ofdword colorNumber=0u)  
*Binds a fragment shader output data location.*
- void `link` ()  
*Links the shader program.*
- void `use` ()  
*Uses this shader program.*
- void `unuse` ()  
*Stops using any shader program that is in use.*
- bool `isInit` () const  
*Checks if shader program was initialized.*
- bool `isLinked` () const  
*Checks if shader program was linked.*
- GLuint `getName` () const  
*Yields the shader program's real name.*
- ofShaderProgram & `operator=` (const ofShaderProgram &program)  
*"Equals" operator for cloning a shader program.*
- ofShaderAttribute `getAttributeLocation` (std::string name)  
*Retrieves the attribute location of a shader attribute.*
- ofShaderUniform `getUniformLocation` (std::string name)  
*Retrieves the uniform location of a shader uniform.*

### 8.19.1 Detailed Description

Represents a shader program.

Definition at line 547 of file `render.hpp`.

### 8.19.2 Member Function Documentation

#### 8.19.2.1 `attach()`

```
void oficina::ofShaderProgram::attach (
 const ofShader & shader)
```

Attaches a shader to the shader program.

#### Parameters

<code>shader</code>	Reference to the shader to be attached.
---------------------	-----------------------------------------

#### Warning

Make sure the shader is already compiled.

### 8.19.2.2 attachUnload()

```
void oficina::ofShaderProgram::attachUnload (
 ofShader & shader)
```

Attaches a shader to the shader program and unloads the shader if attachment was successful.

#### Parameters

<i>shader</i>	Reference to the shader to be attached.
---------------	-----------------------------------------

#### Warning

Make sure the shader is already compiled.

### 8.19.2.3 bindFragmentDataLocation()

```
void oficina::ofShaderProgram::bindFragmentDataLocation (
 std::string name,
 ofdword colorNumber = 0u)
```

Binds a fragment shader output data location.

#### Note

Fragment data location defaults to color 0 on outColor. However, you can pick another fragment data location with this method.

#### Parameters

<i>name</i>	Name of the data location.
<i>colorNumber</i>	Color slot of the fragment shader output data. Defaults to 0.

### 8.19.2.4 getAttributeLocation()

```
ofShaderAttribute oficina::ofShaderProgram::getAttributeLocation (
 std::string name)
```

Retrieves the attribute location of a shader attribute.

#### Parameters

<i>name</i>	Name of the attribute on the attached shaders.
-------------	------------------------------------------------

**Returns**

A reference to the shader attribute.

**8.19.2.5 getName()**

```
GLuint oficina::ofShaderProgram::getName () const
```

Yields the shader program's real name.

**Returns**

An unsigned integer with the shader program's index on the GPU.

**8.19.2.6 getUniformLocation()**

```
ofShaderUniform oficina::ofShaderProgram::getUniformLocation (
 std::string name)
```

Retrieves the uniform location of a shader uniform.

**Parameters**

<i>name</i>	Name of the uniform on the attached shaders.
-------------	----------------------------------------------

**Returns**

A reference to the shader uniform.

**8.19.2.7 isInit()**

```
bool oficina::ofShaderProgram::isInit () const
```

Checks if shader program was initialized.

**Returns**

Whether the shader program was initialized or not.

#### 8.19.2.8 isLinked()

```
bool oficina::ofShaderProgram::isLinked () const
```

Checks if shader program was linked.

##### Returns

Whether the shader program was linked or not.

#### 8.19.2.9 link()

```
void oficina::ofShaderProgram::link ()
```

Links the shader program.

##### Warning

This method should only be called after attaching the desired shaders.

#### 8.19.2.10 operator=()

```
ofShaderProgram& oficina::ofShaderProgram::operator= (
 const ofShaderProgram & program)
```

"Equals" operator for cloning a shader program.

##### Parameters

<i>program</i>	Program to be cloned.
----------------	-----------------------

##### Returns

Reference to this shader program.

#### 8.19.2.11 use()

```
void oficina::ofShaderProgram::use ()
```

Uses this shader program.



**Warning**

This method should only be called after linking the program.

The documentation for this class was generated from the following file:

- [render.hpp](#)

**8.20 oficina::ofShaderUniform Class Reference**

Represents and handles a shader's uniform.

```
#include <render.hpp>
```

**Public Member Functions**

- bool [isValid](#) () const  
*Checks if the uniform is valid.*
- [ofShaderUniform](#) & [operator=](#) (const [ofShaderUniform](#) &uniform)  
*"Equals" operator for cloning the uniform.*
- void [set](#) (float value)  
*Sets the value of the uniform.*
- void [set](#) (glm::vec2 value)  
*Sets the value of the uniform.*
- void [set](#) (glm::vec3 value)  
*Sets the value of the uniform.*
- void [set](#) (glm::vec4 value)  
*Sets the value of the uniform.*
- void [set](#) (int value)  
*Sets the value of the uniform.*
- void [set](#) (glm::ivec2 value)  
*Sets the value of the uniform.*
- void [set](#) (glm::ivec3 value)  
*Sets the value of the uniform.*
- void [set](#) (glm::ivec4 value)  
*Sets the value of the uniform.*
- void [set](#) (unsigned int value)  
*Sets the value of the uniform.*
- void [set](#) (glm::uvec2 value)  
*Sets the value of the uniform.*
- void [set](#) (glm::uvec3 value)  
*Sets the value of the uniform.*
- void [set](#) (glm::uvec4 value)  
*Sets the value of the uniform.*
- void [set](#) (glm::mat2 value, bool transpose=false)  
*Sets the value of the uniform.*
- void [set](#) (glm::mat3 value, bool transpose=false)  
*Sets the value of the uniform.*
- void [set](#) (glm::mat4 value, bool transpose=false)

- Sets the value of the uniform.*
- void [set](#) (glm::mat2x3 value, bool transpose=false)
- Sets the value of the uniform.*
- void [set](#) (glm::mat3x2 value, bool transpose=false)
- Sets the value of the uniform.*
- void [set](#) (glm::mat2x4 value, bool transpose=false)
- Sets the value of the uniform.*
- void [set](#) (glm::mat4x2 value, bool transpose=false)
- Sets the value of the uniform.*
- void [set](#) (glm::mat3x4 value, bool transpose=false)
- Sets the value of the uniform.*
- void [set](#) (glm::mat4x3 value, bool transpose=false)
- Sets the value of the uniform.*

## Friends

- class **ofShaderProgram**

## 8.20.1 Detailed Description

Represents and handles a shader's uniform.

### Warning

When setting uniform values, please notice that literal identifiers matter, specially when handling signed/unsigned values.

Definition at line [452](#) of file [render.hpp](#).

## 8.20.2 Member Function Documentation

### 8.20.2.1 isValid()

```
bool oficina::ofShaderUniform::isValid () const
```

Checks if the uniform is valid.

### Returns

Whether the uniform is valid or not.

### 8.20.2.2 operator=()

```
ofShaderUniform& oficina::ofShaderUniform::operator= (
 const ofShaderUniform & uniform)
```

"Equals" operator for cloning the uniform.

## Parameters

<i>uniform</i>	Uniform to be cloned.
----------------	-----------------------

## Returns

A reference to this uniform.

**8.20.2.3 set()** [1/21]

```
void oficina::ofShaderUniform::set (
 float value)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	Float value.
--------------	--------------

**8.20.2.4 set()** [2/21]

```
void oficina::ofShaderUniform::set (
 glm::vec2 value)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	2D vector of float.
--------------	---------------------

**8.20.2.5 set()** [3/21]

```
void oficina::ofShaderUniform::set (
 glm::vec3 value)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	3D vector of float.
--------------	---------------------

**8.20.2.6 set()** [4/21]

```
void oficina::ofShaderUniform::set (
 glm::vec4 value)
```

Sets the value of the uniform.

**Parameters**

<i>value</i>	4D vector of float.
--------------	---------------------

**8.20.2.7 set()** [5/21]

```
void oficina::ofShaderUniform::set (
 int value)
```

Sets the value of the uniform.

**Parameters**

<i>value</i>	Signed integer value.
--------------	-----------------------

**8.20.2.8 set()** [6/21]

```
void oficina::ofShaderUniform::set (
 glm::ivec2 value)
```

Sets the value of the uniform.

**Parameters**

<i>value</i>	2D vector of signed integer.
--------------	------------------------------

**8.20.2.9 set()** [7/21]

```
void oficina::ofShaderUniform::set (
 glm::ivec3 value)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	3D vector of signed integer.
--------------	------------------------------

8.20.2.10 `set()` [8/21]

```
void oficina::ofShaderUniform::set (
 glm::ivec4 value)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	4D vector of signed integer.
--------------	------------------------------

8.20.2.11 `set()` [9/21]

```
void oficina::ofShaderUniform::set (
 unsigned int value)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	Unsigned integer value.
--------------	-------------------------

8.20.2.12 `set()` [10/21]

```
void oficina::ofShaderUniform::set (
 glm::uvec2 value)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	2D vector of unsigned integer.
--------------	--------------------------------

**8.20.2.13 set()** [11/21]

```
void oficina::ofShaderUniform::set (
 glm::uvec3 value)
```

Sets the value of the uniform.

**Parameters**

<i>value</i>	3D vector of unsigned integer.
--------------	--------------------------------

**8.20.2.14 set()** [12/21]

```
void oficina::ofShaderUniform::set (
 glm::uvec4 value)
```

Sets the value of the uniform.

**Parameters**

<i>value</i>	4D vector of unsigned integer.
--------------	--------------------------------

**8.20.2.15 set()** [13/21]

```
void oficina::ofShaderUniform::set (
 glm::mat2 value,
 bool transpose = false)
```

Sets the value of the uniform.

**Parameters**

<i>value</i>	2x2 matrix of float.
<i>transpose</i>	Whether the matrix should be transposed.

**8.20.2.16 set()** [14/21]

```
void oficina::ofShaderUniform::set (
 glm::mat3 value,
 bool transpose = false)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	3x3 matrix of float.
<i>transpose</i>	Whether the matrix should be transposed.

**8.20.2.17 set()** [15/21]

```
void oficina::ofShaderUniform::set (
 glm::mat4 value,
 bool transpose = false)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	4x4 matrix of float.
<i>transpose</i>	Whether the matrix should be transposed.

**8.20.2.18 set()** [16/21]

```
void oficina::ofShaderUniform::set (
 glm::mat2x3 value,
 bool transpose = false)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	2x3 matrix of float.
<i>transpose</i>	Whether the matrix should be transposed.

**8.20.2.19 set()** [17/21]

```
void oficina::ofShaderUniform::set (
 glm::mat3x2 value,
 bool transpose = false)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	3x2 matrix of float.
<i>transpose</i>	Whether the matrix should be transposed.

**8.20.2.20 set()** [18/21]

```
void oficina::ofShaderUniform::set (
 glm::mat2x4 value,
 bool transpose = false)
```

Sets the value of the uniform.

**Parameters**

<i>value</i>	2x4 matrix of float.
<i>transpose</i>	Whether the matrix should be transposed.

**8.20.2.21 set()** [19/21]

```
void oficina::ofShaderUniform::set (
 glm::mat4x2 value,
 bool transpose = false)
```

Sets the value of the uniform.

**Parameters**

<i>value</i>	4x2 matrix of float.
<i>transpose</i>	Whether the matrix should be transposed.

**8.20.2.22 set()** [20/21]

```
void oficina::ofShaderUniform::set (
 glm::mat3x4 value,
 bool transpose = false)
```

Sets the value of the uniform.

**Parameters**

<i>value</i>	3x4 matrix of float.
<i>transpose</i>	Whether the matrix should be transposed.



8.20.2.23 `set()` [21/21]

```
void oficina::ofShaderUniform::set (
 glm::mat4x3 value,
 bool transpose = false)
```

Sets the value of the uniform.

## Parameters

<i>value</i>	4x3 matrix of float.
<i>transpose</i>	Whether the matrix should be transposed.

The documentation for this class was generated from the following file:

- [render.hpp](#)

## 8.21 oficina::ofTexture Class Reference

Represents a texture on the GPU.

```
#include <render.hpp>
```

## Public Member Functions

- void `bind` (`ofword` currentSampler=0)  
*Binds the texture to be used.*
- void `unbind` (`ofword` currentSampler=0)  
*Unbinds any texture currently in use.*
- `ofTexture` & `operator=` (const `ofTexture` &other)  
*"Equals" operator to clone a texture.*
- `GLuint` `operator()` ()  
*Parenthesis operator to retrieve the texture's real name.*
- bool `isLoading` () const  
*Checks if the texture is loaded.*
- std::string `getFileName` () const  
*Yields the texture location on the game assets.*
- glm::uvec2 `getSize` () const  
*Yields the dimensions of this texture.*

## Friends

- class `ofTexturePool`

## 8.21.1 Detailed Description

Represents a texture on the GPU.

Definition at line 648 of file [render.hpp](#).

## 8.21.2 Member Function Documentation

### 8.21.2.1 bind()

```
void oficina::ofTexture::bind (
 ofword currentSampler = 0)
```

Binds the texture to be used.

#### Parameters

<i>currentSampler</i>	Sampler index to bind the texture to. Use in conjunction with a sampler2D. Defaults to 0.
-----------------------	-------------------------------------------------------------------------------------------

### 8.21.2.2 getFileName()

```
std::string oficina::ofTexture::getFileName () const
```

Yields the texture location on the game assets.

#### Returns

A string containing the texture file path.

### 8.21.2.3 getSize()

```
glm::uvec2 oficina::ofTexture::getSize () const
```

Yields the dimensions of this texture.

#### Returns

A 2D unsigned integer vector containing the texture dimensions in pixels.

### 8.21.2.4 isLoaded()

```
bool oficina::ofTexture::isLoaded () const
```

Checks if the texture is loaded.

#### Returns

Whether the texture is loaded or not.

## 8.21.2.5 operator()

```
GLuint oficina::ofTexture::operator() ()
```

Parenthesis operator to retrieve the texture's real name.

## Returns

An unsigned integer with the texture's real index on the GPU.

## 8.21.2.6 operator=()

```
ofTexture& oficina::ofTexture::operator= (
 const ofTexture & other)
```

"Equals" operator to clone a texture.

## Parameters

<i>other</i>	Texture to be cloned.
--------------	-----------------------

## Returns

A reference to this texture.

## 8.21.2.7 unbind()

```
void oficina::ofTexture::unbind (
 ofword currentSampler = 0)
```

Unbinds any texture currently in use.

## Parameters

<i>currentSampler</i>	Sampler index to unbind a texture from. Use in conjunction with a sampler2D. Defaults to 0.
-----------------------	---------------------------------------------------------------------------------------------

The documentation for this class was generated from the following file:

- [render.hpp](#)

## 8.22 oficina::ofTexturePool Class Reference

Static object for managing textures. Most (if not all) textures should be loaded using this tool.

```
#include <render.hpp>
```

## Static Public Member Functions

- static [ofTexture load](#) (std::string filename)  
*Loads a texture from disk.*
- static [ofTexture load](#) (SDL\_Surface \*surf)  
*Loads a texture from memory.*
- static [ofFont loadDefaultFont](#) ([ofFontFaces](#) fontface=[ofFontFaceGohuFont](#))  
*Loads one of Oficina's default font faces.*
- static void [unload](#) ([ofTexture](#) &t)  
*Unloads a font.*
- static void [clear](#) ()  
*Unloads ALL textures on the pool.*

### 8.22.1 Detailed Description

Static object for managing textures. Most (if not all) textures should be loaded using this tool.

#### Note

The use of this tool for managing textures is so that, when requiring a specific texture, it would never be loaded more than once. Furthermore, closing Oficina will also dispose all textures initialized, so if there is any leak of sorts, Oficina should be able to handle it nonetheless.

Definition at line 711 of file [render.hpp](#).

### 8.22.2 Member Function Documentation

#### 8.22.2.1 [load\(\)](#) [1/2]

```
static ofTexture oficina::ofTexturePool::load (
 std::string filename) [static]
```

Loads a texture from disk.

#### Parameters

<i>filename</i>	File location on the disk.
-----------------	----------------------------

#### Returns

A reference to the loaded texture.

### 8.22.2.2 load() [2/2]

```
static ofTexture oficina::ofTexturePool::load (
 SDL_Surface * surf) [static]
```

Loads a texture from memory.

#### Parameters

<i>surf</i>	SDL surface containing texture data.
-------------	--------------------------------------

#### Returns

A reference to the loaded texture.

### 8.22.2.3 loadDefaultFont()

```
static ofFont oficina::ofTexturePool::loadDefaultFont (
 ofFontFaces fontface = ofFontFaceGohuFont) [static]
```

Loads one of Oficina's default font faces.

#### Parameters

<i>fontface</i>	The default font to be loaded.
-----------------	--------------------------------

#### Returns

A reference to the default font.

### 8.22.2.4 unload()

```
static void oficina::ofTexturePool::unload (
 ofTexture & t) [static]
```

Unloads a font.

#### Parameters

<i>t</i>	Reference to the font to be unloaded.
----------	---------------------------------------

The documentation for this class was generated from the following file:

- [render.hpp](#)

## 8.23 oficina::ofTextureRenderer Class Reference

Tool for easily rendering 2D textures or texture atlases.

```
#include <render.hpp>
```

### Public Member Functions

- void [init](#) ([ofTexture](#) t, glm::uvec2 frameSize=glm::uvec2(0, 0))  
*Initializes the renderer.*
- void [render](#) (glm::vec2 position, glm::mat4 mvp, [ofdword](#) frame=0u, glm::vec4 color=glm::vec4(1.0f))  
*Renders a frame of the texture.*
- void [unload](#) ()  
*Unloads the texture renderer.*
- [ofTextureRenderer](#) & [operator=](#) (const [ofTextureRenderer](#) &other)  
*"Equals" operator for cloning a texture renderer.*
- void [SetTexture](#) ([ofTexture](#) t)  
*Dynamically changes the texture used by the renderer. Particularly useful for handling skins and such.*
- bool [isInit](#) () const  
*Checks for texture renderer initialization.*

### 8.23.1 Detailed Description

Tool for easily rendering 2D textures or texture atlases.

Definition at line 736 of file [render.hpp](#).

### 8.23.2 Member Function Documentation

#### 8.23.2.1 init()

```
void oficina::ofTextureRenderer::init (
 ofTexture t,
 glm::uvec2 frameSize = glm::uvec2(0, 0))
```

Initializes the renderer.

#### Parameters

<i>t</i>	Reference to the texture.
<i>frameSize</i>	2D unsigned integer vector with the size of a frame on the texture. Particularly useful if handling texture atlases. If ignored or passed with null values, the renderer will treat the whole texture as a single frame.

## 8.23.2.2 isInit()

```
bool oficina::ofTextureRenderer::isInit () const
```

Checks for texture renderer initialization.

## Returns

Whether the texture renderer was initialized or not.

## 8.23.2.3 operator=()

```
ofTextureRenderer& oficina::ofTextureRenderer::operator= (
 const ofTextureRenderer & other)
```

"Equals" operator for cloning a texture renderer.

## Parameters

<i>other</i>	Texture renderer to be cloned.
--------------	--------------------------------

## Returns

A reference to this renderer.

## 8.23.2.4 render()

```
void oficina::ofTextureRenderer::render (
 glm::vec2 position,
 glm::mat4 mvp,
 ofdword frame = 0u,
 glm::vec4 color = glm::vec4(1.0f))
```

Renders a frame of the texture.

## Parameters

<i>position</i>	Position of the texture (centered) on the matrix.
<i>mvp</i>	Model-View-Projection matrix to be used when drawing the texture.
<i>frame</i>	Frame to be retrieved from the texture, if it's a texture atlas. Frames are counted left to right, up to down respectively, starting at zero and assuming how many textures of the already given frame size fit on the texture's horizontal size. If texture is not an atlas, value defaults to 0, as the whole texture corresponds to its frame size, and therefore only one frame will fit it.
<i>color</i>	4D vector specifying which color the texture should be tinted with. Corresponds to a format {R, G, B, A}. Default values are {1, 1, 1, 1}.

**Warning**

This rendering process uses a dynamic buffer and uploads vertex buffer data every frame, so it is as efficient as it can get, given that it has a dynamic and general-purpose behaviour.

**8.23.2.5 SetTexture()**

```
void oficina::ofTextureRenderer::SetTexture (
 ofTexture t)
```

Dynamically changes the texture used by the renderer. Particularly useful for handling skins and such.

**Warning**

Be wary that this operation will not change the frame size, therefore both images should have the same padding.

**Parameters**

<i>t</i>	Texture to be now associated with this renderer.
----------	--------------------------------------------------

**8.23.2.6 unload()**

```
void oficina::ofTextureRenderer::unload ()
```

Unloads the texture renderer.

**Warning**

This operation will not unload the texture itself.

The documentation for this class was generated from the following file:

- [render.hpp](#)

**8.24 oficina::ofTimeSpan Class Reference**

Tool for counting and compare fixed amounts of time, independent from the game's time variation.

```
#include <timer.hpp>
```



## Public Member Functions

- void [begin](#) ()  
*Registers current time and begins counting.*
- float [yieldSpan](#) ()  
*Yields the current time from the beginning.*
- float [resetSpan](#) ()  
*Resets the time span, effectively restarting from zero.*
- float [stop](#) ()  
*Stops the time span.*
- bool [isRunning](#) () const  
*Yields the state of the time span.*

### 8.24.1 Detailed Description

Tool for counting and compare fixed amounts of time, independent from the game's time variation.

Definition at line 31 of file [timer.hpp](#).

### 8.24.2 Member Function Documentation

#### 8.24.2.1 isRunning()

```
bool oficina::ofTimeSpan::isRunning () const
```

Yields the state of the time span.

#### Returns

Whether the time span is running or not.

#### 8.24.2.2 resetSpan()

```
float oficina::ofTimeSpan::resetSpan ()
```

Resets the time span, effectively restarting from zero.

#### Returns

Time, in seconds, before the span was reset.

### 8.24.2.3 stop()

```
float oficina::ofTimeSpan::stop ()
```

Stops the time span.

#### Returns

Time, in seconds, before the span was stopped.

### 8.24.2.4 yieldSpan()

```
float oficina::ofTimeSpan::yieldSpan ()
```

Yields the current time from the beginning.

#### Returns

Current time from the beginning of the span, in seconds.

The documentation for this class was generated from the following file:

- [timer.hpp](#)

## 8.25 oficina::ofVertexArray Class Reference

Represents a vertex array for binding shader and vertex data.

```
#include <render.hpp>
```

### Public Member Functions

- void [init](#) ()  
*Initializes (generates) the vertex array.*
- void [unload](#) ()  
*Unloads (deletes) the vertex array.*
- void [bind](#) ()  
*Binds the vertex array.*
- void [unbind](#) ()  
*Unbinds any bound vertex array.*
- void [draw](#) ([ofPrimitiveType](#) mode, int firstVertexIdx, size\_t vertexCount)  
*Draws any primitive based on bound vertex buffer and vertex attributes.*
- [ofVertexArray](#) & [operator=](#) (const [ofVertexArray](#) &other)  
*"Equals" operator for cloning vertex arrays.*

### 8.25.1 Detailed Description

Represents a vertex array for binding shader and vertex data.

Definition at line 613 of file [render.hpp](#).

### 8.25.2 Member Function Documentation

#### 8.25.2.1 draw()

```
void oficina::ofVertexArray::draw (
 ofPrimitiveType mode,
 int firstVertexIdx,
 size_t vertexCount)
```

Draws any primitive based on bound vertex buffer and vertex attributes.

#### Warning

Vertex buffer and vertex attributes must be properly initialized and bound.

#### Parameters

<i>mode</i>	Primitive to be drawn.
<i>firstVertexIdx</i>	Index of the first vertex to be used.
<i>vertexCount</i>	Amount of vertices to be used.

#### 8.25.2.2 operator=()

```
ofVertexArray& oficina::ofVertexArray::operator= (
 const ofVertexArray & other)
```

"Equals" operator for cloning vertex arrays.

#### Parameters

<i>other</i>	Vertex array to be cloned.
--------------	----------------------------

#### Returns

Reference to this vertex array.

The documentation for this class was generated from the following file:

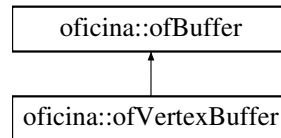
- [render.hpp](#)

## 8.26 oficina::ofVertexBuffer Class Reference

Represents a Vertex Buffer object (VBO). Use this to hold data related to drawing.

```
#include <render.hpp>
```

Inheritance diagram for oficina::ofVertexBuffer:



### Public Member Functions

- [ofVertexBuffer\(\)](#)  
*Buffer constructor.*

### Additional Inherited Members

#### 8.26.1 Detailed Description

Represents a Vertex Buffer object (VBO). Use this to hold data related to drawing.

Definition at line [289](#) of file [render.hpp](#).

The documentation for this class was generated from the following file:

- [render.hpp](#)

## Chapter 9

# File Documentation

### 9.1 benchmark.hpp File Reference

Oficina's default benchmarking utilities.

```
#include <string>
```

#### Functions

- void [oficina::ofBenchmarkStart](#) (float spanTimeS)  
*Starts the benchmarking process.*
- void [oficina::ofBenchmarkUpdateCall](#) ()  
*Updates the benchmarking process, and yields a debriefing if necessary. Must be called every frame.*
- void [oficina::ofBenchmarkEnd](#) ()  
*Stops the benchmarking process.*
- bool [oficina::ofBenchmarkIsRunning](#) ()  
*Shows the benchmarking process status.*

#### 9.1.1 Detailed Description

Oficina's default benchmarking utilities.

Benchmarking utilities for quick usage inside canvases. Uses an internal timer and must be updated by the user's own created canvas. Works better with VSync deactivated.

#### Author

Lucas Vieira

Definition in file [benchmark.hpp](#).

#### 9.1.2 Function Documentation

### 9.1.2.1 ofBenchmarkIsRunning()

```
bool oficina::ofBenchmarkIsRunning ()
```

Shows the benchmarking process status.

#### Returns

Whether benchmarking is active or not.

### 9.1.2.2 ofBenchmarkStart()

```
void oficina::ofBenchmarkStart (
 float spanTimeS)
```

Starts the benchmarking process.

#### Parameters

<i>spanTimeS</i>	Time between each benchmark debriefing.
------------------	-----------------------------------------

## 9.2 benchmark.hpp

```
00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00027 #pragma once
00028
00029 #include <string>
00030
00031 namespace oficina
00032 {
00033 void ofBenchmarkStart(float spanTimeS);
00034
00035 void ofBenchmarkUpdateCall();
00036
00037 void ofBenchmarkEnd();
00038
00039 bool ofBenchmarkIsRunning();
00040 }
00041 }
```

## 9.3 canvas.hpp File Reference

Tools for creating game scenes and manage such scenes.

```
#include <list>
#include <queue>
#include <sstream>
#include "oficina2/types.hpp"
#include "oficina2/render.hpp"
```

## Classes

- class [oficina::ofCanvas](#)  
*Default interface for creating and managing canvases.*
- class [oficina::ofCanvasManager](#)  
*Static class for handling canvases in general.*

### 9.3.1 Detailed Description

Tools for creating game scenes and manage such scenes.

Provides tools for creating canvases (scenes) and managing them. Also includes tools for managing the variable watcher and the repl.

#### Author

Lucas Vieira

Definition in file [canvas.hpp](#).

## 9.4 canvas.hpp

```
00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00026 #pragma once
00027 #include <list>
00028 #include <queue>
00029 #include <sstream>
00030
00031 #include "oficina2/types.hpp"
00032 #include "oficina2/render.hpp"
00033
00034 namespace oficina
00035 {
00036 // Pre-definition of ofCanvasManager so we can refer to it from ofCanvas.
00037 class ofCanvasManager;
00039 class ofCanvas
00040 {
00041 friend class ofCanvasManager;
00042 private:
00043 bool m_init = false;
00044 bool m_load = false;
```

```

00045 bool m_remove = false;
00046 int depth = 0;
00047 std::string m_name = "";
00048 public:
00050 virtual ~ofCanvas() {}
00054 virtual void init() = 0;
00059 virtual void load() = 0;
00061 virtual void unload() = 0;
00067 virtual void update(float dt) = 0;
00070 virtual void draw() = 0;
00074 virtual void remove() final;
00075 };
00076
00086 class ofCanvasManager
00087 {
00088 public:
00090 enum ofDebuggerState
00091 {
00092 ofDebuggerOff = 0u,
00093 ofDebuggerVars = 1u,
00094 ofDebuggerRepl = 2u
00095 };
00096
00098 static void init();
00107 // TODO: Explain canvas depth in documentation
00108 static void add(ofCanvas* c, int depth = 0, std::string name = "");
00113 static void remove(ofCanvas* c);
00118 static void unload();
00128 static void update(float dt);
00131 static void draw();
00132
00136 static std::vector<std::pair<std::string, std::string>> getCanvasList();
00137
00143 static std::ostream& dbg_ReplOutputStream();
00146 static ofDebuggerState dbg_getState();
00153 static void dbg_callEval();
00156 static void dbg_ChangeState();
00158 static void dbg_ReplHistoryPrev();
00160 static void dbg_ReplHistoryNext();
00164 static ofdword dbg_ReplLineNumber();
00169 static void dbg_setFont(oficina::ofFontFaces fontFace);
00170 private:
00171 class ofDebugCanvas : public ofCanvas
00172 {
00173 public:
00174 void init();
00175 void load();
00176 void unload();
00177 void update(float dt);
00178 void draw();
00179 };
00180
00181 static ofDebugCanvas m_debugger;
00182 };
00183 }

```

## 9.5 display.hpp File Reference

Tools for configuring windows for video output.

```

#include <SDL2/SDL.h>
#include <list>
#include <string>
#include "oficina2/types.hpp"
#include "oficina2/timer.hpp"

```

### Classes

- class [oficina::ofDisplay](#)

*Represents a single window prepared for receiving a context.*



## Enumerations

- enum `oficina::ofFrameRateConfig` { `oficina::ofFPSVariable` = 0x00, `oficina::ofFPSCapped` = 0x01, `oficina::ofFPSMinimum` = 0x02, `oficina::ofFPSBoth` = 0x04 }

*Describes the types of framerate.*

### 9.5.1 Detailed Description

Tools for configuring windows for video output.

Provides tools for creating displays (game windows).

#### Author

Lucas Vieira

Definition in file [display.hpp](#).

### 9.5.2 Enumeration Type Documentation

#### 9.5.2.1 ofFrameRateConfig

enum `oficina::ofFrameRateConfig`

Describes the types of framerate.

#### Enumerator

<code>ofFPSVariable</code>	Freely allows FPS to float.
<code>ofFPSCapped</code>	Caps FPS to a maximum number. This will ignore further rendering frames after the nth frame is rendered.
<code>ofFPSMinimum</code>	Caps minimal FPS so that, when FPS goes below the minimum number, the application will force the time variation to assume the minimum number's. The practical explanation is that this will force a slowdown so there are no strange frame jumps.
<code>ofFPSBoth</code>	Induces FPS to assume both behaviours of Capped and Minimum.

Definition at line 36 of file [display.hpp](#).

## 9.6 display.hpp

```

00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *

```

```

00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 * *****/
00015
00024 #pragma once
00025
00026 #include <SDL2/SDL.h>
00027 #include <list>
00028 #include <string>
00029 #include "oficina2/types.hpp"
00030 #include "oficina2/timer.hpp"
00031
00032 namespace oficina
00033 {
00034 // a display may assume when rendering the game.
00035 enum ofFrameRateConfig {
00036 ofFPSVariable = 0x00,
00042 ofFPSCapped = 0x01,
00050 ofFPSMinimum = 0x02,
00053 ofFPSBoth = 0x04
00054 };
00055
00059 class ofDisplay
00060 {
00061 public:
00068 void pushArg(std::string arg);
00072 void open();
00076 void close();
00080 void swap();
00081
00084 SDL_Window* getHandle() const;
00088 glm::uvec2 getSize() const;
00094 bool isOpen() const;
00095
00101 void setSize(glm::uvec2 NewSize);
00106 void setSwapInterval(ofFrameRateConfig cfg, float max = 0.0f
, float min = 0.0f);
00107
00112 bool isFullscreen() const;
00117 void setFullscreen(bool state);
00120 float getDeltaTime() const;
00121 private:
00122 SDL_Window* m_wnd = nullptr;
00123 std::list<std::string> m_confv;
00124 std::string m_title = "OficinaFramework 2.0";
00125 std::string m_iconpath;
00126 glm::uvec2 m_size = glm::uvec2(1280u, 720u);
00127 bool m_full = false;
00128 ofTimeSpan m_dtSpan;
00129 float m_dt = 0.0f;
00130 ofFrameRateConfig m_frmrtt = ofFPSVariable;
00131 float m_frmrt_min = 0.0f;
00132 float m_frmrt_max = 0.0f;
00133 void parseArgs();
00134 };
00135 }

```

## 9.7 entity.hpp File Reference

Interfaces and tools for managing objects ingame.

```

#include "oficina2/types.hpp"
#include <map>

```

### Classes

- class [oficina::ofComponent](#)  
*Defines a single component to be attached to an entity.*
- class [oficina::ofEntity](#)  
*Abstract class representing one ingame entity.*

### 9.7.1 Detailed Description

Interfaces and tools for managing objects ingame.

Provides tools for creating, managing, storing and manipulating ingame objects. Some tools are specially optimized using well-known algorithms.

#### Author

Lucas Vieira

Definition in file [entity.hpp](#).

## 9.8 entity.hpp

```

00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00026 #pragma once
00027
00028 #include "oficina2/types.hpp"
00029 #include <map>
00030
00031 namespace oficina
00032 {
00033 class ofEntity;
00037 class ofIComponent
00038 {
00039 public:
00042 virtual ~ofIComponent() {}
00045 virtual void init() = 0;
00048 virtual void load() {}
00051 virtual void unload() {}
00054 virtual void update(float dt) = 0;
00059 virtual void draw(glm::mat4 ViewProjection) {}
00065 std::string getType() const;
00066 protected:
00070 ofEntity* parent;
00076 void parseType();
00077 private:
00081 std::string m_type = "oficina::ofIComponent";
00082 };
00083
00088 class ofEntity
00089 {
00090 public:
00092 virtual ~ofEntity() {}
00095 virtual void init() = 0;
00098 virtual void load() = 0;
00100 virtual void unload() = 0;
00105 virtual void update(float dt) = 0;
00114 virtual void draw(glm::mat4 ViewProjection) = 0;
00115
00122 void translate(glm::vec3 coord,
00123 bool loadIdentity = false);
00130 void rotate(float theta,
00131 glm::vec3 axis,
00132 bool loadIdentity = false);
00138 void scale(glm::vec3 amount,
00139 bool loadIdentity);
00140

```

```

00144 void SetProperty(ofbyte which, bool state);
00147 void ToggleProperty(ofbyte which);
00152 void setName(std::string name);
00153
00156 glm::mat4 getModelMatrix();
00159 glm::vec3 getPosition() const;
00162 glm::vec3 getEulerAngles() const;
00165 glm::vec3 getScale() const;
00169 bool getProperty(ofbyte which);
00173 ofdword getPropertyMask() const;
00176 std::string getName() const;
00177
00185 void AddComponent(std::string name, ofIComponent* component);
00189 ofIComponent* const GetComponent(std::string name);
00192 void RemoveComponent(std::string name);
00194 void ClearComponents();
00195
00200 void UpdateComponents(float dt);
00205 void DrawComponents(glm::mat4 ViewProjection);
00206 protected:
00210 glm::mat4 translation;
00214 glm::mat4 rotation;
00218 glm::mat4 scaling;
00219
00222 glm::vec3 position;
00225 glm::vec3 eulerangles;
00228 glm::vec3 magnification = glm::vec3(1.0f);
00229
00231 ofdword propertymask = 0x00000000u;
00232
00234 std::map<std::string, ofIComponent*> components;
00235
00237 std::string name = "[unnamed]";
00238 };
00239 }

```

## 9.9 input.hpp File Reference

Special tools for handling player input.

```

#include "oficina2/types.hpp"
#include <SDL2/SDL.h>
#include <string>

```

### Classes

- struct [oficina::ofInputState](#)  
*Holds an input state every frame.*

### Enumerations

- enum [oficina::ofStick](#) { [oficina::ofStickLeft](#) = 0x01u, [oficina::ofStickRight](#) = 0x02u }  
*Enumeration for gamepad sticks.*
- enum [oficina::ofStickAxis](#) { [oficina::ofStickHoriz](#) = 0x04u, [oficina::ofStickVert](#) = 0x08u }  
*Enumeration for gamepad sticks' axis.*
- enum [oficina::ofStickSignal](#) { [oficina::ofStickNegative](#) = 0x10u, [oficina::ofStickPositive](#) = 0x20u }  
*Enumeration for gamepad sticks' axis' signal/direction.*

- enum `oficina::ofPadButton` {  
`oficina::ofPadStart` = 0x0001u, `oficina::ofPadBack` = 0x0002u, `oficina::ofPadA` = 0x0004u, `oficina::ofPadB` = 0x0008u,  
`oficina::ofPadX` = 0x0010u, `oficina::ofPadY` = 0x0020u, `oficina::ofPadLS` = 0x0040u, `oficina::ofPadRS` = 0x0080u,  
`oficina::ofPadDUp` = 0x0100u, `oficina::ofPadDDown` = 0x0200u, `oficina::ofPadDLeft` = 0x0400u, `oficina::ofPadDRight` = 0x0800u,  
`oficina::ofPadLB` = 0x1000u, `oficina::ofPadLT` = 0x2000u, `oficina::ofPadRB` = 0x4000u, `oficina::ofPadRT` = 0x8000u }  
*Enumeration for gamepad buttons. Layout based on Xbox 360 controller.*
- enum `oficina::ofMouseButton` { `oficina::ofMouseLeft` = 0x01u, `oficina::ofMouseMid` = 0x02u, `oficina::ofMouseRight` = 0x04u }  
*Enumeration representing mouse buttons.*
- enum `oficina::ofPlayer` { `oficina::ofPlayerOne` = 0u, `oficina::ofPlayerTwo` = 1u, `oficina::ofPlayerThree` = 2u, `oficina::ofPlayerFour` = 3u }  
*Enumeration representing connected players.*

## Functions

- void `oficina::ofUpdateEventDispatch` ()  
*Updates and dispatches input events.*
- ofInputState `oficina::ofGetInputState` (ofPlayer player=ofPlayerOne)  
*Grabs the whole of the current input state in a single struct.*
- bool `oficina::ofIsGamepadConnected` (ofPlayer player=ofPlayerOne)  
*Yields the state of a player's gamepad.*
- glm::vec2 `oficina::ofGetLeftStick` (ofPlayer player=ofPlayerOne)  
*Yields the gamepad's left stick coordinates. Each coordinate yields from -1.0f (left/up) to 1.0f (right/down).*
- glm::vec2 `oficina::ofGetRightStick` (ofPlayer player=ofPlayerOne)  
*Yields the gamepad's right stick coordinates. Each coordinate yields from -1.0f (left/up) to 1.0f (right/down).*
- float `oficina::ofGetLeftTrigger` (ofPlayer player=ofPlayerOne)  
*Yields a value stating the amount of pressing on the gamepad's left trigger. Value ranges from 0.0f (not pressed) to 1.0f (fully held).*
- float `oficina::ofGetRightTrigger` (ofPlayer player=ofPlayerOne)  
*Yields a value stating the amount of pressing on the gamepad's right trigger. Value ranges from 0.0f (not pressed) to 1.0f (fully held).*
- bool `oficina::ofButtonPress` (ofPadButton button, ofPlayer player=ofPlayerOne)  
*Yields the pressing state of a specific button on the gamepad.*
- bool `oficina::ofButtonTap` (ofPadButton button, ofPlayer player=ofPlayerOne)  
*Yields the tap state of a specific button on the gamepad.*
- bool `oficina::ofStickMovedTowards` (ofbyte stickDirectionMask, ofPlayer player=ofPlayerOne)  
*Checks if a specific stick was moved in a specific direction.*
- glm::vec2 `oficina::ofGetMousePos` ()  
*Yields the mouse position's coordinates inside the display.*
- bool `oficina::ofMouseButtonPress` (ofMouseButton button)  
*Yields the pressing state of a specific mouse button.*
- bool `oficina::ofMouseButtonTap` (ofMouseButton button)  
*Yields the tap state of a specific mouse button.*
- void `oficina::ofMapDefaultsP1` ()  
*Maps default bindings for gamepad buttons on the keyboard - Player 1 only.*
- void `oficina::ofMapKeyToButton` (ofPadButton button, SDL\_Scancode scancode, ofPlayer player=ofPlayerOne)  
*Binds a specific keyboard key to a gamepad button.*

- void `oficina::ofMapKeyToStick` (`ofbyte` stickPositionMask, `SDL_Scancode` scancode, `ofPlayer` player=`ofPlayerOne`)  
*Binds a specific keyboard key to a movement on a gamepad stick.*
- void `oficina::ofMapButtonRemove` (`ofPadButton` button, `ofPlayer` player=`ofPlayerOne`)  
*Remove the binding to a gamepad button by a keyboard key, if such binding exists.*
- void `oficina::ofMapStickRemove` (`ofbyte` stickPositionMask, `ofPlayer` player=`ofPlayerOne`)  
*Remove the binding to a gamepad stick by a keyboard key, if such binding exists.*
- void `oficina::ofMappingClear` (`ofPlayer` player=`ofPlayerOne`)  
*Clear all keyboard key mappings done to a specific player's gamepad.*
- void `oficina::ofStartTextInput` ()  
*Begins text input to the internal keyboard text input logger.  
This will erase all of the previously stored text input.*
- `std::string` `oficina::ofGetTextInput` ()  
*Retrieves all text input that was made between text input's start and end call.*
- void `oficina::ofSetTextInput` (`std::string` str)  
*Redefines the current text input to a specific string.  
Particularly useful if you plan to save your text input after your text control loses focus, which should be called after restarting the text input.*
- bool `oficina::ofIsInputtingText` ()  
*Checks for the state of text input.*
- void `oficina::ofStopTextInput` ()  
*Stops text input, if already started.*
- void `oficina::ofClearTextInput` ()  
*Clears the current text input buffer completely.*
- void `oficina::ofTextInputSetPadding` (`ofdword` padding)  
*Defines a padding of white spaces for the text input, every time the player types a new line (Shift + Enter).*

### 9.9.1 Detailed Description

Special tools for handling player input.

Functions, tools and enumerations for handling input such as keyboard, mouse and gamepad. Also automatically handles typing and gamepad connection management.

#### Author

Lucas Vieira

Definition in file [input.hpp](#).

### 9.9.2 Enumeration Type Documentation

#### 9.9.2.1 ofMouseButton

```
enum oficina::ofMouseButton
```

Enumeration representing mouse buttons.

#### Note

You can cast this to an `ofbyte`.

## Enumerator

ofMouseLeft	Left mouse button.
ofMouseMid	Middle mouse button (wheel, when pressed).
ofMouseRight	Right mouse button.

Definition at line 116 of file [input.hpp](#).

## 9.9.2.2 ofPadButton

```
enum oficina::ofPadButton
```

Enumeration for gamepad buttons. Layout based on Xbox 360 controller.

## Note

You can cast this to an ofword.

## Enumerator

ofPadStart	Gamepad START button.
ofPadBack	Gamepad BACK button.
ofPadA	Gamepad A button.
ofPadB	Gamepad B button.
ofPadX	Gamepad X button.
ofPadY	Gamepad Y button.
ofPadLS	Gamepad LEFT STICK (when pressed).
ofPadRS	Gamepad RIGHT STICK (when pressed).
ofPadDUp	Gamepad DIGITAL UP button.
ofPadDDown	Gamepad DIGITAL DOWN button.
ofPadDLeft	Gamepad DIGITAL LEFT button.
ofPadDRight	Gamepad DIGITAL RIGHT button.
ofPadLB	Gamepad LB (LEFT BUMPER) button.
ofPadLT	Gamepad LT (LEFT TRIGGER).  <b>Note</b>  Although this is a trigger, its usage can also be handled as a button, which will trigger when this trigger is minimally pressed (greater than 0.0f).
ofPadRB	Gamepad RB (RIGHT BUMPER) button.
ofPadRT	Gamepad RT (RIGHT TRIGGER).  <b>Note</b>  Although this is a trigger, its usage can also be handled as a button, which will trigger when this trigger is minimally pressed (greater than 0.0f).

Definition at line 70 of file [input.hpp](#).

### 9.9.2.3 ofPlayer

```
enum oficina::ofPlayer
```

Enumeration representing connected players.

#### Note

Supports up to 4 players connected at once.  
You can cast this to any integer type.

#### Enumerator

ofPlayerOne	Player one (Gamepad #1).
ofPlayerTwo	Player two (Gamepad #2).
ofPlayerThree	Player three (Gamepad #3).
ofPlayerFour	Player four (Gamepad #4).

Definition at line 129 of file [input.hpp](#).

### 9.9.2.4 ofStick

```
enum oficina::ofStick
```

Enumeration for gamepad sticks.

#### Note

You can cast this to an ofbyte.

#### Enumerator

ofStickLeft	Gamepad left stick.
ofStickRight	Gamepad right stick.

Definition at line 36 of file [input.hpp](#).

### 9.9.2.5 ofStickAxis

```
enum oficina::ofStickAxis
```

Enumeration for gamepad sticks' axis.



**Note**

You can cast this to an ofbyte.

**Enumerator**

ofStickHoriz	Gamepad sticks' horizontal axis.
ofStickVert	Gamepad sticks' vertical axis.

Definition at line 46 of file [input.hpp](#).

**9.9.2.6 ofStickSignal**

```
enum oficina::ofStickSignal
```

Enumeration for gamepad sticks' axis' signal/direction.

**Note**

You can cast this to an ofbyte.

**Enumerator**

ofStickNegative	Gamepad stick axis' negative (left/up) direction.
ofStickPositive	Gamepad stick axis' positive (right/down) direction.

Definition at line 57 of file [input.hpp](#).

**9.9.3 Function Documentation****9.9.3.1 ofButtonPress()**

```
bool oficina::ofButtonPress (
 ofPadButton button,
 ofPlayer player = ofPlayerOne)
```

Yields the pressing state of a specific button on the gamepad.

**Note**

This function yields the state of a button when pressed and held. For a single tap, see ofButtonTap.

**See also**

[ofButtonTap](#)

**Parameters**

<i>button</i>	Which gamepad button should be compared.
<i>player</i>	Which player's gamepad should be compared.

**Returns**

Whether the related button is being held down or not.

**9.9.3.2 ofButtonTap()**

```
bool oficina::ofButtonTap (
 ofPadButton button,
 ofPlayer player = ofPlayerOne)
```

Yields the tap state of a specific button on the gamepad.

**Note**

This function yields the state of a button when pressed on a single frame. Holding down the button for more than a frame will not trigger this event more than once per press. For continuously holding the button, see `ofButtonPress`.

**See also**

`ofButtonPress`

**Parameters**

<i>button</i>	Which gamepad button should be compared.
<i>player</i>	Which player's gamepad should be compared.

**Returns**

Whether the related button was pressed on the current frame or not.

**9.9.3.3 ofGetInputState()**

```
ofInputState oficina::ofGetInputState (
 ofPlayer player = ofPlayerOne)
```

Grabs the whole of the current input state in a single struct.

**Parameters**

<i>player</i>	Which player's gamepad state should be yielded.
---------------	-------------------------------------------------

**Returns**

A struct containing the player's input state.

**See also**

ofInputState

**9.9.3.4 ofGetLeftStick()**

```
glm::vec2 oficina::ofGetLeftStick (
 ofPlayer player = ofPlayerOne)
```

Yields the gamepad's left stick coordinates. Each coordinate yields from -1.0f (left/up) to 1.0f (right/down).

**Parameters**

<i>player</i>	Which player's gamepad's left stick should be yielded.
---------------	--------------------------------------------------------

**Returns**

A 2D vector containing the left stick state.

**9.9.3.5 ofGetLeftTrigger()**

```
float oficina::ofGetLeftTrigger (
 ofPlayer player = ofPlayerOne)
```

Yields a value stating the amount of pressing on the gamepad's left trigger. Value ranges from 0.0f (not pressed) to 1.0f (fully held).

**Parameters**

<i>player</i>	Which player's gamepad's left trigger should be yielded.
---------------	----------------------------------------------------------

**Returns**

A floating point containing the left trigger state.

#### 9.9.3.6 ofGetMousePos()

```
glm::vec2 oficina::ofGetMousePos ()
```

Yields the mouse position's coordinates inside the display.

##### Returns

A 2D vector containing the mouse position.

#### 9.9.3.7 ofGetRightStick()

```
glm::vec2 oficina::ofGetRightStick (
 ofPlayer player = ofPlayerOne)
```

Yields the gamepad's right stick coordinates. Each coordinate yields from -1.0f (left/up) to 1.0f (right/down).

##### Parameters

<i>player</i>	Which player's gamepad's right stick should be yielded.
---------------	---------------------------------------------------------

##### Returns

A 2D vector containing the right stick state.

#### 9.9.3.8 ofGetRightTrigger()

```
float oficina::ofGetRightTrigger (
 ofPlayer player = ofPlayerOne)
```

Yields a value stating the amount of pressing on the gamepad's right trigger. Value ranges from 0.0f (not pressed) to 1.0f (fully held).

##### Parameters

<i>player</i>	Which player's gamepad's right trigger should be yielded.
---------------	-----------------------------------------------------------

##### Returns

A floating point containing the right trigger state.

#### 9.9.3.9 ofGetTextInput()

```
std::string oficina::ofGetTextInput ()
```

Retrieves all text input that was made between text input's start and end call.

In case you are displaying text onscreen, the actual text input should always be retrieved; it will modify as needed. The text will also not be erased when text input is stopped.

##### Returns

A string containing the current state of the last text input requirement.

#### 9.9.3.10 ofIsGamepadConnected()

```
bool oficina::ofIsGamepadConnected (
 ofPlayer player = ofPlayerOne)
```

Yields the state of a player's gamepad.

A player which gamepad is not connected will automatically fallback to its keyboard bindings, if registered.

##### Parameters

<i>player</i>	Which player's gamepad connection state should be yielded.
---------------	------------------------------------------------------------

##### Returns

Whether the related player's gamepad is connected or not.

#### 9.9.3.11 ofIsInputtingText()

```
bool oficina::ofIsInputtingText ()
```

Checks for the state of text input.

##### Returns

Whether the player is currently in text input mode or not.

#### 9.9.3.12 ofMapButtonRemove()

```
void oficina::ofMapButtonRemove (
 ofPadButton button,
 ofPlayer player = ofPlayerOne)
```

Remove the binding to a gamepad button by a keyboard key, if such binding exists.

## Parameters

<i>button</i>	Desired button to remove mappings.
<i>player</i>	Which player's gamepad was bound.

## 9.9.3.13 ofMapDefaultsP1()

```
void oficina::ofMapDefaultsP1 ()
```

Maps default bindings for gamepad buttons on the keyboard - Player 1 only.

This function will map default bindings for Player 1, for gamepad buttons and sticks, as per the table below:

Keyboard key	Equivalency
-----;	;-----;
Up Arrow	Left Stick, Up (Vertical, Negative)
Down Arrow	Left Stick, Down (Vertical, Positive)
Left Arrow	Left Stick, Left (Horizontal, Negative)
Right Arrow	Left Stick, Right (Horizontal, Positive)
I	Right Stick, Up (Vertical, Negative)
K	Right Stick, Down (Vertical, Positive)
J	Right Stick, Left (Horizontal, Negative)
L	Right Stick, Right (Horizontal, Positive)
Enter (Return)	ofPadStart
Backspace	ofPadBack
W	ofPadY
A	ofPadX
S	ofPadA
D	ofPadB
Z	ofPadLS
C	ofPadRS
1 (non-numpad)	ofPadDUp
2 (non-numpad)	ofPadDRight
3 (non-numpad)	ofPadDDown
4 (non-numpad)	ofPadDLeft
Q	ofPadLB
E	ofPadRB
Tab	ofPadLT
R	ofPadRT

## See also

ofMapKeyToButton  
ofMapKeyToStick

## 9.9.3.14 ofMapKeyToButton()

```
void oficina::ofMapKeyToButton (
 ofPadButton button,
 SDL_Scancode scancode,
 ofPlayer player = ofPlayerOne)
```

Binds a specific keyboard key to a gamepad button.

## Parameters

<i>button</i>	Desired button to map.
<i>scancode</i>	SDL_Scancode for the key to be mapped. Check SDL2's documentation to see all available scancodes.
<i>player</i>	Which player's gamepad should the key be bound to.

## 9.9.3.15 ofMapKeyToStick()

```
void oficina::ofMapKeyToStick (
 ofbyte stickPositionMask,
 SDL_Scancode scancode,
 ofPlayer player = ofPlayerOne)
```

Binds a specific keyboard key to a movement on a gamepad stick.

## Parameters

<i>stickPositionMask</i>	A bitmask specifying the desired stick, axis and direction to bind to. You can use the enums ofStick, ofStickAxis and ofStickSignal to create a specification. For example:  <code>ofMapKeyToStick(ofStickLeft   ofStickHoriz   ofStickNegative, SDL_SCANCODE_LEFT, ofPlayerOne );</code>
<i>scancode</i>	SDL_Scancode for the key to be mapped. Check SDL2's documentation to see all available scancodes.
<i>player</i>	Which player's gamepad should the key be bound to.

## 9.9.3.16 ofMappingClear()

```
void oficina::ofMappingClear (
 ofPlayer player = ofPlayerOne)
```

Clear all keyboard key mappings done to a specific player's gamepad.

## Parameters

<i>player</i>	Which player's gamepad was bound.
---------------	-----------------------------------

## 9.9.3.17 ofMapStickRemove()

```
void oficina::ofMapStickRemove (
 ofbyte stickPositionMask,
 ofPlayer player = ofPlayerOne)
```

Remove the binding to a gamepad stick by a keyboard key, if such binding exists.

#### Parameters

<i>stickPositionMask</i>	A bitmask specifying the desired stick, axis and direction that was bound. You can use the enums ofStick, ofStickAxis and ofStickSignal to create a specification. For example:  <code>ofMapStickRemove(ofStickLeft   ofStickHoriz   ofStickNegative, ofPlayerOne);</code>
<i>player</i>	Which player's gamepad was bound.

#### 9.9.3.18 ofMouseButtonPress()

```
bool oficina::ofMouseButtonPress (
 ofMouseButton button)
```

Yields the pressing state of a specific mouse button.

#### Note

This function yields the state of a button when pressed and held. For a single tap, see ofMouseButtonTap.

#### Parameters

<i>button</i>	Which mouse button should be compared.
---------------	----------------------------------------

#### Returns

Whether the related button is being held down or not.

#### See also

ofMouseButtonTap

#### 9.9.3.19 ofMouseButtonTap()

```
bool oficina::ofMouseButtonTap (
 ofMouseButton button)
```

Yields the tap state of a specific mouse button.

#### Note

This function yields the state of a button when pressed on a single frame. Holding down the button for more than a frame will not trigger this event more than once per press. For continuously holding the button, see ofMouseButtonPress.

#### See also

ofMouseButtonPress



## Parameters

<i>button</i>	Which mouse button should be compared.
---------------	----------------------------------------

## Returns

Whether the related button was pressed on the current frame or not.

## 9.9.3.20 ofSetTextInput()

```
void oficina::ofSetTextInput (
 std::string str)
```

Redefines the current text input to a specific string.

Particularly useful if you plan to save your text input after your text control loses focus, which should be called after restarting the text input.

## Note

This will erase the currently stored text input and replace it by the string that was fed.

## Parameters

<i>str</i>	Text to be fed to the current text input.
------------	-------------------------------------------

## 9.9.3.21 ofStartTextInput()

```
void oficina::ofStartTextInput ()
```

Begins text input to the internal keyboard text input logger.

This will erase all of the previously stored text input.

## Note

By default, text input will not accept multiline unless you press Shift + Enter.

## 9.9.3.22 ofStickMovedTowards()

```
bool oficina::ofStickMovedTowards (
 ofbyte stickDirectionMask,
 ofPlayer player = ofPlayerOne)
```

Checks if a specific stick was moved in a specific direction.

## Parameters

<i>stickDirectionMask</i>	<p>A bitmask specifying the desired stick, axis and direction to compare for. You can use the enums ofStick, ofStickAxis and ofStickSignal to create a specification. For example:</p> <pre>bool lstickMovedLeft = ofStickMovedTowards(ofStickLeft   ofStickHoriz   ofStickNegative) ;</pre>
<i>player</i>	Which player's gamepad should be compared.

## Returns

Whether the related stick was moved in the related direction or not.

## See also

ofStick  
ofStickAxis  
ofStickSignal

## 9.9.3.23 ofStopTextInput()

```
void oficina::ofStopTextInput ()
```

Stops text input, if already started.

## Note

Calling this function will not erase your text input buffer; you'll still be able to retrieve it until you start text input again.

## 9.9.3.24 ofTextInputSetPadding()

```
void oficina::ofTextInputSetPadding (
 ofdword padding)
```

Defines a padding of white spaces for the text input, every time the player types a new line (Shift + Enter).

## Note

For default reasons, the padding will only appear on the next new line. Padding will also not be output to the buffer on the start of text input.

## Parameters

<i>padding</i>	Unsigned integer specifying the amount of white spaces that should be fed to the text buffer, every time the player inputs a new line.
----------------	----------------------------------------------------------------------------------------------------------------------------------------

## 9.9.3.25 ofUpdateEventDispatch()

```
void oficina::ofUpdateEventDispatch ()
```

Updates and dispatches input events.

Unless automatically called by Oficina's premade game loop, this function should be called to grab the window's events and assign the received events to each input type.

## Note

You should never have to call this yourself, unless you're building your game loop from scratch.

## 9.10 input.hpp

```

00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00026 #pragma once
00027
00028 #include "oficina2/types.hpp"
00029 #include <SDL2/SDL.h>
00030 #include <string>
00031
00032 namespace oficina
00033 {
00036 enum ofStick
00037 {
00039 ofStickLeft = 0x01u,
00041 ofStickRight = 0x02u
00042 };
00043
00046 enum ofStickAxis
00047 {
00049 ofStickHoriz = 0x04u,
00051 ofStickVert = 0x08u
00052 };
00053
00057 enum ofStickSignal
00058 {
00061 ofStickNegative = 0x10u,
00064 ofStickPositive = 0x20u
00065 };
00066
00070 enum ofPadButton
00071 {
00073 ofPadStart = 0x0001u,
00075 ofPadBack = 0x0002u,
00077 ofPadA = 0x0004u,

```

```

00079 ofPadB = 0x0008u,
00081 ofPadX = 0x0010u,
00083 ofPadY = 0x0020u,
00085 ofPadLS = 0x0040u,
00087 ofPadRS = 0x0080u,
00089 ofPadDUp = 0x0100u,
00091 ofPadDDown = 0x0200u,
00093 ofPadDLeft = 0x0400u,
00095 ofPadDRight = 0x0800u,
00097 ofPadLB = 0x1000u,
00103 ofPadLT = 0x2000u,
00105 ofPadRB = 0x4000u,
00111 ofPadRT = 0x8000u
00112 };
00113
00116 enum ofMouseButton
00117 {
00119 ofMouseLeft = 0x01u,
00121 ofMouseMid = 0x02u,
00123 ofMouseRight = 0x04u
00124 };
00125
00129 enum ofPlayer
00130 {
00132 ofPlayerOne = 0u,
00134 ofPlayerTwo = 1u,
00136 ofPlayerThree = 2u,
00138 ofPlayerFour = 3u
00139 };
00140
00142 struct ofInputState
00143 {
00146 ofword padButtons = 0x0000u;
00149 float leftStick[2] = {0.0f, 0.0f};
00152 float rightStick[2] = {0.0f, 0.0f};
00155 float triggers[2] = {0.0f, 0.0f};
00156 };
00157
00165 void ofUpdateEventDispatch();
00170 ofInputState ofGetInputState(ofPlayer player =
ofPlayerOne);
00177 bool ofIsGamepadConnected(ofPlayer player =
ofPlayerOne);
00182 glm::vec2 ofGetLeftStick(ofPlayer player =
ofPlayerOne);
00187 glm::vec2 ofGetRightStick(ofPlayer player =
ofPlayerOne);
00192 float ofGetLeftTrigger(ofPlayer player =
ofPlayerOne);
00197 float ofGetRightTrigger(ofPlayer player =
ofPlayerOne);
00205 bool ofButtonPress(ofPadButton button,
ofPlayer player = ofPlayerOne);
00215 bool ofButtonTap(ofPadButton button, ofPlayer player =
ofPlayerOne);
00228 bool ofStickMovedTowards(ofbyte stickDirectionMask,
ofPlayer player = ofPlayerOne);
00229
00232 glm::vec2 ofGetMousePos();
00239 bool ofMouseButtonPress(ofMouseButton button);
00248 bool ofMouseButtonTap(ofMouseButton button);
00249
00254
00283 void ofMapDefaultsP1();
00292 void ofMapKeyToButton(ofPadButton button, SDL_Scancode scancode,
ofPlayer player = ofPlayerOne);
00305 void ofMapKeyToStick(ofbyte stickPositionMask, SDL_Scancode scancode,
ofPlayer player = ofPlayerOne);
00309 void ofMapButtonRemove(ofPadButton button,
ofPlayer player = ofPlayerOne);
00319 void ofMapStickRemove(ofbyte stickPositionMask,
ofPlayer player = ofPlayerOne);
00322 void ofMappingClear(ofPlayer player =
ofPlayerOne);
00323
00327 void ofStartTextInput();
00334 std::string ofGetTextInput();
00341 void ofSetTextInput(std::string str);
00344 bool ofIsInputtingText();
00348 void ofStopTextInput();
00350 void ofClearTextInput();
00357 void ofTextInputSetPadding(ofdword padding);
00358 }

```

## 9.11 io.hpp File Reference

Tools for handling non-player-related input and output.

```
#include <cstdarg>
#include <string>
#include "oficina2/platform.hpp"
#include <SDL2/SDL.h>
```

### Macros

- `#define OFLOG_NRM ""`  
*(Unix only) Preprocessor macro for concatenation with strings. Defines next outputted text to console's foreground color.*
- `#define OFLOG_RED ""`  
*(Unix only) Preprocessor macro for concatenation with strings. Defines next outputted text to console's red color.*
- `#define OFLOG_GRN ""`  
*(Unix only) Preprocessor macro for concatenation with strings. Defines next outputted text to console's green color.*
- `#define OFLOG_YEL ""`  
*(Unix only) Preprocessor macro for concatenation with strings. Defines next outputted text to console's yellow color.*
- `#define OFLOG_BLU ""`  
*(Unix only) Preprocessor macro for concatenation with strings. Defines next outputted text to console's blue color.*
- `#define OFLOG_MAG ""`  
*(Unix only) Preprocessor macro for concatenation with strings. Defines next outputted text to console's magenta color.*
- `#define OFLOG_CYN ""`  
*(Unix only) Preprocessor macro for concatenation with strings. Defines next outputted text to console's cyan color.*
- `#define OFLOG_WHT ""`  
*(Unix only) Preprocessor macro for concatenation with strings. Defines next outputted text to console's white color.*
- `#define OFLOG_RESET ""`  
*(Unix only) Preprocessor macro for concatenation with strings. Resets a previously defined console color.*

### Enumerations

- `enum oficina::ofLogLevel {`  
`oficina::ofLogCrit = 0, oficina::ofLogErr = 1, oficina::ofLogWarn = 2, oficina::ofLogInfo = 3,`  
`oficina::ofLogNone = 4 }`  
*Represents levels of logging to the log output.*
- `enum oficina::ofLogType { oficina::ofLogDisabled = 0, oficina::ofLogConsole = 1, oficina::ofLogFile = 2 }`  
*Represents types of log output.*

## Functions

- int `oficina::ofLog` (ofLogLevel level, const char \*fmt,...)  
*Logs text to the currently selected log type.*
- void `oficina::ofLogSetLevel` (ofLogLevel level)  
*Defines the minimum log priority level of the log function. Any level below the specified priority will not be output to the log.  
Defaults to ofLogNone.*
- ofLogType `oficina::ofLogGetType` ()  
*Yields the currently used logging type.*
- void `oficina::ofLogUseFile` (std::string filename)  
*Use a text file as logging tool.*
- void `oficina::ofLogUseConsole` ()  
*Use the console as logging tool. If on Windows, output will only be seen if the game was compiled using the CONSOLE subsystem.*
- void `oficina::ofLogDisable` ()  
*Disable logging completely.*
- void `oficina::ofFindAsset` (std::string &filename)  
*Finds a specific file on the folders it should be, then fixes its location.*
- void `oficina::ofSetDataDirectoryName` (std::string dirname)  
*Sets the name of the folder used to enclose the game's assets (outside of project/application run folder).*
- std::string `oficina::ofLoadText` (std::string filename)  
*Load a text file from the filesystem.*
- SDL\_Surface \* `oficina::ofLoadImage` (std::string filename)  
*Loads a surface containing a image from the filesystem.*

### 9.11.1 Detailed Description

Tools for handling non-player-related input and output.

Functions and tools for outputting formatted data to console or a file, loading assets, files, images, sound and misc.

#### Author

Lucas Vieira

Definition in file [io.hpp](#).

### 9.11.2 Enumeration Type Documentation

#### 9.11.2.1 ofLogLevel

```
enum oficina::ofLogLevel
```

Represents levels of logging to the log output.

**Enumerator**

ofLogCrit	"Critical" logging level.
ofLogErr	"Error" logging level.
ofLogWarn	"Warning" logging level.
ofLogInfo	"Info" logging level.
ofLogNone	Unspecified logging level.

Definition at line 94 of file [io.hpp](#).

**9.11.2.2 ofLogType**

```
enum oficina::ofLogType
```

Represents types of log output.

**Enumerator**

ofLogDisabled	Disabled logging.
ofLogConsole	Console-based logging.
ofLogFile	Text file based logging.

Definition at line 108 of file [io.hpp](#).

**9.11.3 Function Documentation****9.11.3.1 ofFindAsset()**

```
void oficina::ofFindAsset (
 std::string & filename)
```

Finds a specific file on the folders it should be, then fixes its location.

**Parameters**

<i>filename</i>	Direct reference to the asset to be found.
-----------------	--------------------------------------------

**9.11.3.2 ofLoadImage()**

```
SDL_Surface* oficina::ofLoadImage (
 std::string filename)
```

Loads a surface containing a image from the filesystem.

#### Parameters

<i>filename</i>	Path to the image to be loaded.
-----------------	---------------------------------

#### Returns

An SDL\_Surface pointer containing all of the image data.

#### 9.11.3.3 ofLoadText()

```
std::string oficina::ofLoadText (
 std::string filename)
```

Load a text file from the filesystem.

#### Parameters

<i>filename</i>	Path to the file to be loaded.
-----------------	--------------------------------

#### Returns

A string containing all of the text file.

#### 9.11.3.4 ofLog()

```
int oficina::ofLog (
 ofLogLvl level,
 const char * fmt,
 ...)
```

Logs text to the currently selected log type.

#### Parameters

<i>level</i>	Logging level of the message.
<i>fmt</i>	Text format of the information to be output to the log, as per printf logic.
...	Arguments to be fed and used by the function's format.

#### Returns

A failure or success code, much like the function printf.



#### 9.11.3.5 ofLogGetType()

```
ofLogType oficina::ofLogGetType ()
```

Yields the currently used logging type.

##### Returns

Type of the current log tool.

#### 9.11.3.6 ofLogSetLevel()

```
void oficina::ofLogSetLevel (
 ofLogLvl level)
```

Defines the minimum log priority level of the log function. Any level below the specified priority will not be output to the log.

Defaults to ofLogNone.

##### Parameters

<i>level</i>	Minimum log priority to be tolerated.
--------------	---------------------------------------

#### 9.11.3.7 ofLogUseFile()

```
void oficina::ofLogUseFile (
 std::string filename)
```

Use a text file as logging tool.

##### Parameters

<i>filename</i>	Path of the file to be used as log.
-----------------	-------------------------------------

##### Warning

If the file already exists, the output will be appended to its end.

#### 9.11.3.8 ofSetDataDirectoryName()

```
void oficina::ofSetDataDirectoryName (
 std::string dirname)
```

Sets the name of the folder used to enclose the game's assets (outside of project/application run folder).

## Warning

If you don't ever call this function nor use the GameArgs to specify the data folder, the game will assume your application should always look for data on the current execution directory.

## Parameters

<i>dirname</i>	Name of the Data folder on disk.
----------------	----------------------------------

## 9.12 io.hpp

```

00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00026 #pragma once
00027
00028 #include <cstdint>
00029 #include <string>
00030 #include "oficina2/platform.hpp"
00031 #include <SDL2/SDL.h>
00032
00033 #if OF_PLATFORM == OF_PLATFORM_WINDOWS
00034 #define OFLOG_NRM ""
00037 #define OFLOG_RED ""
00040 #define OFLOG_GRN ""
00043 #define OFLOG_YEL ""
00046 #define OFLOG_BLU ""
00049 #define OFLOG_MAG ""
00052 #define OFLOG_CYN ""
00055 #define OFLOG_WHT ""
00058 #define OFLOG_RESET ""
00061 #else
00062 #define OFLOG_NRM "\x1B[0m"
00065 #define OFLOG_RED "\x1B[31m"
00068 #define OFLOG_GRN "\x1B[32m"
00071 #define OFLOG_YEL "\x1B[33m"
00074 #define OFLOG_BLU "\x1B[34m"
00077 #define OFLOG_MAG "\x1B[35m"
00080 #define OFLOG_CYN "\x1B[36m"
00083 #define OFLOG_WHT "\x1B[37m"
00086 #define OFLOG_RESET "\033[0m"
00089 #endif
00090
00091 namespace oficina
00092 {
00094 enum ofLogLvl {
00096 ofLogCrit = 0,
00098 ofLogErr = 1,
00100 ofLogWarn = 2,
00102 ofLogInfo = 3,
00104 ofLogNone = 4
00105 };
00106
00108 enum ofLogType {
00110 ofLogDisabled = 0,
00112 ofLogConsole = 1,
00114 ofLogFile = 2
00115 };
00116
00117 int ofLog(ofLogLvl level, const char* fmt, ...);
00132 void ofLogSetLevel(ofLogLvl level);
00135 ofLogType ofLogGetType();
00140 void ofLogUseFile(std::string filename);

```

```

00144 void ofLogUseConsole();
00146 void ofLogDisable();
00147
00151 void ofFindAsset(std::string& filename);
00159 void ofSetDataDirectoryName(std::string dirname);
00163 std::string ofLoadText(std::string filename);
00168 SDL_Surface* ofLoadImage(std::string filename);
00169 }

```

## 9.13 oficina.hpp File Reference

Default tools for easily initializing Oficina.

```

#include "oficina2/display.hpp"
#include "oficina2/io.hpp"
#include "oficina2/input.hpp"
#include "oficina2/render.hpp"
#include "oficina2/canvas.hpp"
#include "oficina2/timer.hpp"
#include "oficina2/ofscheme.hpp"
#include "oficina2/oflua.hpp"
#include "oficina2/entity.hpp"

```

### Macros

- `#define OF_VERSION_STRING "2.0.11"`  
String banner containing the current version of OficinaFramework.

### Enumerations

- enum `oficina::ofReplType` { `oficina::ofReplNone`, `oficina::ofReplScheme`, `oficina::ofReplLua` }  
Specifies the types of REPL.

### Functions

- void `oficina::ofInit` (std::vector< std::string > args)  
Initializes OficinaFramework.
- void `oficina::ofInit` ()  
Initializes OficinaFramework.
- void `oficina::ofGameLoop` ()  
Executes the Game Loop, once the default subsystems are initialized. Finishes when the Soft Stop flag is raised.
- void `oficina::ofSoftStop` ()  
Raises a Soft Stop flag, which will quit the default Game Loop function.
- void `oficina::ofQuit` ()  
De-inits and unloads all subsystems and default display and context initialized by the default initialization function.
- void `oficina::ofSetWindowSize` (ofdword x, ofdword y)  
Sets a new size for the default window.
- bool `oficina::ofQuitFlagRaised` ()  
Yields the state of the Soft Stop flag.
- glm::uvec2 `oficina::ofGetWindowSize` ()

- Yields the size of the window.*
- void `oficina::ofSetFullscreen` (bool state)  
*Changes the application's window state.*
- bool `oficina::ofIsFullscreen` ()  
*Checks for the fullscreen state of the application.*
- void `oficina::ofSetReplType` (ofReplType type)  
*Sets the current type of REPL to be used.*
- ofReplType `oficina::ofGetReplType` ()  
*Gets the current type of REPL.*
- void `oficina::ofReplEval` (std::string str, bool suppressAns=false)  
*Evaluates a string on the REPL, regardless of language.*
- void `oficina::ofReplDumpOutput` ()  
*Dumps REPL output for current REPL type.*
- void `oficina::ofSetSwapInterval` (ofFrameRateConfig cfg, float max=0.0f, float min=0.0f)  
*Changes the way that the global display deals with swap interval.*
- void `oficina::ofSetClearColor` (glm::vec4 color)  
*Sets the background color for the global renderer.*

### 9.13.1 Detailed Description

Default tools for easily initializing Oficina.

Functions and tools for starting and finishing Oficina in its entirety, for a quick and easy game development.

#### Author

Lucas Vieira

Definition in file [oficina.hpp](#).

### 9.13.2 Enumeration Type Documentation

#### 9.13.2.1 ofReplType

enum `oficina::ofReplType`

Specifies the types of REPL.

#### Enumerator

<code>ofReplNone</code>	No REPL language at all.
<code>ofReplScheme</code>	Use IronScheme on REPL.
<code>ofReplLua</code>	Use IronLua on REPL.

Definition at line 107 of file [oficina.hpp](#).

### 9.13.3 Function Documentation

#### 9.13.3.1 ofGameLoop()

```
void oficina::ofGameLoop ()
```

Executes the Game Loop, once the default subsystems are initialized. Finishes when the Soft Stop flag is raised.

##### See also

ofInit  
ofSoftStop

#### 9.13.3.2 ofGetReplType()

```
ofReplType oficina::ofGetReplType ()
```

Gets the current type of REPL.

##### Returns

The currently used REPL language.

#### 9.13.3.3 ofGetWindowSize()

```
glm::uvec2 oficina::ofGetWindowSize ()
```

Yields the size of the window.

##### Note

You should understand "window" as both the display's size and context's viewport. The viewport will always be scaled to fit the display. To maintain the internal resolution, one should handle its own Projection matrix.

##### Returns

A 2D vector containing the window size, in unsigned integer values.

#### 9.13.3.4 ofInit() [1/2]

```
void oficina::ofInit (
 std::vector< std::string > args)
```

Initializes OficinaFramework.

This will automatically initialize a new display and context for your game, and also all necessary subsystems such as canvas manager, debugger, global Scheme interpreter (for Repl), etc.

**Parameters**

<i>args</i>	Vector of arguments to be passed for initialization.
-------------	------------------------------------------------------

**9.13.3.5 ofInit()** [2/2]

```
void oficina::ofInit ()
```

Initializes OficinaFramework.

This will automatically initialize a new display and context for your game, and also all necessary subsystems such as canvas manager, debugger, global Scheme interpreter (for Repl), etc.

**9.13.3.6 ofIsFullscreen()**

```
bool oficina::ofIsFullscreen ()
```

Checks for the fullscreen state of the application.

**Returns**

Whether the application is fullscreen or not.

**9.13.3.7 ofQuit()**

```
void oficina::ofQuit ()
```

De-inits and unloads all subsystems and default display and context initialized by the default initialization function.

**See also**

ofInit  
ofGameLoop  
ofSoftStop

**9.13.3.8 ofQuitFlagRaised()**

```
bool oficina::ofQuitFlagRaised ()
```

Yields the state of the Soft Stop flag.

**Returns**

Whether the Soft Stop flag was raised or not.

**9.13.3.9 ofReplEval()**

```
void oficina::ofReplEval (
 std::string str,
 bool suppressAns = false)
```

Evaluates a string on the REPL, regardless of language.

## Parameters

<i>str</i>	String to be evaluated.
<i>suppressAns</i>	Whether the answer should be suppressed from REPL output.

## 9.13.3.10 ofSetClearColor()

```
void oficina::ofSetClearColor (
 glm::vec4 color)
```

Sets the background color for the global renderer.

## Parameters

<i>color</i>	Four-dimensional vector containing RGBA colors, normalized.
--------------	-------------------------------------------------------------

## 9.13.3.11 ofSetFullscreen()

```
void oficina::ofSetFullscreen (
 bool state)
```

Changes the application's window state.

## Parameters

<i>state</i>	State to be assumed: Fullscreen (true) or Windowed (false).
--------------	-------------------------------------------------------------

## 9.13.3.12 ofSetReplType()

```
void oficina::ofSetReplType (
 ofReplType type)
```

Sets the current type of REPL to be used.

## Parameters

<i>type</i>	Type of REPL to be used from now on.
-------------	--------------------------------------

### 9.13.3.13 ofSetSwapInterval()

```
void oficina::ofSetSwapInterval (
 ofFrameRateConfig cfg,
 float max = 0.0f,
 float min = 0.0f)
```

Changes the way that the global display deals with swap interval.

#### Parameters

<i>cfg</i>	Desired swap interval configuration.
<i>max</i>	Maximum swap interval, in frames per second (FPS). If applicable.
<i>min</i>	Minimum swap interval, in frames per second (FPS). If applicable.

### 9.13.3.14 ofSetWindowSize()

```
void oficina::ofSetWindowSize (
 ofdword x,
 ofdword y)
```

Sets a new size for the default window.

#### Note

You should understand "window" as both the display's size and context's viewport. The viewport will always be scaled to fit the display. To maintain the internal resolution, one should handle its own Projection matrix.

#### Parameters

<i>x</i>	Width of the window, in pixels.
<i>y</i>	Height of the window, in pixels.

### 9.13.3.15 ofSoftStop()

```
void oficina::ofSoftStop ()
```

Raises a Soft Stop flag, which will quit the default Game Loop function.

#### See also

[ofGameLoop](#)



## 9.14 oficina.hpp

```

00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00025 #pragma once
00026
00027 #include "oficina2/display.hpp"
00028 #include "oficina2/io.hpp"
00029 #include "oficina2/input.hpp"
00030 #include "oficina2/render.hpp"
00031 #include "oficina2/canvas.hpp"
00032 #include "oficina2/timer.hpp"
00033 #include "oficina2/ofscheme.hpp"
00034 #include "oficina2/oflua.hpp"
00035 #include "oficina2/entity.hpp"
00036
00039 #define OF_VERSION_STRING "2.0.11"
00040
00041 namespace oficina
00042 {
00050 void ofInit(std::vector<std::string> args);
00057 void ofInit();
00063 void ofGameLoop();
00067 void ofSoftStop();
00074 void ofQuit();
00075
00084 void ofSetWindowSize(ofdword x, ofdword y);
00085
00088 bool ofQuitFlagRaised();
00097 glm::uvec2 ofGetWindowSize();
00098
00101 void ofSetFullscreen(bool state);
00104 bool ofIsFullscreen();
00105
00107 enum ofReplType
00108 {
00110 ofReplNone,
00112 ofReplScheme,
00114 ofReplLua
00115 };
00116
00119 void ofSetReplType(ofReplType type);
00122 ofReplType ofGetReplType();
00126 void ofReplEval(std::string str, bool suppressAns = false);
00128 void ofReplDumpOutput();
00133 void ofSetSwapInterval(ofFrameRateConfig cfg, float max = 0.0f,
float min = 0.0f);
00137 void ofSetClearColor(glm::vec4 color);
00138 }

```

## 9.15 oflua.hpp File Reference

Tools for object scripting and for the Repl, in Lua language.

```

#include <lua.hpp>
#include "oficina2/entity.hpp"
#include <string>

```

### Classes

- class `oficina::ofLua`  
Defines one Lua environment to be used inside an entity.

## Functions

- void `oficina::ofLuaInit ()`  
*Initializes internal Lua Repl.*
- void `oficina::ofLuaDeinit ()`  
*Stops internal Lua Repl.*
- bool `oficina::ofLuaIsInit ()`  
*Yields the state of the Lua Repl.*
- void `oficina::ofLuaEval (std::string strToEval, bool suppressAns=true)`  
*Asks the Repl to evaluate a certain string.*
- void `oficina::ofLuaDefineSymbol (std::string symbol, std::string value)`  
*Defines a symbol and binds it to a value for the Repl.*
- void `oficina::ofLuaDefineSymbol (std::string symbol, double value)`  
*Defines a symbol and binds it to a value for the Repl.*
- void `oficina::ofLuaDefineSymbol (std::string symbol, int value)`  
*Defines a symbol and binds it to a value for the Repl.*
- void `oficina::ofLuaDefineSymbol (std::string symbol, bool value)`  
*Defines a symbol and binds it to a value for the Repl.*
- void `oficina::ofLuaDefineSymbol (std::string symbol, oficina::ofEntity *value)`  
*Defines a symbol and binds it to a value for the Repl.*
- void `oficina::ofLuaDefineSymbol (std::string symbol, oficina::ofComponent *value)`  
*Defines a symbol and binds it to a value for the Repl.*
- void `oficina::ofLuaDefineFunc (std::string symbol, lua_CFunction fun)`  
*Defines a foreign function for the Repl.*
- void `oficina::ofLuaUndefine (std::string symbol)`  
*Undefines a symbol (foreign function/variable) for the Repl.*
- std::string `oficina::ofLuaGetString (std::string symbol)`  
*Retrieves the value of a symbol which holds a string type.*
- double `oficina::ofLuaGetNumber (std::string symbol)`  
*Retrieves the value of a symbol which holds a numeric type.*
- int `oficina::ofLuaGetInteger (std::string symbol)`  
*Retrieves the value of a symbol which holds an integer type.*
- bool `oficina::ofLuaGetBoolean (std::string symbol)`  
*Retrieves the value of a symbol which holds a boolean type.*
- `oficina::ofEntity *` `oficina::ofLuaGetEntity (std::string symbol)`  
*Retrieves the value of a symbol which holds a reference to an entity.*
- `oficina::ofComponent *` `oficina::ofLuaGetComponent (std::string symbol)`  
*Retrieves the value of a symbol which holds a reference to a component.*

### 9.15.1 Detailed Description

Tools for object scripting and for the Repl, in Lua language.

Provides classes and functions for managing the internal Repl, and for executing scripting behavior for entities, on IronLua language, with default OficinaFramework bindings.

#### Author

Lucas Vieira

Definition in file [oflua.hpp](#).

## 9.15.2 Function Documentation

### 9.15.2.1 ofLuaDefineFunc()

```
void oficina::ofLuaDefineFunc (
 std::string symbol,
 lua_CFunction fun)
```

Defines a foreign function for the Repl.

You should use this particularly if there is a specific function you wish to access using the Repl.

#### Parameters

<i>symbol</i>	Name of the function to be defined.
<i>fun</i>	Function to be used. Notice that this function should follow the specifications of the Lua API: It needs to return an integer, containing the number of return values in the function, and accept a <code>lua_State*</code> as parameter, representing the function stack. For more info, consult the Lua 5.3 Reference.

### 9.15.2.2 ofLuaDefineSymbol() [1/6]

```
void oficina::ofLuaDefineSymbol (
 std::string symbol,
 std::string value)
```

Defines a symbol and binds it to a value for the Repl.

You should use this particularly if there is a specific value you wish to access using the Repl.

#### Parameters

<i>symbol</i>	Name of the symbol to be defined.
<i>value</i>	String value to be bound to the symbol.

### 9.15.2.3 ofLuaDefineSymbol() [2/6]

```
void oficina::ofLuaDefineSymbol (
 std::string symbol,
 double value)
```

Defines a symbol and binds it to a value for the Repl.

You should use this particularly if there is a specific value you wish to access using the Repl.

## Parameters

<i>symbol</i>	Name of the symbol to be defined.
<i>value</i>	Double value to be bound to the symbol.

## 9.15.2.4 ofLuaDefineSymbol() [3/6]

```
void oficina::ofLuaDefineSymbol (
 std::string symbol,
 int value)
```

Defines a symbol and binds it to a value for the Repl.

You should use this particularly if there is a specific value you wish to access using the Repl.

## Parameters

<i>symbol</i>	Name of the symbol to be defined.
<i>value</i>	Integer value to be bound to the symbol.

## 9.15.2.5 ofLuaDefineSymbol() [4/6]

```
void oficina::ofLuaDefineSymbol (
 std::string symbol,
 bool value)
```

Defines a symbol and binds it to a value for the Repl.

You should use this particularly if there is a specific value you wish to access using the Repl.

## Parameters

<i>symbol</i>	Name of the symbol to be defined.
<i>value</i>	Boolean value to be bound to the symbol.

## 9.15.2.6 ofLuaDefineSymbol() [5/6]

```
void oficina::ofLuaDefineSymbol (
 std::string symbol,
 oficina::ofEntity * value)
```

Defines a symbol and binds it to a value for the Repl.

You should use this particularly if there is a specific value you wish to access using the Repl.

## Parameters

<i>symbol</i>	Name of the symbol to be defined.
<i>value</i>	Entity reference to be bound to the symbol.

## 9.15.2.7 ofLuaDefineSymbol() [6/6]

```
void oficina::ofLuaDefineSymbol (
 std::string symbol,
 oficina::ofIComponent * value)
```

Defines a symbol and binds it to a value for the Repl.

You should use this particularly if there is a specific value you wish to access using the Repl.

## Parameters

<i>symbol</i>	Name of the symbol to be defined.
<i>value</i>	Component reference to be bound to the symbol.

## 9.15.2.8 ofLuaEval()

```
void oficina::ofLuaEval (
 std::string strToEval,
 bool suppressAns = true)
```

Asks the Repl to evaluate a certain string.

## Parameters

<i>strToEval</i>	String to be evaluated, in Lua language.
<i>suppressAns</i>	Suppresses answer to REPL.

## 9.15.2.9 ofLuaGetBoolean()

```
bool oficina::ofLuaGetBoolean (
 std::string symbol)
```

Retrieves the value of a symbol which holds a boolean type.

## Warning

Make sure that the symbol is actually bound to the type when calling this function.

**Parameters**

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

**Returns**

The value held by said symbol, if compatible.

**9.15.2.10 ofLuaGetComponent()**

```
oficina::ofIComponent* oficina::ofLuaGetComponent (
 std::string symbol)
```

Retrieves the value of a symbol which holds a reference to a component.

**Warning**

Make sure that the symbol is actually bound to the type when calling this function.

**Parameters**

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

**Returns**

A pointer to a component held by said symbol, if compatible.

**9.15.2.11 ofLuaGetEntity()**

```
oficina::ofEntity* oficina::ofLuaGetEntity (
 std::string symbol)
```

Retrieves the value of a symbol which holds a reference to an entity.

**Warning**

Make sure that the symbol is actually bound to the type when calling this function.

**Parameters**

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

**Returns**

A pointer to an entity held by said symbol, if compatible.

**9.15.2.12 ofLuaGetInteger()**

```
int oficina::ofLuaGetInteger (
 std::string symbol)
```

Retrieves the value of a symbol which holds an integer type.

**Warning**

Make sure that the symbol is actually bound to the type when calling this function.

**Parameters**

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

**Returns**

The value held by said symbol, if compatible.

**9.15.2.13 ofLuaGetNumber()**

```
double oficina::ofLuaGetNumber (
 std::string symbol)
```

Retrieves the value of a symbol which holds a numeric type.

**Warning**

Make sure that the symbol is actually bound to the type when calling this function.

**Parameters**

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

**Returns**

The value held by said symbol, if compatible.

#### 9.15.2.14 ofLuaGetString()

```
std::string oficina::ofLuaGetString (
 std::string symbol)
```

Retrieves the value of a symbol which holds a string type.

##### Warning

Make sure that the symbol is actually bound to the type when calling this function.

##### Parameters

<i>symbol</i>	Symbol which value should be retrieved.
---------------	-----------------------------------------

##### Returns

The value held by said symbol, if compatible.

#### 9.15.2.15 ofLuaIsInit()

```
bool oficina::ofLuaIsInit ()
```

Yields the state of the Lua Repl.

##### Returns

Whether the Repl is initialized or not.

#### 9.15.2.16 ofLuaUndefine()

```
void oficina::ofLuaUndefine (
 std::string symbol)
```

Undefines a symbol (foreign function/variable) for the Repl.

Takes a previously defined symbol and binds it to Lua's nil, effectively removing its definition, if existing. This will not make the symbol cease to exist – although Lua's symbols do not technically cease to exist; the actual API recommendation is to bind it to nil –, but will remove its bound behaviour or value.

##### Parameters

<i>symbol</i>	Name of the symbol to be unbound.
---------------	-----------------------------------



## 9.16 oflua.hpp

```

00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00026 #pragma once
00027
00028 #ifndef NO_LUA
00029
00030 #include <lua.hpp>
00031 #include "oficina2/entity.hpp"
00032 #include <string>
00033
00034 namespace oficina
00035 {
00037 void ofLuaInit();
00039 void ofLuaDeinit();
00042 bool ofLuaIsInit();
00046 void ofLuaEval(std::string strToEval, bool suppressAns = true);
00047 //void ofLuaDumpOutput(); // TODO -- needed?
00054 void ofLuaDefineSymbol(std::string symbol, std::string value);
00061 void ofLuaDefineSymbol(std::string symbol, double value);
00068 void ofLuaDefineSymbol(std::string symbol, int value);
00075 void ofLuaDefineSymbol(std::string symbol, bool value);
00082 void ofLuaDefineSymbol(std::string symbol,
oficina::ofEntity* value);
00089 void ofLuaDefineSymbol(std::string symbol,
oficina::ofIComponent* value);
00100 void ofLuaDefineFunc(std::string symbol, lua_CFunction fun);
00109 void ofLuaUndefine(std::string symbol);
00115 std::string ofLuaGetString(std::string symbol);
00121 double ofLuaGetNumber(std::string symbol);
00127 int ofLuaGetInteger(std::string symbol);
00133 bool ofLuaGetBoolean(std::string symbol);
00140 oficina::ofEntity* ofLuaGetEntity(std::string symbol);
00147 oficina::ofIComponent* ofLuaGetComponent(std::string symbol);
00148
00150 class ofLua : public ofIComponent
00151 {
00152 public:
00154 void init();
00160 void loadfile(std::string filename);
00162 void reload();
00164 void unload();
00169 void update(float dt);
00175 void regSym(std::string symbol, std::string value);
00181 void regSym(std::string symbol, double value);
00187 void regSym(std::string symbol, int value);
00193 void regSym(std::string symbol, bool value);
00200 void regSym(std::string symbol,
oficina::ofEntity* value);
00207 void regSym(std::string symbol,
oficina::ofIComponent* value);
00217 void regFunc(std::string symbol, lua_CFunction fun);
00222 std::string getString(std::string symbol);
00227 double getNumber(std::string symbol);
00232 int getInteger(std::string symbol);
00237 bool getBoolean(std::string symbol);
00242 oficina::ofEntity* getEntity(std::string symbol);
00247 oficina::ofIComponent* GetComponent(std::string symbol);
00251 bool isInit() const;
00252 private:
00253 lua_State* L = nullptr;
00254 bool m_initialized = false;
00255 bool m_updt_error = false;
00256 std::string m_filename;
00257 };
00258
00259 }
00260 }
00261 #endif // NO_LUA

```

## 9.17 ofscheme.hpp File Reference

Tools for object scripting and for the Repl, in Scheme language.

```
#include <libguile.h>
#include <string>
#include <functional>
#include "oficina2/entity.hpp"
```

### Classes

- class [oficina::ofScheme](#)  
*Defines one Scheme environment to be used inside an entity.*

### Macros

- `#define` [SCHEME\\_FUNCAST](#)(x) (scm\_t\_subr)&(x)  
*Macro for defining Scheme functions. Pass the function name using this macro when creating functions for both ofScmDefineFunc and ofScheme::regFunc.*

### Functions

- void [oficina::ofScmInit](#) ()  
*Initializes internal Scheme Repl.*
- void [oficina::ofScmDeinit](#) ()  
*Stops internal Scheme Repl.*
- bool [oficina::ofScmIsInit](#) ()  
*Yields the state of the Scheme Repl.*
- void [oficina::ofScmEval](#) (std::string strToEval, bool suppressAns=false)  
*Asks the Repl to evaluate a certain string.*
- void [oficina::ofScmDumpOutput](#) ()  
*Dumps REPL output awaiting the dump call on the output port.*
- void [oficina::ofScmDefineSymbol](#) (std::string symbol, SCM value)  
*Defines a symbol for the Repl.*
- void [oficina::ofScmDefineSymbol](#) (std::string symbol, [oficina::ofEntity](#) \*value)  
*Defines a symbol, containing a reference to an entity, for the Repl.*
- void [oficina::ofScmDefineSymbol](#) (std::string symbol, [oficina::ofIComponent](#) \*value)  
*Defines a symbol, containing a reference to a component, for the Repl.*
- void [oficina::ofScmDefineFunc](#) (std::string symbol, int n\_params, scm\_t\_subr fun, int n\_optional\_params=0)  
*Defines a foreign function for the Repl.*
- void [oficina::ofScmUndefine](#) (std::string symbol)  
*Undefines a symbol (foreign function/variable) for the Repl.*
- SCM [oficina::ofScmGetReference](#) (std::string symbol)  
*Retrieves a reference to a symbol globally defined on the REPL's default module.*

### 9.17.1 Detailed Description

Tools for object scripting and for the Repl, in Scheme language.

Provides classes and functions for managing the internal Repl, and for executing scripting behavior for entities, on IronScheme language, with default OficinaFramework bindings.

#### Author

Lucas Vieira

Definition in file [ofscheme.hpp](#).

### 9.17.2 Function Documentation

#### 9.17.2.1 ofScmDefineFunc()

```
void oficina::ofScmDefineFunc (
 std::string symbol,
 int n_params,
 scm_t_subr fun,
 int n_optional_params = 0)
```

Defines a foreign function for the Repl.

You should use this particularly if there is a specific function you wish to access using the Repl.

#### Parameters

<i>symbol</i>	Name of the function to be defined.
<i>n_params</i>	Number of required/obligatory parameters to be passed to the function.
<i>fun</i>	Function pointer to be used. Pass the function name with the SCHEME_FUNCAST macro to cast it appropriately.

#### See also

[SCHEME\\_FUNCAST](#)

#### Parameters

<i>n_optional_params</i>	Optionally specify the amount of optional parameters which the function should have. Optional parameters should begin right after obligatory parameters.
--------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------

### 9.17.2.2 ofScmDefineSymbol() [1/3]

```
void oficina::ofScmDefineSymbol (
 std::string symbol,
 SCM value)
```

Defines a symbol for the Repl.

You should use this particularly if there is a specific value you wish to access using the Repl.

#### Parameters

<i>symbol</i>	Name of the symbol to be defined.
<i>value</i>	Value to be bound to the symbol.

### 9.17.2.3 ofScmDefineSymbol() [2/3]

```
void oficina::ofScmDefineSymbol (
 std::string symbol,
 oficina::ofEntity * value)
```

Defines a symbol, containing a reference to an entity, for the Repl.

You should use this particularly if there is an entity you wish to access using the Repl.

#### Parameters

<i>symbol</i>	Name of the symbol to be defined.
<i>value</i>	Entity reference to be bound to the symbol.

### 9.17.2.4 ofScmDefineSymbol() [3/3]

```
void oficina::ofScmDefineSymbol (
 std::string symbol,
 oficina::ofIComponent * value)
```

Defines a symbol, containing a reference to a component, for the Repl.

You should use this particularly if there is a component you wish to access using the Repl.

#### Parameters

<i>symbol</i>	Name of the symbol to be defined.
<i>value</i>	Component reference to be bound to the symbol.

### 9.17.2.5 ofScmEval()

```
void oficina::ofScmEval (
 std::string strToEval,
 bool suppressAns = false)
```

Asks the Repl to evaluate a certain string.

#### Parameters

<i>strToEval</i>	String to be evaluated, in Scheme language.
<i>suppressAns</i>	Suppresses answer to REPL.

### 9.17.2.6 ofScmGetReference()

```
SCM oficina::ofScmGetReference (
 std::string symbol)
```

Retrieves a reference to a symbol globally defined on the REPL's default module.

#### Parameters

<i>symbol</i>	Symbol which reference should be retrieved.
---------------	---------------------------------------------

#### Returns

A direct reference to the value or the procedure defined under this symbol.

### 9.17.2.7 ofScmIsInit()

```
bool oficina::ofScmIsInit ()
```

Yields the state of the Scheme Repl.

#### Returns

Whether the Repl is initialized or not.

#### 9.17.2.8 ofScmUndefine()

```
void oficina::ofScmUndefine (
 std::string symbol)
```

Undefines a symbol (foreign function/variable) for the Repl.

Takes a previously defined symbol and binds it to the Scheme's nil, effectively removing its definition, if existing. This will not make the symbol cease to exist, but will remove its bound behaviour or value.

## Parameters

<i>symbol</i>	Name of the symbol to be unbound.
---------------	-----------------------------------

## 9.18 ofscheme.hpp

```

00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00026 #pragma once
00027
00028 #ifndef NO_SCHEME
00029
00030 #include <libguile.h>
00031 #include <string>
00032 #include <functional>
00033 #include "oficina2/entity.hpp"
00034
00039 #define SCHEME_FUNCAST(x) (scm_t_subr)&(x)
00040
00041 namespace oficina
00042 {
00043 void ofScmInit();
00044 void ofScmDeinit();
00045 bool ofScmIsInit();
00046 void ofScmEval(std::string strToEval, bool suppressAns = false);
00047 void ofScmDumpOutput();
00048 void ofScmDefineSymbol(std::string symbol, SCM value);
00049 void ofScmDefineSymbol(std::string symbol, oficina::ofEntity* value);
00050 void ofScmDefineSymbol(std::string symbol, oficina::ofIComponent* value);
00051 void ofScmDefineFunc(std::string symbol, int n_params, scm_t_subr fun, int n_optional_params = 0);
00052 void ofScmUndefine(std::string symbol);
00053 SCM ofScmGetReference(std::string symbol);
00054
00055 class ofScheme : public ofIComponent
00056 {
00057 public:
00058 void init();
00059 void loadfile(std::string module, std::string filename);
00060 void reload();
00061 void unload();
00062 void update(float dt);
00063 void regSym(std::string symbol, SCM value);
00064 void regFunc(std::string symbol, int n_params, scm_t_subr fun, int n_optional_params = 0);
00065 SCM getSymRef(std::string symbol);
00066 bool isInit() const;
00067 private:
00068 bool m_initialized = false;
00069 bool m_loaded = false;
00070 std::string m_modulename;
00071 std::string m_filename;
00072 bool m_updt_error = false;
00073
00074 SCM obj_module = SCM_UNDEFINED,
00075 init_func = SCM_UNDEFINED,
00076 update_func = SCM_UNDEFINED;
00077 };
00078 }
00079
00080 #endif // NO_SCHEME

```

## 9.19 platform.hpp File Reference

Definitions for the platform currently executing the game.

## Macros

- `#define OF_PLATFORM_UNKNOWN 0x000u`  
*Unknown platform.*
- `#define OF_PLATFORM_WINDOWS 0x001u`  
*Windows platform.*
- `#define OF_PLATFORM_LINUX 0x002u`  
*Linux platform.*
- `#define OF_PLATFORM_MACOSX 0x004u`  
*OS X platform.*
- `#define OF_PLATFORM_ANDROID 0x008u`  
*Android platform.*
- `#define OF_PLATFORM_IOS 0x010u`  
*iOS platform.*
- `#define OF_PLATFORM_IOS_SIMULATOR 0x020u`  
*iOS platform (simulator).*
- `#define OF_ARCH_UNKNOWN 0x000u`  
*Unknown processor architecture.*
- `#define OF_ARCH_32BIT 0x002u`  
*32-bit (i386) processor architecture.*
- `#define OF_ARCH_64BIT 0x004u`  
*64-bit (x86\_64) processor architecture.*
- `#define OF_ARCH_ARM 0x008u`  
*ARM processor architecture.*
- `#define OF_ARCH_ARMV7 0x010u`  
*ARMv7 processor architecture.*
- `#define OF_ARCH_ARM64 0x020u`  
*ARM64 processor architecture.*
- `#define OF_SCRIPTING_NONE 0x00u`  
*No scripting language.*
- `#define OF_SCRIPTING_SCHEME 0x01u`  
*Scheme scripting language.*
- `#define OF_SCRIPTING_LUA 0x02u`  
*Lua scripting language.*

### 9.19.1 Detailed Description

Definitions for the platform currently executing the game.

These definitions are given and associated during compile time. You can check the preprocessors `OF_PLATFORM` and `OF_ARCH` for system's platform and architecture.

Other interesting preprocessors are `OF_DESKTOP` and `OF_MOBILE`, which are simply defined for easier use, and therefore are not documented in this file.

#### Author

Lucas Vieira

Definition in file [platform.hpp](#).



## 9.20 platform.hpp

```

00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00029 #pragma once
00030
00032 #define OF_PLATFORM_UNKNOWN 0x000u
00033 #define OF_PLATFORM_WINDOWS 0x001u
00035 #define OF_PLATFORM_LINUX 0x002u
00037 #define OF_PLATFORM_MACOSX 0x004u
00039 #define OF_PLATFORM_ANDROID 0x008u
00041 #define OF_PLATFORM_IOS 0x010u
00043 #define OF_PLATFORM_IOS_SIMULATOR 0x020u
00045
00047 #define OF_ARCH_UNKNOWN 0x000u
00048 #define OF_ARCH_32BIT 0x002u
00050 #define OF_ARCH_64BIT 0x004u
00052 #define OF_ARCH_ARM 0x008u
00054 #define OF_ARCH_ARMV7 0x010u
00056 #define OF_ARCH_ARM64 0x020u
00058
00059 #ifdef _WIN64
00060 #define OF_PLATFORM OF_PLATFORM_WINDOWS
00061 #define OF_ARCH OF_ARCH_64BIT
00062 #define OF_DESKTOP
00063 #elif _WIN32
00064 #define OF_PLATFORM OF_PLATFORM_WINDOWS
00065 #define OF_ARCH OF_ARCH_32BIT
00066 #define OF_DESKTOP
00067 #elif __APPLE__
00068 #if TARGET_OS_IPHONE && TARGET_IPHONE_SIMULATOR
00069 #define OF_PLATFORM (OF_PLATFORM_IOS | OF_PLATFORM_IOS_SIMULATOR)
00070 #define OF_MOBILE
00071 #elif TARGET_OS_IPHONE
00072 #define OF_PLATFORM OF_PLATFORM_IOS
00073 #define OF_MOBILE
00074 #elif TARGET_OS_MAC
00075 #define OF_PLATFORM OF_PLATFORM_MACOSX
00076 #define OF_DESKTOP
00077 #endif
00078 #elif ANDROID
00079 #define OF_PLATFORM OF_PLATFORM_ANDROID
00080 #define OF_MOBILE
00081 #elif __linux__
00082 #define OF_PLATFORM OF_PLATFORM_LINUX
00083 #define OF_DESKTOP
00084 #else
00085 #define OF_PLATFORM OF_PLATFORM_UNKNOWN
00086 #define OF_DESKTOP
00087 #endif
00088
00089 // Check architecture. This will mainly serve for GCC and Clang
00090 #ifndef OF_ARCH
00091 #ifdef __x86_64__
00092 #define OF_ARCH OF_ARCH_64BIT
00093 #elif __ARM_ARCH_7__
00094 #define OF_ARCH OF_ARCH_ARMV7
00095 #elif __arm__
00096 #define OF_ARCH OF_ARCH_ARM
00097 #elif __aarch64__
00098 #define OF_ARCH OF_ARCH_ARM64
00099 #elif __i386__
00100 #define OF_ARCH OF_ARCH_32BIT
00101 #else
00102 #define OF_ARCH OF_ARCH_UNKNOWN
00103 #endif
00104 #endif
00105
00106
00107 // Important platform headers that cannot be
00108 // left out
00109 #if OF_PLATFORM == OF_PLATFORM_WINDOWS
00110 #include <Windows.h>

```

```

00111 #elif OF_PLATFORM == OF_PLATFORM_LINUX
00112 #elif OF_PLATFORM == OF_PLATFORM_MACOSX
00113 #endif
00114
00115 // Definitions for scripting support
00117 #define OF_SCRIPTING_NONE 0x00u
00118 #define OF_SCRIPTING_SCHEME 0x01u
00120 #define OF_SCRIPTING_LUA 0x02u
00122
00123 #if defined(NO_SCHEME) && defined(NO_LUA)
00124 #define OF_SCRIPTING_AVAILABLE OF_SCRIPTING_NONE
00125 #elif defined(NO_SCHEME) && !defined(NO_LUA)
00126 #define OF_SCRIPTING_AVAILABLE OF_SCRIPTING_LUA
00127 #elif !defined(NO_SCHEME) && defined(NO_LUA)
00128 #define OF_SCRIPTING_AVAILABLE OF_SCRIPTING_SCHEME
00129 #else
00130 #define OF_SCRIPTING_AVAILABLE (OF_SCRIPTING_SCHEME | OF_SCRIPTING_LUA)
00131 #endif

```

## 9.21 render.hpp File Reference

Tools and classes for rendering inside a context.

```

#include <SDL2/SDL.h>
#include "oficina2/display.hpp"
#include "oficina2/types.hpp"
#include "oficina2/timer.hpp"
#include "oficina2/entity.hpp"
#include <GL/glew.h>
#include <GL/gl.h>
#include <string>
#include <map>

```

### Classes

- class [oficina::ofContext](#)  
*Describes a context for your display.*
- class [oficina::ofBuffer](#)  
*Specifies a generic buffer. Override this class to create your own buffers.*
- class [oficina::ofVertexBuffer](#)  
*Represents a Vertex Buffer object (VBO). Use this to hold data related to drawing.*
- class [oficina::ofElementBuffer](#)  
*Represents an Element Buffer object (EBO), useful for holding sequences of vertices for drawing on screen.*
- class [oficina::ofShader](#)  
*Describes a shader.*
- class [oficina::ofShaderAttribute](#)  
*Represents the location of an attribute for the program shader.*
- class [oficina::ofShaderUniform](#)  
*Represents and handles a shader's uniform.*
- class [oficina::ofShaderProgram](#)  
*Represents a shader program.*
- class [oficina::ofVertexArray](#)  
*Represents a vertex array for binding shader and vertex data.*
- class [oficina::ofTexture](#)  
*Represents a texture on the GPU.*
- class [oficina::ofTexturePool](#)

*Static object for managing textures. Most (if not all) textures should be loaded using this tool.*

- class [ofTextureRenderer](#)

*Tool for easily rendering 2D textures or texture atlases.*

- class [ofFont](#)

*Represents a font.*

- class [ofAnimator](#)

*Tool for controlling a texture renderer to generate animations.*

- struct [ofPrimitive](#)

*A structure representing a primitive. Can be used for rendering.*

- class [ofPrimitiveRenderer](#)

*A static class containing methods for creating and drawing simple primitives onscreen.*

## Enumerations

- enum [ofContextType](#) { [ofContextNone](#), [ofContextGL](#), [ofContextGLES](#) }

*Describes the type of a rendering context.*

- enum [ofBufferUsage](#) {  
[ofBufferStaticDraw](#) = GL\_STATIC\_DRAW, [ofBufferDynamicDraw](#) = GL\_DYNAMIC\_DRAW,  
[ofBufferStreamDraw](#) = GL\_STREAM\_DRAW, [ofBufferStaticRead](#) = GL\_STATIC\_READ,  
[ofBufferDynamicRead](#) = GL\_DYNAMIC\_READ, [ofBufferStreamRead](#) = GL\_STREAM\_READ,  
[ofBufferStaticCopy](#) = GL\_STATIC\_COPY, [ofBufferDynamicCopy](#) = GL\_DYNAMIC\_COPY,  
[ofBufferStreamCopy](#) = GL\_STREAM\_COPY }

*Describes the usage of a created buffer object.*

- enum [ofShaderType](#) {  
[ofShaderVertex](#) = GL\_VERTEX\_SHADER, [ofShaderGeometry](#) = GL\_GEOMETRY\_SHADER, [ofShaderFragment](#) = GL\_FRAGMENT\_SHADER, [ofShaderTessControl](#) = GL\_TESS\_↵  
CONTROL\_SHADER,  
[ofShaderTessEval](#) = GL\_TESS\_EVALUATION\_SHADER, [ofShaderCompute](#) = GL\_COMP\_↵  
UTE\_SHADER }

*Describes the type of a shader.*

- enum [ofPrimitiveType](#) {  
[ofPoints](#) = GL\_POINTS, [ofLineStrip](#) = GL\_LINE\_STRIP, [ofLineLoop](#) = GL\_LINE\_L\_↵  
OOP, [ofLines](#) = GL\_LINES,  
[ofLineStripAdj](#) = GL\_LINE\_STRIP\_ADJACENCY, [ofLinesAdj](#) = GL\_LINES\_ADJACENCY,  
[ofTriangleStrip](#) = GL\_TRIANGLE\_STRIP, [ofTriangleFan](#) = GL\_TRIANGLE\_FAN,  
[ofQuad](#) = GL\_TRIANGLE\_FAN, [ofTriangles](#) = GL\_TRIANGLES, [ofTriangleStripAdj](#) =  
GL\_TRIANGLE\_STRIP\_ADJACENCY, [ofTrianglesAdj](#) = GL\_TRIANGLES\_ADJACENCY,  
[ofPatches](#) = GL\_PATCHES }

*Describes a type for a primitive.*

- enum [ofDataType](#) {  
[ofDataByte](#) = GL\_BYTE, [ofDataUByte](#) = GL\_UNSIGNED\_BYTE, [ofDataShort](#) = G\_↵  
L\_SHORT, [ofDataUShort](#) = GL\_UNSIGNED\_SHORT,  
[ofDataInt](#) = GL\_INT, [ofDataUInt](#) = GL\_UNSIGNED\_INT, [ofDataFloat](#) = GL\_FLOAT,  
[ofDataDouble](#) = GL\_DOUBLE,  
[ofDataFixed](#) = GL\_FIXED }

*Represents the type of certain data fed to a buffer.*

- enum [ofFontFaces](#) { [ofFontFaceFixedsysExcelsior](#), [ofFontFaceGohuFont](#), [ofFontFaceFantasqueSans](#), [ofFontFaceTerminus](#) }

*Enumeration for default font faces.*

## Functions

- ofShader [oficina::ofLoadDefaultFragShader](#) ()  
*Loads the default fragment shader.*
- ofShader [oficina::ofLoadDefaultVertexShader](#) ()  
*Loads the default vertex shader.*
- ofShaderProgram [oficina::ofLoadDefaultShaderProgram](#) ()  
*Loads the default shader program, with default vertex and fragment shaders.*
- void [oficina::ofSetVSync](#) (bool state)  
*Sets whether the game should vertically sync with the screen or not.*

## Variables

- const char [oficina::ofDefaultShaderSrc\\_VS](#) []  
*Default vertex shader source.*
- const char [oficina::ofDefaultShaderSrc\\_FS](#) []  
*Default fragment shader source.*

### 9.21.1 Detailed Description

Tools and classes for rendering inside a context.

#### Author

Lucas Vieira

Definition in file [render.hpp](#).

### 9.21.2 Enumeration Type Documentation

#### 9.21.2.1 ofBufferUsage

enum [oficina::ofBufferUsage](#)

Describes the usage of a created buffer object.

#### See also

[ofBuffer](#)

#### Enumerator

<a href="#">ofBufferStaticDraw</a>	Store buffer data statically for drawing.
<a href="#">ofBufferDynamicDraw</a>	Store buffer dynamically for drawing.
<a href="#">ofBufferStreamDraw</a>	Store buffer as a stream for drawing.
<a href="#">ofBufferStaticRead</a>	Store buffer statically for reading.
<a href="#">ofBufferDynamicRead</a>	Store buffer dynamically for reading.
<a href="#">ofBufferStreamRead</a>	Store buffer as a stream for reading.
<a href="#">ofBufferStaticCopy</a>	Store buffer statically for copying.
<a href="#">ofBufferDynamicCopy</a>	Store buffer dynamically for copying.

Definition at line 50 of file [render.hpp](#).

### 9.21.2.2 ofContextType

enum [oficina::ofContextType](#)

Describes the type of a rendering context.

#### Warning

Currently, only OpenGL is supported.

#### Enumerator

ofContextNone	No rendering context.
ofContextGL	OpenGL rendering context.
ofContextGLES	OpenGL ES rendering context.

Definition at line 38 of file [render.hpp](#).

### 9.21.2.3 ofDataType

enum [oficina::ofDataType](#)

Represents the type of certain data fed to a buffer.

#### Enumerator

ofDataByte	Signed byte (ofsbyte) data type.
ofDataUByte	Unsigned byte (ofbyte) data type.
ofDataShort	Signed short (ofsword) data type.
ofDataUShort	Unsigned short (ofword) data type.
ofDataInt	Signed int (ofsdword) data type.
ofDataUInt	Unsigned int (ofdword) data type.
ofDataFloat	Floating point (float).
ofDataDouble	Double-precision floating point (double).
ofDataFixed	Fixed floating point. Particularly useful for older Android devices with no float support.

Definition at line 123 of file [render.hpp](#).

#### 9.21.2.4 ofFontFaces

```
enum oficina::ofFontFaces
```

Enumeration for default font faces.

##### Enumerator

ofFontFaceFixedsysExcelsior	Fixedsys Excelsior font (8x16px), by Darien Gavin Valentine. License: OFL.
ofFontFaceGohuFont	GohuFont Font (8x13px), by Hugo Chargois. License: WTFPL.
ofFontFaceFantasqueSans	Fantasque Sans Mono Font (7x15px), by Jany Belluz. License: OFL.
ofFontFaceTerminus	Terminus (6x12px). License: OFL.

Definition at line 686 of file [render.hpp](#).

#### 9.21.2.5 ofPrimitiveType

```
enum oficina::ofPrimitiveType
```

Describes a type for a primitive.

##### Enumerator

ofPoints	A set of points.
ofLineStrip	A line strip.
ofLineLoop	A looping line.
ofLines	A set of lines.
ofLineStripAdj	A line strip formed by the lines' adjacency.
ofLinesAdj	A set of lines formed by the lines' adjacency.
ofTriangleStrip	A triangle strip.
ofTriangleFan	A triangle fan.
ofQuad	A quad. In reality, nothing more than a triangle fan.
ofTriangles	A set of triangles.
ofTriangleStripAdj	A triangle strip formed by the triangles' adjacency.
ofTrianglesAdj	A set of triangles formed by the triangles' adjacency.
ofPatches	A set of patches.

Definition at line 92 of file [render.hpp](#).

#### 9.21.2.6 ofShaderType

```
enum oficina::ofShaderType
```

Describes the type of a shader.

## Enumerator

ofShaderVertex	Vertex Shader.
ofShaderGeometry	Geometry Shader.
ofShaderFragment	Fragment Shader.
ofShaderTessControl	Tessellation Control Shader.
ofShaderTessEval	Tessellation Evaluation Shader.
ofShaderCompute	Compute Shader.

Definition at line 75 of file [render.hpp](#).

### 9.21.3 Function Documentation

#### 9.21.3.1 ofLoadDefaultFragShader()

```
ofShader oficina::ofLoadDefaultFragShader ()
```

Loads the default fragment shader.

**Returns**

Reference to the default fragment shader.

#### 9.21.3.2 ofLoadDefaultShaderProgram()

```
ofShaderProgram oficina::ofLoadDefaultShaderProgram ()
```

Loads the default shader program, with default vertex and fragment shaders.

**Returns**

Reference to the default shader program.

#### 9.21.3.3 ofLoadDefaultVertexShader()

```
ofShader oficina::ofLoadDefaultVertexShader ()
```

Loads the default vertex shader.

**Returns**

Reference to the default vertex shader.

#### 9.21.3.4 ofSetVSync()

```
void oficina::ofSetVSync (
 bool state)
```

Sets whether the game should vertically sync with the screen or not.

## Parameters

<i>state</i>	Default VSync state.
--------------	----------------------

## 9.21.4 Variable Documentation

## 9.21.4.1 ofDefaultShaderSrc\_FS

```
const char oficina::ofDefaultShaderSrc_FS[]
```

## Initial value:

```
=
 R" (#version 330

 in vec3 Color;
 in vec2 Texcoord;

 out vec4 outColor;

 uniform sampler2D tex;

 void main()
 {
 vec4 texColor = texture(tex, Texcoord);
 outColor = texColor * vec4(Color, 1.0);
 }) "
```

Default fragment shader source.

By default, receives color and texture coordinates from the default vertex shader, and asks for a texture to be bound on unit 0 so it can access the texture using a uniform sampler2D.

Definition at line 176 of file [render.hpp](#).

## 9.21.4.2 ofDefaultShaderSrc\_VS

```
const char oficina::ofDefaultShaderSrc_VS[]
```

## Initial value:

```
=
 R" (#version 330

 in vec3 position;
 in vec3 color;
 in vec2 texcoord;

 out vec3 Color;
 out vec2 Texcoord;

 uniform mat4 mvp;

 void main()
 {
 Color = color;
 Texcoord = texcoord;
 gl_Position = mvp * vec4(position, 1.0);
 }) "
```

Default vertex shader source.

By default, asks for position, color and texture coordinates to be fed using a vertex buffer, and an MVP matrix fed by an uniform.

Definition at line 152 of file [render.hpp](#).



## 9.22 render.hpp

```

00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00022 #pragma once
00023
00024 #include <SDL2/SDL.h>
00025 #include "oficina2/display.hpp"
00026 #include "oficina2/types.hpp"
00027 #include "oficina2/timer.hpp"
00028 #include "oficina2/entity.hpp"
00029 #include <GL/glew.h>
00030 #include <GL/gl.h>
00031 #include <string>
00032 #include <map>
00033
00034 namespace oficina
00035 {
00038 enum ofContextType
00039 {
00041 ofContextNone,
00043 ofContextGL,
00045 ofContextGLES
00046 };
00047
00050 enum ofBufferUsage
00051 {
00053 ofBufferStaticDraw = GL_STATIC_DRAW,
00055 ofBufferDynamicDraw = GL_DYNAMIC_DRAW,
00057 ofBufferStreamDraw = GL_STREAM_DRAW,
00058
00060 ofBufferStaticRead = GL_STATIC_READ,
00062 ofBufferDynamicRead = GL_DYNAMIC_READ,
00064 ofBufferStreamRead = GL_STREAM_READ,
00065
00067 ofBufferStaticCopy = GL_STATIC_COPY,
00069 ofBufferDynamicCopy = GL_DYNAMIC_COPY,
00071 ofBufferStreamCopy = GL_STREAM_COPY,
00072 };
00073
00075 enum ofShaderType
00076 {
00078 ofShaderVertex = GL_VERTEX_SHADER,
00080 ofShaderGeometry = GL_GEOMETRY_SHADER,
00082 ofShaderFragment = GL_FRAGMENT_SHADER,
00084 ofShaderTessControl = GL_TESS_CONTROL_SHADER,
00086 ofShaderTessEval = GL_TESS_EVALUATION_SHADER,
00088 ofShaderCompute = GL_COMPUTE_SHADER
00089 };
00090
00092 enum ofPrimitiveType
00093 {
00095 ofPoints = GL_POINTS,
00097 ofLineStrip = GL_LINE_STRIP,
00099 ofLineLoop = GL_LINE_LOOP,
00101 ofLines = GL_LINES,
00103 ofLineStripAdj = GL_LINE_STRIP_ADJACENCY,
00105 ofLinesAdj = GL_LINES_ADJACENCY,
00107 ofTriangleStrip = GL_TRIANGLE_STRIP,
00109 ofTriangleFan = GL_TRIANGLE_FAN,
00111 ofQuad = GL_TRIANGLE_FAN,
00113 ofTriangles = GL_TRIANGLES,
00115 ofTriangleStripAdj = GL_TRIANGLE_STRIP_ADJACENCY,
00117 ofTrianglesAdj = GL_TRIANGLES_ADJACENCY,
00119 ofPatches = GL_PATCHES
00120 };
00121
00123 enum ofDataType
00124 {
00126 ofDataByte = GL_BYTE,
00128 ofDataUByte = GL_UNSIGNED_BYTE,
00130 ofDataShort = GL_SHORT,
00132 ofDataUShort = GL_UNSIGNED_SHORT,

```

```

00134 ofDataInt = GL_INT,
00135 ofDataUInt = GL_UNSIGNED_INT,
00136 ofDataFloat = GL_FLOAT,
00137 ofDataDouble = GL_DOUBLE,
00138 ofDataFixed = GL_FIXED
00139 };
00140
00141 const char ofDefaultShaderSrc_VS[] =
00142 R"(#version 330
00143
00144 in vec3 position;
00145 in vec3 color;
00146 in vec2 texcoord;
00147
00148 out vec3 Color;
00149 out vec2 Texcoord;
00150
00151 uniform mat4 mvp;
00152
00153 void main()
00154 {
00155 Color = color;
00156 Texcoord = texcoord;
00157 gl_Position = mvp * vec4(position, 1.0);
00158 })";
00159
00160 const char ofDefaultShaderSrc_FS[] =
00161 R"(#version 330
00162
00163 in vec3 Color;
00164 in vec2 Texcoord;
00165
00166 out vec4 outColor;
00167
00168 uniform sampler2D tex;
00169
00170 void main()
00171 {
00172 vec4 texColor = texture(tex, Texcoord);
00173 outColor = texColor * vec4(Color, 1.0);
00174 })";
00175
00176 class ofContext
00177 {
00178 public:
00179 void pushArg(std::string arg);
00180 void open(ofContextType type, const ofDisplay& hwnd);
00181 void close();
00182
00183 bool isInit() const;
00184 void setViewportSize(glm::uvec2 sz);
00185 glm::uvec2 getViewportSize();
00186 void setClearColor(glm::vec4 color);
00187 private:
00188 ofContextType m_type = ofContextNone;
00189 glm::uvec2 m_vwpsz;
00190 glm::vec4 m_clearcolor = glm::vec4(0.0f, 0.0f, 0.0f, 1.0f);
00191 SDL_GLContext ctx;
00192 bool m_initialized = false;
00193 std::list<std::string> m_confv;
00194 void parseArgs();
00195 };
00196
00197 class ofBuffer
00198 {
00199 public:
00200 virtual void init() final;
00201 virtual void unload() final;
00202 virtual void bind() final;
00203 virtual void unbind() final;
00204
00205 virtual void setData(size_t dataSize,
00206 void* data,
00207 ofBufferUsage usage);
00208
00209 ofBuffer& operator=(const ofBuffer& other);
00210
00211 virtual bool isInit() const final;
00212 virtual GLuint getName() const final;
00213 protected:
00214 GLenum m_type = GL_ARRAY_BUFFER;
00215 GLuint m_name = 0u;
00216 };

```

```

00285
00286
00289 class ofVertexBuffer final : public ofBuffer
00290 {
00291 public:
00293 ofVertexBuffer();
00294 };
00295
00299 class ofElementBuffer final : public ofBuffer
00300 {
00301 public:
00303 ofElementBuffer();
00304
00308 void setCount(GLsizei count);
00312 void setType(ofDataType type);
00317 void setProps(GLsizei count, ofDataType type);
00318
00321 GLsizei getCount() const;
00324 ofDataType getType() const;
00325
00332 void draw(ofPrimitiveType mode);
00333 private:
00334 GLsizei m_count = -1;
00335 ofDataType m_dataType = ofDataUInt;
00336 };
00337
00338
00339
00341 class ofShader
00342 {
00343 public:
00346 virtual void init(ofShaderType type) final;
00348 virtual void unload() final;
00352 virtual void setSource(const char* src) final;
00356 virtual void compile() final;
00357
00360 virtual bool isInit() const final;
00363 virtual bool isCompiled() const final;
00367 virtual GLuint getName() const final;
00368
00372 ofShader& operator=(const ofShader& shader);
00373 protected:
00375 ofShaderType m_type = ofShaderFragment;
00377 GLuint m_name = 0u;
00379 bool m_srcassign = false;
00381 bool m_compiled = false;
00382 };
00383
00384 class ofShaderProgram;
00386 class ofShaderAttribute final
00387 {
00388 friend class ofShaderProgram;
00389 public:
00392 void setSize(GLint s);
00395 void setType(ofDataType t);
00398 void setStride(GLsizei stride);
0401 void setAutoNormalize(bool state);
0407 void setProps(GLint size, ofDataType type, GLsizei stride, bool normalize = false);
0408
0410 void enable();
0411
0414 int getSize();
0417 ofDataType getType();
0420 size_t getStride();
0423 bool isAutoNormalizing();
0426 bool isValid() const;
0427
0433 void bindVertexArrayData(void* byteOffset = nullptr);
0434
0438 ofShaderAttribute& operator=(const ofShaderAttribute& attr);
0439 private:
0440 GLint m_name = -1;
0441 GLint m_size = 1;
0442 GLsizei m_stride = 0;
0443 bool m_normalize = false;
0444 ofDataType m_type = ofDataFloat;
0445 };
0446
0447 class ofTexture;
00452 class ofShaderUniform final
00453 {
00454 friend class ofShaderProgram;
00455 public:
00458 bool isValid() const;
00462 ofShaderUniform& operator=(const ofShaderUniform& uniform);
00463
00466 void set(float value);

```

```

00469 void set(glm::vec2 value);
00472 void set(glm::vec3 value);
00475 void set(glm::vec4 value);
00476
00479 void set(int value);
00482 void set(glm::ivec2 value);
00485 void set(glm::ivec3 value);
00488 void set(glm::ivec4 value);
00489
00492 void set(unsigned int value);
00495 void set(glm::uvec2 value);
00498 void set(glm::uvec3 value);
00501 void set(glm::uvec4 value);
00502
00506 void set(glm::mat2 value, bool transpose = false);
00510 void set(glm::mat3 value, bool transpose = false);
00514 void set(glm::mat4 value, bool transpose = false);
00515
00519 void set(glm::mat2x3 value, bool transpose = false);
00523 void set(glm::mat3x2 value, bool transpose = false);
00524
00528 void set(glm::mat2x4 value, bool transpose = false);
00532 void set(glm::mat4x2 value, bool transpose = false);
00533
00537 void set(glm::mat3x4 value, bool transpose = false);
00541 void set(glm::mat4x3 value, bool transpose = false);
00542 private:
00543 GLint m_name = -1;
00544 };
00545
00547 class ofShaderProgram final
00548 {
00549 public:
00551 void init();
00553 void unload();
00557 void attach(const ofShader& shader);
00562 void attachUnload(ofShader& shader);
00570 void bindFragmentDataLocation(std::string name, ofdword colorNumber = 0u);
00574 void link();
00577 void use();
00579 void unuse();
00580
00583 bool isInit() const;
00586 bool isLinked() const;
00589 GLuint getName() const;
00590
00594 ofShaderProgram& operator=(const ofShaderProgram& program);
00595
00599 ofShaderAttribute getAttributeLocation(std::string name);
00603 ofShaderUniform getUniformLocation(std::string name);
00604 private:
00605 bool shaderProgramVerify(const ofShader&) const;
00606 GLuint m_name = 0u;
00607 bool m_linked = false;
00608 };
00609
00610
00611
00613 class ofVertexArray
00614 {
00615 public:
00617 void init();
00619 void unload();
00621 void bind();
00623 void unbind();
00624
00632 void draw(ofPrimitiveType mode, int firstVertexIdx, size_t vertexCount);
00633
00637 ofVertexArray& operator=(const ofVertexArray& other);
00638 private:
00639 GLuint m_name = 0u;
00640 };
00641
00642
00643
00644
00645 // Textures
00646 class ofTexturePool;
00648 class ofTexture
00649 {
00650 friend class ofTexturePool;
00651 public:
00655 void bind(ofword currentSampler = 0);
00659 void unbind(ofword currentSampler = 0);
00660
00664 ofTexture& operator=(const ofTexture& other);
00667 GLuint operator()();

```

```

00668
00671 bool isLoading() const;
00674 std::string getFileName() const;
00678 glm::uvec2 getSize() const;
00679 private:
00680 GLuint m_name = 0u;
00681 glm::uvec2 m_size;
00682 std::string m_filename;
00683 };
00684
00686 enum ofFontFaces
00687 {
00690 ofFontFaceFixedsysExcelsior,
00693 ofFontFaceGohuFont,
00696 ofFontFaceFantasqueSans,
00698 ofFontFaceTerminus
00699 };
00700
00701 class ofFont;
00711 class ofTexturePool
00712 {
00713 public:
00717 static ofTexture load(std::string filename);
00721 static ofTexture load(SDL_Surface* surf);
00725 static ofFont loadDefaultFont(ofFontFaces fontface =
ofFontFaceGohuFont);
00728 static void unload(ofTexture& t);
00730 static void clear();
00731 };
00732
00733
00734
00736 class ofTextureRenderer
00737 {
00738 public:
00744 void init(ofTexture t, glm::uvec2 frameSize = glm::uvec2(0, 0));
00757 void render(glm::vec2 position, glm::mat4 mvp, ofdword frame = 0u, glm::vec4 color =
glm::vec4(1.0f));
00760 void unload();
00761
00765 ofTextureRenderer& operator=(const ofTextureRenderer& other);
00766
00772 void SetTexture(ofTexture t);
00773
00776 bool isInit() const;
00777 private:
00778 bool m_initialized = false;
00779 ofTexture m_texture;
00780 glm::vec2 m_frameSize;
00781 glm::vec2 m_frameSizeTx1;
00782 ofVertexArray vao;
00783 ofVertexBuffer vbo;
00784 ofShaderAttribute attrQuadpos,
00785 attrTexcoord,
00786 attrFrSz,
00787 attrFrSzTx1;
00788
00789 ofShaderUniform uniPos,
00790 uniFrpostx1,
00791 uniColor,
00792 uniMVP,
00793 uniTexSampler;
00794 };
00795
00802 class ofFont
00803 {
00804 public:
00811 void init(ofTexture fontTexture, glm::uvec2 glyphSize, bool manageTexture = false);
00820 void write(std::string text, glm::vec2 position, glm::mat4 mvp, glm::vec4 color = glm::vec4(1.0f));
00823 void unload();
00827 glm::vec2 measure(std::string text);
00828
00832 ofFont& operator=(const ofFont& other);
00833
00836 glm::uvec2 getGlyphSize() const;
00837
00840 bool isInit() const;
00841 private:
00842 bool m_unloadtexture = false;
00843 bool m_initialized = false;
00844 glm::uvec2 m_glyphsize;
00845 ofTexture m_texture;
00846 ofTextureRenderer m_renderer;
00847 };
00848
00851 class ofAnimator : public ofIComponent
00852 {

```

```

00853 public:
00854 void init();
00860 void unload();
00864 void update(float dt);
00871 void draw(glm::mat4 ViewProjection);
00872
00891 void reg(std::string animName, ofdword nFrames, const ofdword* animFrames, float
speed, bool loops = false, ofdword loopBackTo = 0u);
00894 void unreg(std::string animName);
00899 void SetAnimation(std::string animName);
00903 void SyncToFrameRate(bool state);
00910 void SetAnimationSpeed(float spd);
00915 float GetAnimationSpeed() const;
00920 float GetDefaultAnimationSpeed() const;
00924 void SetAnimationRunning(bool state);
00927 std::string GetCurrentAnimationName() const;
00928
00934 void SetRenderer(ofTextureRenderer renderer, bool manage = false);
00940 void SetAnimationTexture(ofTexture t);
00941
00944 bool isInit() const;
00947 glm::vec2 getPosition();
00950 void setPosition(glm::vec2 pos);
00953 bool GetAnimationRunning() const;
00954 private:
00955 struct ofAnimProps
00956 {
00957 ofdword num_frames;
00958 ofdword loopback;
00959 ofdword* frames = nullptr;
00960 bool loops;
00961 float speed;
00962 };
00963
00964 bool m_unloadtexture = false;
00965 bool m_initialized = false;
00966 bool m_sync = true;
00967 bool m_playing = true;
00968 const ofAnimProps* m_current = nullptr;
00969 ofdword m_currentframe = 0u;
00970 ofFrameSpan m_framespan;
00971 ofTimeSpan m_timespan;
00972 glm::uvec2 m_framesize;
00973 glm::vec2 m_position;
00974 ofTextureRenderer m_renderer;
00975 std::string m_animname;
00976 float m_animspd;
00977 std::map<std::string, ofAnimProps> m_animations;
00978 };
00979
00980 class ofPrimitiveRenderer;
00982 struct ofPrimitive
00983 {
00984 friend class ofPrimitiveRenderer;
00985 public:
00987 ofPrimitiveType type;
00989 ofVertexBuffer vbo;
00991 ofdword NumberOfVertices;
00993 ~ofPrimitive();
00994 private:
00997 ofPrimitive() {}
00998 };
00999
01002 class ofPrimitiveRenderer
01003 {
01004 public:
01015 static ofPrimitive* makePrimitive(ofPrimitiveType type,
ofdword verticesAmount, size_t verticesSize, float* vertices);
01021 static void draw(ofPrimitive* p, glm::vec4 color, glm::mat4 mvp);
01022 };
01023
01024
01027 ofShader ofLoadDefaultFragShader();
01030 ofShader ofLoadDefaultVertexShader();
01034 ofShaderProgram ofLoadDefaultShaderProgram();
01038 void ofSetVSync(bool state);
01039 }

```

## 9.23 timer.hpp File Reference

Tools for counting and processing time-related events.

```
#include <cstdint>
```

## Classes

- class [oficina::ofTimeSpan](#)  
*Tool for counting and compare fixed amounts of time, independent from the game's time variation.*
- class [oficina::ofFrameSpan](#)  
*Tool for counting and comparing frames, depending of the game's time variation.*

### 9.23.1 Detailed Description

Tools for counting and processing time-related events.

#### Author

Lucas Vieira

Definition in file [timer.hpp](#).

## 9.24 timer.hpp

```
00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00022 #pragma once
00023
00024 #include <cstdint>
00025
00026 namespace oficina
00027 {
00031 class ofTimeSpan
00032 {
00033 public:
00036 void begin();
00041 float yieldSpan();
00046 float resetSpan();
00050 float stop();
00054 bool isRunning() const;
00055 private:
00056 bool m_started = false;
00057 uint32_t m_timer = 0u;
00058 };
00059
00062 class ofFrameSpan
00063 {
00064 public:
00066 void begin();
00068 void update();
00074 uint32_t yieldSpan();
00078 uint32_t resetSpan();
00082 uint32_t stop();
00086 bool isRunning() const;
00087 private:
00088 bool m_started = false;
00089 uint32_t m_timer = 0u;
00090 };
00091 }
```

## 9.25 types.hpp File Reference

Tools for predefining default types and math tools used by OficinaFramework.

```
#include "oficina2/platform.hpp"
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <cmath>
#include <cstdint>
```

### Typedefs

- typedef uint8\_t [ofbyte](#)  
*Unsigned integer with size of at least one byte.*
- typedef uint16\_t [ofword](#)  
*Unsigned integer with size of at least two bytes.*
- typedef uint32\_t [ofdword](#)  
*Unsigned integer with size of at least four bytes.*
- typedef uint64\_t [ofqword](#)  
*Unsigned integer with size of at least eight bytes.*
- typedef int8\_t [ofsbyte](#)  
*Signed integer with size of at least one byte.*
- typedef int16\_t [ofsword](#)  
*Signed integer with size of at least two bytes.*
- typedef int32\_t [ofsdword](#)  
*Signed integer with size of at least four bytes.*
- typedef int64\_t [ofsqword](#)  
*Signed integer with size of at least eight bytes.*
- typedef uintptr\_t [ofaword](#)  
*Unsigned integer with enough size to hold a memory pointer. Size varies according to processor architecture.*
- typedef intptr\_t [ofsaword](#)  
*Signed integer with enough size to hold a memory pointer. Size varies according to processor architecture.*

### Functions

- float [ofClamp](#) (float value, float min, float max)  
*Clamps a floating point between two other values.*

#### 9.25.1 Detailed Description

Tools for predefining default types and math tools used by OficinaFramework.

#### Author

Lucas Vieira

Definition in file [types.hpp](#).



## 9.25.2 Function Documentation

### 9.25.2.1 ofClamp()

```
float ofClamp (
 float value,
 float min,
 float max)
```

Clamps a floating point between two other values.

#### Parameters

<i>value</i>	Value to be compared.
<i>min</i>	Minimum value tolerated by the clamping operation.
<i>max</i>	Maximum value tolerated by the clamping operation.

#### Returns

The given value, accordingly clamped between the given minimum and maximum values.

## 9.26 types.hpp

```
00001 /*****
00002 * Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com> *
00003 * This file is part of OficinaFramework v2.x *
00004 * *
00005 * OficinaFramework is free software: you can redistribute *
00006 * it and/or modify it under the terms of the GNU Lesser *
00007 * General Public License as published by the Free Software *
00008 * Foundation, either version 3 of the License, or (at your *
00009 * option) any later version. *
00010 * *
00011 * You should have received a copy of the GNU Lesser General *
00012 * Public License along with OficinaFramework. If not, see *
00013 * <http://www.gnu.org/licenses/>. *
00014 *****/
00015
00023 //define GLM_FORCE_SWIZZLE
00024
00025 #include "oficina2/platform.hpp"
00026 #include <glm/glm.hpp>
00027 #include <glm/gtc/matrix_transform.hpp>
00028 #include <glm/gtc/type_ptr.hpp>
00029 #include <cmath>
00030 #include <cstdint>
00031
00032 #pragma once
00033
00035 typedef uint8_t ofbyte;
00037 typedef uint16_t ofword;
00039 typedef uint32_t ofdword;
00041 typedef uint64_t ofqword;
00042
00044 typedef int8_t ofsbyte;
00046 typedef int16_t ofsword;
00048 typedef int32_t ofsdword;
00050 typedef int64_t ofsqword;
00051
00054 typedef uintptr_t ofaword;
00057 typedef intptr_t ofsaword;
00058
00065 float ofClamp(float value, float min, float max);
```



# Index

- add
  - oficina::ofCanvasManager, [62](#)
- AddComponent
  - oficina::ofEntity, [77](#)
- attach
  - oficina::ofShaderProgram, [113](#)
- attachUnload
  - oficina::ofShaderProgram, [113](#)
- benchmark.hpp, [137](#), [138](#)
  - ofBenchmarkIsRunning, [137](#)
  - ofBenchmarkStart, [138](#)
- bind
  - oficina::ofTexture, [126](#)
- bindFragmentDataLocation
  - oficina::ofShaderProgram, [114](#)
- bindVertexArrayData
  - oficina::ofShaderAttribute, [109](#)
- canvas.hpp, [138](#), [139](#)
- close
  - oficina::ofDisplay, [69](#)
- compile
  - oficina::ofShader, [106](#)
- dbg\_ChangeState
  - oficina::ofCanvasManager, [63](#)
- dbg\_ReplLineNumber
  - oficina::ofCanvasManager, [63](#)
- dbg\_ReplOutputStream
  - oficina::ofCanvasManager, [64](#)
- dbg\_callEval
  - oficina::ofCanvasManager, [63](#)
- dbg\_getState
  - oficina::ofCanvasManager, [63](#)
- dbg\_setFont
  - oficina::ofCanvasManager, [64](#)
- display.hpp, [140](#), [141](#)
  - ofFrameRateConfig, [141](#)
- draw
  - oficina::ofAnimator, [50](#)
  - oficina::ofCanvasManager, [64](#)
  - oficina::ofElementBuffer, [73](#)
  - oficina::ofEntity, [77](#)
  - oficina::ofIComponent, [91](#)
  - oficina::ofPrimitiveRenderer, [100](#)
  - oficina::ofVertexArray, [135](#)
- DrawComponents
  - oficina::ofEntity, [78](#)
- entity.hpp, [142](#), [143](#)

- GetAnimationRunning
  - oficina::ofAnimator, [51](#)
- GetAnimationSpeed
  - oficina::ofAnimator, [51](#)
- getAttributeLocation
  - oficina::ofShaderProgram, [114](#)
- getBoolean
  - oficina::ofLua, [94](#)
- getCanvasList
  - oficina::ofCanvasManager, [64](#)
- GetComponent
  - oficina::ofEntity, [78](#)
- GetComponent
  - oficina::ofLua, [94](#)
- getCount
  - oficina::ofElementBuffer, [74](#)
- GetCurrentAnimationName
  - oficina::ofAnimator, [51](#)
- GetDefaultAnimationSpeed
  - oficina::ofAnimator, [51](#)
- getDeltaTime
  - oficina::ofDisplay, [69](#)
- getEntity
  - oficina::ofLua, [94](#)
- getEulerAngles
  - oficina::ofEntity, [78](#)
- getFileName
  - oficina::ofTexture, [126](#)
- getGlyphSize
  - oficina::ofFont, [86](#)
- getHandle
  - oficina::ofDisplay, [70](#)
- getInteger
  - oficina::ofLua, [95](#)
- getModelMatrix
  - oficina::ofEntity, [79](#)
- getName
  - oficina::ofBuffer, [57](#)
  - oficina::ofEntity, [79](#)
  - oficina::ofShader, [106](#)
  - oficina::ofShaderProgram, [115](#)
- getNumber
  - oficina::ofLua, [95](#)
- getPosition
  - oficina::ofAnimator, [52](#)
  - oficina::ofEntity, [79](#)
- getProperty
  - oficina::ofEntity, [79](#)
- getPropertyMask

- oficina::ofEntity, 80
- getScale
  - oficina::ofEntity, 80
- getSize
  - oficina::ofDisplay, 70
  - oficina::ofShaderAttribute, 109
  - oficina::ofTexture, 126
- getStride
  - oficina::ofShaderAttribute, 109
- getString
  - oficina::ofLua, 95
- getSymRef
  - oficina::ofScheme, 102
- getType
  - oficina::ofElementBuffer, 74
  - oficina::ofIComponent, 91
  - oficina::ofShaderAttribute, 109
- getUniformLocation
  - oficina::ofShaderProgram, 115
- getViewportSize
  - oficina::ofContext, 66
- init
  - oficina::ofAnimator, 52
  - oficina::ofCanvas, 59
  - oficina::ofEntity, 80
  - oficina::ofFont, 86
  - oficina::ofShader, 106
  - oficina::ofTextureRenderer, 130
- input.hpp, 144, 159
  - ofButtonPress, 149
  - ofButtonTap, 150
  - ofGetInputState, 150
  - ofGetLeftStick, 151
  - ofGetLeftTrigger, 151
  - ofGetMousePos, 151
  - ofGetRightStick, 152
  - ofGetRightTrigger, 152
  - ofGetTextInput, 152
  - ofIsGamepadConnected, 153
  - ofIsInputtingText, 153
  - ofMapButtonRemove, 153
  - ofMapDefaultsP1, 154
  - ofMapKeyToButton, 154
  - ofMapKeyToStick, 155
  - ofMapStickRemove, 155
  - ofMappingClear, 155
  - ofMouseButton, 146
  - ofMouseButtonPress, 156
  - ofMouseButtonTap, 156
  - ofPadButton, 147
  - ofPlayer, 148
  - ofSetTextInput, 157
  - ofStartTextInput, 157
  - ofStick, 148
  - ofStickAxis, 148
  - ofStickMovedTowards, 157
  - ofStickSignal, 149
  - ofStopTextInput, 158
  - ofTextInputSetPadding, 158
  - ofUpdateEventDispatch, 159
- io.hpp, 161, 166
  - ofFindAsset, 163
  - ofLoadImage, 163
  - ofLoadText, 164
  - ofLog, 164
  - ofLogGetType, 164
  - ofLogLevel, 162
  - ofLogSetLevel, 165
  - ofLogType, 163
  - ofLogUseFile, 165
  - ofSetDataDirectoryName, 165
- isAutoNormalizing
  - oficina::ofShaderAttribute, 110
- isCompiled
  - oficina::ofShader, 106
- isFullscreen
  - oficina::ofDisplay, 70
- isInit
  - oficina::ofAnimator, 52
  - oficina::ofBuffer, 58
  - oficina::ofContext, 67
  - oficina::ofFont, 86
  - oficina::ofLua, 96
  - oficina::ofScheme, 103
  - oficina::ofShader, 107
  - oficina::ofShaderProgram, 115
  - oficina::ofTextureRenderer, 130
- isLinked
  - oficina::ofShaderProgram, 115
- isLoading
  - oficina::ofTexture, 126
- isOpen
  - oficina::ofDisplay, 70
- isRunning
  - oficina::ofFrameSpan, 89
  - oficina::ofTimeSpan, 133
- isValid
  - oficina::ofShaderAttribute, 110
  - oficina::ofShaderUniform, 118
- link
  - oficina::ofShaderProgram, 116
- load
  - oficina::ofCanvas, 60
  - oficina::ofEntity, 80
  - oficina::ofTexturePool, 128
- loadDefaultFont
  - oficina::ofTexturePool, 129
- loadfile
  - oficina::ofLua, 96
  - oficina::ofScheme, 103
- makePrimitive
  - oficina::ofPrimitiveRenderer, 101
- measure
  - oficina::ofFont, 87

- ofBenchmarksRunning
  - benchmark.hpp, 137
- ofBenchmarkStart
  - benchmark.hpp, 138
- ofBufferUsage
  - render.hpp, 192
- ofButtonPress
  - input.hpp, 149
- ofButtonTap
  - input.hpp, 150
- ofClamp
  - types.hpp, 205
- ofContextType
  - render.hpp, 193
- ofDataType
  - render.hpp, 193
- ofDebuggerState
  - oficina::ofCanvasManager, 62
- ofDefaultShaderSrc\_FS
  - render.hpp, 196
- ofDefaultShaderSrc\_VS
  - render.hpp, 196
- ofFindAsset
  - io.hpp, 163
- ofFontFaces
  - render.hpp, 193
- ofFrameRateConfig
  - display.hpp, 141
- ofGameLoop
  - oficina.hpp, 169
- ofGetInputState
  - input.hpp, 150
- ofGetLeftStick
  - input.hpp, 151
- ofGetLeftTrigger
  - input.hpp, 151
- ofGetMousePos
  - input.hpp, 151
- ofGetReplType
  - oficina.hpp, 169
- ofGetRightStick
  - input.hpp, 152
- ofGetRightTrigger
  - input.hpp, 152
- ofGetTextInput
  - input.hpp, 152
- ofGetWindowSize
  - oficina.hpp, 169
- ofInit
  - oficina.hpp, 169, 170
- ofIsFullscreen
  - oficina.hpp, 170
- ofIsGamepadConnected
  - input.hpp, 153
- ofIsInputtingText
  - input.hpp, 153
- ofLoadDefaultFragShader
  - render.hpp, 195
- ofLoadDefaultShaderProgram
  - render.hpp, 195
- ofLoadDefaultVertexShader
  - render.hpp, 195
- ofLoadImage
  - io.hpp, 163
- ofLoadText
  - io.hpp, 164
- ofLog
  - io.hpp, 164
- ofLogGetType
  - io.hpp, 164
- ofLogLvl
  - io.hpp, 162
- ofLogSetLevel
  - io.hpp, 165
- ofLogType
  - io.hpp, 163
- ofLogUseFile
  - io.hpp, 165
- ofLuaDefineFunc
  - oflua.hpp, 175
- ofLuaDefineSymbol
  - oflua.hpp, 175–177
- ofLuaEval
  - oflua.hpp, 177
- ofLuaGetBoolean
  - oflua.hpp, 177
- ofLuaGetComponent
  - oflua.hpp, 178
- ofLuaGetEntity
  - oflua.hpp, 178
- ofLuaGetInteger
  - oflua.hpp, 179
- ofLuaGetNumber
  - oflua.hpp, 179
- ofLuaGetString
  - oflua.hpp, 179
- ofLuaIsInit
  - oflua.hpp, 180
- ofLuaUndefine
  - oflua.hpp, 180
- ofMapButtonRemove
  - input.hpp, 153
- ofMapDefaultsP1
  - input.hpp, 154
- ofMapKeyToButton
  - input.hpp, 154
- ofMapKeyToStick
  - input.hpp, 155
- ofMapStickRemove
  - input.hpp, 155
- ofMappingClear
  - input.hpp, 155
- ofMouseButton
  - input.hpp, 146
- ofMouseButtonPress
  - input.hpp, 156

- ofMouseButtonTap
  - input.hpp, 156
- ofPadButton
  - input.hpp, 147
- ofPlayer
  - input.hpp, 148
- ofPrimitiveType
  - render.hpp, 194
- ofQuit
  - oficina.hpp, 170
- ofQuitFlagRaised
  - oficina.hpp, 170
- ofReplEval
  - oficina.hpp, 170
- ofReplType
  - oficina.hpp, 168
- ofScmDefineFunc
  - ofscheme.hpp, 183
- ofScmDefineSymbol
  - ofscheme.hpp, 183, 184
- ofScmEval
  - ofscheme.hpp, 185
- ofScmGetReference
  - ofscheme.hpp, 185
- ofScmIsInit
  - ofscheme.hpp, 185
- ofScmUndefine
  - ofscheme.hpp, 185
- ofSetClearColor
  - oficina.hpp, 171
- ofSetDataDirectoryName
  - io.hpp, 165
- ofSetFullscreen
  - oficina.hpp, 171
- ofSetReplType
  - oficina.hpp, 171
- ofSetSwapInterval
  - oficina.hpp, 171
- ofSetTextInput
  - input.hpp, 157
- ofSetVSync
  - render.hpp, 195
- ofSetWindowSize
  - oficina.hpp, 172
- ofShaderType
  - render.hpp, 194
- ofSoftStop
  - oficina.hpp, 172
- ofStartTextInput
  - input.hpp, 157
- ofStick
  - input.hpp, 148
- ofStickAxis
  - input.hpp, 148
- ofStickMovedTowards
  - input.hpp, 157
- ofStickSignal
  - input.hpp, 149
- ofStopTextInput
  - input.hpp, 158
- ofTextInputSetPadding
  - input.hpp, 158
- ofUpdateEventDispatch
  - input.hpp, 159
- oficina.hpp, 167, 173
  - ofGameLoop, 169
  - ofGetReplType, 169
  - ofGetWindowSize, 169
  - ofInit, 169, 170
  - ofIsFullscreen, 170
  - ofQuit, 170
  - ofQuitFlagRaised, 170
  - ofReplEval, 170
  - ofReplType, 168
  - ofSetClearColor, 171
  - ofSetFullscreen, 171
  - ofSetReplType, 171
  - ofSetSwapInterval, 171
  - ofSetWindowSize, 172
  - ofSoftStop, 172
- oficina::ofAnimator, 49
  - draw, 50
  - GetAnimationRunning, 51
  - GetAnimationSpeed, 51
  - GetCurrentAnimationName, 51
  - GetDefaultAnimationSpeed, 51
  - getPosition, 52
  - init, 52
  - isInit, 52
  - reg, 53
  - SetAnimation, 53
  - SetAnimationRunning, 54
  - SetAnimationSpeed, 54
  - SetAnimationTexture, 55
  - setPosition, 55
  - SetRenderer, 55
  - SyncToFrameRate, 55
  - unreg, 56
  - update, 56
- oficina::ofBuffer, 56
  - getName, 57
  - isInit, 58
  - operator=, 58
  - setData, 58
- oficina::ofCanvas, 59
  - init, 59
  - load, 60
  - remove, 60
  - update, 60
- oficina::ofCanvasManager, 61
  - add, 62
  - dbg\_ChangeState, 63
  - dbg\_ReplLineNumber, 63
  - dbg\_ReplOutputStream, 64
  - dbg\_callEval, 63
  - dbg\_getState, 63

- dbg\_setFont, 64
- draw, 64
- getCanvasList, 64
- ofDebuggerState, 62
- remove, 65
- unload, 65
- update, 65
- oficina::ofContext, 66
  - getViewportSize, 66
  - isInit, 67
  - open, 67
  - pushArg, 67
  - setClearColor, 68
  - setViewportSize, 68
- oficina::ofDisplay, 68
  - close, 69
  - getDeltaTime, 69
  - getHandle, 70
  - getSize, 70
  - isFullscreen, 70
  - isOpen, 70
  - open, 71
  - pushArg, 71
  - setFullscreen, 71
  - setSize, 72
  - setSwapInterval, 72
  - swap, 72
- oficina::ofElementBuffer, 73
  - draw, 73
  - getCount, 74
  - getType, 74
  - setCount, 74
  - setProps, 74
  - setType, 75
- oficina::ofEntity, 75
  - AddComponent, 77
  - draw, 77
  - DrawComponents, 78
  - GetComponent, 78
  - getEulerAngles, 78
  - getModelMatrix, 79
  - getName, 79
  - getPosition, 79
  - getProperty, 79
  - getPropertyMask, 80
  - getScale, 80
  - init, 80
  - load, 80
  - RemoveComponent, 81
  - rotate, 82
  - rotation, 84
  - scale, 82
  - scaling, 84
  - setName, 82
  - setProperty, 83
  - toggleProperty, 83
  - translate, 83
  - translation, 85
  - update, 84
  - UpdateComponents, 84
- oficina::ofFont, 85
  - getGlyphSize, 86
  - init, 86
  - isInit, 86
  - measure, 87
  - operator=, 87
  - write, 87
- oficina::ofFrameSpan, 88
  - isRunning, 89
  - resetSpan, 89
  - stop, 89
  - yieldSpan, 89
- oficina::ofIComponent, 90
  - draw, 91
  - getType, 91
- oficina::ofInputState, 92
- oficina::ofLua, 92
  - getBoolean, 94
  - GetComponent, 94
  - getEntity, 94
  - getInteger, 95
  - getNumber, 95
  - getString, 95
  - isInit, 96
  - loadfile, 96
  - regFunc, 96
  - regSym, 97, 98
  - update, 99
- oficina::ofPrimitive, 99
- oficina::ofPrimitiveRenderer, 100
  - draw, 100
  - makePrimitive, 101
- oficina::ofScheme, 101
  - getSymRef, 102
  - isInit, 103
  - loadfile, 103
  - regFunc, 103
  - regSym, 104
  - update, 104
- oficina::ofShader, 105
  - compile, 106
  - getName, 106
  - init, 106
  - isCompiled, 106
  - isInit, 107
  - operator=, 107
  - setSource, 107
- oficina::ofShaderAttribute, 108
  - bindVertexArrayData, 109
  - getSize, 109
  - getStride, 109
  - getType, 109
  - isAutoNormalizing, 110
  - isValid, 110
  - operator=, 110
  - setAutoNormalize, 111

- setProps, 111
  - setSize, 111
  - setStride, 112
  - setType, 112
- oficina::ofShaderProgram, 112
  - attach, 113
  - attachUnload, 113
  - bindFragmentDataLocation, 114
  - getAttributeLocation, 114
  - getName, 115
  - getUniformLocation, 115
  - isInit, 115
  - isLinked, 115
  - link, 116
  - operator=, 116
  - use, 116
- oficina::ofShaderUniform, 117
  - isValid, 118
  - operator=, 118
  - set, 119–124
- oficina::ofTexture, 125
  - bind, 126
  - getFileName, 126
  - getSize, 126
  - isLoading, 126
  - operator(), 126
  - operator=, 127
  - unbind, 127
- oficina::ofTexturePool, 127
  - load, 128
  - loadDefaultFont, 129
  - unload, 129
- oficina::ofTextureRenderer, 130
  - init, 130
  - isInit, 130
  - operator=, 131
  - render, 131
  - SetTexture, 132
  - unload, 132
- oficina::ofTimeSpan, 132
  - isRunning, 133
  - resetSpan, 133
  - stop, 133
  - yieldSpan, 134
- oficina::ofVertexArray, 134
  - draw, 135
  - operator=, 135
- oficina::ofVertexBuffer, 136
- oflua.hpp, 173, 181
  - ofLuaDefineFunc, 175
  - ofLuaDefineSymbol, 175–177
  - ofLuaEval, 177
  - ofLuaGetBoolean, 177
  - ofLuaGetComponent, 178
  - ofLuaGetEntity, 178
  - ofLuaGetInteger, 179
  - ofLuaGetNumber, 179
  - ofLuaGetString, 179
  - ofLuaInit, 180
  - ofLuaUndefine, 180
- ofscheme.hpp, 182, 187
  - ofScmDefineFunc, 183
  - ofScmDefineSymbol, 183, 184
  - ofScmEval, 185
  - ofScmGetReference, 185
  - ofScmIsInit, 185
  - ofScmUndefine, 185
- open
  - oficina::ofContext, 67
  - oficina::ofDisplay, 71
- operator()
  - oficina::ofTexture, 126
- operator=
  - oficina::ofBuffer, 58
  - oficina::ofFont, 87
  - oficina::ofShader, 107
  - oficina::ofShaderAttribute, 110
  - oficina::ofShaderProgram, 116
  - oficina::ofShaderUniform, 118
  - oficina::ofTexture, 127
  - oficina::ofTextureRenderer, 131
  - oficina::ofVertexArray, 135
- platform.hpp, 187, 189
- pushArg
  - oficina::ofContext, 67
  - oficina::ofDisplay, 71
- reg
  - oficina::ofAnimator, 53
- regFunc
  - oficina::ofLua, 96
  - oficina::ofScheme, 103
- regSym
  - oficina::ofLua, 97, 98
  - oficina::ofScheme, 104
- remove
  - oficina::ofCanvas, 60
  - oficina::ofCanvasManager, 65
- RemoveComponent
  - oficina::ofEntity, 81
- render
  - oficina::ofTextureRenderer, 131
- render.hpp, 190, 197
  - ofBufferUsage, 192
  - ofContextType, 193
  - ofDataType, 193
  - ofDefaultShaderSrc\_FS, 196
  - ofDefaultShaderSrc\_VS, 196
  - ofFontFaces, 193
  - ofLoadDefaultFragShader, 195
  - ofLoadDefaultShaderProgram, 195
  - ofLoadDefaultVertexShader, 195
  - ofPrimitiveType, 194
  - ofSetVSync, 195
  - ofShaderType, 194
- resetSpan



- oficina::ofFrameSpan, [89](#)
  - oficina::ofTimeSpan, [133](#)
- rotate
  - oficina::ofEntity, [82](#)
- rotation
  - oficina::ofEntity, [84](#)
- scale
  - oficina::ofEntity, [82](#)
- scaling
  - oficina::ofEntity, [84](#)
- set
  - oficina::ofShaderUniform, [119–124](#)
- SetAnimation
  - oficina::ofAnimator, [53](#)
- SetAnimationRunning
  - oficina::ofAnimator, [54](#)
- SetAnimationSpeed
  - oficina::ofAnimator, [54](#)
- SetAnimationTexture
  - oficina::ofAnimator, [55](#)
- setAutoNormalize
  - oficina::ofShaderAttribute, [111](#)
- setClearColor
  - oficina::ofContext, [68](#)
- setCount
  - oficina::ofElementBuffer, [74](#)
- setData
  - oficina::ofBuffer, [58](#)
- setFullscreen
  - oficina::ofDisplay, [71](#)
- setName
  - oficina::ofEntity, [82](#)
- setPosition
  - oficina::ofAnimator, [55](#)
- setProperty
  - oficina::ofEntity, [83](#)
- setProps
  - oficina::ofElementBuffer, [74](#)
  - oficina::ofShaderAttribute, [111](#)
- SetRenderer
  - oficina::ofAnimator, [55](#)
- setSize
  - oficina::ofDisplay, [72](#)
  - oficina::ofShaderAttribute, [111](#)
- setSource
  - oficina::ofShader, [107](#)
- setStride
  - oficina::ofShaderAttribute, [112](#)
- setSwapInterval
  - oficina::ofDisplay, [72](#)
- SetTexture
  - oficina::ofTextureRenderer, [132](#)
- setType
  - oficina::ofElementBuffer, [75](#)
  - oficina::ofShaderAttribute, [112](#)
- setViewportSize
  - oficina::ofContext, [68](#)
- stop
  - oficina::ofFrameSpan, [89](#)
  - oficina::ofTimeSpan, [133](#)
- swap
  - oficina::ofDisplay, [72](#)
- SyncToFrameRate
  - oficina::ofAnimator, [55](#)
- timer.hpp, [202](#), [203](#)
- toggleProperty
  - oficina::ofEntity, [83](#)
- translate
  - oficina::ofEntity, [83](#)
- translation
  - oficina::ofEntity, [85](#)
- types.hpp, [204](#), [205](#)
  - ofClamp, [205](#)
- unbind
  - oficina::ofTexture, [127](#)
- unload
  - oficina::ofCanvasManager, [65](#)
  - oficina::ofTexturePool, [129](#)
  - oficina::ofTextureRenderer, [132](#)
- unreg
  - oficina::ofAnimator, [56](#)
- update
  - oficina::ofAnimator, [56](#)
  - oficina::ofCanvas, [60](#)
  - oficina::ofCanvasManager, [65](#)
  - oficina::ofEntity, [84](#)
  - oficina::ofLua, [99](#)
  - oficina::ofScheme, [104](#)
- UpdateComponents
  - oficina::ofEntity, [84](#)
- use
  - oficina::ofShaderProgram, [116](#)
- write
  - oficina::ofFont, [87](#)
- yieldSpan
  - oficina::ofFrameSpan, [89](#)
  - oficina::ofTimeSpan, [134](#)