OficinaFramework

2.0.0a

Generated by Doxygen 1.8.13

Thu Feb 23 2017 15:46:07

Contents

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 Dependancies

- 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

This engine also uses code from TinyScheme project by Dimitrios Souflis, (c) 2000. See src/oficina2/scheme/COPYING for details.

1.4 Building

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

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

2 ofScheme API Reference

2.1 General Scheme Syntax

ofScheme is a custom Scheme, based on the TinyScheme API. Therefore, all of R5RS Scheme specifications are already built-in. You can refer to the R5RS paper or to TinyScheme's manual for more information.

2.2 ofScheme 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-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
```

2.2.2 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).

2.2.2.1 Output

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

2.2.2.1.1 display

Displays a string on the REPL's output.

```
(display string)
```

2.2.2.1.2 print-hex

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

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

2.2.2.1.3 newline

Inputs new line on REPL's output

```
(newline)
```

2.2.2.1.4 clear

Clears REPL's output

```
(clear)
```

2.2.2.1.5 canvas-list

Shows information on currently loaded canvases.

```
(canvas-list)
```

2.2.2.1.6 quit

Soft stops the entire engine and quits game.

```
(quit)
```

2.2.2.2 Input

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

2.2.2.2.1 lstick?

Gets player's left stick. Returns an actual vector with two real coordinates ranging from -1.0 to 1.0. Use vector-ref to access each coordinate.

```
(lstick? player)
```

2.2.2.2.2 rstick?

Gets player's right stick. Refer to lstick? for usage tips.

```
(rstick? player)
```

2.2.2.2.3 trigger?

Gets a controller's trigger pressing ratio value, for a specific player's controller. Ranges from 0.0 to 1.0, depending on how much the trigger is being pressed.

```
(trigger? which player)
```

2.2.2.2.4 btnpress?

Gets whether a button is being held at a specific player's controller. Returns #t or #f.

```
(btnpress? which player)
```

2.2.2.2.5 btntap?

Gets whether a button was pressed on the current frame. Different from btnpress?, a btntap? only lasts for one single frame. Also returns #t or #f.

```
(btntap? which player)
```

2.2.2.2.6 mousepos?

Gets the current mouse position. Returns a vector with two real values representing screen coordinates.

```
(mousepos?)
```

2.2.2.2.7 mousepress?

Gets whether a mouse button is being held. Returns #t or #f.

```
(mousepress? which)
```

2.2.2.2.8 mousetap?

Gets whether a mouse button was tapped. To understand the difference between a press and a tap, please refer to btntap?. Also returns #t or #f.

```
(mousetap? which)
```

2.2.2.3 Display

Display-related stuff to get useful information regarding stuff, such as screen size, etc.

2.2.2.3.1 vwprt?

Gets a vector of two integers containing the current viewport size.

```
(vwptr?)
```

2.2.2.3.2 set-fullscreen!

Sets the fullscreen state of the global display. State can be #t or #f.

```
(set-fullscreen! state)
```

2.2.2.3.3 fullscreen?

Gets the fullscreen state of the global display. Returns #t or #f.

```
(fullscreen?)
```

2.2.3 Object API

These functions are only available for Schemes executing within entities (class oficina::ofScheme).

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+

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.

```
+this+
```

2.2.3.2 Object transformation

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

2.2.3.2.1 trl!

Translates object to/by a coordinate.

Parameters

coord	A LIST of exactly three numeric values.
load-identity	Whether the positioning matrix must be reset before positioning.
objref	Reference to object or +this+.

```
(trl! coord load-identity objref)
```

2.2.3.2.2 rot!

Rotates object by an angle around a specified axis.

Parameters

theta	Angle of rotation, in radians.
vector	A LIST with three numbers representing the axis of rotation.
load-identity	Whether the positioning matrix must be reset before positioning.
objref	Reference to object or +this+.

```
(rot! theta vector load-identity objref)
```

2.2.3.2.3 scl!

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

vector	A LIST of exactly three numeric values.
load-identity	Whether the positioning matrix must be reset before positioning.
objref	Reference to object or +this+.

```
(scl! vector load-identity objref)
```

2.2.3.2.4 pos?

Gets an object's position.

Parameters

objref	Reference to object or +this+.

Returns

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

```
(pos? objref)
```

2.2.3.2.5 eulerangle?

Gets the Euler angle related to a specific rotated axis.

Parameters

axis	Desired axis of rotation to reference.
objref	Reference to object or +this+.

Returns

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

```
(eulerangle? axis objref)
```

2.2.3.2.6 mag?

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

Parameters

axis	Desired axis of rotation to reference.
objref	Reference to object or +this+.

Returns

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

```
(mag? axis objref)
```

2.2.3.2.7 propset!

Sets a specific property to true or false.

Parameters

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

```
(propset! which state objref)
```

2.2.3.2.8 proptog!

Toggles a specific property's state.

Parameters

which	Property index, ranging from 0 to 31
objref	Reference to object or +this+.

```
(proptog! which objref)
```

2.2.3.2.9 propget?

Gets whether a property is active or inactive.

Parameters

which	Property index, ranging from 0 to 31
objref	Reference to object or +this+.

Returns

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

```
(propget? which objref)
```

2.2.3.2.10 propmask?

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

Parameters

objref	Reference to object or +this+.

Returns

An integer value containing the properties mask of an object.

```
(propmask? objref)
```

2.3 Usage Guide

2.3.1 Basic Example

Every script needs two functions defined to work properly: (init) and (update dt). Below is an example of an empty script with those requirements:

```
(define init
  (lambda ()
    #t))

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

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

```
(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:

```
(define *rotation-speed* 0.5)

(define init
  (lambda ()
    #t))

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

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.

3 Hierarchical Index

3.1 Class Hierarchy

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

4 Class Index

4.1 Class List

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

5 File Index

5.1 File List

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

6 Class Documentation

6.1 oficina::ofAnimator Class Reference

Tool for controlling a texture renderer to generate animations.

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

Public Member Functions

- void init (ofTexture t, glm::uvec2 frameSize, bool manageTexture=false)

  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, float magnification=1.0f)

  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.
- 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.

6.1.1 Detailed Description

Tool for controlling a texture renderer to generate animations.

Definition at line 808 of file render.hpp.

6.1.2 Member Function Documentation

6.1.2.1 draw()

```
void oficina::ofAnimator::draw (
            glm::mat4 ViewProjection,
            float magnification = 1.0f )
```

Draws the animation.

Parameters

ViewProjection	View-Projection matrix.

Note

If you wish to ignore the animator's own positioning controls, you can pass a whole Model-View-Projection matrix here.

Parameters

magnification	Scaling of the animation. Defaults to 1.0 (default frame size).

6.1.2.2 GetAnimationRunning()

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

Checks if the animation is currently running.

Returns

Whether the animation is running or not.

6.1.2.3 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

6.1.2.4 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

6.1.2.5 getPosition()

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

Yields the position of the animation on the matrix.

Returns

A 2D vector containing the animation position.

6.1.2.6 init()

```
void oficina::ofAnimator::init (
            ofTexture t,
            glm::uvec2 frameSize,
            bool manageTexture = false )
```

Initializes the animator.

Parameters

t	Texture atlas containing the animation frames.
frameSize	Frame size, as per ofTextureRenderer specification.

See also

ofTextureRenderer

Parameters

manageTexture	Whether the given texture should be managed by this tool (disposed when the animator is disposed).

6.1.2.7 isInit()

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

Checks if the animator was initialized.

Returns

Whether the animator was initialized or not.

6.1.2.8 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

6.1.2.9 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.

6.1.2.10 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.

6.1.2.11 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.

6.1.2.12 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.

6.1.2.13 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.

6.1.2.14 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.

6.1.2.15 unreg()

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

Unregisters an animation.

Parameters

animName	Desired animation to unregister.

6.1.2.16 update()

```
void oficina::ofAnimator::update (
            float dt )
```

Updates the animation step.

Parameters

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

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

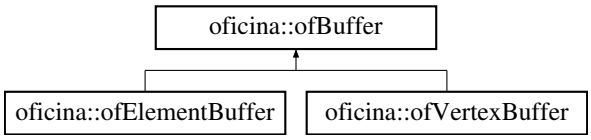
- render.hpp

6.2 oficina::ofBuffer Class Reference

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

`#include <render.hpp>`

Inheritance diagram for oficina::ofBuffer:



Public Member Functions

- virtual void init () final

  Initializes (generates) the buffer.
- virtual void unload () final

  Unloads (deletes) the buffer.
- virtual void bind () final

  Binds the buffer.
- virtual void unbind () final

  Unbinds all buffers of this type.
- virtual void setData (size_t dataSize, void ∗data, ofBufferUsage usage)

  Sets the data present on this buffer.
- ofBuffer & operator= (const ofBuffer &other)

  "Equals" operator for cloning the buffer.
- virtual bool isInit () const final

  Checks for buffer's initialization.
- virtual GLuint getName () const final

  Gets the buffer's real name on the GPU.

Protected Attributes

- GLenum m_type = GL_ARRAY_BUFFER

  Type of this buffer. Redefine this on the constructor if you need a different type of buffer.
- GLuint m_name = 0u

  Buffer's real name (on the GPU).

6.2.1 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 229 of file render.hpp.

6.2.2 Member Function Documentation

6.2.2.1 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.

6.2.2.2 isInit()

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

Checks for buffer's initialization.

Returns

Whether the buffer was initialized or not.

6.2.2.3 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.

6.2.2.4 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

6.3 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**

6.3.1 Detailed Description

Default interface for creating and managing canvases.

Definition at line 38 of file canvas.hpp.

6.3.2 Member Function Documentation

6.3.2.1 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.

6.3.2.2 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.

6.3.2.3 remove()

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

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

See also

ofCanvasManager

6.3.2.4 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

6.4 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::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.

6.4.1 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 85 of file canvas.hpp.

6.4.2 Member Enumeration Documentation

6.4.2.1 ofDebuggerState

```
enum oficina::ofCanvasManager::ofDebuggerState
```

State of the Debugger.

Enumerator

ofDebuggerVars	Disabled.
ofDebuggerRepl	Variable Watcher Mode.

Definition at line 89 of file canvas.hpp.

6.4.3 Member Function Documentation

6.4.3.1 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.

6.4.3.2 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

6.4.3.3 dbg_ChangeState()

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

Cycles through the debugger's state orderly.

See also

ofDebuggerState

6.4.3.4 dbg_getState()

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

Current state of the debugger.

See also

ofDebuggerState

6.4.3.5 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.

6.4.3.6 draw()

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

Draws all canvases registered within the manager.

Note

This method should always be called after "update".

6.4.3.7 getCanvasList()

```
static std::string oficina::ofCanvasManager::getCanvasList ( )  [static]
```

Yields text information regarding the canvas list.

Returns

A multiline string containing info on the canvas list.

6.4.3.8 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.

6.4.3.9 unload()

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

Unloads the manager.

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

6.4.3.10 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

6.5 oficina::ofContext Class Reference

Describes a context for your display.

`#include <render.hpp>`

Public Member Functions

- 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.

6.5.1 Detailed Description

Describes a context for your display.

Definition at line 191 of file render.hpp.

6.5.2 Member Function Documentation

6.5.2.1 getViewportSize()

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

Yields the current viewport size.

Returns

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

6.5.2.2 isInit()

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

Checks for context initialization.

Returns

Whether the context was opened or not.

6.5.2.3 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.

6.5.2.4 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

6.6 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.
- bool isFullscreen () const
  Gets the state of the window (fullscreen/windowed).
- void setFullscreen (bool state)
  Sets the state of the window on screen.

6.6.1 Detailed Description

Represents a single window prepared for receiving a context.

See also

ofContext

Definition at line 36 of file display.hpp.

6.6.2 Member Function Documentation

6.6.2.1 close()

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

Closes the display.

Closes the display, effectively closing the window.

6.6.2.2 getHandle()

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

Retrieves a low-level handle for the display.

Returns

an SDL2 window pointer.

6.6.2.3 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.

6.6.2.4 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.

6.6.2.5 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.

6.6.2.6 open()

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

Opens the display.

Opens the display, effectively initializing the window.

6.6.2.7 pushArg()

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

Handles display arguments.

Handles display arguments for display configuration, such as size, name, etc.

Parameters

arg	Argument to be treated and added to the configuration.

6.6.2.8 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.

6.6.2.9 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.

6.6.2.10 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:

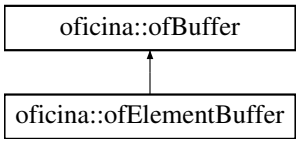
- display.hpp

6.7 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

6.7.1 Detailed Description

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

Definition at line 282 of file render.hpp.

6.7.2 Member Function Documentation

6.7.2.1 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.

6.7.2.2 getCount()

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

Yields the amount of elements stored.

Returns

Amount of buffer elements.

6.7.2.3 getType()

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

Yields the type of data stored.

Returns

Type of data used by the elements.

6.7.2.4 setCount()

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

Defines the amount of elements fed to the object.

Parameters

count	Amount of elements.

6.7.2.5 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.

6.7.2.6 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

6.8 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 ()

  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

  String holding the entity's actual name.

6.8.1 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 70 of file entity.hpp.

6.8.2 Member Function Documentation

6.8.2.1 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.

6.8.2.2 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.

6.8.2.3 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.

6.8.2.4 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.

6.8.2.5 getModelMatrix()

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

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

Returns

This entity's model matrix.

6.8.2.6 getName()

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

Yields the name of this entity.

Returns

A string containing this entity's name.

6.8.2.7 getPosition()

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

Yields the entity's position.

Returns

This entity's position in a 3D vector.

6.8.2.8 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.

6.8.2.9 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.

6.8.2.10 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.

6.8.2.11 init()

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

Initializes logic for this entity.

Note

This method should be called before "load".

6.8.2.12 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".

6.8.2.13 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.

6.8.2.14 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.

6.8.2.15 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.

6.8.2.16 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.

6.8.2.17 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.

6.8.2.18 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.

6.8.2.19 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.

6.8.2.20 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.

6.8.2.21 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.

6.8.3 Member Data Documentation

6.8.3.1 rotation

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

The rotation matrix.

Note

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

Definition at line 194 of file entity.hpp.

6.8.3.2 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 198 of file entity.hpp.

6.8.3.3 translation

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

The translation matrix.

Note

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

Definition at line 190 of file entity.hpp.

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

- entity.hpp

6.9 oficina::ofFont Class Reference

Represents a font.

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

Public Member Functions

- void init (ofTexture fontTexture, glm::uvec2 glyphSize, bool manageTexture=false)
  Initializes the font.
- void write (std::string text, glm::vec2 position, glm::mat4 mvp, glm::vec4 color=glm::vec4(1.0f), float mag=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.
- ofFont & operator= (const ofFont &other)
  "Equals" operator for cloning fonts.
- bool isInit () const
  Checks if the font was initialized.

6.9.1 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 765 of file render.hpp.

6.9.2 Member Function Documentation

6.9.2.1 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.

6.9.2.2 isInit()

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

Checks if the font was initialized.

Returns

Whether the font was initialized or not.

6.9.2.3 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.

6.9.2.4 write()

```
void oficina::ofFont::write (
            std::string text,
            glm::vec2 position,
            glm::mat4 mvp,
            glm::vec4 color = glm::vec4(1.0f),
            float mag = 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}.
mag	Magnitude (scaling) of the drawn text glyphs. Defaults to 1.0f (default glyph size).

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

- render.hpp

6.10 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.

6.10.1 Detailed Description

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

Definition at line 62 of file timer.hpp.

6.10.2 Member Function Documentation

6.10.2.1 isRunning()

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

Yields the state of the frame count.

Returns

Whether the frame count is running or not.

6.10.2.2 resetSpan()

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

Resets the frame counting.

Returns

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

6.10.2.3 stop()

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

Stops the frame counting.

Returns

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

6.10.2.4 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:

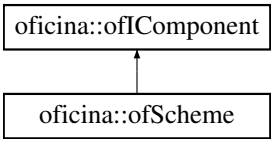
- timer.hpp

6.11 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 ()
  Draws the component. Overriding is optional.

Protected Attributes

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

Friends

- class **ofEntity**

6.11.1 Detailed Description

Defines a single component to be attached to an entity.

See also

ofEntity

Definition at line 38 of file entity.hpp.

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

- entity.hpp

6.12 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.

6.12.1 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:

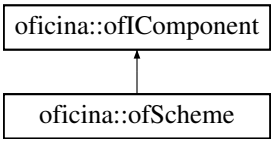
- input.hpp

6.13 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 filename)
  Loads and evaluates an actual script file. You can also reload your script at runtime with this function, if needed.
- void unload ()
  Disposes the script object.
- void update (float dt)
  Calls the script object's update function, if existing.
- void regFunc (std::string symbol, foreign_func fun)
  Defines/registers a foreign function on the script object.

Additional Inherited Members

6.13.1 Detailed Description

Defines one Scheme environment to be used inside an entity.

Definition at line 75 of file ofscheme.hpp.

6.13.2 Member Function Documentation

6.13.2.1 loadfile()

```
void oficina::ofScheme::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 ofScheme API Reference for details.

6.13.2.2 regFunc()

```
void oficina::ofScheme::regFunc (
            std::string symbol,
            foreign_func fun )
```

Defines/registers a foreign function on the script object.

Parameters

symbol	Name of the function to be defined.
fun	Function pointer to be used. Also accepts lambdas, but not closures (e.g. lambdas with captures).

6.13.2.3 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

6.14 oficina::ofShader Class Reference

Describes a shader.

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

Public Member Functions

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

Protected Attributes

- ofShaderType m_type = ofShaderFragment
  Type of shader.
- GLuint m_name = 0u
  True name of shader on the GPU.
- bool m_srcassign = false
  Whether the shader source code was assigned.
- bool m_compiled = false
  Whether the shader was compiled.

6.14.1 Detailed Description

Describes a shader.

Definition at line 324 of file render.hpp.

6.14.2 Member Function Documentation

6.14.2.1 compile()

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

Compiles the shader.

Warning

You must define a source for the shader before.

6.14.2.2 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.

6.14.2.3 init()

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

Initializes (generates) the shader.

Parameters

type	Type of shader to be used.

6.14.2.4 isCompiled()

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

Checks if the shader was compiled.

Returns

Whether the shader was compiled or not.

6.14.2.5 isInit()

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

Checks if the shader was initialized.

Returns

Whether the shader was initialized or not.

6.14.2.6 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.

6.14.2.7 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

6.15 oficina::ofShaderAttribute Class Reference

Represents the location of an attribute for the program shader.

`#include <render.hpp>`

Public Member Functions

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

Friends

- class **ofShaderProgram**

6.15.1 Detailed Description

Represents the location of an attribute for the program shader.

Definition at line 369 of file render.hpp.

6.15.2 Member Function Documentation

6.15.2.1 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.

6.15.2.2 getSize()

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

Yields the size of the attribute.

Returns

Attribute size.

6.15.2.3 getStride()

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

Yields the stride of the attribute.

Returns

Attribute stride.

6.15.2.4 getType()

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

Yields the data type of the attribute.

Returns

Attribute data type.

6.15.2.5 isAutoNormalizing()

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

Yields the automatic normalization state of the attribute.

Returns

Whether the attribute automatically normalizes or not.

6.15.2.6 isValid()

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

Checks if the attribute is valid.

Returns

Whether the attribute is valid or not.

6.15.2.7 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.

6.15.2.8 setAutoNormalize()

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

Defines if the attribute should be automatically normalized.

Parameters

state	Whether the attribute should be automatically normalized.

6.15.2.9 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.

6.15.2.10 setSize()

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

Defines the size of the attribute.

Parameters

s	Size to be given to the attribute.

6.15.2.11 setStride()

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

Defines the stride of the attribute on the vertex data.

Parameters

stride	Stride of the attribute.

6.15.2.12 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

6.16 oficina::ofShaderProgram Class Reference

Represents a shader program.

`#include <render.hpp>`

Public Member Functions

- void init ()
  Initializes (generates) the shader program.
- void unload ()
  Unloads (deletes) the shader program.
- void attach (const ofShader &shader)
  Attaches a shader to the shader program.
- void attachUnload (ofShader &shader)
  Attaches a shader to the shader program and unloads the shader if attachment was successful.
- void bindFragmentDataLocation (std::string name, ofdword colorNumber=0u)
  Binds a fragment shader output data location.
- void link ()
  Links the shader program.
- void use ()
  Uses this shader program.
- void unuse ()
  Stops using any shader program that is in use.
- bool isInit () const
  Checks if shader program was initialized.
- bool isLinked () const
  Checks if shader program was linked.
- GLuint getName () const
  Yields the shader program's real name.
- ofShaderProgram & operator= (const ofShaderProgram &program)
  "Equals" operator for cloning a shader program.
- ofShaderAttribute getAttributeLocation (std::string name)
  Retrieves the attribute location of a shader attribute.
- ofShaderUniform getUniformLocation (std::string name)
  Retrieves the uniform location of a shader uniform.

6.16.1 Detailed Description

Represents a shader program.

Definition at line 530 of file render.hpp.

6.16.2 Member Function Documentation

6.16.2.1 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.

6.16.2.2 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.

6.16.2.3 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.

6.16.2.4 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.

6.16.2.5 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.

6.16.2.6 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.

6.16.2.7 isInit()

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

Checks if shader program was initialized.

Returns

Whether the shader program was initialized or not.

6.16.2.8 isLinked()

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

Checks if shader program was linked.

Returns

Whether the shader program was linked or not.

6.16.2.9 link()

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

Links the shader program.

Warning

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

6.16.2.10 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.

6.16.2.11 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

6.17 oficina::ofShaderUniform Class Reference

Represents and handles a shader's uniform.

`#include <render.hpp>`

Public Member Functions

- bool isValid () const
  Checks if the uniform is valid.
- ofShaderUniform & operator= (const ofShaderUniform &uniform)
  "Equals" operator for cloning the uniform.
- void set (float value)
  Sets the value of the uniform.
- void set (glm::vec2 value)
  Sets the value of the uniform.
- void set (glm::vec3 value)
  Sets the value of the uniform.
- void set (glm::vec4 value)
  Sets the value of the uniform.
- void set (int value)
  Sets the value of the uniform.
- void set (glm::ivec2 value)
  Sets the value of the uniform.
- void set (glm::ivec3 value)
  Sets the value of the uniform.
- void set (glm::ivec4 value)
  Sets the value of the uniform.
- void set (unsigned int value)
  Sets the value of the uniform.
- void set (glm::uvec2 value)
  Sets the value of the uniform.
- void set (glm::uvec3 value)
  Sets the value of the uniform.
- void set (glm::uvec4 value)
  Sets the value of the uniform.
- void set (glm::mat2 value, bool transpose=false)
  Sets the value of the uniform.
- void set (glm::mat3 value, bool transpose=false)
  Sets the value of the uniform.
- void set (glm::mat4 value, bool transpose=false)
  Sets the value of the uniform.
- void set (glm::mat2x3 value, bool transpose=false)
  Sets the value of the uniform.
- void set (glm::mat3x2 value, bool transpose=false)
  Sets the value of the uniform.
- void set (glm::mat2x4 value, bool transpose=false)
  Sets the value of the uniform.
- void set (glm::mat4x2 value, bool transpose=false)
  Sets the value of the uniform.
- void set (glm::mat3x4 value, bool transpose=false)
  Sets the value of the uniform.
- void set (glm::mat4x3 value, bool transpose=false)
  Sets the value of the uniform.

Friends

- class **ofShaderProgram**

6.17.1 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 435 of file render.hpp.

6.17.2 Member Function Documentation

6.17.2.1 isValid()

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

Checks if the uniform is valid.

Returns

Whether the uniform is valid or not.

6.17.2.2 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.

6.17.2.3 set() [1/21]

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

Sets the value of the uniform.

Parameters

value	Float value.

6.17.2.4 set() [2/21]

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

Sets the value of the uniform.

Parameters

value	2D vector of float.

6.17.2.5 set() [3/21]

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

Sets the value of the uniform.

Parameters

value	3D vector of float.

6.17.2.6 set() [4/21]

```
void oficina::ofShaderUniform::set (
            glm::vec4 value )
```

Sets the value of the uniform.

Parameters

value	4D vector of float.

6.17.2.7 set() [5/21]

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

Sets the value of the uniform.

Parameters

value	Signed integer value.

6.17.2.8 set() [6/21]

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

Sets the value of the uniform.

Parameters

value	2D vector of signed integer.

6.17.2.9 set() [7/21]

```
void oficina::ofShaderUniform::set (
            glm::ivec3 value )
```

Sets the value of the uniform.

Parameters

value	3D vector of signed integer.

6.17.2.10 set() [8/21]

```
void oficina::ofShaderUniform::set (
            glm::ivec4 value )
```

Sets the value of the uniform.

Parameters

value	4D vector of signed integer.

6.17.2.11 set() [9/21]

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

Sets the value of the uniform.

Parameters

value	Unsigned integer value.

6.17.2.12 set() [10/21]

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

Sets the value of the uniform.

Parameters

value	2D vector of unsigned integer.

6.17.2.13 set() [11/21]

```
void oficina::ofShaderUniform::set (
            glm::uvec3 value )
```

Sets the value of the uniform.

Parameters

value	3D vector of unsigned integer.

6.17.2.14 set() [12/21]

```
void oficina::ofShaderUniform::set (
            glm::uvec4 value )
```

Sets the value of the uniform.

Parameters

value	4D vector of unsigned integer.

6.17.2.15 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.

6.17.2.16 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.

6.17.2.17 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.

6.17.2.18 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.

6.17.2.19 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.

6.17.2.20 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.

6.17.2.21 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.

6.17.2.22 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.

6.17.2.23 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

6.18 oficina::ofTexture Class Reference

Represents a texture on the GPU.

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

Public Member Functions

- void bind (ofword currentSampler=0)

  Binds the texture to be used.
- void unbind (ofword currentSampler=0)

  Unbinds any texture currently in use.
- ofTexture & operator= (const ofTexture &other)

  "Equals" operator to clone a texture.
- GLuint operator() ()

Parenthesis operator to retrieve the texture's real name.

- bool 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**

6.18.1 Detailed Description

Represents a texture on the GPU.

Definition at line 631 of file render.hpp.

6.18.2 Member Function Documentation

6.18.2.1 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.

6.18.2.2 getFileName()

```
std::string oficina::ofTexture::getFileName ( ) const
```

Yields the texture location on the game assets.

Returns

A string containing the texture file path.

6.18.2.3 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.

6.18.2.4 isLoaded()

```
bool oficina::ofTexture::isLoaded ( ) const
```

Checks if the texture is loaded.

Returns

Whether the texture is loaded or not.

6.18.2.5 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.

6.18.2.6 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.

6.18.2.7 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

6.19 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 ()
  Loads Oficina's default font (GohuFont 11).
- static void unload (ofTexture &t)
  Unloads a font.
- static void clear ()
  Unloads ALL textures on the pool.

6.19.1 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 678 of file render.hpp.

6.19.2 Member Function Documentation

6.19.2.1 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.

6.19.2.2 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.

6.19.2.3 loadDefaultFont()

```
static ofFont oficina::ofTexturePool::loadDefaultFont ( )  [static]
```

Loads Oficina's default font (GohuFont 11).

Returns

A reference to the default font.

6.19.2.4 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

6.20 oficina::ofTextureRenderer Class Reference

Tool for easily rendering 2D textures or texture atlases.

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

Public Member Functions

- void init (ofTexture t, glm::uvec2 frameSize=glm::uvec2(0, 0))

  Initializes the renderer.
- void render (glm::vec2 position, glm::mat4 mvp, ofdword frame=0u, glm::vec4 color=glm::vec4(1.0f), float mag=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.

6.20.1 Detailed Description

Tool for easily rendering 2D textures or texture atlases.

Definition at line 702 of file render.hpp.

6.20.2 Member Function Documentation

6.20.2.1 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.

6.20.2.2 isInit()

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

Checks for texture renderer initialization.

Returns

Whether the texture renderer was initialized or not.

6.20.2.3 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.

6.20.2.4 render()

```
void oficina::ofTextureRenderer::render (
            glm::vec2 position,
            glm::mat4 mvp,
            ofdword frame = 0u,
            glm::vec4 color = glm::vec4(1.0f),
            float mag = 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}.
mag	Magnitude (scaling) of the drawn texture or frame. Defaults to 1.0f (default frame size).

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.

6.20.2.5 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.

6.20.2.6 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

6.21 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.

6.21.1 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.

6.21.2 Member Function Documentation

6.21.2.1 isRunning()

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

Yields the state of the time span.

Returns

Whether the time span is running or not.

6.21.2.2 resetSpan()

```
float oficina::ofTimeSpan::resetSpan ( )
```

Resets the time span, effectively restarting from zero.

Returns

Time, in seconds, before the span was reset.

6.21.2.3 stop()

```
float oficina::ofTimeSpan::stop ( )
```

Stops the time span.

Returns

Time, in seconds, before the span was stopped.

6.21.2.4 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

6.22 oficina::ofVertexArray Class Reference

Represents a vertex array for binding shader and vertex data.

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

Public Member Functions

- void init ()
  Initializes (generates) the vertex array.
- void unload ()
  Unloads (deletes) the vertex array.
- void bind ()
  Binds the vertex array.
- void unbind ()
  Unbinds any bound vertex array.
- void draw (ofPrimitiveType mode, int firstVertexIdx, size_t vertexCount)
  Draws any primitive based on bound vertex buffer and vertex attributes.
- ofVertexArray & operator= (const ofVertexArray &other)
  "Equals" operator for cloning vertex arrays.

6.22.1 Detailed Description

Represents a vertex array for binding shader and vertex data.

Definition at line 596 of file render.hpp.

6.22.2 Member Function Documentation

6.22.2.1 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.

6.22.2.2 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:

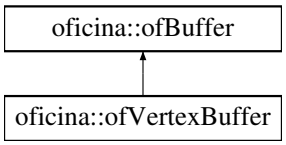
- render.hpp

6.23 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

6.23.1 Detailed Description

Represents a Vertex Buffer object (VBO). Use this to hold data related to drawing.

Definition at line 272 of file render.hpp.

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

- render.hpp

7 File Documentation

7.1 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.

7.1.1 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.

7.1.2 Function Documentation

7.1.2.1 ofBenchmarkIsRunning()

```
bool oficina::ofBenchmarkIsRunning ( )
```

Shows the benchmarking process status.

Returns

Whether benchmarking is active or not.

7.1.2.2 ofBenchmarkStart()

```
void oficina::ofBenchmarkStart (
            float spanTimeS )
```

Starts the benchmarking process.

Parameters

spanTimeS	Time between each benchmark debriefing.

7.2 benchmark.hpp

```
00001 /*************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                               *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
```

```
00010  *                                                                   *
00011  *  You should have received a copy of the GNU Lesser General        *
00012  *  Public License along with OficinaFramework.  If not, see         *
00013  *  <http://www.gnu.org/licenses/>.                                  *
00014  ********************************************************************/
00015
00027 #pragma once
00028
00029 #include <string>
00030
00031 namespace oficina
00032 {
00035     void ofBenchmarkStart(float spanTimeS);
00036
00039     void ofBenchmarkUpdateCall();
00040
00042     void ofBenchmarkEnd();
00043
00046     bool ofBenchmarkIsRunning();
00047 }
```

7.3 canvas.hpp File Reference

Tools for creating game scenes and manage such scenes.

```
#include <list>
#include <queue>
#include <sstream>
#include "oficina2/types.hpp"
```

Classes

- class oficina::ofCanvas

  Default interface for creating and managing canvases.
- class oficina::ofCanvasManager

  Static class for handling canvases in general.

7.3.1 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.

7.4 canvas.hpp

```
00001 /*****************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *****************************************************************/
00015
00026 #pragma once
00027 #include <list>
00028 #include <queue>
00029 #include <sstream>
00030
00031 #include "oficina2/types.hpp"
00032
00033 namespace oficina
00034 {
00035     // Pre-definition of ofCanvasManager so we can refer to it from ofCanvas.
00036     class ofCanvasManager;
00038     class ofCanvas
00039     {
00040         friend class ofCanvasManager;
00041     private:
00042         bool          m_init   = false;
00043         bool          m_load   = false;
00044         bool          m_remove = false;
00045         int           depth     = 0;
00046         std::string   m_name = "";
00047     public:
00049         virtual ~ofCanvas() {}
00053         virtual void init() = 0;
00058         virtual void load() = 0;
00060         virtual void unload() = 0;
00066         virtual void update(float dt) = 0;
00069         virtual void draw() = 0;
00073         virtual void remove() final;
00074     };
00075
00085     class ofCanvasManager
00086     {
00087     public:
00089         enum ofDebuggerState
00090         {
00091             ofDebuggerOff  = 0u,
00092             ofDebuggerVars = 1u,
00093             ofDebuggerRepl = 2u
00094         };
00095
00097         static void init();
00106         // TODO: Explain canvas depth in documentation
00107         static void add(ofCanvas* c, int depth = 0, std::string name = "");
00112         static void remove(ofCanvas* c);
00113         //
00118         static void unload();
00128         static void update(float dt);
00131         static void draw();
00132
00135         static std::string getCanvasList();
00136
00142         static std::ostringstream& dbg_ReplOutStream();
00145         static ofDebuggerState     dbg_getState();
00152         static void                dbg_callEval();
00155         static void                dbg_ChangeState();
00157         static void                dbg_ReplHistoryPrev();
00159         static void                dbg_ReplHistoryNext();
00160     private:
00161         class ofDebugCanvas : public ofCanvas
00162         {
00163         public:
00164             void init();
00165             void load();
00166             void unload();
00167             void update(float dt);
00168             void draw();
00169         };
00170
00171         static ofDebugCanvas m_debugger;
```

```
00172     };
00173 }
```

7.5 display.hpp File Reference

Tools for configuring windows for video output.

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

Classes

- class oficina::ofDisplay

  Represents a single window prepared for receiving a context.

7.5.1 Detailed Description

Tools for configuring windows for video output.

Provides tools for creating displays (game windows).

Author

Lucas Vieira

Definition in file display.hpp.

7.6 display.hpp

```
00001 /*************************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *************************************************************************/
00015
00024 #pragma once
00025
00026 #include <SDL2/SDL.h>
00027 #include <list>
00028 #include <string>
00029 #include "oficina2/types.hpp"
00030
00031 namespace oficina
00032 {
00036     class ofDisplay
00037     {
00038     public:
00039         // TODO: Actually handle the display args...
00046         // TODO: Write docs for display config.
```

```
00047         void        pushArg(std::string arg);
00051         void        open();
00055         void        close();
00059         void        swap();
00060
00063         SDL_Window* getHandle() const;
00067         glm::uvec2  getSize() const;
00073         bool        isOpen() const;
00074
00080         void        setSize(glm::uvec2 NewSize);
00081
00086         bool isFullscreen() const;
00091         void setFullscreen(bool state);
00092     private:
00093         SDL_Window*            m_wnd = nullptr;
00094         std::list<std::string> m_confv;
00095         std::string            m_title = "OficinaFramework 2.0";
00096         glm::uvec2             m_size = glm::uvec2(1280u,
00097                                                    720u);
00098         bool                   m_full = false;
00099     };
00100 }
```

7.7 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.

7.7.1 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.

7.8 entity.hpp

```
00001 /*************************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *************************************************************************/
00015
00026 #pragma once
00027
00028 #include "oficina2/types.hpp"
00029 //#include "oficina2/ofscheme.hpp"
00030 #include <map>
00031
00032 namespace oficina
00033 {
00034     class ofEntity;
00038     class ofIComponent
00039     {
00040         friend class ofEntity;
00041     public:
00043         virtual ~ofIComponent()       {}
00046         virtual void init()           = 0;
00049         virtual void load()           {}
00052         virtual void unload()         {}
00055         virtual void update(float dt) = 0;
00058         virtual void draw()           {}
00059     protected:
00063         ofEntity* parent;
00064     };
00065
00070     class ofEntity
00071     {
00072     public:
00074         virtual ~ofEntity() {}
00077         virtual void init()           = 0;
00080         virtual void load()           = 0;
00082         virtual void unload()         = 0;
00087         virtual void update(float dt) = 0;
00096         virtual void draw(glm::mat4 ViewProjection) = 0;
00097
00104         void translate(glm::vec3 coord,
00105                        bool loadIdentity = false);
00112         void rotate(float theta,
00113                     glm::vec3 axis,
00114                     bool loadIdentity = false);
00120         void scale(glm::vec3 amount,
00121                    bool loadIdentity);
00122
00126         void setProperty(ofbyte which, bool state);
00129         void toggleProperty(ofbyte which);
00134         void setName(std::string name);
00135
00138         glm::mat4 getModelMatrix();
00141         glm::vec3 getPosition() const;
00144         glm::vec3 getEulerAngles() const;
00147         glm::vec3 getScale() const;
00151         bool getProperty(ofbyte which);
00155         ofdword getPropertyMask() const;
00158         std::string getName() const;
00159
00167         void AddComponent(std::string name, ofIComponent* component);
00171         ofIComponent* const GetComponent(std::string name);
00174         void RemoveComponent(std::string name);
00176         void ClearComponents();
00177
00182         void UpdateComponents(float dt);
00185         void DrawComponents();
00186     protected:
00190         glm::mat4 translation;
00194         glm::mat4 rotation;
00198         glm::mat4 scaling;
00199
00202         glm::vec3 position;
00205         glm::vec3 eulerangles;
00208         glm::vec3 magnification = glm::vec3(1.0f);
00209
```

```
00211         ofdword   propertymask = 0x00000000u;
00212
00214         std::map<std::string, ofIComponent*> components;
00215
00217         std::string name;
00218     };
00219
00220     /*class ofUniformHashGrid : public ofEntityCollection
00221     {
00222     public:
00223         void init(glm::uvec2 CellSize,
00224                   ofsdword hash_1 = 0x8da6b343,
00225                   ofsdword hash_2 = 0xd8163841,
00226                   ofsdword hash_3 = 0xcb1ab31f);
00227     private:
00228         bool m_initialized = false;
00229         glm::uvec2 m_cellsz;
00230         ofsdword h1, h2, h3;
00231     };*/
00232 }
```

7.9 input.hpp File Reference

Special tools for handling player input.

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

Classes

- struct oficina::ofInputState

  Holds an input state every frame.

Enumerations

- enum oficina::ofStick { oficina::ofStickLeft = 0x01u, oficina::ofStickRight = 0x02u }

  Enumeration for gamepad sticks.
- enum oficina::ofStickAxis { oficina::ofStickHoriz = 0x04u, oficina::ofStickVert = 0x08u }

  Enumeration for gamepad sticks' axis.
- enum oficina::ofStickSignal { oficina::ofStickNegative = 0x10u, oficina::ofStickPositive = 0x20u }

  Enumeration for gamepad sticks' axis' signal/direction.
- enum oficina::ofPadButton {
  oficina::ofPadStart = 0x0001u, oficina::ofPadBack = 0x0002u, oficina::ofPadA = 0x0004u, oficina::ofPadB = 0x0008u,
  oficina::ofPadX = 0x0010u, oficina::ofPadY = 0x0020u, oficina::ofPadLS = 0x0040u, oficina::ofPadRS = 0x0080u,
  oficina::ofPadDUp = 0x0100u, oficina::ofPadDDown = 0x0200u, oficina::ofPadDLeft = 0x0400u, oficina::ofPadDRight = 0x0800u,
  oficina::ofPadLB = 0x1000u, oficina::ofPadLT = 0x2000u, oficina::ofPadRB = 0x4000u, oficina::ofPadRT = 0x8000u }

  Enumeration for gamepad buttons. Layout based on Xbox 360 controller.
- enum oficina::ofMouseButton { oficina::ofMouseLeft = 0x01u, oficina::ofMouseMid = 0x02u, oficina::ofMouseRight = 0x04u }

  Enumeration representing mouse buttons.
- enum oficina::ofPlayer { oficina::ofPlayerOne = 0u, oficina::ofPlayerTwo = 1u, oficina::ofPlayerThree = 2u, oficina::ofPlayerFour = 3u }

  Enumeration representing connected players.

Functions

- void oficina::ofUpdateEventDispatch ()

  Updates and dispatches input events.
- ofInputState oficina::ofGetInputState (ofPlayer player=ofPlayerOne)

  Grabs the whole of the current input state in a single struct.
- bool oficina::ofIsGamepadConnected (ofPlayer player=ofPlayerOne)

  Yields the state of a player's gamepad.
- glm::vec2 oficina::ofGetLeftStick (ofPlayer player=ofPlayerOne)

  Yields the gamepad's left stick coordinates. Each coordinate yields from -1.0f (left/up) to 1.0f (right/down).
- glm::vec2 oficina::ofGetRightStick (ofPlayer player=ofPlayerOne)

  Yields the gamepad's right stick coordinates. Each coordinate yields from -1.0f (left/up) to 1.0f (right/down).
- float oficina::ofGetLeftTrigger (ofPlayer player=ofPlayerOne)

  Yields a value stating the amount of pressing on the gamepad's left trigger. Value ranges from 0.0f (not pressed) to 1.0f (fully held).
- float oficina::ofGetRightTrigger (ofPlayer player=ofPlayerOne)

  Yields a value stating the amount of pressing on the gamepad's right trigger. Value ranges from 0.0f (not pressed) to 1.0f (fully held).
- bool oficina::ofButtonPress (ofPadButton button, ofPlayer player=ofPlayerOne)

  Yields the pressing state of a specific button on the gamepad.
- bool oficina::ofButtonTap (ofPadButton button, ofPlayer player=ofPlayerOne)

  Yields the tap state of a specific button on the gamepad.
- bool oficina::ofStickMovedTowards (ofbyte stickDirectionMask, ofPlayer player=ofPlayerOne)

  Checks if a specific stick was moved in a specific direction.
- glm::vec2 oficina::ofGetMousePos ()

  Yields the mouse position's coordinates inside the display.
- bool oficina::ofMouseButtonPress (ofMouseButton button)

  Yields the pressing state of a specific mouse button.
- bool oficina::ofMouseButtonTap (ofMouseButton button)

  Yields the tap state of a specific mouse button.
- void oficina::ofMapDefaultsP1 ()

  Maps default bindings for gamepad buttons on the keyboard - Player 1 only.
- void oficina::ofMapKeyToButton (ofPadButton button, SDL_Scancode scancode, ofPlayer player=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).

7.9.1 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.

7.9.2 Enumeration Type Documentation

7.9.2.1 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.

7.9.2.2 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.

7.9.2.3 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.

7.9.2.4 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.

7.9.2.5 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.

7.9.2.6 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.

7.9.3 Function Documentation

7.9.3.1 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.

7.9.3.2 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.

7.9.3.3 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

7.9.3.4 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.

7.9.3.5 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.

7.9.3.6 ofGetMousePos()

```
glm::vec2 oficina::ofGetMousePos ( )
```

Yields the mouse position's coordinates inside the display.

Returns

A 2D vector containing the mouse position.

7.9.3.7 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.

7.9.3.8 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.

7.9.3.9 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.

7.9.3.10 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.

7.9.3.11 ofIsInputtingText()

```
bool oficina::ofIsInputtingText ( )
```

Checks for the state of text input.

Returns

Whether the player is currently in text input mode or not.

7.9.3.12 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.

7.9.3.13 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

7.9.3.14 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.

7.9.3.15 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.

7.9.3.16 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.

7.9.3.17 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.

7.9.3.18 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

7.9.3.19 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.

7.9.3.20 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.

7.9.3.21 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.

7.9.3.22 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

7.9.3.23 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.

7.9.3.24 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.

7.9.3.25 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.

7.10 input.hpp

```
00001 /*************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
```

```
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  ****************************************************************/
00015
00026 #pragma once
00027
00028 #include "oficina2/types.hpp"
00029 #include <SDL2/SDL.h>
00030 #include <string>
00031
00032 namespace oficina
00033 {
00036     enum ofStick
00037     {
00039         ofStickLeft  = 0x01u,
00041         ofStickRight = 0x02u
00042     };
00043
00046     enum ofStickAxis
00047     {
00049         ofStickHoriz = 0x04u,
00051         ofStickVert  = 0x08u
00052     };
00053
00057     enum ofStickSignal
00058     {
00061         ofStickNegative = 0x10u,
00064         ofStickPositive = 0x20u
00065     };
00066
00070     enum ofPadButton
00071     {
00073         ofPadStart  = 0x0001u,
00075         ofPadBack   = 0x0002u,
00077         ofPadA      = 0x0004u,
00079         ofPadB      = 0x0008u,
00081         ofPadX      = 0x0010u,
00083         ofPadY      = 0x0020u,
00085         ofPadLS     = 0x0040u,
00087         ofPadRS     = 0x0080u,
00089         ofPadDUp    = 0x0100u,
00091         ofPadDDown  = 0x0200u,
00093         ofPadDLeft  = 0x0400u,
00095         ofPadDRight = 0x0800u,
00097         ofPadLB     = 0x1000u,
00103         ofPadLT     = 0x2000u,
00105         ofPadRB     = 0x4000u,
00111         ofPadRT     = 0x8000u
00112     };
00113
00116     enum ofMouseButton
00117     {
00119         ofMouseLeft  = 0x01u,
00121         ofMouseMid   = 0x02u,
00123         ofMouseRight = 0x04u
00124     };
00125
00129     enum ofPlayer
00130     {
00132         ofPlayerOne    = 0u,
00134         ofPlayerTwo    = 1u,
00136         ofPlayerThree  = 2u,
00138         ofPlayerFour   = 3u
00139     };
00140
00142     struct ofInputState
00143     {
00146         ofword    padButtons    = 0x0000u;
00149         float     leftStick[2]  = {0.0f, 0.0f};
00152         float     rightStick[2] = {0.0f, 0.0f};
00155         float     triggers[2]   = {0.0f, 0.0f};
00156     };
00157
00165     void         ofUpdateEventDispatch();
00170     ofInputState ofGetInputState(ofPlayer player =
      ofPlayerOne);
00177     bool         ofIsGamepadConnected(ofPlayer player =
      ofPlayerOne);
00182     glm::vec2    ofGetLeftStick(ofPlayer player =
      ofPlayerOne);
00187     glm::vec2    ofGetRightStick(ofPlayer player =
      ofPlayerOne);
```

```
00192     float        ofGetLeftTrigger(ofPlayer player =
      ofPlayerOne);
00197     float        ofGetRightTrigger(ofPlayer player =
      ofPlayerOne);
00205     bool         ofButtonPress(ofPadButton button,
      ofPlayer player = ofPlayerOne);
00215     bool         ofButtonTap(ofPadButton button, ofPlayer player =
      ofPlayerOne);
00228     bool         ofStickMovedTowards(ofbyte stickDirectionMask,
      ofPlayer player = ofPlayerOne);
00229
00232     glm::vec2    ofGetMousePos();
00239     bool         ofMouseButtonPress(ofMouseButton button);
00248     bool         ofMouseButtonTap(ofMouseButton button);
00249
00254
00283     void         ofMapDefaultsP1();
00292     void         ofMapKeyToButton(ofPadButton button, SDL_Scancode scancode,
      ofPlayer player = ofPlayerOne);
00305     void         ofMapKeyToStick(ofbyte stickPositionMask, SDL_Scancode scancode,
      ofPlayer player = ofPlayerOne);
00309     void         ofMapButtonRemove(ofPadButton button,
      ofPlayer player = ofPlayerOne);
00319     void         ofMapStickRemove(ofbyte stickPositionMask,
      ofPlayer player = ofPlayerOne);
00322     void         ofMappingClear(ofPlayer player =
      ofPlayerOne);
00323
00327     void         ofStartTextInput();
00334     std::string  ofGetTextInput();
00341     void         ofSetTextInput(std::string str);
00344     bool         ofIsInputtingText();
00348     void         ofStopTextInput();
00350     void         ofClearTextInput();
00357     void         ofTextInputSetPadding(ofdword padding);
00358 }
```

7.11 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 CONSOLE subsystem.
- void oficina::ofLogDisable ()
  Disable logging completely.
- 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.

7.11.1 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.

7.11.2 Enumeration Type Documentation

7.11.2.1 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.

7.11.2.2 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.

7.11.3 Function Documentation

7.11.3.1 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.

7.11.3.2 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.

7.11.3.3 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.

7.11.3.4 ofLogGetType()

```
ofLogType oficina::ofLogGetType ( )
```

Yields the currently used logging type.

Returns

Type of the current log tool.

7.11.3.5 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.

7.11.3.6 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.

7.12 io.hpp

```
00001 /*************************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *************************************************************************/
00015
00026 #pragma once
00027
00028 #include <cstdarg>
00029 #include <string>
00030 #include "oficina2/platform.hpp"
00031 #include <SDL2/SDL.h>
00032
00033 #if OF_PLATFORM == OF_PLATFORM_WINDOWS
00034         #define OFLOG_NRM   ""
00037         #define OFLOG_RED   ""
00040         #define OFLOG_GRN   ""
00043         #define OFLOG_YEL   ""
00046         #define OFLOG_BLU   ""
00049         #define OFLOG_MAG   ""
00052         #define OFLOG_CYN   ""
00055         #define OFLOG_WHT   ""
00058         #define OFLOG_RESET ""
00061 #else
00062         #define OFLOG_NRM   "\x1B[0m"
00065         #define OFLOG_RED   "\x1B[31m"
00068         #define OFLOG_GRN   "\x1B[32m"
00071         #define OFLOG_YEL   "\x1B[33m"
00074         #define OFLOG_BLU   "\x1B[34m"
00077         #define OFLOG_MAG   "\x1B[35m"
00080         #define OFLOG_CYN   "\x1B[36m"
00083         #define OFLOG_WHT   "\x1B[37m"
00086         #define OFLOG_RESET "\033[0m"
00089 #endif
00090
00091 namespace oficina
```

```
00092 {
00094     enum ofLogLvl {
00096         ofLogCrit = 0,
00098         ofLogErr  = 1,
00100         ofLogWarn = 2,
00102         ofLogInfo = 3,
00104         ofLogNone = 4
00105     };
00106
00108     enum ofLogType {
00110         ofLogDisabled = 0,
00112         ofLogConsole  = 1,
00114         ofLogFile     = 2
00115     };
00116
00117
00126     int  ofLog(ofLogLvl level, const char* fmt, ...);
00132     void ofLogSetLevel(ofLogLvl level);
00135     ofLogType ofLogGetType();
00140     void ofLogUseFile(std::string filename);
00144     void ofLogUseConsole();
00146     void ofLogDisable();
00147
00148
00152     std::string  ofLoadText(std::string filename);
00157     SDL_Surface* ofLoadImage(std::string filename);
00158 }
```

7.13 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/entity.hpp"
```

Macros

- #define OF_VERSION_STRING "2.0.0a"

  String banner containing the current version of OficinaFramework.

Functions

- void oficina::ofInit ()

  Initialized 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.

7.13.1 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.

7.13.2 Function Documentation

7.13.2.1 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

7.13.2.2 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.

7.13.2.3 ofInit()

```
void oficina::ofInit ( )
```

Initialized 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.

7.13.2.4 ofIsFullscreen()

```
bool oficina::ofIsFullscreen ( )
```

Checks for the fullscreen state of the application.

Returns

Whether the application is fullscreen or not.

7.13.2.5 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

7.13.2.6 ofQuitFlagRaised()

```
bool oficina::ofQuitFlagRaised ( )
```

Yields the state of the Soft Stop flag.

Returns

Whether the Soft Stop flag was raised or not.

7.13.2.7 ofSetFullscreen()

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

Changes the application's window state.

Parameters

state	State to be assumed: Fullscreen (true) or Windowed (false).

7.13.2.8 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.

7.13.2.9 ofSoftStop()

```
void oficina::ofSoftStop ( )
```

Raises a Soft Stop flag, which will quit the default Game Loop function.

See also

ofGameLoop

7.14 oficina.hpp

```
00001 /*************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *************************************************************/
00015
00025 #pragma once
00026
00027 #include "oficina2/display.hpp"
00028 #include "oficina2/io.hpp"
```

```
00029 #include "oficina2/input.hpp"
00030 #include "oficina2/render.hpp"
00031 #include "oficina2/canvas.hpp"
00032 #include "oficina2/timer.hpp"
00033 #include "oficina2/ofscheme.hpp"
00034 #include "oficina2/entity.hpp"
00035
00038 #define OF_VERSION_STRING "2.0.0a"
00039
00040 namespace oficina
00041 {
00048     void ofInit();
00054     void ofGameLoop();
00058     void ofSoftStop();
00065     void ofQuit();
00066
00075     void ofSetWindowSize(ofdword x, ofdword y);
00076
00079     bool ofQuitFlagRaised();
00088     glm::uvec2 ofGetWindowSize();
00089
00092     void ofSetFullscreen(bool state);
00095     bool ofIsFullscreen();
00096 }
```

7.15 ofscheme.hpp File Reference

Tools for object scripting and for the Repl.

```
#include "oficina2/scheme/scheme.h"
#include "oficina2/scheme/scheme-private.h"
#include "oficina2/scheme/dynload.h"
#include <string>
#include <functional>
#include "oficina2/entity.hpp"
```

Classes

- class oficina::ofScheme

  Defines one Scheme environment to be used inside an entity.

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)

  Asks the Repl to evaluate a certain string.
- char * oficina::ofScmGetOutputPtr ()

  Yields a pointer to the Repl's Error output string.
- void oficina::ofScmResetOutput (scheme *scm=nullptr)

  Resets the error output of the Repl or of a Scheme script.
- void oficina::ofScmDefineFunc (std::string symbol, foreign_func fun)

  Defines a foreign function for the Repl.
- void oficina::ofScmUndefineFunc (std::string symbol)

  Undefines a foreign function for the Repl.

Variables

- const char oficina::ofScmInitSrc [ ]

  Initialization source code for each and any Scheme; namely the init.scm file.

7.15.1 Detailed Description

Tools for object scripting and for the Repl.

Provides classes and functions for managing the internal Repl, and for executing scripting behavior for entities, both on Scheme language, with default OficinaFramework bindings.

Author

Lucas Vieira

Definition in file ofscheme.hpp.

7.15.2 Function Documentation

7.15.2.1 ofScmDefineFunc()

```
void oficina::ofScmDefineFunc (
            std::string symbol,
            foreign_func 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 pointer to be used. Also accepts lambdas, but not closures (e.g. lambdas with captures).

7.15.2.2 ofScmEval()

```
void oficina::ofScmEval (
            std::string strToEval )
```

Asks the Repl to evaluate a certain string.

Parameters

strToEval	String to be evaluated, in Scheme language.

7.15.2.3 ofScmGetOutputPtr()

```
char* oficina::ofScmGetOutputPtr ( )
```

Yields a pointer to the Repl's Error output string.

Warning

Please handle this pointer with care; you should not ever have to dispose it manually.

7.15.2.4 ofScmIsInit()

```
bool oficina::ofScmIsInit ( )
```

Yields the state of the Scheme Repl.

Returns

Whether the Repl is initialized or not.

7.15.2.5 ofScmResetOutput()

```
void oficina::ofScmResetOutput (
            scheme * scm = nullptr )
```

Resets the error output of the Repl or of a Scheme script.

Parameters

scm	Pointer to the actual scheme structure, or NULL/nullptr if you wish to reset the Repl's error output.

7.15.2.6 ofScmUndefineFunc()

```
void oficina::ofScmUndefineFunc (
            std::string symbol )
```

Undefines a foreign function for the Repl.

Takes a previously defined function and binds it to the Scheme's nil, effectively removing its lambda definition, if existing. This will not make the symbol cease to exist, but will remove its bound behaviour.

Parameters

symbol	Name of the function to be unbound.

7.16 ofscheme.hpp

```
00001 /*****************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *****************************************************************/
00015
00026 #pragma once
00027
00028 #include "oficina2/scheme/scheme.h"
00029 #include "oficina2/scheme/scheme-private.h"
00030 #include "oficina2/scheme/dynload.h"
00031 #include <string>
00032 #include <functional>
00033 #include "oficina2/entity.hpp"
00034
00035 namespace oficina
00036 {
00038     void ofScmInit();
00040     void ofScmDeinit();
00043     bool ofScmIsInit();
00046     void ofScmEval(std::string strToEval);
00050     char* ofScmGetOutputPtr();
00054     void ofScmResetOutput(scheme* scm = nullptr);
00062     void ofScmDefineFunc(std::string symbol, foreign_func fun);
00070     void ofScmUndefineFunc(std::string symbol);
00071
00072
00075     class ofScheme : public ofIComponent
00076     {
00077     public:
00079         void init();
00085         void loadfile(std::string filename);
00087         void unload();
00093         void update(float dt);
00099         void regFunc(std::string symbol, foreign_func fun);
00100     private:
00101         bool m_initialized = false;
00102         bool m_loaded = false;
00103         scheme* scm = nullptr;
00104         char error_buffer[255] = "\0";
00105     };
00106
00109     extern const char ofScmInitSrc[];
00110 }
```

7.17 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.

7.17.1 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.

7.18 platform.hpp

```
00001 /*************************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *************************************************************************/
00015
```

```
00029 #pragma once
00030
00032 #define OF_PLATFORM_UNKNOWN       0x000u
00033 #define OF_PLATFORM_WINDOWS       0x001u
00035 #define OF_PLATFORM_LINUX         0x002u
00037 #define OF_PLATFORM_MACOSX        0x004u
00039 #define OF_PLATFORM_ANDROID       0x008u
00041 #define OF_PLATFORM_IOS           0x010u
00043 #define OF_PLATFORM_IOS_SIMULATOR 0x020u
00045
00047 #define OF_ARCH_UNKNOWN   0x000u
00048 #define OF_ARCH_32BIT     0x002u
00050 #define OF_ARCH_64BIT     0x004u
00052 #define OF_ARCH_ARM       0x008u
00054 #define OF_ARCH_ARMV7     0x010u
00056 #define OF_ARCH_ARM64     0x020u
00058
00059 #ifdef _WIN64
00060     #define OF_PLATFORM     OF_PLATFORM_WINDOWS
00061     #define OF_ARCH         OF_ARCH_64BIT
00062     #define OF_DESKTOP
00063 #elif  _WIN32
00064     #define OF_PLATFORM     OF_PLATFORM_WINDOWS
00065     #define OF_ARCH         OF_ARCH_32BIT
00066     #define OF_DESKTOP
00067 #elif  __APPLE__
00068     #if TARGET_OS_IPHONE && TARGET_IPHONE_SIMULATOR
00069         #define OF_PLATFORM (OF_PLATFORM_IOS | OF_PLATFORM_IOS_SIMULATOR)
00070         #define OF_MOBILE
00071     #elif TARGET_OS_IPHONE
00072         #define OF_PLATFORM OF_PLATFORM_IOS
00073         #define OF_MOBILE
00074     #elif TARGET_OS_MAC
00075         #define OF_PLATFORM OF_PLATFORM_MACOSX
00076         #define OF_DESKTOP
00077     #endif
00078 #elif ANDROID
00079     #define OF_PLATFORM     OF_PLATFORM_ANDROID
00080     #define OF_MOBILE
00081 #elif  __linux__
00082     #define OF_PLATFORM     OF_PLATFORM_LINUX
00083     #define OF_DESKTOP
00084 #else
00085     #define OF_PLATFORM     OF_PLATFORM_UNKNOWN
00086     #define OF_DESKTOP
00087 #endif
00088
00089 // Check architecture. This will mainly serve for GCC and Clang
00090 #ifndef OF_ARCH
00091     #ifdef __x86_64__
00092         #define OF_ARCH  OF_ARCH_64BIT
00093     #elif __ARM_ARCH_7__
00094         #define OF_ARCH  OF_ARCH_ARMV7
00095     #elif __arm__
00096         #define OF_ARCH  OF_ARCH_ARM
00097     #elif __aarch64__
00098         #define OF_ARCH  OF_ARCH_ARM64
00099     #elif __i386__
00100         #define OF_ARCH  OF_ARCH_32BIT
00101     #else
00102         #define OF_ARCH  OF_ARCH_UNKNOWN
00103     #endif
00104 #endif
00105
00106
00107 // Important platform headers that cannot be
00108 // left out
00109 #if   OF_PLATFORM == OF_PLATFORM_WINDOWS
00110     #include <Windows.h>
00111 #elif OF_PLATFORM == OF_PLATFORM_LINUX
00112 #elif OF_PLATFORM == OF_PLATFORM_MACOSX
00113 #endif
```

7.19 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 <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.

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::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.

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.

7.19.1 Detailed Description

Tools and classes for rendering inside a context.

Author

Lucas Vieira

Definition in file render.hpp.

7.19.2 Enumeration Type Documentation

7.19.2.1 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.
ofBufferStreamCopy	Store buffer as a stream for copying.

Definition at line 49 of file render.hpp.

7.19.2.2 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 37 of file render.hpp.

7.19.2.3 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 120 of file render.hpp.

7.19.2.4 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.
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 91 of file render.hpp.

7.19.2.5 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 74 of file render.hpp.

7.19.3 Function Documentation

7.19.3.1 ofLoadDefaultFragShader()

```
ofShader oficina::ofLoadDefaultFragShader ( )
```

Loads the default fragment shader.

Returns

Reference to the default fragment shader.

7.19.3.2 ofLoadDefaultShaderProgram()

```
ofShaderProgram oficina::ofLoadDefaultShaderProgram ( )
```

Loads the default shader program, with default vertex and fragment shaders.

Returns

Reference to the default shader program.

7.19.3.3 ofLoadDefaultVertexShader()

```
ofShader oficina::ofLoadDefaultVertexShader ( )
```

Loads the default vertex shader.

Returns

Reference to the default vertex shader.

7.19.3.4 ofSetVSync()

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

Sets whether the game should vertically sync with the screen or not.

Parameters

state	Default VSync state.

7.19.4 Variable Documentation

7.19.4.1 ofDefaultShaderSrc_FS

```
const char oficina::ofDefaultShaderSrc_FS[]
```

Initial value:

```
=
        "#version 330                                   \n"

        "in vec3 Color;                                 \n"
        "in vec2 Texcoord;                              \n"

        "out vec4 outColor;                             \n"

        "uniform sampler2D tex;                         \n"

        "void main()                                    \n"
        "{                                              \n"
        "    vec4 texColor = texture(tex, Texcoord);    \n"
        "    outColor = texColor * vec4(Color, 1.0);    \n"
        "}                                              \n"
```

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 173 of file render.hpp.

7.19.4.2 ofDefaultShaderSrc_VS

```
const char oficina::ofDefaultShaderSrc_VS[]
```

Initial value:

```
=
        "#version 330                                   \n"

        "in vec3 position;                              \n"
        "in vec3 color;                                 \n"
        "in vec2 texcoord;                              \n"

        "out vec3 Color;                                \n"
        "out vec2 Texcoord;                             \n"

        "uniform mat4 mvp;                              \n"

        "void main()                                    \n"
        "{                                              \n"
        "    Color = color;                             \n"
        "    Texcoord = texcoord;                       \n"
        "    gl_Position = mvp * vec4(position, 1.0);   \n"
        "}                                              \n"
```

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 149 of file render.hpp.

7.20 render.hpp

```
00001 /*****************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *****************************************************************/
00015
00022 #pragma once
00023
00024 #include <SDL2/SDL.h>
00025 #include "oficina2/display.hpp"
00026 #include "oficina2/types.hpp"
00027 #include "oficina2/timer.hpp"
00028 #include <GL/glew.h>
00029 #include <GL/gl.h>
00030 #include <string>
00031 #include <map>
00032
00033 namespace oficina
00034 {
00037     enum ofContextType
00038     {
00040         ofContextNone,
00042         ofContextGL,
00044         ofContextGLES
00045     };
00046
00049     enum ofBufferUsage
00050     {
00052         ofBufferStaticDraw  = GL_STATIC_DRAW,
00054         ofBufferDynamicDraw = GL_DYNAMIC_DRAW,
00056         ofBufferStreamDraw  = GL_STREAM_DRAW,
00057
00059         ofBufferStaticRead  = GL_STATIC_READ,
00061         ofBufferDynamicRead = GL_DYNAMIC_READ,
00063         ofBufferStreamRead  = GL_STREAM_READ,
00064
00066         ofBufferStaticCopy  = GL_STATIC_COPY,
00068         ofBufferDynamicCopy = GL_DYNAMIC_COPY,
00070         ofBufferStreamCopy  = GL_STREAM_COPY,
00071     };
00072
00074     enum ofShaderType
00075     {
00077         ofShaderVertex      = GL_VERTEX_SHADER,
00079         ofShaderGeometry    = GL_GEOMETRY_SHADER,
00081         ofShaderFragment    = GL_FRAGMENT_SHADER,
00083         ofShaderTessControl = GL_TESS_CONTROL_SHADER,
00085         ofShaderTessEval    = GL_TESS_EVALUATION_SHADER,
00087         ofShaderCompute     = GL_COMPUTE_SHADER
00088     };
00089
00091     enum ofPrimitiveType
00092     {
00094         ofPoints           = GL_POINTS,
00096         ofLineStrip        = GL_LINE_STRIP,
00098         ofLineLoop         = GL_LINE_LOOP,
00100         ofLines            = GL_LINES,
00102         ofLineStripAdj     = GL_LINE_STRIP_ADJACENCY,
00104         ofLinesAdj         = GL_LINES_ADJACENCY,
00106         ofTriangleStrip    = GL_TRIANGLE_STRIP,
00108         ofTriangleFan      = GL_TRIANGLE_FAN,
00110         ofTriangles        = GL_TRIANGLES,
00112         ofTriangleStripAdj = GL_TRIANGLE_STRIP_ADJACENCY,
00114         ofTrianglesAdj     = GL_TRIANGLES_ADJACENCY,
00116         ofPatches          = GL_PATCHES
00117     };
00118
00120     enum ofDataType
00121     {
00123         ofDataByte   = GL_BYTE,
00125         ofDataUByte  = GL_UNSIGNED_BYTE,
00127         ofDataShort  = GL_SHORT,
00129         ofDataUShort = GL_UNSIGNED_SHORT,
00131         ofDataInt    = GL_INT,
00133         ofDataUInt   = GL_UNSIGNED_INT,
```

```
00135         ofDataFloat  = GL_FLOAT,
00137         ofDataDouble = GL_DOUBLE,
00140         ofDataFixed  = GL_FIXED
00141     };
00142
00143
00149     const char ofDefaultShaderSrc_VS[] =
00150         "#version 330                                   \n"
00151
00152         "in vec3 position;                              \n"
00153         "in vec3 color;                                 \n"
00154         "in vec2 texcoord;                              \n"
00155
00156         "out vec3 Color;                                \n"
00157         "out vec2 Texcoord;                             \n"
00158
00159         "uniform mat4 mvp;                              \n"
00160
00161         "void main()                                    \n"
00162         "{                                              \n"
00163         "    Color = color;                             \n"
00164         "    Texcoord = texcoord;                       \n"
00165         "    gl_Position = mvp * vec4(position, 1.0);   \n"
00166         "}                                              \n";
00167
00173     const char ofDefaultShaderSrc_FS[] =
00174         "#version 330                                   \n"
00175
00176         "in vec3 Color;                                 \n"
00177         "in vec2 Texcoord;                              \n"
00178
00179         "out vec4 outColor;                             \n"
00180
00181         "uniform sampler2D tex;                         \n"
00182
00183         "void main()                                    \n"
00184         "{                                              \n"
00185         "    vec4 texColor = texture(tex, Texcoord);    \n"
00186         "    outColor = texColor * vec4(Color, 1.0);    \n"
00187         "}                                              \n";
00188
00189
00191     class ofContext
00192     {
00193     public:
00199         void open(ofContextType type, const ofDisplay& hwnd);
00201         void close();
00202
00205         bool       isInit() const;
00210         void       setViewportSize(glm::uvec2 sz);
00214         glm::uvec2 getViewportSize();
00215     private:
00216         ofContextType m_type = ofContextNone;
00217         glm::uvec2    m_vwpsz;
00218         SDL_GLContext ctx;
00219         bool          m_initialized = false;
00220     };
00221
00222
00223
00229     class ofBuffer
00230     {
00231     public:
00233         virtual void init() final;
00235         virtual void unload() final;
00237         virtual void bind() final;
00239         virtual void unbind() final;
00240
00245         virtual void setData(size_t dataSize,
00246                              void* data,
00247                              ofBufferUsage usage);
00248
00252         ofBuffer& operator=(const ofBuffer& other);
00253
00256         virtual bool isInit() const final;
00259         virtual GLuint getName() const final;
00260     protected:
00264         GLenum m_type = GL_ARRAY_BUFFER;
00266         GLuint m_name = 0u;
00267     };
00268
00269
00272     class ofVertexBuffer final : public ofBuffer
00273     {
00274     public:
00276         ofVertexBuffer();
00277     };
```

```
00278
00282     class ofElementBuffer final : public ofBuffer
00283     {
00284     public:
00286         ofElementBuffer();
00287
00291         void setCount(GLsizei count);
00295         void setType(ofDataType type);
00300         void setProps(GLsizei count, ofDataType type);
00301
00304         GLsizei getCount() const;
00307         ofDataType getType() const;
00308
00315         void draw(ofPrimitiveType mode);
00316     private:
00317         GLsizei    m_count = -1;
00318         ofDataType m_dataType  = ofDataUInt;
00319     };
00320
00321
00322
00324     class ofShader
00325     {
00326     public:
00329         virtual void init(ofShaderType type) final;
00331         virtual void unload() final;
00335         virtual void setSource(const char* src) final;
00339         virtual void compile() final;
00340
00343         virtual bool isInit() const final;
00346         virtual bool isCompiled() const final;
00350         virtual GLuint getName() const final;
00351
00355         ofShader& operator=(const ofShader& shader);
00356     protected:
00358         ofShaderType m_type = ofShaderFragment;
00360         GLuint       m_name = 0u;
00362         bool         m_srcassign = false;
00364         bool         m_compiled = false;
00365     };
00366
00367     class ofShaderProgram;
00369     class ofShaderAttribute final
00370     {
00371         friend class ofShaderProgram;
00372     public:
00375         void setSize(GLint s);
00378         void setType(ofDataType t);
00381         void setStride(GLsizei stride);
00384         void setAutoNormalize(bool state);
00390         void setProps(GLint size, ofDataType type, GLsizei stride, bool normalize = false);
00391
00393         void enable();
00394
00397         int getSize();
00400         ofDataType getType();
00403         size_t getStride();
00406         bool isAutoNormalizing();
00409         bool isValid() const;
00410
00416         void bindVertexArrayData(void* byteOffset = nullptr);
00417
00421         ofShaderAttribute& operator=(const ofShaderAttribute& attr);
00422     private:
00423         GLint   m_name         = -1;
00424         GLint   m_size         = 1;
00425         GLsizei m_stride       = 0;
00426         bool    m_normalize    = false;
00427         ofDataType m_type = ofDataFloat;
00428     };
00429
00430     class ofTexture;
00435     class ofShaderUniform final
00436     {
00437         friend class ofShaderProgram;
00438     public:
00441         bool isValid() const;
00445         ofShaderUniform& operator=(const ofShaderUniform& uniform);
00446
00449         void set(float value);
00452         void set(glm::vec2 value);
00455         void set(glm::vec3 value);
00458         void set(glm::vec4 value);
00459
00462         void set(int value);
00465         void set(glm::ivec2 value);
00468         void set(glm::ivec3 value);
```

```
00471         void set(glm::ivec4 value);
00472
00475         void set(unsigned int value);
00478         void set(glm::uvec2 value);
00481         void set(glm::uvec3 value);
00484         void set(glm::uvec4 value);
00485
00489         void set(glm::mat2 value, bool transpose = false);
00493         void set(glm::mat3 value, bool transpose = false);
00497         void set(glm::mat4 value, bool transpose = false);
00498
00502         void set(glm::mat2x3 value, bool transpose = false);
00506         void set(glm::mat3x2 value, bool transpose = false);
00507
00511         void set(glm::mat2x4 value, bool transpose = false);
00515         void set(glm::mat4x2 value, bool transpose = false);
00516
00520         void set(glm::mat3x4 value, bool transpose = false);
00524         void set(glm::mat4x3 value, bool transpose = false);
00525     private:
00526         GLint m_name          = -1;
00527     };
00528
00530     class ofShaderProgram final
00531     {
00532     public:
00534         void init();
00536         void unload();
00540         void attach(const ofShader& shader);
00545         void attachUnload(ofShader& shader);
00553         void bindFragmentDataLocation(std::string name, ofdword colorNumber = 0u);
00557         void link();
00560         void use();
00562         void unuse();
00563
00566         bool isInit() const;
00569         bool isLinked() const;
00572         GLuint getName() const;
00573
00577         ofShaderProgram& operator=(const ofShaderProgram& program);
00578
00582         ofShaderAttribute getAttributeLocation(std::string name);
00586         ofShaderUniform   getUniformLocation(std::string name);
00587     private:
00588         bool shaderProgramVerify(const ofShader&) const;
00589         GLuint m_name = 0u;
00590         bool   m_linked = false;
00591     };
00592
00593
00594
00596     class ofVertexArray
00597     {
00598     public:
00600         void init();
00602         void unload();
00604         void bind();
00606         void unbind();
00607
00615         void draw(ofPrimitiveType mode, int firstVertexIdx, size_t vertexCount);
00616
00620         ofVertexArray& operator=(const ofVertexArray& other);
00621     private:
00622         GLuint m_name = 0u;
00623     };
00624
00625
00626
00627
00628     // Textures
00629     class ofTexturePool;
00631     class ofTexture
00632     {
00633         friend class ofTexturePool;
00634     public:
00638         void bind(ofword currentSampler = 0);
00642         void unbind(ofword currentSampler = 0);
00643
00647         ofTexture& operator=(const ofTexture& other);
00650         GLuint     operator()();
00651
00654         bool isLoaded() const;
00657         std::string getFileName() const;
00661         glm::uvec2  getSize() const;
00662     private:
00663         GLuint      m_name = 0u;
00664         glm::uvec2  m_size;
```

```
00665         std::string m_filename;
00666     };
00667
00668     class ofFont;
00678     class ofTexturePool
00679     {
00680     public:
00684         static ofTexture load(std::string filename);
00688         static ofTexture load(SDL_Surface* surf);
00691         static ofFont    loadDefaultFont();
00694         static void      unload(ofTexture& t);
00696         static void      clear();
00697     };
00698
00699
00700
00702     class ofTextureRenderer
00703     {
00704     public:
00710         void init(ofTexture t, glm::uvec2 frameSize = glm::uvec2(0, 0));
00725         void render(glm::vec2 position, glm::mat4 mvp, ofdword frame = 0u, glm::vec4 color =
      glm::vec4(1.0f), float mag = 1.0f);
00728         void unload();
00729
00733         ofTextureRenderer& operator=(const ofTextureRenderer& other);
00734
00740         void SetTexture(ofTexture t);
00741
00744         bool isInit() const;
00745     private:
00746         bool m_initialized = false;
00747         ofTexture m_texture;
00748         glm::vec2 m_frameSize;
00749         ofVertexArray vao;
00750         ofVertexBuffer vbo;
00751         ofShaderProgram program;
00752         ofShaderAttribute attrPosition,
00753                           attrTexcoord;
00754         ofShaderUniform uniColor,
00755                         uniMVP,
00756                         uniTexSampler;
00757     };
00758
00765     class ofFont
00766     {
00767     public:
00774         void init(ofTexture fontTexture, glm::uvec2 glyphSize, bool manageTexture = false);
00785         void write(std::string text, glm::vec2 position, glm::mat4 mvp, glm::vec4 color = glm::vec4(1.0f),
      float mag = 1.0f);
00788         void unload();
00789
00793         ofFont& operator=(const ofFont& other);
00794
00797         bool isInit() const;
00798     private:
00799         bool m_unloadtexture = false;
00800         bool m_initialized = false;
00801         glm::uvec2 m_glyphsize;
00802         ofTexture m_texture;
00803         ofTextureRenderer m_renderer;
00804     };
00805
00808     class ofAnimator
00809     {
00810     public:
00817         void init(ofTexture t, glm::uvec2 frameSize, bool manageTexture = false);
00820         void unload();
00824         void update(float dt);
00831         void draw(glm::mat4 ViewProjection, float magnification = 1.0f);
00832
00851         void reg(std::string animName, ofdword nFrames, const ofdword* animFrames, float
      speed, bool loops = false, ofdword loopBackTo = 0u);
00854         void unreg(std::string animName);
00859         void SetAnimation(std::string animName);
00863         void SyncToFrameRate(bool state);
00870         void SetAnimationSpeed(float spd);
00875         float GetAnimationSpeed() const;
00880         float GetDefaultAnimationSpeed() const;
00884         void SetAnimationRunning(bool state);
00885
00891         void SetAnimationTexture(ofTexture t);
00892
00895         bool isInit() const;
00898         glm::vec2 getPosition();
00901         void setPosition(glm::vec2 pos);
00904         bool GetAnimationRunning() const;
00905     private:
```

```
00906         struct ofAnimProps
00907         {
00908             ofdword  num_frames;
00909             ofdword  loopback;
00910             ofdword* frames = nullptr;
00911             bool     loops;
00912             float    speed;
00913         };
00914
00915         bool m_unloadtexture = false;
00916         bool m_initialized   = false;
00917         bool m_sync          = true;
00918         bool m_playing       = true;
00919         const ofAnimProps*  m_current = nullptr;
00920         ofdword             m_currentframe = 0u;
00921         ofFrameSpan         m_framespan;
00922         ofTimeSpan          m_timespan;
00923         glm::uvec2          m_framesize;
00924         glm::vec2           m_position;
00925         ofTexture           m_t;
00926         ofTextureRenderer   m_renderer;
00927         std::string         m_animname;
00928         float               m_animspd;
00929         std::map<std::string, ofAnimProps> m_animations;
00930     };
00931
00932
00935     ofShader        ofLoadDefaultFragShader();
00938     ofShader        ofLoadDefaultVertexShader();
00942     ofShaderProgram ofLoadDefaultShaderProgram();
00946     void            ofSetVSync(bool state);
00947 }
```

7.21 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.

7.21.1 Detailed Description

Tools for counting and processing time-related events.

Author

Lucas Vieira

Definition in file timer.hpp.

7.22 timer.hpp

```
00001 /*************************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *************************************************************************/
00015
00022 #pragma once
00023
00024 #include <cstdint>
00025
00026 namespace oficina
00027 {
00031     class ofTimeSpan
00032     {
00033     public:
00036         void  begin();
00041         float yieldSpan();
00046         float resetSpan();
00050         float stop();
00054         bool  isRunning() const;
00055     private:
00056         bool     m_started = false;
00057         uint32_t m_timer   = 0u;
00058     };
00059
00062     class ofFrameSpan
00063     {
00064     public:
00066         void     begin();
00068         void     update();
00074         uint32_t yieldSpan();
00078         uint32_t resetSpan();
00082         uint32_t stop();
00086         bool     isRunning() const;
00087     private:
00088         bool     m_started = false;
00089         uint32_t m_timer   = 0u;
00090     };
00091 }
```

7.23 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.

7.23.1 Detailed Description

Tools for predefining default types and math tools used by OficinaFramework.

Author

Lucas Vieira

Definition in file types.hpp.

7.23.2 Function Documentation

7.23.2.1 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.

7.24 types.hpp

```
00001 /*************************************************************
00002  *  Copyright (c) 2017 Lucas Vieira <lucas.samuel2002@gmail.com>  *
00003  *  This file is part of OficinaFramework v2.x                    *
00004  *                                                                *
00005  *  OficinaFramework is free software: you can redistribute       *
00006  *  it and/or modify it under the terms of the GNU Lesser         *
00007  *  General Public License as published by the Free Software      *
00008  *  Foundation, either version 3 of the License, or (at your      *
00009  *  option) any later version.                                    *
00010  *                                                                *
00011  *  You should have received a copy of the GNU Lesser General     *
00012  *  Public License along with OficinaFramework.  If not, see      *
00013  *  <http://www.gnu.org/licenses/>.                               *
00014  *************************************************************/
00015
00023 //#define GLM_FORCE_SWIZZLE
00024
00025 #include "oficina2/platform.hpp"
00026 #include <glm/glm.hpp>
00027 #include <glm/gtc/matrix_transform.hpp>
00028 #include <glm/gtc/type_ptr.hpp>
00029 #include <cmath>
00030 #include <cstdint>
00031
00032 #pragma once
00033
00035 typedef uint8_t  ofbyte;
00037 typedef uint16_t ofword;
00039 typedef uint32_t ofdword;
00041 typedef uint64_t ofqword;
00042
00044 typedef int8_t   ofsbyte;
00046 typedef int16_t  ofsword;
00048 typedef int32_t  ofsdword;
00050 typedef int64_t  ofsqword;
00051
00054 typedef uintptr_t ofaword;
00057 typedef intptr_t  ofsaword;
00058
00065 float ofClamp(float value, float min, float max);
```

Index