OficinaFramework

2.0.12

Generated by Doxygen 1.8.13

Wed Apr 26 2017 17:02:16

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 . . . 2

1.5.1 Building on Linux . . . 2

1.5.1.1 Building Without GNU Guile or Lua . . . 2

1.5.2 Building on Windows . . . 3

1.6 Learn . . . 3

2 IronScheme API Reference v0.3.5b **5**

2.1 General Scheme Syntax . . . 5

2.2 IronScheme Specific Syntax . . . 5

2.2.1 Global symbols . . . 5

2.2.2 Oficina Common API . . . 6

2.2.2.1 Output . . . 6

2.2.2.2 Input . . . 8

2.2.2.3 Display . . . 10

2.2.3 Oficina Entity API . . . 11

2.2.3.1 Referencing objects . . . 11

2.2.3.2 Retrieving components . . . 12

2.2.3.3 Object transformation . . . 12

2.2.4 Oficina Rendering API . . . 15

2.2.4.1 General Rendering API . . . 16

2.2.4.2 Animation API . . . 16

2.3 Usage Guide . . . 19

2.3.1 Basic Example . . . 19

2.3.2 A More Complex Example . . . 19

2.4 IronScheme Example . . . 20

3 IronLua API Reference v0.1.5b **23**

3.1 General Lua Syntax 23

3.2 IronLua Specific Syntax 23

3.2.1 Global symbols 23

3.2.2 Oficina Common API 24

3.2.2.1 Output 24

3.2.2.2 Input 26

3.2.2.3 Display 28

3.2.3 Oficina Entity API 29

3.2.3.1 Referencing objects 29

3.2.3.2 Retrieving components 29

3.2.3.3 Object transformation 30

3.2.4 Oficina Rendering API 33

3.2.4.1 General Rendering API 33

3.2.4.2 Animation API 34

3.3 Usage Guide 36

3.3.1 Basic Example 37

3.3.2 A More Complex Example 37

3.4 IronLua Example 37

4 GameArgs Reference **41**

4.1 What Are GameArgs? 41

4.1.1 General GameArgs 41

4.1.1.1 datad 41

4.1.2 Display GameArgs 42

4.1.2.1 wname 42

4.1.2.2 wicon 42

4.1.2.3 winsz 42

4.1.2.4 frmrt 43

4.1.3 Context GameArgs 43

4.1.3.1 vsync 43

5 Hierarchical Index 45

5.1 Class Hierarchy 45

6 Class Index 47

6.1 Class List 47

7 File Index 49

7.1 File List 49

8 Class Documentation 51

8.1 oficina::ofAnimator Class Reference 51

8.1.1 Detailed Description 52

8.1.2 Member Function Documentation 52

8.1.2.1 draw() 52

8.1.2.2 GetAnimationRunning() 53

8.1.2.3 GetAnimationSpeed() 53

8.1.2.4 GetCurrentAnimationName() 53

8.1.2.5 GetDefaultAnimationSpeed() 54

8.1.2.6 getPosition() 54

8.1.2.7 init() 54

8.1.2.8 isInit() 55

8.1.2.9 reg() 55

8.1.2.10 SetAnimation() 56

8.1.2.11 SetAnimationRunning() 56

8.1.2.12 SetAnimationSpeed() 56

8.1.2.13 SetAnimationTexture() 57

8.1.2.14 setPosition() 57

8.1.2.15 SetRenderer() 57

8.1.2.16 SyncToFrameRate() 58

8.1.2.17 unreg() 58

8.1.2.18 update() 58

8.2 oficina::ofBuffer Class Reference 58

8.2.1 Detailed Description 59
8.2.2 Member Function Documentation 60
8.2.2.1 getName() 60
8.2.2.2 isInit() 60
8.2.2.3 operator=() 60
8.2.2.4 setData() 60
8.3 oficina::ofCanvas Class Reference 61
8.3.1 Detailed Description 61
8.3.2 Member Function Documentation 62
8.3.2.1 init() 62
8.3.2.2 load() 62
8.3.2.3 remove() 62
8.3.2.4 update() 62
8.4 oficina::ofCanvasManager Class Reference 63
8.4.1 Detailed Description 64
8.4.2 Member Enumeration Documentation 64
8.4.2.1 ofDebuggerState 64
8.4.3 Member Function Documentation 64
8.4.3.1 add() 64
8.4.3.2 dbg_callEval() 65
8.4.3.3 dbg_ChangeState() 65
8.4.3.4 dbg_getState() 65
8.4.3.5 dbg_ReplLineNumber() 66
8.4.3.6 dbg_ReplOutStream() 66
8.4.3.7 dbg_setFont() 66
8.4.3.8 draw() 66
8.4.3.9 getCanvasList() 67
8.4.3.10 remove() 67
8.4.3.11 unload() 67
8.4.3.12 update() 67

8.5 oficina::ofContext Class Reference 68
8.5.1 Detailed Description 68
8.5.2 Member Function Documentation 68
8.5.2.1 getViewportSize() 69
8.5.2.2 isInit() 69
8.5.2.3 open() 69
8.5.2.4 pushArg() 69
8.5.2.5 setClearColor() 70
8.5.2.6 setViewportSize() 70
8.6 oficina::ofDisplay Class Reference 70
8.6.1 Detailed Description 71
8.6.2 Member Function Documentation 71
8.6.2.1 close() 71
8.6.2.2 getDeltaTime() 72
8.6.2.3 getHandle() 72
8.6.2.4 getSize() 72
8.6.2.5 isFullscreen() 72
8.6.2.6 isOpen() 73
8.6.2.7 open() 73
8.6.2.8 pushArg() 73
8.6.2.9 setFullscreen() 73
8.6.2.10 setSize() 74
8.6.2.11 setSwapInterval() 74
8.6.2.12 swap() 74
8.7 oficina::ofElementBuffer Class Reference 75
8.7.1 Detailed Description 75
8.7.2 Member Function Documentation 75
8.7.2.1 draw() 75
8.7.2.2 getCount() 76
8.7.2.3 getType() 76

8.7.2.4 setCount() 76
8.7.2.5 setProps() 76
8.7.2.6 setType() 77
8.8 oficina::ofEntity Class Reference 77
8.8.1 Detailed Description 79
8.8.2 Member Function Documentation 79
8.8.2.1 AddComponent() 79
8.8.2.2 draw() 80
8.8.2.3 DrawComponents() 80
8.8.2.4 GetComponent() 80
8.8.2.5 getEulerAngles() 80
8.8.2.6 getModelMatrix() 81
8.8.2.7 getName() 81
8.8.2.8 getPosition() 81
8.8.2.9 getProperty() 81
8.8.2.10 getPropertyMask() 82
8.8.2.11 getScale() 82
8.8.2.12 init() 82
8.8.2.13 load() 83
8.8.2.14 RemoveComponent() 83
8.8.2.15 rotate() 84
8.8.2.16 scale() 84
8.8.2.17 setName() 84
8.8.2.18 setProperty() 85
8.8.2.19 toggleProperty() 85
8.8.2.20 translate() 85
8.8.2.21 update() 86
8.8.2.22 UpdateComponents() 86
8.8.3 Member Data Documentation 86
8.8.3.1 rotation 86

8.8.3.2 scaling 87

8.8.3.3 translation 87

8.9 oficina::ofFont Class Reference 87

8.9.1 Detailed Description 88

8.9.2 Member Function Documentation 88

8.9.2.1 getGlyphSize() 88

8.9.2.2 init() 88

8.9.2.3 isInit() 89

8.9.2.4 measure() 89

8.9.2.5 operator=() 89

8.9.2.6 write() 90

8.10 oficina::ofFrameSpan Class Reference 90

8.10.1 Detailed Description 91

8.10.2 Member Function Documentation 91

8.10.2.1 isRunning() 91

8.10.2.2 resetSpan() 91

8.10.2.3 stop() 91

8.10.2.4 yieldSpan() 92

8.11 oficina::ofIComponent Class Reference 92

8.11.1 Detailed Description 93

8.11.2 Member Function Documentation 93

8.11.2.1 draw() 93

8.11.2.2 getType() 93

8.12 oficina::ofInputState Struct Reference 94

8.12.1 Detailed Description 94

8.13 oficina::ofLua Class Reference 94

8.13.1 Detailed Description 95

8.13.2 Member Function Documentation 96

8.13.2.1 getBoolean() 96

8.13.2.2 getComponent() 96

8.13.2.3 getEntity() 96
8.13.2.4 getInteger() 97
8.13.2.5 getNumber() 97
8.13.2.6 getString() 97
8.13.2.7 isInit() 98
8.13.2.8 loadfile() 98
8.13.2.9 regFunc() 98
8.13.2.10 regSym() [1/6] 99
8.13.2.11 regSym() [2/6] 99
8.13.2.12 regSym() [3/6] 99
8.13.2.13 regSym() [4/6] 100
8.13.2.14 regSym() [5/6] 100
8.13.2.15 regSym() [6/6] 100
8.13.2.16 update() 101
8.14 oficina::ofPrimitive Struct Reference 101
8.14.1 Detailed Description 102
8.15 oficina::ofPrimitiveRenderer Class Reference 102
8.15.1 Detailed Description 102
8.15.2 Member Function Documentation 102
8.15.2.1 draw() 102
8.15.2.2 makePrimitive() 103
8.16 oficina::ofScheme Class Reference 103
8.16.1 Detailed Description 104
8.16.2 Member Function Documentation 104
8.16.2.1 getSymRef() 104
8.16.2.2 isInit() 105
8.16.2.3 loadfile() 105
8.16.2.4 regFunc() 105
8.16.2.5 regSym() 106
8.16.2.6 update() 106

8.17 oficina::ofShader Class Reference 107
8.17.1 Detailed Description 108
8.17.2 Member Function Documentation 108
8.17.2.1 compile() 108
8.17.2.2 getName() 108
8.17.2.3 init() 108
8.17.2.4 isCompiled() 109
8.17.2.5 isInit() 109
8.17.2.6 operator=() 109
8.17.2.7 setSource() 109
8.18 oficina::ofShaderAttribute Class Reference 110
8.18.1 Detailed Description 111
8.18.2 Member Function Documentation 111
8.18.2.1 bindVertexArrayData() 111
8.18.2.2 getSize() 111
8.18.2.3 getStride() 111
8.18.2.4 getType() 112
8.18.2.5 isAutoNormalizing() 112
8.18.2.6 isValid() 112
8.18.2.7 operator=() 112
8.18.2.8 setAutoNormalize() 113
8.18.2.9 setProps() 113
8.18.2.10 setSize() 113
8.18.2.11 setStride() 114
8.18.2.12 setType() 114
8.19 oficina::ofShaderProgram Class Reference 114
8.19.1 Detailed Description 115
8.19.2 Member Function Documentation 115
8.19.2.1 attach() 115
8.19.2.2 attachUnload() 116

8.19.2.3 bindFragmentDataLocation() 116
8.19.2.4 getAttributeLocation() 116
8.19.2.5 getName() 117
8.19.2.6 getUniformLocation() 117
8.19.2.7 isInit() 117
8.19.2.8 isLinked() 118
8.19.2.9 link() 118
8.19.2.10 operator=() 118
8.19.2.11 use() 118
8.20 oficina::ofShaderUniform Class Reference 119
8.20.1 Detailed Description 120
8.20.2 Member Function Documentation 120
8.20.2.1 isValid() 120
8.20.2.2 operator=() 120
8.20.2.3 set() [1/21] 121
8.20.2.4 set() [2/21] 121
8.20.2.5 set() [3/21] 121
8.20.2.6 set() [4/21] 122
8.20.2.7 set() [5/21] 122
8.20.2.8 set() [6/21] 122
8.20.2.9 set() [7/21] 122
8.20.2.10 set() [8/21] 123
8.20.2.11 set() [9/21] 123
8.20.2.12 set() [10/21] 123
8.20.2.13 set() [11/21] 124
8.20.2.14 set() [12/21] 124
8.20.2.15 set() [13/21] 124
8.20.2.16 set() [14/21] 124
8.20.2.17 set() [15/21] 125
8.20.2.18 set() [16/21] 125

8.20.2.19 set() [17/21] 125
8.20.2.20 set() [18/21] 126
8.20.2.21 set() [19/21] 126
8.20.2.22 set() [20/21] 126
8.20.2.23 set() [21/21] 127
8.21 oficina::ofTexture Class Reference 127
8.21.1 Detailed Description 128
8.21.2 Member Function Documentation 128
8.21.2.1 bind() 128
8.21.2.2 getFileName() 128
8.21.2.3 getSize() 128
8.21.2.4 isLoaded() 129
8.21.2.5 operator()() 129
8.21.2.6 operator=() 129
8.21.2.7 unbind() 129
8.22 oficina::ofTexturePool Class Reference 130
8.22.1 Detailed Description 130
8.22.2 Member Function Documentation 130
8.22.2.1 load() [1/2] 130
8.22.2.2 load() [2/2] 131
8.22.2.3 loadDefaultFont() 131
8.22.2.4 unload() 131
8.23 oficina::ofTextureRenderer Class Reference 133
8.23.1 Detailed Description 133
8.23.2 Member Function Documentation 133
8.23.2.1 init() 133
8.23.2.2 isInit() 134
8.23.2.3 operator=() 134
8.23.2.4 render() 134
8.23.2.5 SetTexture() 135

8.23.2.6 unload() 135

8.24 oficina::ofTimeSpan Class Reference 136

8.24.1 Detailed Description 136

8.24.2 Member Function Documentation 136

8.24.2.1 isRunning() 136

8.24.2.2 resetSpan() 137

8.24.2.3 stop() 137

8.24.2.4 yieldSpan() 137

8.25 oficina::ofVertexArray Class Reference 137

8.25.1 Detailed Description 138

8.25.2 Member Function Documentation 138

8.25.2.1 draw() 138

8.25.2.2 operator=() 139

8.26 oficina::ofVertexBuffer Class Reference 139

8.26.1 Detailed Description 139

9 File Documentation 141

9.1 benchmark.hpp File Reference 141

9.1.1 Detailed Description 141

9.1.2 Function Documentation 141

9.1.2.1 ofBenchmarkIsRunning() 142

9.1.2.2 ofBenchmarkStart() 142

9.2 benchmark.hpp 142

9.3 canvas.hpp File Reference 142

9.3.1 Detailed Description 143

9.4 canvas.hpp 143

9.5 display.hpp File Reference 144

9.5.1 Detailed Description 145

9.5.2 Enumeration Type Documentation 145

9.5.2.1 ofFrameRateConfig 145

9.6 display.hpp 145

9.7 entity.hpp File Reference 146
9.7.1 Detailed Description 147
9.8 entity.hpp 147
9.9 input.hpp File Reference 148
9.9.1 Detailed Description 150
9.9.2 Enumeration Type Documentation 150
9.9.2.1 ofMouseButton 150
9.9.2.2 ofPadButton 151
9.9.2.3 ofPlayer 152
9.9.2.4 ofStick 152
9.9.2.5 ofStickAxis 152
9.9.2.6 ofStickSignal 153
9.9.3 Function Documentation 153
9.9.3.1 ofButtonPress() 153
9.9.3.2 ofButtonTap() 154
9.9.3.3 ofGetInputState() 154
9.9.3.4 ofGetLeftStick() 155
9.9.3.5 ofGetLeftTrigger() 155
9.9.3.6 ofGetMousePos() 156
9.9.3.7 ofGetRightStick() 156
9.9.3.8 ofGetRightTrigger() 156
9.9.3.9 ofGetTextInput() 157
9.9.3.10 ofIsGamepadConnected() 157
9.9.3.11 ofIsInputtingText() 157
9.9.3.12 ofMapButtonRemove() 157
9.9.3.13 ofMapDefaultsP1() 158
9.9.3.14 ofMapKeyToButton() 158
9.9.3.15 ofMapKeyToStick() 159
9.9.3.16 ofMappingClear() 159
9.9.3.17 ofMapStickRemove() 159

9.9.3.18 ofMouseButtonPress() 160
9.9.3.19 ofMouseButtonTap() 160
9.9.3.20 ofSetTextInput() 161
9.9.3.21 ofStartTextInput() 161
9.9.3.22 ofStickMovedTowards() 161
9.9.3.23 ofStopTextInput() 162
9.9.3.24 ofTextInputSetPadding() 162
9.9.3.25 ofUpdateEventDispatch() 163
9.10 input.hpp 163
9.11 io.hpp File Reference 165
9.11.1 Detailed Description 166
9.11.2 Enumeration Type Documentation 166
9.11.2.1 ofLogLvl 166
9.11.2.2 ofLogType 167
9.11.3 Function Documentation 167
9.11.3.1 ofFindAsset() 167
9.11.3.2 ofLoadImage() 167
9.11.3.3 ofLoadText() 168
9.11.3.4 ofLog() 168
9.11.3.5 ofLogGetType() 169
9.11.3.6 ofLogSetLevel() 169
9.11.3.7 ofLogUseFile() 169
9.11.3.8 ofSetDataDirectoryName() 169
9.12 io.hpp 170
9.13 oficina.hpp File Reference 171
9.13.1 Detailed Description 172
9.13.2 Enumeration Type Documentation 172
9.13.2.1 ofReplType 172
9.13.3 Function Documentation 173
9.13.3.1 ofGameLoop() 173

9.13.3.2 ofGetReplType() 173
9.13.3.3 ofGetWindowSize() 173
9.13.3.4 ofInit() [1/2] 173
9.13.3.5 ofInit() [2/2] 174
9.13.3.6 ofIsFullscreen() 174
9.13.3.7 ofQuit() 174
9.13.3.8 ofQuitFlagRaised() 174
9.13.3.9 ofReplEval() 174
9.13.3.10 ofSetClearColor() 175
9.13.3.11 ofSetFullscreen() 175
9.13.3.12 ofSetReplType() 175
9.13.3.13 ofSetSwapInterval() 176
9.13.3.14 ofSetWindowSize() 176
9.13.3.15 ofSoftStop() 176
9.14 oficina.hpp 177
9.15 oflua.hpp File Reference 177
9.15.1 Detailed Description 178
9.15.2 Function Documentation 179
9.15.2.1 ofLuaDefineFunc() 179
9.15.2.2 ofLuaDefineSymbol() [1/6] 179
9.15.2.3 ofLuaDefineSymbol() [2/6] 179
9.15.2.4 ofLuaDefineSymbol() [3/6] 180
9.15.2.5 ofLuaDefineSymbol() [4/6] 180
9.15.2.6 ofLuaDefineSymbol() [5/6] 180
9.15.2.7 ofLuaDefineSymbol() [6/6] 181
9.15.2.8 ofLuaEval() 181
9.15.2.9 ofLuaGetBoolean() 181
9.15.2.10 ofLuaGetComponent() 182
9.15.2.11 ofLuaGetEntity() 182
9.15.2.12 ofLuaGetInteger() 183

9.15.2.13 ofLuaGetNumber() 183
9.15.2.14 ofLuaGetString() 184
9.15.2.15 ofLuaIsInit() 184
9.15.2.16 ofLuaUndefine() 184
9.16 oflua.hpp 185
9.17 ofscheme.hpp File Reference 186
9.17.1 Detailed Description 187
9.17.2 Function Documentation 187
9.17.2.1 ofScmDefineFunc() 187
9.17.2.2 ofScmDefineSymbol() [1/3] 188
9.17.2.3 ofScmDefineSymbol() [2/3] 188
9.17.2.4 ofScmDefineSymbol() [3/3] 188
9.17.2.5 ofScmEval() 189
9.17.2.6 ofScmGetReference() 189
9.17.2.7 ofScmIsInit() 189
9.17.2.8 ofScmUndefine() 190
9.18 ofscheme.hpp 191
9.19 platform.hpp File Reference 191
9.19.1 Detailed Description 192
9.20 platform.hpp 193
9.21 render.hpp File Reference 194
9.21.1 Detailed Description 196
9.21.2 Enumeration Type Documentation 196
9.21.2.1 ofBufferUsage 196
9.21.2.2 ofContextType 197
9.21.2.3 ofDataType 197
9.21.2.4 ofFontFaces 198
9.21.2.5 ofPrimitiveType 198
9.21.2.6 ofShaderType 198
9.21.3 Function Documentation 199

9.21.3.1 ofLoadDefaultFragShader() 199
9.21.3.2 ofLoadDefaultShaderProgram() 199
9.21.3.3 ofLoadDefaultVertexShader() 199
9.21.3.4 ofSetVSync() 199
9.21.4 Variable Documentation 200
9.21.4.1 ofDefaultShaderSrc_FS 200
9.21.4.2 ofDefaultShaderSrc_VS 200
9.22 render.hpp 201
9.23 timer.hpp File Reference 207
9.23.1 Detailed Description 207
9.24 timer.hpp 207
9.25 types.hpp File Reference 208
9.25.1 Detailed Description 209
9.25.2 Function Documentation 209
9.25.2.1 ofClamp() 209
9.26 types.hpp 209

Index **211**

Chapter 1

Oficina Framework

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 $>$= 2.0.5
- SDL2_Image $>$= 2.0.0
- OpenGL 3.3 support or higher
- GLEW $>$= 2.0.0
- GL Mathematics (GLM) $>$= 0.9.8
- (Optional) GNU Guile $>$= 2.0.11
- (Optional) Lua $>$= 5.3

1.4 Building Your Own Projects With Oficina2

Building your own game, which uses Oficina2, is made really easy by using the program `oficina2-config`, which is provided when building Oficina. This program offers three arguments, to be used when invoked:

- `--version`: Shows Oficina's build version;
- `--cppflags`: Shows the flags needed to build a C++ game or application based on Oficina;
- `--libs`: Shows the flags to link libraries on a C++ game or application based on Oficina.

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

```
g++ game.cpp `oficina2-config --cppflags` `oficina2-config --libs` -o MyGameBinary
```

If you're using Windows, you may want to simply catch the output of commands `oficina2-config --cppflags` and `oficina2-config --libs` and compile your program in that order.

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.

1.5 Building

1.5.1 Building on Linux

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.

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

1.5.1.1 Building Without GNU Guile or Lua

1.5.1.1.1 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:

```
cmake .. -DNO_SCHEME=on
```

1.5.1.1.2 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:

```
cmake .. -DNO_LUA=on
```

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

1.5.1.1.3 Without Both

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

```
cmake .. -DNO_SCHEME=on -DNO_LUA=on
```

1.5.2 Building on Windows

Building Oficina2 on Windows is easy, but is not as intuitive as with a Unix system.

Currently, Oficina2 supports MinGW (32-bit) and, to compile it with CMake, you'll need to make sure you:

- Have the proper library dependencies installed on your MinGW path;
- Have `mingw32-make` installed on your MinGW installation (you can easily select this package on mingw-get);
- Make sure you have installed Git without `cmd` support, only with Git Bash (or CMake will give an error regarding Git's own `sh`).

Because of the last situation, you also probably won't be able to compile using a Git Bash or Cygwin.

After that, open the Command Prompt, `cd` to the folder you have cloned this repository, then build the program:

```
mkdir build
cd build
cmake -G "MinGW Makefiles" ..
make
```

If you're running the Command Prompt in Administrator Mode, you can use `make install` to directly install OficinaFramework in the folder `C:\Program Files (x86)\OficinaFramework2`.

For selectively building with or without Lua, please refer to the Linux section.

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), which does not work well with Windows.

It is also advisable to add the directory `C:\Program Files (x86)\OficinaFramework2\bin` to your PATH so you can use Oficina2's config tool and test tool.

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

number	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

number	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

typename-enum	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

player	(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

player	(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

which	Specification for which trigger should be compared. You may use :left-trigger or :right-trigger.
player	(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

which	Button to be compared. All atoms similar to :pad-start, :pad-back and etcetera may be used.
player	(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

which	Button to be compared. All atoms similar to :pad-start, :pad-back and etcetera may be used.
player	(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

which	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

which	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

state	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::ofIComponent.

See also

oficina::ofIComponent

2.2.3.2.1 get-component

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

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

Parameters

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

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

compref	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

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

theta	Angle of rotation, in radians.
axis	List describing the rotation axis in X, Y and Z axis.
load-identity	(Optional) Whether the rotation matrix must be reset before rotating. Defaults to #f.
objref	(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

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

2.2.3.3.4 pos?

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

Gets an object's position.

Parameters

objref	(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

axis	Desired axis coordinate (:x, :y or :z) of rotation to reference.
objref	(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

axis	Desired axis coordinate (:x, :y or :z) of magnification to reference.
objref	(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

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

2.2.3.3.8 proptog!

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

Toggles a specific property's state.

Parameters

which	Property index, ranging from 0 to 31.
objref	(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

which	Property index, ranging from 0 to 31.
objref	(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

objref	(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

color	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 oficina::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

animator	Reference to an oficina::ofAnimator component.
name	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

animator	Reference to an oficina::ofAnimator component.
speed	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

animator	Reference to an oficina::ofAnimator component.
state	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

animator	Reference to an oficina::ofAnimator component.
position	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

animator	Reference to an oficina::ofAnimator 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

animator	Reference to an oficina::ofAnimator 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

animator	Reference to an oficina::ofAnimator 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

animator	Reference to an oficina::ofAnimator 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

animator	Reference to an oficina::ofAnimator 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 fullfill 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

format	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

number	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

typename	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

player	(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

player	(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

which	Specification for which trigger should be compared. You may use LEFT_TRIGGER or RIGHT_TRIGGER.
player	(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

which	Button to be compared. All values similar to PAD_START, PAD_BACK and etcetera may be used.
player	(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

which	Button to be compared. All values similar to PAD_START, PAD_BACK and etcetera may be used.
player	(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

which	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

which	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

state	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

Moust 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::ofIComponent.

See also

oficina::ofIComponent

3.2.3.2.1 getComponent

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

Retrieves a component from a specified entity or from this.

Parameters

name	Name of the component, as registered on the Entity.
objref	(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

compref	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

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

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

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

3.2.3.3.4 getPosition

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

Gets an object's position.

Parameters

objref	(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

axis	Desired axis coordinate (X, Y or Z) of rotation to reference.
objref	(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

axis	Desired axis coordinate (X, Y or Z) of magnification to reference.
objref	(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

which	Property index, ranging from 1 to 32, as per Lua's 1-based numbering.
state	Active (true) or inactive (false).
objref	(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

which	Property index, ranging from 1 to 32, as per Lua's 1-based numbering.
objref	(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

which	Property index, ranging from 1 to 32, as per Lua's 1-based numbering.
objref	(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

objref	(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

color	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

animator	Reference to an oficina::ofAnimator component.
name	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

animator	Reference to an oficina::ofAnimator component.
speed	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

animator	Reference to an oficina::ofAnimator component.
state	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

animator	Reference to an oficina::ofAnimator component.
position	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

animator	Reference to an oficina::ofAnimator 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

animator	Reference to an oficina::ofAnimator component.

Returns

A numeric value representing the default animation speed.

3.2.4.2.7 getPosition

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

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

Parameters

animator	Reference to an oficina::ofAnimator 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(animator)
```

Gets whether the animation is currently running.

Parameters

animator	Reference to an oficina::ofAnimator component.

Returns

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

3.2.4.2.9 getName

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

Gets the name of the currently playing animation.

Parameters

animator	Reference to an oficina::ofAnimator 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 acessing 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, dwe 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",
        "frmrt=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

DirName	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

Name	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

path	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

size	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 frmrt

```
frmrt=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

config	Configuration string for the framerate. This argument is case-insensitive. • If you wish to let the window swap freely, just use the literal VARIABLE; • 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 C; • 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 deltaTime), input the value (floating point or integer) with the suffix M. • You can also fix a capped FPS and a minimum FPS by inputting two values, each one with a suffix C or M.

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

state	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 58
  - oficina::ofElementBuffer 75
  - oficina::ofVertexBuffer 139
- oficina::ofCanvas 61
- oficina::ofCanvasManager 63
- oficina::ofContext 68
- oficina::ofDisplay 70
- oficina::ofEntity 77
- oficina::ofFont 87
- oficina::ofFrameSpan 90
- oficina::ofIComponent 92
  - oficina::ofAnimator 51
  - oficina::ofLua 94
  - oficina::ofScheme 103
- oficina::ofInputState 94
- oficina::ofPrimitive 101
- oficina::ofPrimitiveRenderer 102
- oficina::ofShader 107
- oficina::ofShaderAttribute 110
- oficina::ofShaderProgram 114
- oficina::ofShaderUniform 119
- oficina::ofTexture 127
- oficina::ofTexturePool 130
- oficina::ofTextureRenderer 133
- oficina::ofTimeSpan 136
- oficina::ofVertexArray 137

Chapter 6

Class Index

6.1 Class List

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

oficina::ofAnimator
Tool for controlling a texture renderer to generate animations 51
oficina::ofBuffer
Specifies a generic buffer. Override this class to create your own buffers 58
oficina::ofCanvas
Default interface for creating and managing canvases 61
oficina::ofCanvasManager
Static class for handling canvases in general 63
oficina::ofContext
Describes a context for your display 68
oficina::ofDisplay
Represents a single window prepared for receiving a context 70
oficina::ofElementBuffer
Represents an Element Buffer object (EBO), useful for holding sequences of vertices for drawing on screen 75
oficina::ofEntity
Abstract class representing one ingame entity 77
oficina::ofFont
Represents a font 87
oficina::ofFrameSpan
Tool for counting and comparing frames, depending of the game's time variation 90
oficina::ofIComponent
Defines a single component to be attached to an entity 92
oficina::ofInputState
Holds an input state every frame 94
oficina::ofLua
Defines one Lua environment to be used inside an entity 94
oficina::ofPrimitive
A structure representing a primitive. Can be used for rendering 101
oficina::ofPrimitiveRenderer
A static class containing methods for creating and drawing simple primitives onscreen 102
oficina::ofScheme
Defines one Scheme environment to be used inside an entity 103
oficina::ofShader
Describes a shader 107

oficina::ofShaderAttribute
Represents the location of an attribute for the program shader 110
oficina::ofShaderProgram
Represents a shader program 114
oficina::ofShaderUniform
Represents and handles a shader's uniform 119
oficina::ofTexture
Represents a texture on the GPU 127
oficina::ofTexturePool
Static object for managing textures. Most (if not all) textures should be loaded using this tool . 130
oficina::ofTextureRenderer
Tool for easily rendering 2D textures or texture atlases 133
oficina::ofTimeSpan
Tool for counting and compare fixed amounts of time, independent from the game's time variation 136
oficina::ofVertexArray
Represents a vertex array for binding shader and vertex data 137
oficina::ofVertexBuffer
Represents a Vertex Buffer object (VBO). Use this to hold data related to drawing 139

Chapter 7

File Index

7.1 File List

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

benchmark.hpp
Oficina's default benchmarking utilities 141
canvas.hpp
Tools for creating game scenes and manage such scenes 142
display.hpp
Tools for configuring windows for video output 144
entity.hpp
Interfaces and tools for managing objects ingame 146
input.hpp
Special tools for handling player input 148
io.hpp
Tools for handling non-player-related input and output 165
oficina.hpp
Default tools for easily initializing Oficina 171
oflua.hpp
Tools for object scripting and for the Repl, in Lua language 177
ofscheme.hpp
Tools for object scripting and for the Repl, in Scheme language 186
platform.hpp
Definitions for the platform currently executing the game 191
render.hpp
Tools and classes for rendering inside a context 194
timer.hpp
Tools for counting and processing time-related events 207
types.hpp
Tools for predefining default types and math tools used by OficinaFramework 208

Chapter 8

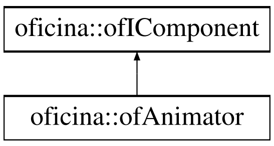
Class Documentation

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 887 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

ViewProjection	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::ofIComponent.

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::ofIComponent.

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

animName	Desired animation name.
nFrames	amount of frames on the animation.
animFrames	Pointer to an array containing all animation frames, numbered.
speed	Speed of the animation. Animation speed handling changes depending on the syncing type. • 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. • 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.
loops	Optionally set the animation to loop, jumping to the looping frame.
loopBackTo	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

animName	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

state	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

spd	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

t	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

pos	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

renderer	Instantiated Texture Renderer to be used.
manage	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

state	State of syncing. Defaults to true.

8.1.2.17 unreg()

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

Unregisters an animation.

Parameters

animName	Desired animation to unregister.

8.1.2.18 update()

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

Updates the animation step.

Parameters

dt	Time difference, in seconds, from the last drawn frame to the currently drawn frame (delta time).

Implements oficina::ofIComponent.

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

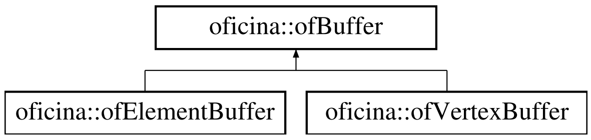
- render.hpp

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

- ofBuffer ()

  Constructor.
- ofBuffer (const ofBuffer &)

  Copy constructor.
- 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

other	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

dataSize	Size of the data to be fed, in bytes.
data	Pointer to the beginning of data.
usage	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

dt	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::ostringstream & dbg_ReplOutStream ()

  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

ofDebuggerVars	Disabled.
ofDebuggerRepl	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

c	Pointer to the newly-initialized canvas.
depth	Optional canvas depth.
name	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_ReplOutStream()

```
static std::ostringstream& oficina::ofCanvasManager::dbg_ReplOutStream ( )  [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

fontFace	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::get↩
CanvasList ( ) [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

c	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

dt	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

type	Type of context. Currently, only OpenGL is supported.
hwnd	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

arg	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

color	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

sz	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

arg	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

state	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

cfg	Desired swap interval configuration.
max	Maximum swap interval, in frames per second (FPS). If applicable.
min	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:

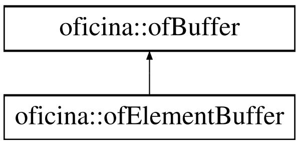
- display.hpp

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 303 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

mode	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

count	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

count	Amount of elements.
type	Type of data.

8.7.2.6 setType()

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

Defines the type of data fed to the object.

Parameters

type	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 overriden).

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

name	Name of the component to be added.
component	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

ViewProjection	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 overriden).

Parameters

ViewProjection	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

name	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

which	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

name	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

theta	Angle to rotate the entity, in radians.
axis	Axis of the Euler rotation.
loadIdentity	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

amount	3D Vector containing how much should the object be scaled. Use positive numbers to scale up, and negative to scale down.
loadIdentity	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

name	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

which	A property, ranging from 0 to 31.
state	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

which	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

coord	3D Vector containing the coordinates for the object.
loadIdentity	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

dt	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

dt	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

- ofFont ()
  Constructor.
- ofFont (const ofFont &)
  Copy constructor.
- 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 834 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

fontTexture	Texture atlas containing the font characters.
glyphSize	2D unsigned integer vector containing the size of each glyph frame on the atlas.
manageTexture	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

text	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

other	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

text	Text to be written.
position	Position of the first text glyph (centered) on the matrix.
mvp	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

color	4D vector specifying which color the textshould 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:

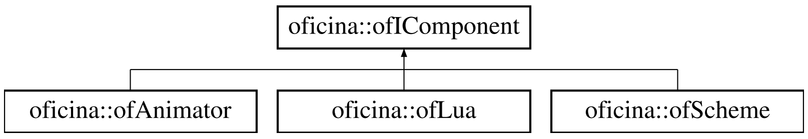
- timer.hpp

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

ViewProjection	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::of↩ IComponent".

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:

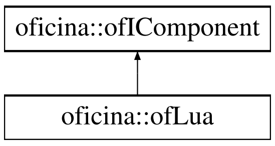
- input.hpp

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::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.
- 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::ofIComponent ∗ 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

symbol	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

symbol	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

symbol	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

symbol	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

symbol	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

symbol	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

filename	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

symbol	Name of the function to be defined.
fun	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

symbol	Name of the variable to be defined.
value	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

symbol	Name of the variable to be defined.
value	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

symbol	Name of the variable to be defined.
value	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

symbol	Name of the variable to be defined.
value	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

symbol	Name of the variable to be defined.
value	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

symbol	Name of the variable to be defined.
value	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

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

Implements oficina::ofIComponent.

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 1018 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 1038 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

p	Primitive to be rendered.
color	A vector specifying the color of the primitive to be rendered. Should contain RGBA values.
mvp	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

type	Type of the primitive.
verticesAmount	Number of vertices to be used on creation.
verticesSize	Total size, in bytes, of the vector array to be fed on the next argument.
vertices	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:

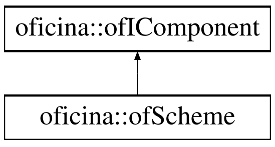
- render.hpp

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

symbol	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

module	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

filename	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

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

See also

SCHEME_FUNCAST

Parameters

n_optional_params	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

symbol	Name of the variable to be defined.
value	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

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

Implements oficina::ofIComponent.

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

- ofShader ()
  Constructor.
- ofShader (const ofShader &)
  Copy constructor.
- 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 345 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

type	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

shader	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

- ofShaderAttribute ()

  Constructor.
- ofShaderAttribute (const ofShaderAttribute &)

  Copy constructor.
- 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 394 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

byteOffset	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

attr	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

state	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

size	Size of the attribute.
type	Type of attribute data.
stride	Stride of the attribute on vertex data.
normalize	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

s	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

stride	Stride of the attribute.

8.18.2.12 setType()

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

Defines the type of data of the attribute.

Parameters

t	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

- ofShaderProgram ()

  Constructor.
- ofShaderProgram (const ofShaderProgram &)

  Copy constructor.
- 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 563 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

shader	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

shader	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

name	Name of the data location.
colorNumber	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

name	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

name	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

program	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

- ofShaderUniform ()
  Constructor.
- ofShaderUniform (const ofShaderUniform &)
  Copy constructor.
- 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 464 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

uniform	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

value	Float value.

8.20.2.4 set() [2/21]

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

Sets the value of the uniform.

Parameters

value	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

value	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

value	4D vector of float.

8.20.2.7 set() [5/21]

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

Sets the value of the uniform.

Parameters

value	Signed integer value.

8.20.2.8 set() [6/21]

```
void oficina::ofShaderUniform::set (
             glm::ivec2 value )
```

Sets the value of the uniform.

Parameters

value	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

value	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

value	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

value	Unsigned integer value.

8.20.2.12 set() [10/21]

```
void oficina::ofShaderUniform::set (
             glm::uvec2 value )
```

Sets the value of the uniform.

Parameters

value	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

value	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

value	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

value	2x2 matrix of float.
transpose	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

value	3x3 matrix of float.
transpose	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

value	4x4 matrix of float.
transpose	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

value	2x3 matrix of float.
transpose	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

value	3x2 matrix of float.
transpose	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

value	2x4 matrix of float.
transpose	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

value	4x2 matrix of float.
transpose	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

value	3x4 matrix of float.
transpose	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

value	4x3 matrix of float.
transpose	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

- ofTexture ()
  Constructor.
- ofTexture (const ofTexture &)
  Copy constructor.
- 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 isLoaded () 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 672 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

currentSampler	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

other	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

currentSampler	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 739 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

filename	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

surf	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

fontface	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

t	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

- ofTextureRenderer ()

  Constructor.
- ofTextureRenderer (const ofTextureRenderer &)

  Copy constructor.
- 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 764 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

t	Reference to the texture.
frameSize	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 reat 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

other	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

position	Position of the texture (centered) on the matrix.
mvp	Model-View-Projection matrix to be used when drawing the texture.
frame	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.
color	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

t	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

- ofVertexArray ()
  Constructor.
- ofVertexArray (const ofVertexArray &)
  Copy constructor.
- 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 633 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

mode	Primitive to be drawn.
firstVertexIdx	Index of the first vertex to be used.
vertexCount	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

other	Vertex array to be cloned.

Returns

Reference to this vertex array.

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

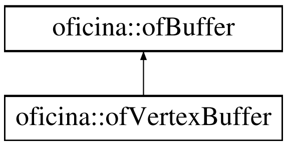
- render.hpp

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 293 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

spanTimeS	Time between each benchmark debriefing.

9.2 benchmark.hpp

```
/********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                    *
 *                                                                *
 *  OficinaFramework is free software: you can redistribute       *
 *  it and/or modify it under the terms of the GNU Lesser         *
 *  General Public License as published by the Free Software      *
 *  Foundation, either version 3 of the License, or (at your      *
 *  option) any later version.                                    *
 *                                                                *
 *  You should have received a copy of the GNU Lesser General     *
 *  Public License along with OficinaFramework.  If not, see      *
 *  <http://www.gnu.org/licenses/>.                               *
 ********************************************************************/

#pragma once

#include <string>

namespace oficina
{
    void ofBenchmarkStart(float spanTimeS);

    void ofBenchmarkUpdateCall();

    void ofBenchmarkEnd();

    bool ofBenchmarkIsRunning();
}
```

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

```
/********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                      *
 *                                                                  *
 *  OficinaFramework is free software: you can redistribute         *
 *  it and/or modify it under the terms of the GNU Lesser           *
 *  General Public License as published by the Free Software        *
 *  Foundation, either version 3 of the License, or (at your        *
 *  option) any later version.                                      *
 *                                                                  *
 *  You should have received a copy of the GNU Lesser General       *
 *  Public License along with OficinaFramework.  If not, see        *
 *  <http://www.gnu.org/licenses/>.                                 *
 ********************************************************************/

#pragma once
#include <list>
#include <queue>
#include <sstream>

#include "oficina2/types.hpp"
#include "oficina2/render.hpp"

namespace oficina
{
    // Pre-definition of ofCanvasManager so we can refer to it from ofCanvas.
    class ofCanvasManager;
    class ofCanvas
    {
        friend class ofCanvasManager;
    private:
        bool            m_init    = false;
        bool            m_load    = false;
        bool           m_remove = false;
        int            depth     = 0;
        std::string    m_name = "";
    public:
        virtual ~ofCanvas() {}
        virtual void init() = 0;
        virtual void load() = 0;
        virtual void unload() = 0;
        virtual void update(float dt) = 0;
        virtual void draw() = 0;
        virtual void remove() final;
    };

    class ofCanvasManager
    {
    public:
        enum ofDebuggerState
        {
            ofDebuggerOff  = 0u,
            ofDebuggerVars = 1u,
            ofDebuggerRepl = 2u
        };

        static void init();
        // TODO: Explain canvas depth in documentation
        static void add(ofCanvas* c, int depth = 0, std::string name = "");
        static void remove(ofCanvas* c);
        static void unload();
        static void update(float dt);
        static void draw();

        static std::vector<std::pair<std::string, std::string>> getCanvasList();

        static std::ostringstream& dbg_ReplOutStream();
        static ofDebuggerState     dbg_getState();
        static void                dbg_callEval();
        static void                dbg_ChangeState();
        static void                dbg_ReplHistoryPrev();
        static void                dbg_ReplHistoryNext();
        static ofdword             dbg_ReplLineNumber();
        static void                dbg_setFont(oficina::ofFontFaces fontFace);
    private:
        class ofDebugCanvas : public ofCanvas
        {
        public:
            void init();
            void load();
            void unload();
            void update(float dt);
            void draw();
        };

        static ofDebugCanvas m_debugger;
    };
}
```

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

ofFPSVariable	Freely allows FPS to float.
ofFPSCapped	Caps FPS to a maximum number. This will ignore further rendering frames after the nth frame is rendered.
ofFPSMinimum	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.
ofFPSBoth	Induces FPS to assume both behaviours of Capped and Minimum.

Definition at line 36 of file display.hpp.

9.6 display.hpp

```
/**********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>   *
 *  This file is part of OficinaFramework v2.x                      *
 *                                                                   *
 *  OficinaFramework is free software: you can redistribute         *
 *  it and/or modify it under the terms of the GNU Lesser           *
 *  General Public License as published by the Free Software      *
 *  Foundation, either version 3 of the License, or (at your      *
 *  option) any later version.                                    *
 *                                                                *
 *  You should have received a copy of the GNU Lesser General     *
 *  Public License along with OficinaFramework.  If not, see      *
 *  <http://www.gnu.org/licenses/>.                               *
 ******************************************************************/

#pragma once

#include <SDL2/SDL.h>
#include <list>
#include <string>
#include "oficina2/types.hpp"
#include "oficina2/timer.hpp"

namespace oficina
{
    // a display may assume when rendering the game.
    enum ofFrameRateConfig {
        ofFPSVariable = 0x00,
        ofFPSCapped   = 0x01,
        ofFPSMinimum  = 0x02,
        ofFPSBoth     = 0x04
    };

    class ofDisplay
    {
    public:
        void        pushArg(std::string arg);
        void        open();
        void        close();
        void        swap();

        SDL_Window* getHandle() const;
        glm::uvec2  getSize() const;
        bool        isOpen() const;

        void        setSize(glm::uvec2 NewSize);
        void        setSwapInterval(ofFrameRateConfig cfg, float max = 0.0f
, float min = 0.0f);

        bool isFullscreen() const;
        void setFullscreen(bool state);
        float getDeltaTime() const;
    private:
        SDL_Window*             m_wnd = nullptr;
        std::list<std::string> m_confv;
        std::string             m_title = "OficinaFramework 2.0";
        std::string             m_iconpath;
        glm::uvec2              m_size = glm::uvec2(1280u, 720u);
        bool                    m_full = false;
        ofTimeSpan              m_dtSpan;
        float                   m_dt = 0.0f;
        ofFrameRateConfig       m_frmrtt = ofFPSVariable;
        float                   m_frmrt_min = 0.0f;
        float                   m_frmrt_max = 0.0f;
        void                    parseArgs();
    };
}
```

9.7 entity.hpp File Reference

Interfaces and tools for managing objects ingame.

```
#include "oficina2/types.hpp"
#include <map>
```

Classes

- class oficina::ofIComponent

  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

```
/********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                     *
 *                                                                 *
 *  OficinaFramework is free software: you can redistribute        *
 *  it and/or modify it under the terms of the GNU Lesser          *
 *  General Public License as published by the Free Software       *
 *  Foundation, either version 3 of the License, or (at your       *
 *  option) any later version.                                     *
 *                                                                 *
 *  You should have received a copy of the GNU Lesser General      *
 *  Public License along with OficinaFramework.  If not, see       *
 *  <http://www.gnu.org/licenses/>.                                *
 ********************************************************************/

#pragma once

#include "oficina2/types.hpp"
#include <map>

namespace oficina
{
    class ofEntity;
    class ofIComponent
    {
        friend class ofEntity;
    public:
        virtual ~ofIComponent()       {}
        virtual void init()           = 0;
        virtual void load()           {}
        virtual void unload()         {}
        virtual void update(float dt) = 0;
        virtual void draw(glm::mat4 ViewProjection) {}
        std::string getType() const;
    protected:
        ofEntity* parent;
        void parseType();
    private:
        std::string m_type = "oficina::ofIComponent";
    };

    class ofEntity
    {
    public:
        virtual ~ofEntity() {}
        virtual void init()           = 0;
        virtual void load()           = 0;
        virtual void unload()         = 0;
        virtual void update(float dt) = 0;
        virtual void draw(glm::mat4 ViewProjection) = 0;

        void translate(glm::vec3 coord,
                       bool loadIdentity = false);
        void rotate(float theta,
                    glm::vec3 axis,
                    bool loadIdentity = false);
        void scale(glm::vec3 amount,
                   bool loadIdentity);

        void setProperty(ofbyte which, bool state);
        void toggleProperty(ofbyte which);
        void setName(std::string name);

        glm::mat4 getModelMatrix();
        glm::vec3 getPosition() const;
        glm::vec3 getEulerAngles() const;
        glm::vec3 getScale() const;
        bool getProperty(ofbyte which);
        ofdword getPropertyMask() const;
        std::string getName() const;

        void AddComponent(std::string name, ofIComponent* component);
        ofIComponent* const GetComponent(std::string name);
        void RemoveComponent(std::string name);
        void ClearComponents();

        void UpdateComponents(float dt);
        void DrawComponents(glm::mat4 ViewProjection);
    protected:
        glm::mat4 translation;
        glm::mat4 rotation;
        glm::mat4 scaling;

        glm::vec3 position;
        glm::vec3 eulerangles;
        glm::vec3 magnification = glm::vec3(1.0f);

        ofdword   propertymask = 0x00000000u;

        std::map<std::string, ofIComponent*> components;

        std::string name = "[unnamed]";
    };
}
```

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::of↩
PadDRight = 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::of↩
MouseRight = 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=ofPlayer↩
One)

  Binds a specific keyboard key to a gamepad button.

- void oficina::ofMapKeyToStick (ofbyte stickPositionMask, SDL_Scancode scancode, ofPlayer player=of↩ PlayerOne)

    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). **Note** 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). **Note** 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

button	Which gamepad button should be compared.
player	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

button	Which gamepad button should be compared.
player	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

player	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

player	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

player	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

player	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

player	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

player	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

button	Desired button to remove mappings.
player	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

button	Desired button to map.
scancode	SDL_Scancode for the key to be mapped. Check SDL2's documentation to see all available scancodes.
player	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

stickPositionMask	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: `ofMapKeyToStick(ofStickLeft | ofStickHoriz | ofStickNegative, SDL_SCANCODE_LEFT, ofPlayerOne );`
scancode	SDL_Scancode for the key to be mapped. Check SDL2's documentation to see all available scancodes.
player	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

player	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

stickPositionMask	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: `ofMapStickRemove(ofStickLeft | ofStickHoriz | ofStickNegative, ofPlayerOne);`
player	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

button	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

button	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

str	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

stickDirectionMask	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: `bool lstickMovedLeft = ofStickMovedTowards(ofStickLeft | ofStickHoriz | ofStickNegative) ;`
player	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

padding	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

```
/*******************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                    *
 *                                                                *
 *  OficinaFramework is free software: you can redistribute       *
 *  it and/or modify it under the terms of the GNU Lesser         *
 *  General Public License as published by the Free Software      *
 *  Foundation, either version 3 of the License, or (at your      *
 *  option) any later version.                                    *
 *                                                                *
 *  You should have received a copy of the GNU Lesser General     *
 *  Public License along with OficinaFramework.  If not, see      *
 *  <http://www.gnu.org/licenses/>.                               *
 ******************************************************************/

#pragma once

#include "oficina2/types.hpp"
#include <SDL2/SDL.h>
#include <string>

namespace oficina
{
    enum ofStick
    {
        ofStickLeft  = 0x01u,
        ofStickRight = 0x02u
    };

    enum ofStickAxis
    {
        ofStickHoriz = 0x04u,
        ofStickVert  = 0x08u
    };

    enum ofStickSignal
    {
        ofStickNegative = 0x10u,
        ofStickPositive = 0x20u
    };

    enum ofPadButton
    {
        ofPadStart  = 0x0001u,
        ofPadBack   = 0x0002u,
        ofPadA      = 0x0004u,
        ofPadB      = 0x0008u,
        ofPadX      = 0x0010u,
        ofPadY      = 0x0020u,
        ofPadLS     = 0x0040u,
        ofPadRS     = 0x0080u,
        ofPadDUp    = 0x0100u,
        ofPadDDown  = 0x0200u,
        ofPadDLeft  = 0x0400u,
        ofPadDRight = 0x0800u,
        ofPadLB     = 0x1000u,
        ofPadLT     = 0x2000u,
        ofPadRB     = 0x4000u,
        ofPadRT     = 0x8000u
    };

    enum ofMouseButton
    {
        ofMouseLeft  = 0x01u,
        ofMouseMid   = 0x02u,
        ofMouseRight = 0x04u
    };

    enum ofPlayer
    {
        ofPlayerOne    = 0u,
        ofPlayerTwo    = 1u,
        ofPlayerThree  = 2u,
        ofPlayerFour   = 3u
    };

    struct ofInputState
    {
        ofword    padButtons    = 0x0000u;
        float     leftStick[2]  = {0.0f, 0.0f};
        float     rightStick[2] = {0.0f, 0.0f};
        float     triggers[2]   = {0.0f, 0.0f};
    };

    void         ofUpdateEventDispatch();
    ofInputState ofGetInputState(ofPlayer player =
ofPlayerOne);
    bool         ofIsGamepadConnected(ofPlayer player =
ofPlayerOne);
    glm::vec2    ofGetLeftStick(ofPlayer player =
ofPlayerOne);
    glm::vec2    ofGetRightStick(ofPlayer player =
ofPlayerOne);
    float        ofGetLeftTrigger(ofPlayer player =
ofPlayerOne);
    float        ofGetRightTrigger(ofPlayer player =
ofPlayerOne);
    bool         ofButtonPress(ofPadButton button,
ofPlayer player = ofPlayerOne);
    bool         ofButtonTap(ofPadButton button, ofPlayer player =
ofPlayerOne);
    bool         ofStickMovedTowards(ofbyte stickDirectionMask,
ofPlayer player = ofPlayerOne);

    glm::vec2    ofGetMousePos();
    bool         ofMouseButtonPress(ofMouseButton button);
    bool         ofMouseButtonTap(ofMouseButton button);


    void         ofMapDefaultsP1();
    void         ofMapKeyToButton(ofPadButton button, SDL_Scancode scancode,
ofPlayer player = ofPlayerOne);
    void         ofMapKeyToStick(ofbyte stickPositionMask, SDL_Scancode scancode,
ofPlayer player = ofPlayerOne);
    void         ofMapButtonRemove(ofPadButton button,
ofPlayer player = ofPlayerOne);
    void         ofMapStickRemove(ofbyte stickPositionMask,
ofPlayer player = ofPlayerOne);
    void         ofMappingClear(ofPlayer player =
ofPlayerOne);

    void         ofStartTextInput();
    std::string  ofGetTextInput();
    void         ofSetTextInput(std::string str);
    bool         ofIsInputtingText();
    void         ofStopTextInput();
    void         ofClearTextInput();
    void         ofTextInputSetPadding(ofdword padding);
}
```

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::ofLogLvl {
  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 (ofLogLvl level, const char ∗fmt,...)

  Logs text to the currently selected log type.
- 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.
- 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 CON←↩ SOLE 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 ofLogLvl

```
enum oficina::ofLogLvl
```

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

filename	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

filename	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

filename	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

level	Logging level of the message.
fmt	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

level	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

filename	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

dirname	Name of the Data folder on disk.

9.12 io.hpp

```
/*********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                    *
 *                                                                *
 *  OficinaFramework is free software: you can redistribute       *
 *  it and/or modify it under the terms of the GNU Lesser         *
 *  General Public License as published by the Free Software      *
 *  Foundation, either version 3 of the License, or (at your      *
 *  option) any later version.                                    *
 *                                                                *
 *  You should have received a copy of the GNU Lesser General     *
 *  Public License along with OficinaFramework.  If not, see      *
 *  <http://www.gnu.org/licenses/>.                               *
 *********************************************************************/

#pragma once

#include <cstdarg>
#include <string>
#include "oficina2/platform.hpp"
#include <SDL2/SDL.h>

#if OF_PLATFORM == OF_PLATFORM_WINDOWS
        #define OFLOG_NRM   ""
        #define OFLOG_RED   ""
        #define OFLOG_GRN   ""
        #define OFLOG_YEL   ""
        #define OFLOG_BLU   ""
        #define OFLOG_MAG   ""
        #define OFLOG_CYN   ""
        #define OFLOG_WHT   ""
        #define OFLOG_RESET ""
#else
        #define OFLOG_NRM   "\x1B[0m"
        #define OFLOG_RED   "\x1B[31m"
        #define OFLOG_GRN   "\x1B[32m"
        #define OFLOG_YEL   "\x1B[33m"
        #define OFLOG_BLU   "\x1B[34m"
        #define OFLOG_MAG   "\x1B[35m"
        #define OFLOG_CYN   "\x1B[36m"
        #define OFLOG_WHT   "\x1B[37m"
        #define OFLOG_RESET "\033[0m"
#endif

namespace oficina
{
    enum ofLogLvl {
        ofLogCrit = 0,
        ofLogErr  = 1,
        ofLogWarn = 2,
        ofLogInfo = 3,
        ofLogNone = 4
    };

    enum ofLogType {
        ofLogDisabled = 0,
        ofLogConsole  = 1,
        ofLogFile     = 2
    };


    int  ofLog(ofLogLvl level, const char* fmt, ...);
    void ofLogSetLevel(ofLogLvl level);
    ofLogType ofLogGetType();
    void ofLogUseFile(std::string filename);
    void ofLogUseConsole();
    void ofLogDisable();

    void         ofFindAsset(std::string& filename);
    void         ofSetDataDirectoryName(std::string& dirname);
    std::string  ofLoadText(std::string filename);
    SDL_Surface* ofLoadImage(std::string filename);
}
```

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 "2.0.12"

  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

ofReplNone	No REPL language at all.
ofReplScheme	Use IronScheme on REPL.
ofReplLua	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 intepreter (for Repl), etc.

Parameters

args	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 intepreter (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

str	String to be evaluated.
suppressAns	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

color	Four-dimensional vector containing RGBA colors, normalized.

9.13.3.11 ofSetFullscreen()

```
void oficina::ofSetFullscreen (
             bool state )
```

Changes the application's window state.

Parameters

state	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

type	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

cfg	Desired swap interval configuration.
max	Maximum swap interval, in frames per second (FPS). If applicable.
min	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

x	Width of the window, in pixels.
y	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

```
/*********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                    *
 *                                                                 *
 *  OficinaFramework is free software: you can redistribute        *
 *  it and/or modify it under the terms of the GNU Lesser          *
 *  General Public License as published by the Free Software       *
 *  Foundation, either version 3 of the License, or (at your       *
 *  option) any later version.                                     *
 *                                                                 *
 *  You should have received a copy of the GNU Lesser General      *
 *  Public License along with OficinaFramework.  If not, see       *
 *  <http://www.gnu.org/licenses/>.                                *
 ********************************************************************/

#pragma once

#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"

#define OF_VERSION_STRING "2.0.12"

namespace oficina
{
    void ofInit(std::vector<std::string> args);
    void ofInit();
    void ofGameLoop();
    void ofSoftStop();
    void ofQuit();

    void ofSetWindowSize(ofdword x, ofdword y);

    bool ofQuitFlagRaised();
    glm::uvec2 ofGetWindowSize();

    void ofSetFullscreen(bool state);
    bool ofIsFullscreen();

    enum ofReplType
    {
        ofReplNone,
        ofReplScheme,
        ofReplLua
    };

    void       ofSetReplType(ofReplType type);
    ofReplType ofGetReplType();
    void       ofReplEval(std::string str, bool suppressAns = false);
    void       ofReplDumpOutput();
    void       ofSetSwapInterval(ofFrameRateConfig cfg, float max = 0.0f,
 float min = 0.0f);
    void       ofSetClearColor(glm::vec4 color);
}
```

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::ofIComponent ∗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::ofIComponent ∗ 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

symbol	Name of the function to be defined.
fun	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.

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

symbol	Name of the symbol to be defined.
value	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

symbol	Name of the symbol to be defined.
value	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

symbol	Name of the symbol to be defined.
value	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

symbol	Name of the symbol to be defined.
value	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

symbol	Name of the symbol to be defined.
value	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

symbol	Name of the symbol to be defined.
value	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

strToEval	String to be evaluated, in Lua language.
suppressAns	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

symbol	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

symbol	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

symbol	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

symbol	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

symbol	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

symbol	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

symbol	Name of the symbol to be unbound.

9.16 oflua.hpp

```
/******************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                    *
 *                                                                *
 *  OficinaFramework is free software: you can redistribute       *
 *  it and/or modify it under the terms of the GNU Lesser         *
 *  General Public License as published by the Free Software      *
 *  Foundation, either version 3 of the License, or (at your      *
 *  option) any later version.                                    *
 *                                                                *
 *  You should have received a copy of the GNU Lesser General     *
 *  Public License along with OficinaFramework.  If not, see      *
 *  <http://www.gnu.org/licenses/>.                               *
 ******************************************************************/

#pragma once

#ifndef NO_LUA

#include <lua.hpp>
#include "oficina2/entity.hpp"
#include <string>

namespace oficina
{
    void                       ofLuaInit();
    void                       ofLuaDeinit();
    bool                       ofLuaIsInit();
    void                       ofLuaEval(std::string strToEval, bool suppressAns = true);
    //void                       ofLuaDumpOutput(); // TODO -- needed?
    void                       ofLuaDefineSymbol(std::string symbol, std::string value);
    void                       ofLuaDefineSymbol(std::string symbol, double value);
    void                       ofLuaDefineSymbol(std::string symbol, int value);
    void                       ofLuaDefineSymbol(std::string symbol, bool value);
    void                       ofLuaDefineSymbol(std::string symbol,
oficina::ofEntity* value);
    void                       ofLuaDefineSymbol(std::string symbol,
oficina::ofIComponent* value);
    void                       ofLuaDefineFunc(std::string symbol, lua_CFunction fun);
    void                       ofLuaUndefine(std::string symbol);
    std::string                ofLuaGetString(std::string symbol);
    double                     ofLuaGetNumber(std::string symbol);
    int                        ofLuaGetInteger(std::string symbol);
    bool                       ofLuaGetBoolean(std::string symbol);
    oficina::ofEntity*         ofLuaGetEntity(std::string symbol);
    oficina::ofIComponent*     ofLuaGetComponent(std::string symbol);

    class ofLua : public ofIComponent
    {
    public:
        void                       init();
        void                       loadfile(std::string filename);
        void                       reload();
        void                       unload();
        void                       update(float dt);
        void                       regSym(std::string symbol, std::string value);
        void                       regSym(std::string symbol, double value);
        void                       regSym(std::string symbol, int value);
        void                       regSym(std::string symbol, bool value);
        void                       regSym(std::string symbol,
oficina::ofEntity* value);
        void                       regSym(std::string symbol,
oficina::ofIComponent* value);
        void                       regFunc(std::string symbol, lua_CFunction fun);
        std::string                getString(std::string symbol);
        double                     getNumber(std::string symbol);
        int                        getInteger(std::string symbol);
        bool                       getBoolean(std::string symbol);
        oficina::ofEntity*         getEntity(std::string symbol);
        oficina::ofIComponent*     getComponent(std::string symbol);
        bool        isInit() const;
    private:
        lua_State*  L             = nullptr;
        bool        m_initialized = false;
        bool        m_updt_error  = false;
        std::string m_filename;
    };


}
#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

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

See also

SCHEME_FUNCAST

Parameters

n_optional_params	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

symbol	Name of the symbol to be defined.
value	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

symbol	Name of the symbol to be defined.
value	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

symbol	Name of the symbol to be defined.
value	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

strToEval	String to be evaluated, in Scheme language.
suppressAns	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

symbol	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

symbol	Name of the symbol to be unbound.

9.18 ofscheme.hpp

```
/**********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                     *
 *                                                                  *
 *  OficinaFramework is free software: you can redistribute        *
 *  it and/or modify it under the terms of the GNU Lesser          *
 *  General Public License as published by the Free Software       *
 *  Foundation, either version 3 of the License, or (at your       *
 *  option) any later version.                                     *
 *                                                                  *
 *  You should have received a copy of the GNU Lesser General      *
 *  Public License along with OficinaFramework.  If not, see       *
 *  <http://www.gnu.org/licenses/>.                                *
 ********************************************************************/

#pragma once

#ifndef NO_SCHEME

#include <libguile.h>
#include <string>
#include <functional>
#include "oficina2/entity.hpp"

#define SCHEME_FUNCAST(x) (scm_t_subr)&(x)

namespace oficina
{
    void ofScmInit();
    void ofScmDeinit();
    bool ofScmIsInit();
    void ofScmEval(std::string strToEval, bool suppressAns = false);
    void ofScmDumpOutput();
    void ofScmDefineSymbol(std::string symbol, SCM value);
    void ofScmDefineSymbol(std::string symbol, oficina::ofEntity* value);
    void ofScmDefineSymbol(std::string symbol, oficina::ofIComponent* value);
    void ofScmDefineFunc(std::string symbol, int n_params, scm_t_subr fun, int n_optional_params = 0);
    void ofScmUndefine(std::string symbol);
    SCM  ofScmGetReference(std::string symbol);

    class ofScheme : public ofIComponent
    {
    public:
        void init();
        void loadfile(std::string module, std::string filename);
        void reload();
        void unload();
        void update(float dt);
        void regSym(std::string symbol, SCM value);
        void regFunc(std::string symbol, int n_params, scm_t_subr fun, int n_optional_params = 0);
        SCM  getSymRef(std::string symbol);
        bool isInit() const;
    private:
        bool m_initialized = false;
        bool m_loaded = false;
        std::string m_modulename;
        std::string m_filename;
        bool        m_updt_error = false;

        SCM obj_module  = SCM_UNDEFINED,
            init_func   = SCM_UNDEFINED,
            update_func = SCM_UNDEFINED;
    };
}

#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

```
/********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                    *
 *                                                                *
 *  OficinaFramework is free software: you can redistribute       *
 *  it and/or modify it under the terms of the GNU Lesser         *
 *  General Public License as published by the Free Software      *
 *  Foundation, either version 3 of the License, or (at your      *
 *  option) any later version.                                    *
 *                                                                *
 *  You should have received a copy of the GNU Lesser General     *
 *  Public License along with OficinaFramework.  If not, see      *
 *  <http://www.gnu.org/licenses/>.                               *
 ********************************************************************/

#pragma once

#define OF_PLATFORM_UNKNOWN       0x000u
#define OF_PLATFORM_WINDOWS       0x001u
#define OF_PLATFORM_LINUX         0x002u
#define OF_PLATFORM_MACOSX        0x004u
#define OF_PLATFORM_ANDROID       0x008u
#define OF_PLATFORM_IOS           0x010u
#define OF_PLATFORM_IOS_SIMULATOR 0x020u

#define OF_ARCH_UNKNOWN   0x000u
#define OF_ARCH_32BIT     0x002u
#define OF_ARCH_64BIT     0x004u
#define OF_ARCH_ARM       0x008u
#define OF_ARCH_ARMV7     0x010u
#define OF_ARCH_ARM64     0x020u

#ifdef _WIN64
    #define OF_PLATFORM     OF_PLATFORM_WINDOWS
    #define OF_ARCH         OF_ARCH_64BIT
    #define OF_DESKTOP
#elif  _WIN32
    #define OF_PLATFORM     OF_PLATFORM_WINDOWS
    #define OF_ARCH         OF_ARCH_32BIT
    #define OF_DESKTOP
#elif  __APPLE__
    #if TARGET_OS_IPHONE && TARGET_IPHONE_SIMULATOR
        #define OF_PLATFORM (OF_PLATFORM_IOS | OF_PLATFORM_IOS_SIMULATOR)
        #define OF_MOBILE
    #elif TARGET_OS_IPHONE
        #define OF_PLATFORM OF_PLATFORM_IOS
        #define OF_MOBILE
    #elif TARGET_OS_MAC
        #define OF_PLATFORM OF_PLATFORM_MACOSX
        #define OF_DESKTOP
    #endif
#elif ANDROID
    #define OF_PLATFORM     OF_PLATFORM_ANDROID
    #define OF_MOBILE
#elif  __linux__
    #define OF_PLATFORM     OF_PLATFORM_LINUX
    #define OF_DESKTOP
#else
    #define OF_PLATFORM     OF_PLATFORM_UNKNOWN
    #define OF_DESKTOP
#endif

// Check architecture. This will mainly serve for GCC and Clang
#ifndef OF_ARCH
    #ifdef __x86_64__
        #define OF_ARCH  OF_ARCH_64BIT
    #elif __ARM_ARCH_7__
        #define OF_ARCH  OF_ARCH_ARMV7
    #elif __arm__
        #define OF_ARCH  OF_ARCH_ARM
    #elif __aarch64__
        #define OF_ARCH  OF_ARCH_ARM64
    #elif __i386__
        #define OF_ARCH  OF_ARCH_32BIT
    #else
        #define OF_ARCH  OF_ARCH_UNKNOWN
    #endif
#endif


// Important platform headers that cannot be
// left out
#if   OF_PLATFORM == OF_PLATFORM_WINDOWS
    #include <Windows.h>
```

```
#elif OF_PLATFORM == OF_PLATFORM_LINUX
#elif OF_PLATFORM == OF_PLATFORM_MACOSX
#endif

// Definitions for scripting support
#define OF_SCRIPTING_NONE      0x00u
#define OF_SCRIPTING_SCHEME    0x01u
#define OF_SCRIPTING_LUA       0x02u

#if defined(NO_SCHEME) && defined(NO_LUA)
    #define OF_SCRIPTING_AVAILABLE OF_SCRIPTING_NONE
#elif defined(NO_SCHEME) && !defined(NO_LUA)
    #define OF_SCRIPTING_AVAILABLE OF_SCRIPTING_LUA
#elif !defined(NO_SCHEME) && defined(NO_LUA)
    #define OF_SCRIPTING_AVAILABLE OF_SCRIPTING_SCHEME
#else
    #define OF_SCRIPTING_AVAILABLE (OF_SCRIPTING_SCHEME | OF_SCRIPTING_LUA)
#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 oficina::ofTextureRenderer

    Tool for easily rendering 2D textures or texture atlases.
- class oficina::ofFont

    Represents a font.
- class oficina::ofAnimator

    Tool for controlling a texture renderer to generate animations.
- struct oficina::ofPrimitive

    A structure representing a primitive. Can be used for rendering.
- class oficina::ofPrimitiveRenderer

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

Enumerations

- enum oficina::ofContextType { oficina::ofContextNone, oficina::ofContextGL, oficina::ofContextGLES }

    Describes the type of a rendering context.
- enum oficina::ofBufferUsage {
oficina::ofBufferStaticDraw = GL_STATIC_DRAW, oficina::ofBufferDynamicDraw = GL_DYNAMIC_DRAW, oficina::ofBufferStreamDraw = GL_STREAM_DRAW, oficina::ofBufferStaticRead = GL_STATIC_READ, oficina::ofBufferDynamicRead = GL_DYNAMIC_READ, oficina::ofBufferStreamRead = GL_STREAM_READ, oficina::ofBufferStaticCopy = GL_STATIC_COPY, oficina::ofBufferDynamicCopy = GL_DYNAMIC_COPY, oficina::ofBufferStreamCopy = GL_STREAM_COPY }

    Describes the usage of a created buffer object.
- enum oficina::ofShaderType {
oficina::ofShaderVertex = GL_VERTEX_SHADER, oficina::ofShaderGeometry = GL_GEOMETRY_SHAD↩ER, oficina::ofShaderFragment = GL_FRAGMENT_SHADER, oficina::ofShaderTessControl = GL_TESS_↩CONTROL_SHADER,
oficina::ofShaderTessEval = GL_TESS_EVALUATION_SHADER, oficina::ofShaderCompute = GL_COMP↩UTE_SHADER }

    Describes the type of a shader.
- enum oficina::ofPrimitiveType {
oficina::ofPoints = GL_POINTS, oficina::ofLineStrip = GL_LINE_STRIP, oficina::ofLineLoop = GL_LINE_L↩OOP, oficina::ofLines = GL_LINES,
oficina::ofLineStripAdj = GL_LINE_STRIP_ADJACENCY, oficina::ofLinesAdj = GL_LINES_ADJACENCY, oficina::ofTriangleStrip = GL_TRIANGLE_STRIP, oficina::ofTriangleFan = GL_TRIANGLE_FAN,
oficina::ofQuad = GL_TRIANGLE_FAN, oficina::ofTriangles = GL_TRIANGLES, oficina::ofTriangleStripAdj = GL_TRIANGLE_STRIP_ADJACENCY, oficina::ofTrianglesAdj = GL_TRIANGLES_ADJACENCY,
oficina::ofPatches = GL_PATCHES }

    Describes a type for a primitive.
- enum oficina::ofDataType {
oficina::ofDataByte = GL_BYTE, oficina::ofDataUByte = GL_UNSIGNED_BYTE, oficina::ofDataShort = G↩L_SHORT, oficina::ofDataUShort = GL_UNSIGNED_SHORT,
oficina::ofDataInt = GL_INT, oficina::ofDataUInt = GL_UNSIGNED_INT, oficina::ofDataFloat = GL_FLOAT, oficina::ofDataDouble = GL_DOUBLE,
oficina::ofDataFixed = GL_FIXED }

    Represents the type of certain data fed to a buffer.
- enum oficina::ofFontFaces { oficina::ofFontFaceFixedsysExcelsior, oficina::ofFontFaceGohuFont, oficina↩::ofFontFaceFantasqueSans, oficina::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

ofBufferStaticDraw	Store buffer data statically for drawing.
ofBufferDynamicDraw	Store buffer dynamically for drawing.
ofBufferStreamDraw	Store buffer as a stream for drawing.
ofBufferStaticRead	Store buffer statically for reading.
ofBufferDynamicRead	Store buffer dynamically for reading.
ofBufferStreamRead	Store buffer as a stream for reading.
ofBufferStaticCopy	Store buffer statically for copying.
ofBufferDynamicCopy	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 714 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	Tesselation Control Shader.
ofShaderTessEval	Tesselation 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

state	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

```
/*********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                    *
 *                                                                 *
 *  OficinaFramework is free software: you can redistribute       *
 *  it and/or modify it under the terms of the GNU Lesser         *
 *  General Public License as published by the Free Software      *
 *  Foundation, either version 3 of the License, or (at your      *
 *  option) any later version.                                    *
 *                                                                 *
 *  You should have received a copy of the GNU Lesser General     *
 *  Public License along with OficinaFramework.  If not, see      *
 *  <http://www.gnu.org/licenses/>.                               *
 ********************************************************************/

#pragma once

#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>

namespace oficina
{
    enum ofContextType
    {
        ofContextNone,
        ofContextGL,
        ofContextGLES
    };

    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,
    };

    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_COMPUTE_SHADER
    };

    enum ofPrimitiveType
    {
        ofPoints           = GL_POINTS,
        ofLineStrip        = GL_LINE_STRIP,
        ofLineLoop         = GL_LINE_LOOP,
        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
    };

    enum ofDataType
    {
        ofDataByte   = GL_BYTE,
        ofDataUByte  = GL_UNSIGNED_BYTE,
        ofDataShort  = GL_SHORT,
        ofDataUShort = GL_UNSIGNED_SHORT,
        ofDataInt     = GL_INT,
        ofDataUInt    = GL_UNSIGNED_INT,
        ofDataFloat   = GL_FLOAT,
        ofDataDouble  = GL_DOUBLE,
        ofDataFixed   = GL_FIXED
    };


    const char ofDefaultShaderSrc_VS[] =
        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);
        })";

    const char ofDefaultShaderSrc_FS[] =
        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);
        })";


    class ofContext
    {
    public:
        void        pushArg(std::string arg);
        void open(ofContextType type, const ofDisplay& hwnd);
        void close();

        bool       isInit() const;
        void       setViewportSize(glm::uvec2 sz);
        glm::uvec2 getViewportSize();
        void       setClearColor(glm::vec4 color);
    private:
        ofContextType          m_type = ofContextNone;
        glm::uvec2             m_vwpsz;
        glm::vec4              m_clearcolor = glm::vec4(0.0f, 0.0f, 0.0f, 1.0f);
        SDL_GLContext          ctx;
        bool                   m_initialized = false;
        std::list<std::string> m_confv;
        void                   parseArgs();
    };



    class ofBuffer
    {
    public:
        ofBuffer();
        ofBuffer(const ofBuffer&);
        virtual void init() final;
        virtual void unload() final;
        virtual void bind() final;
        virtual void unbind() final;

        virtual void setData(size_t dataSize,
                             void* data,
                             ofBufferUsage usage);

        ofBuffer& operator=(const ofBuffer& other);

        virtual bool isInit() const final;
        virtual GLuint getName() const final;
    protected:
        GLenum m_type = GL_ARRAY_BUFFER;
```

```
        GLuint m_name = 0u;
    };


    class ofVertexBuffer final : public ofBuffer
    {
    public:
        ofVertexBuffer();
    };

    class ofElementBuffer final : public ofBuffer
    {
    public:
        ofElementBuffer();

        void setCount(GLsizei count);
        void setType(ofDataType type);
        void setProps(GLsizei count, ofDataType type);

        GLsizei getCount() const;
        ofDataType getType() const;

        void draw(ofPrimitiveType mode);
    private:
        GLsizei    m_count = -1;
        ofDataType m_dataType  = ofDataUInt;
    };



    class ofShader
    {
    public:
        ofShader();
        ofShader(const ofShader&);
        virtual void init(ofShaderType type) final;
        virtual void unload() final;
        virtual void setSource(const char* src) final;
        virtual void compile() final;

        virtual bool isInit() const final;
        virtual bool isCompiled() const final;
        virtual GLuint getName() const final;

        ofShader& operator=(const ofShader& shader);
    protected:
        ofShaderType m_type = ofShaderFragment;
        GLuint       m_name = 0u;
        bool         m_srcassign = false;
        bool         m_compiled = false;
    };

    class ofShaderProgram;
    class ofShaderAttribute final
    {
        friend class ofShaderProgram;
    public:
        ofShaderAttribute();
        ofShaderAttribute(const ofShaderAttribute&);
        void setSize(GLint s);
        void setType(ofDataType t);
        void setStride(GLsizei stride);
        void setAutoNormalize(bool state);
        void setProps(GLint size, ofDataType type, GLsizei stride, bool normalize = false);

        void enable();

        int getSize();
        ofDataType getType();
        size_t getStride();
        bool isAutoNormalizing();
        bool isValid() const;

        void bindVertexArrayData(void* byteOffset = nullptr);

        ofShaderAttribute& operator=(const ofShaderAttribute& attr);
    private:
        GLint   m_name         = -1;
        GLint   m_size         = 1;
        GLsizei m_stride       = 0;
        bool    m_normalize    = false;
        ofDataType m_type = ofDataFloat;
    };

    class ofTexture;
    class ofShaderUniform final
    {
        friend class ofShaderProgram;
    public:
        ofShaderUniform();
        ofShaderUniform(const ofShaderUniform&);
        bool isValid() const;
        ofShaderUniform& operator=(const ofShaderUniform& uniform);

        void set(float value);
        void set(glm::vec2 value);
        void set(glm::vec3 value);
        void set(glm::vec4 value);

        void set(int value);
        void set(glm::ivec2 value);
        void set(glm::ivec3 value);
        void set(glm::ivec4 value);

        void set(unsigned int value);
        void set(glm::uvec2 value);
        void set(glm::uvec3 value);
        void set(glm::uvec4 value);

        void set(glm::mat2 value, bool transpose = false);
        void set(glm::mat3 value, bool transpose = false);
        void set(glm::mat4 value, bool transpose = false);

        void set(glm::mat2x3 value, bool transpose = false);
        void set(glm::mat3x2 value, bool transpose = false);

        void set(glm::mat2x4 value, bool transpose = false);
        void set(glm::mat4x2 value, bool transpose = false);

        void set(glm::mat3x4 value, bool transpose = false);
        void set(glm::mat4x3 value, bool transpose = false);
    private:
        GLint m_name          = -1;
    };

    class ofShaderProgram final
    {
    public:
        ofShaderProgram();
        ofShaderProgram(const ofShaderProgram&);
        void init();
        void unload();
        void attach(const ofShader& shader);
        void attachUnload(ofShader& shader);
        void bindFragmentDataLocation(std::string name, ofdword colorNumber = 0u);
        void link();
        void use();
        void unuse();

        bool isInit() const;
        bool isLinked() const;
        GLuint getName() const;

        ofShaderProgram& operator=(const ofShaderProgram& program);

        ofShaderAttribute getAttributeLocation(std::string name);
        ofShaderUniform   getUniformLocation(std::string name);
    private:
        bool shaderProgramVerify(const ofShader&) const;
        GLuint m_name = 0u;
        bool   m_linked = false;
    };



    class ofVertexArray
    {
    public:
        ofVertexArray();
        ofVertexArray(const ofVertexArray&);
        void init();
        void unload();
        void bind();
        void unbind();

        void draw(ofPrimitiveType mode, int firstVertexIdx, size_t vertexCount);

        ofVertexArray& operator=(const ofVertexArray& other);
    private:
        GLuint m_name = 0u;
    };




    // Textures
    class ofTexturePool;
    class ofTexture
    {
        friend class ofTexturePool;
    public:
        ofTexture();
        ofTexture(const ofTexture&);
        void bind(ofword currentSampler = 0);
        void unbind(ofword currentSampler = 0);

        ofTexture& operator=(const ofTexture& other);
        GLuint     operator()();

        bool isLoaded() const;
        std::string getFileName() const;
        glm::uvec2  getSize() const;
    private:
        GLuint      m_name = 0u;
        glm::uvec2  m_size;
        std::string m_filename;
    };

    enum ofFontFaces
    {
        ofFontFaceFixedsysExcelsior,
        ofFontFaceGohuFont,
        ofFontFaceFantasqueSans,
        ofFontFaceTerminus
    };

    class ofFont;
    class ofTexturePool
    {
    public:
        static ofTexture load(std::string filename);
        static ofTexture load(SDL_Surface* surf);
        static ofFont    loadDefaultFont(ofFontFaces fontface = ofFontFaceGohuFont);
        static void      unload(ofTexture& t);
        static void      clear();
    };



    class ofTextureRenderer
    {
    public:
        ofTextureRenderer();
        ofTextureRenderer(const ofTextureRenderer&);
        void init(ofTexture& t, glm::uvec2 frameSize = glm::uvec2(0, 0));
        void render(glm::vec2 position, glm::mat4 mvp, ofdword frame = 0u, glm::vec4 color = glm::vec4(1.0f));
        void unload();

        ofTextureRenderer& operator=(const ofTextureRenderer& other);

        void SetTexture(ofTexture& t);

        bool isInit() const;
    private:
        bool m_initialized = false;
        ofTexture m_texture;
        glm::vec2 m_frameSize;
        glm::vec2 m_frameSizeTxl;
        ofVertexArray vao;
        ofVertexBuffer vbo;
        ofShaderAttribute attrQuadpos,
                          attrTexcoord,
                          attrFrsz,
                          attrFrsztxl;

        ofShaderUniform uniPos,
                        uniFrpostxl,
                        uniColor,
                        uniMVP,
                        uniTexSampler;
    };

    class ofFont
    {
    public:
        ofFont();
        ofFont(const ofFont&);
        void init(ofTexture fontTexture, glm::uvec2 glyphSize, bool manageTexture = false);
        void write(std::string text, glm::vec2 position, glm::mat4 mvp, glm::vec4 color = glm::vec4(1.0f));
        void unload();
        glm::vec2 measure(std::string text);

        ofFont& operator=(const ofFont& other);

        glm::uvec2 getGlyphSize() const;

        bool isInit() const;
    private:
        bool m_unloadtexture = false;
        bool m_initialized = false;
        glm::uvec2 m_glyphsize;
        ofTexture m_texture;
        ofTextureRenderer m_renderer;
    };

    class ofAnimator : public ofIComponent
    {
    public:
        void init();
        void unload();
        void update(float dt);
        void draw(glm::mat4 ViewProjection);

        void reg(std::string animName, ofdword nFrames, const ofdword* animFrames, float speed, bool loops = false, ofdword loopBackTo = 0u);
        void unreg(std::string animName);
        void SetAnimation(std::string animName);
        void SyncToFrameRate(bool state);
        void SetAnimationSpeed(float spd);
        float GetAnimationSpeed() const;
        float GetDefaultAnimationSpeed() const;
        void SetAnimationRunning(bool state);
        std::string GetCurrentAnimationName() const;

        void SetRenderer(ofTextureRenderer renderer, bool manage = false);
        void SetAnimationTexture(ofTexture t);

        bool isInit() const;
        glm::vec2 getPosition();
        void setPosition(glm::vec2 pos);
        bool GetAnimationRunning() const;
    private:
        struct ofAnimProps
        {
            ofdword  num_frames;
            ofdword  loopback;
            ofdword* frames = nullptr;
            bool     loops;
            float    speed;
        };

        bool m_unloadtexture = false;
        bool m_initialized   = false;
        bool m_sync          = true;
        bool m_playing       = true;
        const ofAnimProps*  m_current = nullptr;
        ofdword             m_currentframe = 0u;
        ofFrameSpan         m_framespan;
        ofTimeSpan          m_timespan;
        glm::uvec2          m_framesize;
        glm::vec2           m_position;
        ofTextureRenderer   m_renderer;
        std::string         m_animname;
        float               m_animspd;
        std::map<std::string, ofAnimProps> m_animations;
    };

    class ofPrimitiveRenderer;
    struct ofPrimitive
    {
        friend class ofPrimitiveRenderer;
    public:
        ofPrimitiveType type;
        ofVertexBuffer vbo;
        ofdword NumberOfVertices;
        ~ofPrimitive();
    private:
        ofPrimitive() {}
    };

    class ofPrimitiveRenderer
    {
    public:
        static ofPrimitive* makePrimitive(ofPrimitiveType type, ofdword verticesAmount, size_t verticesSize, float* vertices);
        static void draw(ofPrimitive* p, glm::vec4 color, glm::mat4 mvp);
    };



    ofShader        ofLoadDefaultFragShader();
    ofShader        ofLoadDefaultVertexShader();
    ofShaderProgram ofLoadDefaultShaderProgram();
    void            ofSetVSync(bool state);
}
```

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

```
/********************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                    *
 *                                                                *
 *  OficinaFramework is free software: you can redistribute       *
 *  it and/or modify it under the terms of the GNU Lesser         *
 *  General Public License as published by the Free Software      *
 *  Foundation, either version 3 of the License, or (at your      *
 *  option) any later version.                                    *
 *                                                                *
 *  You should have received a copy of the GNU Lesser General     *
 *  Public License along with OficinaFramework.  If not, see      *
 *  <http://www.gnu.org/licenses/>.                               *
 ********************************************************************/

#pragma once

#include <cstdint>

namespace oficina
{
    class ofTimeSpan
    {
    public:
        void  begin();
        float yieldSpan();
        float resetSpan();
        float stop();
        bool  isRunning() const;
    private:
        bool     m_started = false;
        uint32_t m_timer   = 0u;
    };

    class ofFrameSpan
    {
    public:
        void     begin();
        void     update();
        uint32_t yieldSpan();
        uint32_t resetSpan();
        uint32_t stop();
        bool     isRunning() const;
    private:
        bool     m_started = false;
        uint32_t m_timer   = 0u;
    };
}
```

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

value	Value to be compared.
min	Minimum value tolerated by the clamping operation.
max	Maximum value tolerated by the clamping operation.

Returns

The given value, accordingly clamped between the given minimum and maximum values.

9.26 types.hpp

```
/*******************************************************************
 *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
 *  This file is part of OficinaFramework v2.x                    *
 *                                                                *
 *  OficinaFramework is free software: you can redistribute       *
 *  it and/or modify it under the terms of the GNU Lesser         *
 *  General Public License as published by the Free Software      *
 *  Foundation, either version 3 of the License, or (at your      *
 *  option) any later version.                                    *
 *                                                                *
 *  You should have received a copy of the GNU Lesser General     *
 *  Public License along with OficinaFramework.  If not, see      *
 *  <http://www.gnu.org/licenses/>.                               *
 *****************************************************************/

//#define GLM_FORCE_SWIZZLE

#include "oficina2/platform.hpp"
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <cmath>
#include <cstdint>

#pragma once

typedef uint8_t  ofbyte;
typedef uint16_t ofword;
typedef uint32_t ofdword;
typedef uint64_t ofqword;

typedef int8_t   ofsbyte;
typedef int16_t  ofsword;
typedef int32_t  ofsdword;
typedef int64_t  ofsqword;

typedef uintptr_t ofaword;
typedef intptr_t  ofsaword;

float ofClamp(float value, float min, float max);
```

Index

add
    oficina::ofCanvasManager, 64
AddComponent
    oficina::ofEntity, 79
attach
    oficina::ofShaderProgram, 115
attachUnload
    oficina::ofShaderProgram, 116

benchmark.hpp, 141, 142
    ofBenchmarkIsRunning, 141
    ofBenchmarkStart, 142
bind
    oficina::ofTexture, 128
bindFragmentDataLocation
    oficina::ofShaderProgram, 116
bindVertexArrayData
    oficina::ofShaderAttribute, 111

canvas.hpp, 142, 143
close
    oficina::ofDisplay, 71
compile
    oficina::ofShader, 108

dbg_ChangeState
    oficina::ofCanvasManager, 65
dbg_ReplLineNumber
    oficina::ofCanvasManager, 65
dbg_ReplOutStream
    oficina::ofCanvasManager, 66
dbg_callEval
    oficina::ofCanvasManager, 65
dbg_getState
    oficina::ofCanvasManager, 65
dbg_setFont
    oficina::ofCanvasManager, 66
display.hpp, 144, 145
    ofFrameRateConfig, 145
draw
    oficina::ofAnimator, 52
    oficina::ofCanvasManager, 66
    oficina::ofElementBuffer, 75
    oficina::ofEntity, 79
    oficina::ofIComponent, 93
    oficina::ofPrimitiveRenderer, 102
    oficina::ofVertexArray, 138
DrawComponents
    oficina::ofEntity, 80

entity.hpp, 146, 147

GetAnimationRunning
    oficina::ofAnimator, 53
GetAnimationSpeed
    oficina::ofAnimator, 53
getAttributeLocation
    oficina::ofShaderProgram, 116
getBoolean
    oficina::ofLua, 96
getCanvasList
    oficina::ofCanvasManager, 66
GetComponent
    oficina::ofEntity, 80
getComponent
    oficina::ofLua, 96
getCount
    oficina::ofElementBuffer, 76
GetCurrentAnimationName
    oficina::ofAnimator, 53
GetDefaultAnimationSpeed
    oficina::ofAnimator, 53
getDeltaTime
    oficina::ofDisplay, 71
getEntity
    oficina::ofLua, 96
getEulerAngles
    oficina::ofEntity, 80
getFileName
    oficina::ofTexture, 128
getGlyphSize
    oficina::ofFont, 88
getHandle
    oficina::ofDisplay, 72
getInteger
    oficina::ofLua, 97
getModelMatrix
    oficina::ofEntity, 81
getName
    oficina::ofBuffer, 60
    oficina::ofEntity, 81
    oficina::ofShader, 108
    oficina::ofShaderProgram, 117
getNumber
    oficina::ofLua, 97
getPosition
    oficina::ofAnimator, 54
    oficina::ofEntity, 81
getProperty
    oficina::ofEntity, 81
getPropertyMask

oficina::ofEntity, 82
getScale
oficina::ofEntity, 82
getSize
oficina::ofDisplay, 72
oficina::ofShaderAttribute, 111
oficina::ofTexture, 128
getStride
oficina::ofShaderAttribute, 111
getString
oficina::ofLua, 97
getSymRef
oficina::ofScheme, 104
getType
oficina::ofElementBuffer, 76
oficina::ofIComponent, 93
oficina::ofShaderAttribute, 111
getUniformLocation
oficina::ofShaderProgram, 117
getViewportSize
oficina::ofContext, 68

init
oficina::ofAnimator, 54
oficina::ofCanvas, 62
oficina::ofEntity, 82
oficina::ofFont, 88
oficina::ofShader, 108
oficina::ofTextureRenderer, 133
input.hpp, 148, 163
ofButtonPress, 153
ofButtonTap, 154
ofGetInputState, 154
ofGetLeftStick, 155
ofGetLeftTrigger, 155
ofGetMousePos, 155
ofGetRightStick, 156
ofGetRightTrigger, 156
ofGetTextInput, 156
ofIsGamepadConnected, 157
ofIsInputtingText, 157
ofMapButtonRemove, 157
ofMapDefaultsP1, 158
ofMapKeyToButton, 158
ofMapKeyToStick, 159
ofMapStickRemove, 159
ofMappingClear, 159
ofMouseButton, 150
ofMouseButtonPress, 160
ofMouseButtonTap, 160
ofPadButton, 151
ofPlayer, 152
ofSetTextInput, 161
ofStartTextInput, 161
ofStick, 152
ofStickAxis, 152
ofStickMovedTowards, 161
ofStickSignal, 153
ofStopTextInput, 162
ofTextInputSetPadding, 162
ofUpdateEventDispatch, 163
io.hpp, 165, 170
ofFindAsset, 167
ofLoadImage, 167
ofLoadText, 168
ofLog, 168
ofLogGetType, 168
ofLogLvl, 166
ofLogSetLevel, 169
ofLogType, 167
ofLogUseFile, 169
ofSetDataDirectoryName, 169
isAutoNormalizing
oficina::ofShaderAttribute, 112
isCompiled
oficina::ofShader, 108
isFullscreen
oficina::ofDisplay, 72
isInit
oficina::ofAnimator, 54
oficina::ofBuffer, 60
oficina::ofContext, 69
oficina::ofFont, 88
oficina::ofLua, 98
oficina::ofScheme, 105
oficina::ofShader, 109
oficina::ofShaderProgram, 117
oficina::ofTextureRenderer, 134
isLinked
oficina::ofShaderProgram, 117
isLoaded
oficina::ofTexture, 128
isOpen
oficina::ofDisplay, 72
isRunning
oficina::ofFrameSpan, 91
oficina::ofTimeSpan, 136
isValid
oficina::ofShaderAttribute, 112
oficina::ofShaderUniform, 120

link
oficina::ofShaderProgram, 118
load
oficina::ofCanvas, 62
oficina::ofEntity, 82
oficina::ofTexturePool, 130, 131
loadDefaultFont
oficina::ofTexturePool, 131
loadfile
oficina::ofLua, 98
oficina::ofScheme, 105

makePrimitive
oficina::ofPrimitiveRenderer, 103
measure
oficina::ofFont, 89

ofBenchmarkIsRunning
benchmark.hpp, 141
ofBenchmarkStart
benchmark.hpp, 142
ofBufferUsage
render.hpp, 196
ofButtonPress
input.hpp, 153
ofButtonTap
input.hpp, 154
ofClamp
types.hpp, 209
ofContextType
render.hpp, 197
ofDataType
render.hpp, 197
ofDebuggerState
oficina::ofCanvasManager, 64
ofDefaultShaderSrc_FS
render.hpp, 200
ofDefaultShaderSrc_VS
render.hpp, 200
ofFindAsset
io.hpp, 167
ofFontFaces
render.hpp, 197
ofFrameRateConfig
display.hpp, 145
ofGameLoop
oficina.hpp, 173
ofGetInputState
input.hpp, 154
ofGetLeftStick
input.hpp, 155
ofGetLeftTrigger
input.hpp, 155
ofGetMousePos
input.hpp, 155
ofGetReplType
oficina.hpp, 173
ofGetRightStick
input.hpp, 156
ofGetRightTrigger
input.hpp, 156
ofGetTextInput
input.hpp, 156
ofGetWindowSize
oficina.hpp, 173
ofInit
oficina.hpp, 173, 174
ofIsFullscreen
oficina.hpp, 174
ofIsGamepadConnected
input.hpp, 157
ofIsInputtingText
input.hpp, 157
ofLoadDefaultFragShader
render.hpp, 199
ofLoadDefaultShaderProgram
render.hpp, 199
ofLoadDefaultVertexShader
render.hpp, 199
ofLoadImage
io.hpp, 167
ofLoadText
io.hpp, 168
ofLog
io.hpp, 168
ofLogGetType
io.hpp, 168
ofLogLvl
io.hpp, 166
ofLogSetLevel
io.hpp, 169
ofLogType
io.hpp, 167
ofLogUseFile
io.hpp, 169
ofLuaDefineFunc
oflua.hpp, 179
ofLuaDefineSymbol
oflua.hpp, 179–181
ofLuaEval
oflua.hpp, 181
ofLuaGetBoolean
oflua.hpp, 181
ofLuaGetComponent
oflua.hpp, 182
ofLuaGetEntity
oflua.hpp, 182
ofLuaGetInteger
oflua.hpp, 183
ofLuaGetNumber
oflua.hpp, 183
ofLuaGetString
oflua.hpp, 183
ofLuaIsInit
oflua.hpp, 184
ofLuaUndefine
oflua.hpp, 184
ofMapButtonRemove
input.hpp, 157
ofMapDefaultsP1
input.hpp, 158
ofMapKeyToButton
input.hpp, 158
ofMapKeyToStick
input.hpp, 159
ofMapStickRemove
input.hpp, 159
ofMappingClear
input.hpp, 159
ofMouseButton
input.hpp, 150
ofMouseButtonPress
input.hpp, 160

ofMouseButtonTap
    input.hpp, 160
ofPadButton
    input.hpp, 151
ofPlayer
    input.hpp, 152
ofPrimitiveType
    render.hpp, 198
ofQuit
    oficina.hpp, 174
ofQuitFlagRaised
    oficina.hpp, 174
ofReplEval
    oficina.hpp, 174
ofReplType
    oficina.hpp, 172
ofScmDefineFunc
    ofscheme.hpp, 187
ofScmDefineSymbol
    ofscheme.hpp, 187, 188
ofScmEval
    ofscheme.hpp, 189
ofScmGetReference
    ofscheme.hpp, 189
ofScmIsInit
    ofscheme.hpp, 189
ofScmUndefine
    ofscheme.hpp, 189
ofSetClearColor
    oficina.hpp, 175
ofSetDataDirectoryName
    io.hpp, 169
ofSetFullscreen
    oficina.hpp, 175
ofSetReplType
    oficina.hpp, 175
ofSetSwapInterval
    oficina.hpp, 175
ofSetTextInput
    input.hpp, 161
ofSetVSync
    render.hpp, 199
ofSetWindowSize
    oficina.hpp, 176
ofShaderType
    render.hpp, 198
ofSoftStop
    oficina.hpp, 176
ofStartTextInput
    input.hpp, 161
ofStick
    input.hpp, 152
ofStickAxis
    input.hpp, 152
ofStickMovedTowards
    input.hpp, 161
ofStickSignal
    input.hpp, 153
ofStopTextInput
    input.hpp, 162
ofTextInputSetPadding
    input.hpp, 162
ofUpdateEventDispatch
    input.hpp, 163
oficina.hpp, 171, 177
    ofGameLoop, 173
    ofGetReplType, 173
    ofGetWindowSize, 173
    ofInit, 173, 174
    ofIsFullscreen, 174
    ofQuit, 174
    ofQuitFlagRaised, 174
    ofReplEval, 174
    ofReplType, 172
    ofSetClearColor, 175
    ofSetFullscreen, 175
    ofSetReplType, 175
    ofSetSwapInterval, 175
    ofSetWindowSize, 176
    ofSoftStop, 176
oficina::ofAnimator, 51
    draw, 52
    GetAnimationRunning, 53
    GetAnimationSpeed, 53
    GetCurrentAnimationName, 53
    GetDefaultAnimationSpeed, 53
    getPosition, 54
    init, 54
    isInit, 54
    reg, 55
    SetAnimation, 55
    SetAnimationRunning, 56
    SetAnimationSpeed, 56
    SetAnimationTexture, 57
    setPosition, 57
    SetRenderer, 57
    SyncToFrameRate, 57
    unreg, 58
    update, 58
oficina::ofBuffer, 58
    getName, 60
    isInit, 60
    operator=, 60
    setData, 60
oficina::ofCanvas, 61
    init, 62
    load, 62
    remove, 62
    update, 62
oficina::ofCanvasManager, 63
    add, 64
    dbg_ChangeState, 65
    dbg_ReplLineNumber, 65
    dbg_ReplOutStream, 66
    dbg_callEval, 65
    dbg_getState, 65

dbg_setFont, 66
draw, 66
getCanvasList, 66
ofDebuggerState, 64
remove, 67
unload, 67
update, 67
oficina::ofContext, 68
getViewportSize, 68
isInit, 69
open, 69
pushArg, 69
setClearColor, 70
setViewportSize, 70
oficina::ofDisplay, 70
close, 71
getDeltaTime, 71
getHandle, 72
getSize, 72
isFullscreen, 72
isOpen, 72
open, 73
pushArg, 73
setFullscreen, 73
setSize, 74
setSwapInterval, 74
swap, 74
oficina::ofElementBuffer, 75
draw, 75
getCount, 76
getType, 76
setCount, 76
setProps, 76
setType, 77
oficina::ofEntity, 77
AddComponent, 79
draw, 79
DrawComponents, 80
GetComponent, 80
getEulerAngles, 80
getModelMatrix, 81
getName, 81
getPosition, 81
getProperty, 81
getPropertyMask, 82
getScale, 82
init, 82
load, 82
RemoveComponent, 83
rotate, 84
rotation, 86
scale, 84
scaling, 86
setName, 84
setProperty, 85
toggleProperty, 85
translate, 85
translation, 87
update, 86
UpdateComponents, 86
oficina::ofFont, 87
getGlyphSize, 88
init, 88
isInit, 88
measure, 89
operator=, 89
write, 89
oficina::ofFrameSpan, 90
isRunning, 91
resetSpan, 91
stop, 91
yieldSpan, 91
oficina::ofIComponent, 92
draw, 93
getType, 93
oficina::ofInputState, 94
oficina::ofLua, 94
getBoolean, 96
getComponent, 96
getEntity, 96
getInteger, 97
getNumber, 97
getString, 97
isInit, 98
loadfile, 98
regFunc, 98
regSym, 99, 100
update, 101
oficina::ofPrimitive, 101
oficina::ofPrimitiveRenderer, 102
draw, 102
makePrimitive, 103
oficina::ofScheme, 103
getSymRef, 104
isInit, 105
loadfile, 105
regFunc, 105
regSym, 106
update, 106
oficina::ofShader, 107
compile, 108
getName, 108
init, 108
isCompiled, 108
isInit, 109
operator=, 109
setSource, 109
oficina::ofShaderAttribute, 110
bindVertexArrayData, 111
getSize, 111
getStride, 111
getType, 111
isAutoNormalizing, 112
isValid, 112
operator=, 112
setAutoNormalize, 113

setProps, 113
setSize, 113
setStride, 114
setType, 114
oficina::ofShaderProgram, 114
attach, 115
attachUnload, 116
bindFragmentDataLocation, 116
getAttributeLocation, 116
getName, 117
getUniformLocation, 117
isInit, 117
isLinked, 117
link, 118
operator=, 118
use, 118
oficina::ofShaderUniform, 119
isValid, 120
operator=, 120
set, 121–126
oficina::ofTexture, 127
bind, 128
getFileName, 128
getSize, 128
isLoaded, 128
operator(), 129
operator=, 129
unbind, 129
oficina::ofTexturePool, 130
load, 130, 131
loadDefaultFont, 131
unload, 131
oficina::ofTextureRenderer, 133
init, 133
isInit, 134
operator=, 134
render, 134
SetTexture, 135
unload, 135
oficina::ofTimeSpan, 136
isRunning, 136
resetSpan, 136
stop, 137
yieldSpan, 137
oficina::ofVertexArray, 137
draw, 138
operator=, 139
oficina::ofVertexBuffer, 139
oflua.hpp, 177, 185
ofLuaDefineFunc, 179
ofLuaDefineSymbol, 179–181
ofLuaEval, 181
ofLuaGetBoolean, 181
ofLuaGetComponent, 182
ofLuaGetEntity, 182
ofLuaGetInteger, 183
ofLuaGetNumber, 183
ofLuaGetString, 183
ofLuaIsInit, 184
ofLuaUndefine, 184
ofscheme.hpp, 186, 191
ofScmDefineFunc, 187
ofScmDefineSymbol, 187, 188
ofScmEval, 189
ofScmGetReference, 189
ofScmIsInit, 189
ofScmUndefine, 189
open
oficina::ofContext, 69
oficina::ofDisplay, 73
operator()
oficina::ofTexture, 129
operator=
oficina::ofBuffer, 60
oficina::ofFont, 89
oficina::ofShader, 109
oficina::ofShaderAttribute, 112
oficina::ofShaderProgram, 118
oficina::ofShaderUniform, 120
oficina::ofTexture, 129
oficina::ofTextureRenderer, 134
oficina::ofVertexArray, 139

platform.hpp, 191, 193
pushArg
oficina::ofContext, 69
oficina::ofDisplay, 73

reg
oficina::ofAnimator, 55
regFunc
oficina::ofLua, 98
oficina::ofScheme, 105
regSym
oficina::ofLua, 99, 100
oficina::ofScheme, 106
remove
oficina::ofCanvas, 62
oficina::ofCanvasManager, 67
RemoveComponent
oficina::ofEntity, 83
render
oficina::ofTextureRenderer, 134
render.hpp, 194, 201
ofBufferUsage, 196
ofContextType, 197
ofDataType, 197
ofDefaultShaderSrc_FS, 200
ofDefaultShaderSrc_VS, 200
ofFontFaces, 197
ofLoadDefaultFragShader, 199
ofLoadDefaultShaderProgram, 199
ofLoadDefaultVertexShader, 199
ofPrimitiveType, 198
ofSetVSync, 199
ofShaderType, 198
resetSpan

oficina::ofFrameSpan, 91
oficina::ofTimeSpan, 136
rotate
oficina::ofEntity, 84
rotation
oficina::ofEntity, 86

scale
oficina::ofEntity, 84
scaling
oficina::ofEntity, 86
set
oficina::ofShaderUniform, 121–126
SetAnimation
oficina::ofAnimator, 55
SetAnimationRunning
oficina::ofAnimator, 56
SetAnimationSpeed
oficina::ofAnimator, 56
SetAnimationTexture
oficina::ofAnimator, 57
setAutoNormalize
oficina::ofShaderAttribute, 113
setClearColor
oficina::ofContext, 70
setCount
oficina::ofElementBuffer, 76
setData
oficina::ofBuffer, 60
setFullscreen
oficina::ofDisplay, 73
setName
oficina::ofEntity, 84
setPosition
oficina::ofAnimator, 57
setProperty
oficina::ofEntity, 85
setProps
oficina::ofElementBuffer, 76
oficina::ofShaderAttribute, 113
SetRenderer
oficina::ofAnimator, 57
setSize
oficina::ofDisplay, 74
oficina::ofShaderAttribute, 113
setSource
oficina::ofShader, 109
setStride
oficina::ofShaderAttribute, 114
setSwapInterval
oficina::ofDisplay, 74
SetTexture
oficina::ofTextureRenderer, 135
setType
oficina::ofElementBuffer, 77
oficina::ofShaderAttribute, 114
setViewportSize
oficina::ofContext, 70
stop
oficina::ofFrameSpan, 91
oficina::ofTimeSpan, 137
swap
oficina::ofDisplay, 74
SyncToFrameRate
oficina::ofAnimator, 57

timer.hpp, 207
toggleProperty
oficina::ofEntity, 85
translate
oficina::ofEntity, 85
translation
oficina::ofEntity, 87
types.hpp, 208, 209
ofClamp, 209

unbind
oficina::ofTexture, 129
unload
oficina::ofCanvasManager, 67
oficina::ofTexturePool, 131
oficina::ofTextureRenderer, 135
unreg
oficina::ofAnimator, 58
update
oficina::ofAnimator, 58
oficina::ofCanvas, 62
oficina::ofCanvasManager, 67
oficina::ofEntity, 86
oficina::ofLua, 101
oficina::ofScheme, 106
UpdateComponents
oficina::ofEntity, 86
use
oficina::ofShaderProgram, 118

write
oficina::ofFont, 89

yieldSpan
oficina::ofFrameSpan, 91
oficina::ofTimeSpan, 137