TinyWindow

1.3.8

Generated by Doxygen 1.8.11

Contents

Chapter 1

TinyWindow

a cross platform single header window management API

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 windowManager::window_t Struct Reference

Public Member Functions

- window_t (const char ∗name=nullptr, unsigned int iD=0, unsigned int colorBits=0, unsigned int depthBits=0, unsigned int stencilBits=0, bool shouldClose=false, tinyWindowState_t currentState=tinyWindowState_↩ t::NORMAL, std::function< void(unsigned int, tinyWindowKeyState_t)> keyEvent=nullptr, std::function< void(tinyWindowMouseButton_t, tinyWindowButtonState_t)> mouseButtonEvent=nullptr, std::function< void(tinyWindowMouseScroll_t)> mouseWheelEvent=nullptr, std::function< void(void)> destroyed↩ Event=nullptr, std::function< void(void)> maximizedEvent=nullptr, std::function< void(void)> minimized↩ Event=nullptr, std::function< void(bool)> focusEvent=nullptr, std::function< void(unsigned int, unsigned int)> movedEvent=nullptr, std::function< void(unsigned int, unsigned int)> resizeEvent=nullptr, std::function< void(unsigned int, unsigned int, unsigned int, unsigned int)> mouseMoveEvent=nullptr)

Public Attributes

- const char ∗ name
- unsigned int iD
- int colorBits
- int depthBits
- int stencilBits
- tinyWindowKeyState_t keys [KEY_LAST]
- tinyWindowButtonState_t mouseButton [(unsigned int) tinyWindowMouseButton_t::LAST]
- unsigned int resolution [2]
- unsigned int position [2]
- unsigned int mousePosition [2]
- bool shouldClose
- bool inFocus
- bool initialized
- bool contextCreated
- bool isCurrentContext
- tinyWindowState_t currentState
- unsigned int currentWindowStyle
- std::function< void(unsigned int, tinyWindowKeyState_t)> keyEvent
- std::function< void(tinyWindowMouseButton_t, tinyWindowButtonState_t)> mouseButtonEvent
- std::function< void(tinyWindowMouseScroll_t)> mouseWheelEvent

- std::function< void(void)> destroyedEvent
- std::function< void(void)> maximizedEvent
- std::function< void(void)> minimizedEvent
- std::function< void(bool)> focusEvent
- std::function< void(unsigned int, unsigned int)> movedEvent
- std::function< void(unsigned int, unsigned int)> resizeEvent
- std::function< void(unsigned int, unsigned int, unsigned int, unsigned int)> mouseMoveEvent
- Window windowHandle
- GLXContext context
- XVisualInfo ∗ visualInfo
- int ∗ attributes
- XSetWindowAttributes setAttributes
- unsigned int decorators
- Atom AtomState
- Atom AtomHidden
- Atom AtomFullScreen
- Atom AtomMaxHorz
- Atom AtomMaxVert
- Atom AtomClose
- Atom AtomActive
- Atom AtomDemandsAttention
- Atom AtomFocused
- Atom AtomCardinal
- Atom AtomIcon
- Atom AtomHints
- Atom AtomWindowType
- Atom AtomWindowTypeDesktop
- Atom AtomWindowTypeSplash
- Atom AtomWindowTypeNormal
- Atom AtomAllowedActions
- Atom AtomActionResize
- Atom AtomActionMinimize
- Atom AtomActionShade
- Atom AtomActionMaximizeHorz
- Atom AtomActionMaximizeVert
- Atom AtomActionClose
- Atom AtomDesktopGeometry

4.1.1 Detailed Description

Definition at line 2292 of file TinyWindow.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 windowManager::window_t::window_t (const char ∗ *name* = `nullptr`, unsigned int *iD* = `0`, unsigned int *colorBits* = `0`, unsigned int *depthBits* = `0`, unsigned int *stencilBits* = `0`, bool *shouldClose* = `false`, tinyWindowState_t *currentState* = tinyWindowState_t::NORMAL, std::function< void(unsigned int, tinyWindowKeyState_t)> *keyEvent* = `nullptr`, std::function< void(tinyWindowMouseButton_t, tinyWindowButtonState_t)> *mouseButtonEvent* = `nullptr`, std::function< void(tinyWindowMouseScroll_t)> *mouseWheelEvent* = `nullptr`, std::function< void(void)> *destroyedEvent* = `nullptr`, std::function< void(void)> *maximizedEvent* = `nullptr`, std::function< void(void)> *minimizedEvent* = `nullptr`, std::function< void(bool)> *focusEvent* = `nullptr`, std::function< void(unsigned int, unsigned int)> *movedEvent* = `nullptr`, std::function< void(unsigned int, unsigned int)> *resizeEvent* = `nullptr`, std::function< void(unsigned int, unsigned int, unsigned int, unsigned int)> *mouseMoveEvent* = `nullptr`) `[inline]`

Definition at line 2374 of file TinyWindow.h.

References colorBits, contextCreated, currentState, currentWindowStyle, DEFAULT, depthBits, iD, initialized, name, shouldClose, and stencilBits.

```
        {
            this->name = name;
            this->iD = iD;
            this->colorBits = colorBits;
            this->depthBits = depthBits;
            this->stencilBits = stencilBits;
            this->shouldClose = shouldClose;
            this->currentState = currentState;

            this->keyEvent = keyEvent;
            this->mouseButtonEvent = mouseButtonEvent;
            this->mouseWheelEvent = mouseWheelEvent;
            this->destroyedEvent = destroyedEvent;
            this->maximizedEvent = maximizedEvent;
            this->minimizedEvent = minimizedEvent;
            this->focusEvent = focusEvent;
            this->movedEvent = movedEvent;
            this->resizeEvent = resizeEvent;
            this->mouseMoveEvent = mouseMoveEvent;

            initialized = false;
            contextCreated = false;
            currentWindowStyle = (unsigned int)
 tinyWindowStyle_t::DEFAULT;

#if defined( __linux )
            context = 0;
#endif
        }
```

4.1.3 Member Data Documentation

4.1.3.1 Atom windowManager::window_t::AtomActionClose

Atom for allowing the window to be closed

Definition at line 2369 of file TinyWindow.h.

4.1.3.2 Atom windowManager::window_t::AtomActionMaximizeHorz

Atom for allowing the window to be maximized horizontally

Definition at line 2367 of file TinyWindow.h.

4.1.3.3 Atom windowManager::window_t::AtomActionMaximizeVert

Atom for allowing the window to be maximized vertically

Definition at line 2368 of file TinyWindow.h.

4.1.3.4 Atom windowManager::window_t::AtomActionMinimize

Atom for allowing the window to be minimized

Definition at line 2365 of file TinyWindow.h.

4.1.3.5 Atom windowManager::window_t::AtomActionResize

Atom for allowing the window to be resized

Definition at line 2364 of file TinyWindow.h.

4.1.3.6 Atom windowManager::window_t::AtomActionShade

Atom for allowing the window to be shaded

Definition at line 2366 of file TinyWindow.h.

4.1.3.7 Atom windowManager::window_t::AtomActive

Atom for the active window

Definition at line 2351 of file TinyWindow.h.

4.1.3.8 Atom windowManager::window_t::AtomAllowedActions

Atom for allowed window actions

Definition at line 2363 of file TinyWindow.h.

4.1.3.9 Atom windowManager::window_t::AtomCardinal

Atom for cardinal coordinates

Definition at line 2354 of file TinyWindow.h.

4.1.3.10 Atom windowManager::window_t::AtomClose

Atom for closing the window

Definition at line 2350 of file TinyWindow.h.

4.1.3.11 Atom windowManager::window_t::AtomDemandsAttention

Atom for when the window demands attention

Definition at line 2352 of file TinyWindow.h.

4.1.3.12 Atom windowManager::window_t::AtomDesktopGeometry

Atom for Desktop Geometry

Definition at line 2371 of file TinyWindow.h.

Referenced by windowManager::AddWindow(), windowManager::DisableWindowDecoratorByIndex(), window↩ Manager::DisableWindowDecoratorByName(), windowManager::EnableWindowDecoratorsByIndex(), window↩ Manager::EnableWindowDecoratorsByName(), windowManager::FocusWindowByIndex(), windowManager↩ ::FocusWindowByName(), windowManager::MakeWindowCurrentContextByIndex(), windowManager::Make↩ WindowCurrentContextByName(), windowManager::MaximizeWindowByIndex(), windowManager::Maximize↩ WindowByName(), windowManager::MinimizeWindowByIndex(), windowManager::MinimizeWindowByName(), windowManager::RestoreWindowByIndex(), windowManager::RestoreWindowByName(), windowManager↩ ::SetFullScreenByIndex(), windowManager::SetFullScreenByName(), windowManager::SetMousePosition↩ InWindowByIndex(), windowManager::SetMousePositionInWindowByName(), windowManager::SetWindow↩ PositionByIndex(), windowManager::SetWindowPositionByName(), windowManager::SetWindowResolutionBy↩ Index(), windowManager::SetWindowResolutionByName(), windowManager::SetWindowStyleByIndex(), window↩ Manager::SetWindowStyleByName(), windowManager::SetWindowTitleBarByIndex(), windowManager::Set↩ WindowTitleBarByName(), windowManager::WindowSwapBuffersByIndex(), and windowManager::WindowSwap↩ BuffersByName().

4.1.3.13 Atom windowManager::window_t::AtomFocused

Atom for the focused state of the window

Definition at line 2353 of file TinyWindow.h.

4.1.3.14 Atom windowManager::window_t::AtomFullScreen

Atom for the full screen state of the window

Definition at line 2347 of file TinyWindow.h.

4.1.3.15 Atom windowManager::window_t::AtomHidden

Atom for the current hidden state of the window

Definition at line 2346 of file TinyWindow.h.

4.1.3.16 Atom windowManager::window_t::AtomHints

Atom for the window decorations

Definition at line 2356 of file TinyWindow.h.

4.1.3.17 Atom windowManager::window_t::AtomIcon

Atom for the icon of the window

Definition at line 2355 of file TinyWindow.h.

4.1.3.18 Atom windowManager::window_t::AtomMaxHorz

Atom for the maximized horizontally state of the window

Definition at line 2348 of file TinyWindow.h.

4.1.3.19 Atom windowManager::window_t::AtomMaxVert

Atom for the maximized vertically state of the window

Definition at line 2349 of file TinyWindow.h.

4.1.3.20 Atom windowManager::window_t::AtomState

Atom for the state of the window

Definition at line 2345 of file TinyWindow.h.

4.1.3.21 Atom windowManager::window_t::AtomWindowType

Atom for the type of window

Definition at line 2358 of file TinyWindow.h.

4.1.3.22 Atom windowManager::window_t::AtomWindowTypeDesktop

Atom for the desktop window type

Definition at line 2359 of file TinyWindow.h.

4.1.3.23 Atom windowManager::window_t::AtomWindowTypeNormal

Atom for the normal splash screen window type

Definition at line 2361 of file TinyWindow.h.

4.1.3.24 Atom windowManager::window_t::AtomWindowTypeSplash

Atom for the splash screen window type

Definition at line 2360 of file TinyWindow.h.

4.1.3.25 int∗ windowManager::window_t::attributes

Attributes of the window. RGB, depth, stencil, etc

Definition at line 2339 of file TinyWindow.h.

4.1.3.26 int windowManager::window_t::colorBits

color format of the window. (defaults to 32 bit color)

Definition at line 2296 of file TinyWindow.h.

Referenced by windowManager::AddWindow(), and window_t().

4.1.3.27 GLXContext windowManager::window_t::context

The handle to the GLX rendering context

Definition at line 2337 of file TinyWindow.h.

4.1.3.28 bool windowManager::window_t::contextCreated

Whether the OpenGL context has been successfully created

Definition at line 2308 of file TinyWindow.h.

Referenced by windowManager::Platform_InitializeGL(), and window_t().

4.1.3.29 tinyWindowState_t windowManager::window_t::currentState

The current state of the window. these states include Normal, Minimized, Maximized and Full screen

Definition at line 2311 of file TinyWindow.h.

Referenced by windowManager::GetWindowIsFullScreenByIndex(), windowManager::GetWindowIsFullScreenBy↵
Name(), windowManager::GetWindowIsMaximizedByIndex(), windowManager::GetWindowIsMaximizedByName(),
windowManager::GetWindowIsMinimizedByIndex(), windowManager::GetWindowIsMinimizedByName(), window↵
Manager::Platform_MaximizeWindow(), windowManager::Platform_MinimizeWindow(), windowManager::SetFull↵
ScreenByIndex(), windowManager::SetFullScreenByName(), and window_t().

4.1.3.30 unsigned int windowManager::window_t::currentWindowStyle

The current style of the window

Definition at line 2312 of file TinyWindow.h.

Referenced by windowManager::Platform_DisableWindowDecorators(), windowManager::Platform_Enable↵
WindowDecorators(), and window_t().

4.1.3.31 unsigned int windowManager::window_t::decorators

Enabled window decorators

Definition at line 2341 of file TinyWindow.h.

4.1.3.32 int windowManager::window_t::depthBits

Size of the Depth buffer. (defaults to 8 bit depth)

Definition at line 2297 of file TinyWindow.h.

Referenced by windowManager::AddWindow(), and window_t().

4.1.3.33 std::function<void(void)> windowManager::window_t::destroyedEvent

This is the callback to be used when the window has been closed in a non-programmatic fashion

Definition at line 2317 of file TinyWindow.h.

4.1.3.34 std::function<void(bool)> windowManager::window_t::focusEvent

This is the callback to be used when the window has been given focus in a non-programmatic fashion

Definition at line 2320 of file TinyWindow.h.

4.1.3.35 unsigned int windowManager::window_t::iD

ID of the Window. (where it belongs in the window manager)

Definition at line 2295 of file TinyWindow.h.

Referenced by windowManager::AddWindow(), windowManager::GetWindowIndexByName(), and window_t().

4.1.3.36 bool windowManager::window_t::inFocus

Whether the Window is currently in focus(if it is the current window be used)

Definition at line 2305 of file TinyWindow.h.

Referenced by windowManager::GetWindowIsInFocusByIndex(), and windowManager::GetWindowIsInFocusBy↩
Name().

4.1.3.37 bool windowManager::window_t::initialized

Whether the window has been successfully initialized

Definition at line 2307 of file TinyWindow.h.

Referenced by window_t().

4.1.3.38 bool windowManager::window_t::isCurrentContext

Whether the window is the current window being drawn to

Definition at line 2309 of file TinyWindow.h.

4.1.3.39 std::function< void(unsigned int, tinyWindowKeyState_t)> windowManager::window_t::keyEvent

This is the callback to be used when a key has been pressed

Definition at line 2314 of file TinyWindow.h.

4.1.3.40 tinyWindowKeyState_t windowManager::window_t::keys[KEY_LAST]

Record of keys that are either pressed or released in the respective window

Definition at line 2299 of file TinyWindow.h.

Referenced by windowManager::WindowGetKeyByIndex(), and windowManager::WindowGetKeyByName().

4.1.3.41 std::function< void(void)> windowManager::window_t::maximizedEvent

This is the callback to be used when the window has been maximized in a non-programmatic fashion

Definition at line 2318 of file TinyWindow.h.

4.1.3.42 std::function< void(void)> windowManager::window_t::minimizedEvent

This is the callback to be used when the window has been minimized in a non-programmatic fashion

Definition at line 2319 of file TinyWindow.h.

4.1.3.43 tinyWindowButtonState_t windowManager::window_t::mouseButton[(unsigned int) tinyWindowMouseButton_t::LAST]

Record of mouse buttons that are either presses or released

Definition at line 2300 of file TinyWindow.h.

4.1.3.44 std::function< void(tinyWindowMouseButton_t, tinyWindowButtonState_t)> windowManager::window_t::mouseButtonEvent

This is the callback to be used when a mouse button has been pressed

Definition at line 2315 of file TinyWindow.h.

4.1.3.45 std::function<void(unsigned int, unsigned int, unsigned int, unsigned int)> windowManager::window_t::mouse↩ MoveEvent

This is a callback to be used when the mouse has been moved

Definition at line 2323 of file TinyWindow.h.

4.1.3.46 unsigned int windowManager::window_t::mousePosition[2]

Position of the Mouse cursor relative to the window co-ordinates

Definition at line 2303 of file TinyWindow.h.

Referenced by windowManager::GetMousePositionInWindowByIndex(), windowManager::GetMousePositionIn↩ WindowByName(), windowManager::SetMousePositionInWindowByIndex(), and windowManager::SetMouse↩ PositionInWindowByName().

4.1.3.47 std::function<void(tinyWindowMouseScroll_t)> windowManager::window_t::mouseWheelEvent

This is the callback to be used when the mouse wheel has been scrolled.

Definition at line 2316 of file TinyWindow.h.

4.1.3.48 std::function<void(unsigned int, unsigned int)> windowManager::window_t::movedEvent

This is the callback to be used the window has been moved in a non-programmatic fashion

Definition at line 2321 of file TinyWindow.h.

4.1.3.49 const char* windowManager::window_t::name

Name of the window

Definition at line 2294 of file TinyWindow.h.

Referenced by windowManager::AddWindow(), windowManager::GetWindowNameByIndex(), windowManager::↩ Platform_SetWindowStyle(), windowManager::ShutdownWindow(), and window_t().

4.1.3.50 unsigned int windowManager::window_t::position[2]

Position of the Window relative to the screen co-ordinates

Definition at line 2302 of file TinyWindow.h.

Referenced by windowManager::GetWindowPositionByIndex(), windowManager::GetWindowPositionByName(), windowManager::Platform_SetWindowResolution(), windowManager::SetWindowPositionByIndex(), and window↩ Manager::SetWindowPositionByName().

4.1.3.51 std::function<void(unsigned int, unsigned int)> windowManager::window_t::resizeEvent

This is a callback to be used when the window has been resized in a non-programmatic fashion

Definition at line 2322 of file TinyWindow.h.

4.1.3.52 unsigned int windowManager::window_t::resolution[2]

Resolution/Size of the window stored in an array

Definition at line 2301 of file TinyWindow.h.

Referenced by windowManager::AddWindow(), windowManager::GetWindowResolutionByIndex(), window↩ Manager::GetWindowResolutionByName(), windowManager::Platform_SetWindowPosition(), windowManager::↩ Platform_SetWindowResolution(), windowManager::SetWindowResolutionByIndex(), and windowManager::Set↩ WindowResolutionByName().

4.1.3.53 XSetWindowAttributes windowManager::window_t::setAttributes

The attributes to be set for the window

Definition at line 2340 of file TinyWindow.h.

4.1.3.54 bool windowManager::window_t::shouldClose

Whether the Window should be closing

Definition at line 2304 of file TinyWindow.h.

Referenced by windowManager::GetWindowShouldCloseByIndex(), windowManager::GetWindowShouldClose↩ ByName(), and window_t().

4.1.3.55 int windowManager::window_t::stencilBits

Size of the stencil buffer, (defaults to 8 bit)

Definition at line 2298 of file TinyWindow.h.

Referenced by windowManager::AddWindow(), and window_t().

4.1.3.56 XVisualInfo* windowManager::window_t::visualInfo

The handle to the Visual Information. similar purpose to PixelformatDesriptor

Definition at line 2338 of file TinyWindow.h.

4.1.3.57 Window windowManager::window_t::windowHandle

The X11 handle to the window. I wish they didn't name the type 'Window'

Definition at line 2336 of file TinyWindow.h.

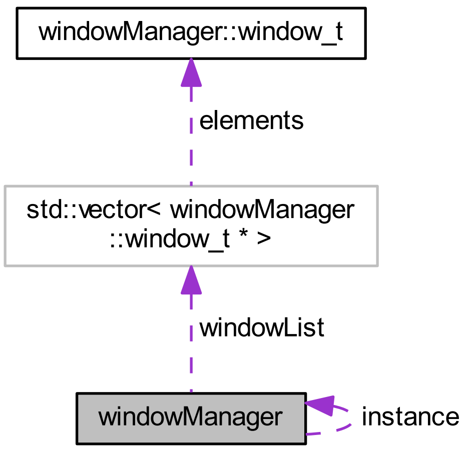
The documentation for this struct was generated from the following file:

- Include/TinyWindow.h

4.2 windowManager Class Reference

```
#include <TinyWindow.h>
```

Collaboration diagram for windowManager:



Classes

- struct window_t

Public Member Functions

- windowManager ()
- ∼windowManager (void)

Static Public Member Functions

- static void ShutDown (void)
- static windowManager ∗ AddWindow (const char ∗windowName, unsigned int width=DEFAULT_WIND↩ OW_WIDTH, unsigned int height=DEFAULT_WINDOW_HEIGHT, unsigned int colourBits=8, unsigned int depthBits=8, unsigned int stencilBits=8)
- static int GetNumWindows (void)
- static bool GetMousePositionInScreen (unsigned int &x, unsigned int &y)
- static unsigned int ∗ GetMousePositionInScreen (void)
- static bool SetMousePositionInScreen (unsigned int x, unsigned int y)
- static unsigned int ∗ GetScreenResolution (void)
- static bool GetScreenResolution (unsigned int &width, unsigned int &Height)
- static bool GetWindowResolutionByName (const char ∗windowName, unsigned int &width, unsigned int &height)
- static bool GetWindowResolutionByIndex (unsigned int windowIndex, unsigned int &width, unsigned int &height)
- static unsigned int ∗ GetWindowResolutionByName (const char ∗windowName)
- static unsigned int ∗ GetWindowResolutionByIndex (unsigned int windowIndex)
- static bool SetWindowResolutionByName (const char ∗windowName, unsigned int width, unsigned int height)
- static bool SetWindowResolutionByIndex (unsigned int windowIndex, unsigned int width, unsigned int height)
- static bool GetWindowPositionByName (const char ∗windowName, unsigned int &x, unsigned int &y)
- static bool GetWindowPositionByIndex (unsigned int windowIndex, unsigned int &x, unsigned int &y)
- static unsigned int ∗ GetWindowPositionByName (const char ∗windowName)
- static unsigned int ∗ GetWindowPositionByIndex (unsigned int windowIndex)
- static bool SetWindowPositionByName (const char ∗windowName, unsigned int x, unsigned int y)
- static bool SetWindowPositionByIndex (unsigned int windowIndex, unsigned int x, unsigned int y)
- static bool GetMousePositionInWindowByName (const char ∗windowName, unsigned int &x, unsigned int &y)
- static bool GetMousePositionInWindowByIndex (unsigned int windowIndex, unsigned int &x, unsigned int &y)
- static unsigned int ∗ GetMousePositionInWindowByName (const char ∗windowName)
- static unsigned int ∗ GetMousePositionInWindowByIndex (unsigned int windowIndex)
- static bool SetMousePositionInWindowByName (const char ∗windowName, unsigned int x, unsigned int y)
- static bool SetMousePositionInWindowByIndex (unsigned int windowIndex, unsigned int x, unsigned int y)
- static tinyWindowKeyState_t WindowGetKeyByName (const char ∗windowName, unsigned int key)
- static tinyWindowKeyState_t WindowGetKeyByIndex (unsigned int windowIndex, unsigned int key)
- static bool GetWindowShouldCloseByName (const char ∗windowName)
- static bool GetWindowShouldCloseByIndex (unsigned int windowIndex)
- static bool WindowSwapBuffersByName (const char ∗windowName)
- static bool WindowSwapBuffersByIndex (unsigned int windowIndex)
- static bool MakeWindowCurrentContextByName (const char ∗windowName)
- static bool MakeWindowCurrentContextByIndex (unsigned int windowIndex)
- static bool GetWindowIsFullScreenByName (const char ∗windowName)
- static bool GetWindowIsFullScreenByIndex (unsigned int windowIndex)
- static bool SetFullScreenByName (const char ∗windowName, bool newState)
- static bool SetFullScreenByIndex (unsigned int windowIndex, bool newState)
- static bool GetWindowIsMinimizedByName (const char ∗windowName)
- static bool GetWindowIsMinimizedByIndex (unsigned int windowIndex)
- static bool MinimizeWindowByName (const char ∗windowName, bool newState)
- static bool MinimizeWindowByIndex (unsigned int windowIndex, bool newState)
- static bool GetWindowIsMaximizedByName (const char ∗windowName)
- static bool GetWindowIsMaximizedByIndex (unsigned int windowIndex)
- static bool MaximizeWindowByName (const char ∗windowName, bool newState)
- static bool MaximizeWindowByIndex (unsigned int windowIndex, bool newState)
- static const char ∗ GetWindowNameByIndex (unsigned int windowIndex)
- static unsigned int GetWindowIndexByName (const char ∗windowName)
- static bool SetWindowTitleBarByName (const char ∗windowName, const char ∗newTitle)

- static bool SetWindowTitleBarByIndex (unsigned int windowIndex, const char ∗newName)
- static bool SetWindowIconByName (void)
- static bool SetWindowIconByIndex (void)
- static bool GetWindowIsInFocusByName (const char ∗windowName)
- static bool GetWindowIsInFocusByIndex (unsigned int windowIndex)
- static bool FocusWindowByName (const char ∗windowName, bool newState)
- static bool FocusWindowByIndex (unsigned int windowIndex, bool newState)
- static bool RestoreWindowByName (const char ∗windowName)
- static bool RestoreWindowByIndex (unsigned int windowIndex)
- static bool Initialize (void)
- static bool IsInitialized (void)
- static void PollForEvents (void)
- static void WaitForEvents (void)
- static bool RemoveWindowByName (const char ∗windowName)
- static bool RemoveWindowByIndex (unsigned int windowIndex)
- static bool SetWindowStyleByName (const char ∗windowName, tinyWindowStyle_t windowStyle)
- static bool SetWindowStyleByIndex (unsigned int windowIndex, tinyWindowStyle_t windowStyle)
- static bool EnableWindowDecoratorsByName (const char ∗windowName, unsigned int decorators)
- static bool EnableWindowDecoratorsByIndex (unsigned int windowIndex, unsigned int decorators)
- static bool DisableWindowDecoratorByName (const char ∗windowName, unsigned int decorators)
- static bool DisableWindowDecoratorByIndex (unsigned int windowIndex, unsigned int decorators)
- static bool SetWindowOnKeyEventByName (const char ∗windowName, std::function< void(unsigned int, tinyWindowKeyState_t)> onKey)
- static bool SetWindowOnKeyEventByIndex (unsigned int windowIndex, std::function< void(unsigned int, tinyWindowKeyState_t)> onKey)
- static bool SetWindowOnMouseButtonEventByName (const char ∗windowName, std::function< void(tiny←WindowMouseButton_t, tinyWindowButtonState_t)> onMouseButton)
- static bool SetWindowOnMouseButtonEventByIndex (unsigned int windowIndex, std::function< void(tiny←WindowMouseButton_t, tinyWindowButtonState_t)> onMouseButton)
- static bool SetWindowOnMouseWheelEventByName (const char ∗windowName, std::function< void(tiny←WindowMouseScroll_t)> onMouseWheel)
- static bool SetWindowOnMouseWheelEventByIndex (unsigned int windowIndex, std::function< void(tiny←WindowMouseScroll_t)> onMouseWheel)
- static bool SetWindowOnDestroyedByName (const char ∗windowName, std::function< void(void)> on←Destroyed)
- static bool SetWindowOnDestroyedByIndex (unsigned int windowIndex, std::function< void(void)> on←Destroyed)
- static bool SetWindowOnMaximizedByName (const char ∗windowName, std::function< void(void)> on←Maximized)
- static bool SetWindowOnMaximizedByIndex (unsigned int windowIndex, std::function< void(void)> on←Maximized)
- static bool SetWindowOnMinimizedByName (const char ∗windowName, std::function< void(void)> on←Minimized)
- static bool SetWindowOnMinimizedByIndex (unsigned int windowIndex, std::function< void(void)> on←Minimized)
- static bool SetWindowOnFocusByName (const char ∗windowName, std::function< void(bool)> onFocus)
- static bool SetWindowOnFocusByIndex (unsigned int windowIndex, std::function< void(bool)> onFocus)
- static bool SetWindowOnMovedByName (const char ∗windowName, std::function< void(unsigned int, unsigned int)> onMoved)
- static bool SetWindowOnMovedByIndex (unsigned int windowIndex, std::function< void(unsigned int, unsigned int)> onMoved)
- static bool SetWindowOnResizeByName (const char ∗windowName, std::function< void(unsigned int, unsigned int)> onResize)
- static bool SetWindowOnResizeByIndex (unsigned int windowIndex, std::function< void(unsigned int, unsigned int)> onResize)

- static bool SetWindowOnMouseMoveByName (const char ∗windowName, std::function< void(unsigned int, unsigned int, unsigned int, unsigned int)> onMouseMove)
- static bool SetWindowOnMouseMoveByIndex (unsigned int windowIndex, std::function< void(unsigned int, unsigned int, unsigned int, unsigned int)> onMouseMove)

Static Private Member Functions

- static bool IsValid (const char ∗stringParameter)
- static bool WindowExists (unsigned int windowIndex)
- static windowManager ∗ GetInstance (void)
- static void Platform_InitializeWindow (window_t ∗window)
- static bool Platform_InitializeGL (window_t ∗window)
- static void Platform_SetWindowResolution (window_t ∗window)
- static void Platform_SetWindowPosition (window_t ∗window, unsigned int x, unsigned int y)
- static void Platform_SetMousePositionInWindow (window_t ∗window, unsigned int x, unsigned int y)
- static void Platform_SwapBuffers (window_t ∗window)
- static void Platform_MakeCurrentContext (window_t ∗window)
- static void Platform_SetFullScreen (window_t ∗window)
- static void Platform_MinimizeWindow (window_t ∗window, bool newState)
- static void Platform_MaximizeWindow (window_t ∗window, bool newState)
- static void Platform_SetWindowTitleBar (window_t ∗window, const char ∗newTitle)
- static void Platform_FocusWindow (window_t ∗window, bool newState)
- static void Platform_RestoreWindow (window_t ∗window)
- static void Platform_SetWindowStyle (window_t ∗window, tinyWindowStyle_t windowStyle)
- static void Platform_EnableWindowDecorators (window_t ∗window, unsigned int decorators)
- static void Platform_DisableWindowDecorators (window_t ∗window, unsigned int decorators)
- static void ShutdownWindow (window_t ∗window)
- static bool DoesExistByName (const char ∗windowName)
- static bool DoesExistByIndex (unsigned int windowIndex)
- static window_t ∗ GetWindowByName (const char ∗windowName)
- static window_t ∗ GetWindowByIndex (unsigned int windowIndex)

Private Attributes

- std::vector< window_t ∗ > windowList
- unsigned int screenResolution [2]
- unsigned int screenMousePosition [2]
- bool isInitialized

Static Private Attributes

- static windowManager ∗ instance = nullptr

4.2.1 Detailed Description

Definition at line 342 of file TinyWindow.h.

4.2.2 Constructor & Destructor Documentation

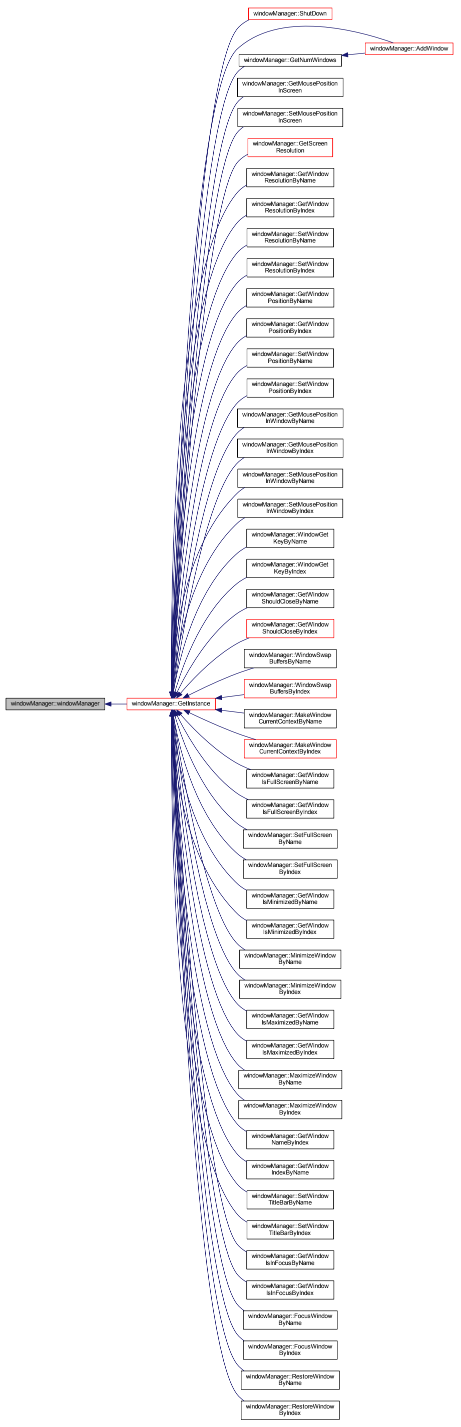
4.2.2.1 windowManager::windowManager ( ) `[inline]`

Definition at line 348 of file TinyWindow.h.

Referenced by GetInstance().

```
348 {}
```

Here is the caller graph for this function:


windowManager::ShutDown
windowManager::AddWindow
windowManager::GetNumWindows
windowManager::GetMousePosition InScreen
windowManager::SetMousePosition InScreen
windowManager::GetScreen Resolution
windowManager::GetWindow ResolutionByName
windowManager::GetWindow ResolutionByIndex
windowManager::SetWindow ResolutionByName
windowManager::SetWindow ResolutionByIndex
windowManager::GetWindow PositionByName
windowManager::GetWindow PositionByIndex
windowManager::SetWindow PositionByName
windowManager::SetWindow PositionByIndex
windowManager::GetMousePosition InWindowByName
windowManager::GetMousePosition InWindowByIndex
windowManager::SetMousePosition InWindowByName
windowManager::SetMousePosition InWindowByIndex
windowManager::WindowGet KeyByName
windowManager::WindowGet KeyByIndex
windowManager::GetWindow ShouldCloseByName
windowManager::GetWindow ShouldCloseByIndex
windowManager::WindowSwap BuffersByName
windowManager::WindowSwap BuffersByIndex
windowManager::windowManager
windowManager::GetInstance
windowManager::MakeWindow CurrentContextByName
windowManager::MakeWindow CurrentContextByIndex
windowManager::GetWindow IsFullScreenByName
windowManager::GetWindow IsFullScreenByIndex
windowManager::SetFullScreen ByName
windowManager::SetFullScreen ByIndex
windowManager::GetWindow IsMinimizedByName
windowManager::GetWindow IsMinimizedByIndex
windowManager::MinimizeWindow ByName
windowManager::MinimizeWindow ByIndex
windowManager::GetWindow IsMaximizedByName
windowManager::GetWindow IsMaximizedByIndex
windowManager::MaximizeWindow ByName
windowManager::MaximizeWindow ByIndex
windowManager::GetWindow NameByIndex
windowManager::GetWindow IndexByName
windowManager::SetWindow TitleBarByName
windowManager::SetWindow TitleBarByIndex
windowManager::GetWindow IsInFocusByName
windowManager::GetWindow IsInFocusByIndex
windowManager::FocusWindow ByName
windowManager::FocusWindow ByIndex
windowManager::RestoreWindow ByName
windowManager::RestoreWindow ByIndex

4.2.2.2 windowManager::∼windowManager (void) [inline]

Shutdown and delete all windows in the manager

Definition at line 353 of file TinyWindow.h.

```
    {
        if ( !GetInstance()->windowList.empty() )
        {
            for ( auto CurrentWindow : GetInstance()->windowList )
            {
                delete CurrentWindow;
            }
            GetInstance()->windowList.clear();
        }
    }
```

4.2.3 Member Function Documentation

4.2.3.1 static windowManager∗ windowManager::AddWindow (const char ∗ *windowName*, unsigned int *width* = DEFAULT_WINDOW_WIDTH, unsigned int *height* = DEFAULT_WINDOW_HEIGHT, unsigned int *colourBits* = 8, unsigned int *depthBits* = 8, unsigned int *stencilBits* = 8) [inline], [static]

Use this to add a window to the manager. returns a pointer to the manager which allows for the easy creation of multiple windows

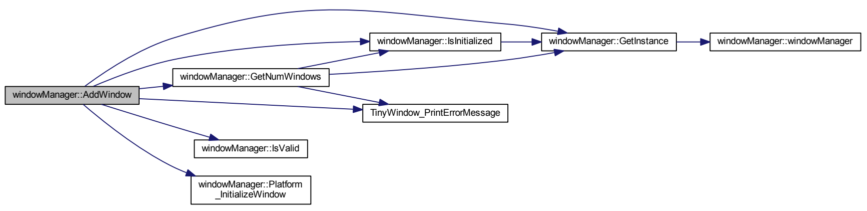
Definition at line 392 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, windowManager::window_t::colorBits, window↩ Manager::window_t::depthBits, GetInstance(), GetNumWindows(), windowManager::window_t::iD, instance, INV↩ ALID_WINDOW_NAME, IsInitialized(), IsValid(), windowManager::window_t::name, NOT_INITIALIZED, Platform↩ _InitializeWindow(), windowManager::window_t::resolution, windowManager::window_t::stencilBits, and Tiny↩ Window_PrintErrorMessage().

Referenced by main().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( IsValid( windowName ) )
            {
                window_t* newWindow = new window_t;
                newWindow->name = windowName;
                newWindow->resolution[ 0 ] = width;
                newWindow->resolution[ 1 ] = height;
                newWindow->colorBits = colourBits;
                newWindow->depthBits = depthBits;
                newWindow->stencilBits = stencilBits;

                instance->windowList.push_back( newWindow );
                newWindow->iD = GetNumWindows() - 1;

                Platform_InitializeWindow( newWindow );

                return instance;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::INVALID_WINDOW_NAME);
            return nullptr;
        }

        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return nullptr;
    }
```

Here is the call graph for this function:


windowManager::AddWindow
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::GetNumWindows
TinyWindow_PrintErrorMessage
windowManager::IsValid
windowManager::Platform
_InitializeWindow

Here is the caller graph for this function:

windowManager::AddWindow
main

4.2.3.2 static bool windowManager::DisableWindowDecoratorByIndex (unsigned int *windowIndex,* unsigned int *decorators*) [inline], [static]

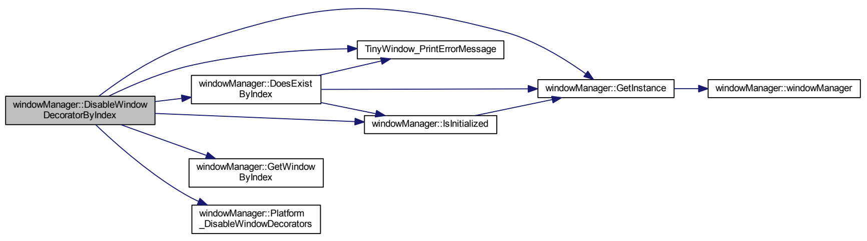
Disable windows decorators by index

Definition at line 1801 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_DisableWindowDecorators(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                window_t* window = GetWindowByIndex(windowIndex);
                Platform_DisableWindowDecorators(window, decorators);
                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::DisableWindow DecoratorByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _DisableWindowDecorators

4.2.3.3 static bool windowManager::DisableWindowDecoratorByName (const char ∗ *windowName,* unsigned int *decorators*) [inline], [static]

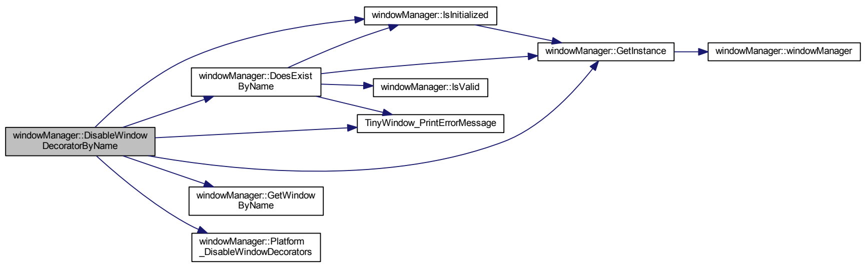
Disable windows decorators by name

Definition at line 1782 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get↩ WindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_DisableWindowDecorators(), TinyWindow_Print↩ ErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                window_t* window = GetWindowByName(windowName);
                Platform_DisableWindowDecorators(window, decorators);
                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::DisableWindow DecoratorByName
windowManager::GetWindow ByName
windowManager::Platform _DisableWindowDecorators

4.2.3.4 static bool windowManager::DoesExistByIndex (unsigned int *windowIndex*) [inline], [static], [private]

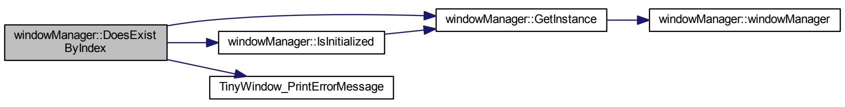
Definition at line 3073 of file TinyWindow.h.

References GetInstance(), INVALID_WINDOW_INDEX, IsInitialized(), NOT_INITIALIZED, and TinyWindow_Print↩ ErrorMessage().

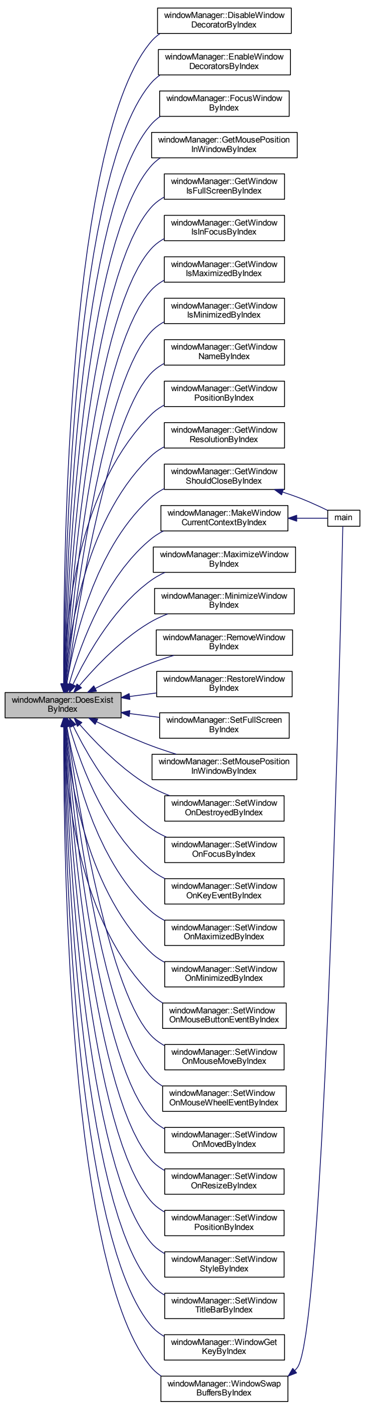
Referenced by DisableWindowDecoratorByIndex(), EnableWindowDecoratorsByIndex(), FocusWindowByIndex(), GetMousePositionInWindowByIndex(), GetWindowIsFullScreenByIndex(), GetWindowIsInFocusByIndex(), Get↩ WindowIsMaximizedByIndex(), GetWindowIsMinimizedByIndex(), GetWindowNameByIndex(), GetWindow↩ PositionByIndex(), GetWindowResolutionByIndex(), GetWindowShouldCloseByIndex(), MakeWindowCurrent↩ ContextByIndex(), MaximizeWindowByIndex(), MinimizeWindowByIndex(), RemoveWindowByIndex(), Restore↩ WindowByIndex(), SetFullScreenByIndex(), SetMousePositionInWindowByIndex(), SetWindowOnDestroyedBy↩ Index(), SetWindowOnFocusByIndex(), SetWindowOnKeyEventByIndex(), SetWindowOnMaximizedByIndex(), SetWindowOnMinimizedByIndex(), SetWindowOnMouseButtonEventByIndex(), SetWindowOnMouseMoveBy↩ Index(), SetWindowOnMouseWheelEventByIndex(), SetWindowOnMovedByIndex(), SetWindowOnResizeBy↩ Index(), SetWindowPositionByIndex(), SetWindowStyleByIndex(), SetWindowTitleBarByIndex(), WindowGetKey↩ ByIndex(), and WindowSwapBuffersByIndex().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( windowIndex <= ( instance->windowList.size() - 1 ) )
            {
                return true;
            }

            TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_WINDOW_INDEX );
            return false;
        }

        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED);
        return false;
    }
```

Here is the call graph for this function:


windowManager::DoesExist ByIndex
windowManager::IsInitialized
TinyWindow_PrintErrorMessage
windowManager::GetInstance
windowManager::windowManager

Here is the caller graph for this function:


windowManager::DisableWindow DecoratorByIndex
windowManager::EnableWindow DecoratorsByIndex
windowManager::FocusWindow ByIndex
windowManager::GetMousePosition InWindowByIndex
windowManager::GetWindow IsFullScreenByIndex
windowManager::GetWindow IsInFocusByIndex
windowManager::GetWindow IsMaximizedByIndex
windowManager::GetWindow IsMinimizedByIndex
windowManager::GetWindow NameByIndex
windowManager::GetWindow PositionByIndex
windowManager::GetWindow ResolutionByIndex
windowManager::GetWindow ShouldCloseByIndex
windowManager::MakeWindow CurrentContextByIndex
main
windowManager::MaximizeWindow ByIndex
windowManager::MinimizeWindow ByIndex
windowManager::RemoveWindow ByIndex
windowManager::RestoreWindow ByIndex
windowManager::DoesExist ByIndex
windowManager::SetFullScreen ByIndex
windowManager::SetMousePosition InWindowByIndex
windowManager::SetWindow OnDestroyedByIndex
windowManager::SetWindow OnFocusByIndex
windowManager::SetWindow OnKeyEventByIndex
windowManager::SetWindow OnMaximizedByIndex
windowManager::SetWindow OnMinimizedByIndex
windowManager::SetWindow OnMouseButtonEventByIndex
windowManager::SetWindow OnMouseMoveByIndex
windowManager::SetWindow OnMouseWheelEventByIndex
windowManager::SetWindow OnMovedByIndex
windowManager::SetWindow OnResizeByIndex
windowManager::SetWindow PositionByIndex
windowManager::SetWindow StyleByIndex
windowManager::SetWindow TitleBarByIndex
windowManager::WindowGet KeyByIndex
windowManager::WindowSwap BuffersByIndex

4.2.3.5 static bool windowManager::DoesExistByName (const char ∗ *windowName*) `[inline]`, `[static]`, `[private]`

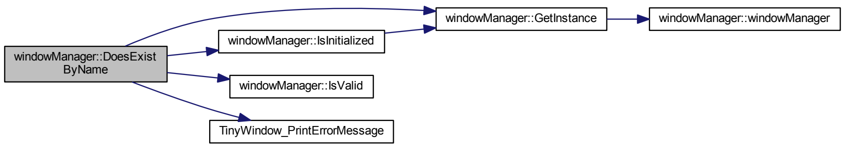
Definition at line 3052 of file TinyWindow.h.

References GetInstance(), INVALID_WINDOW_NAME, IsInitialized(), IsValid(), NOT_INITIALIZED, and Tiny↩ Window_PrintErrorMessage().

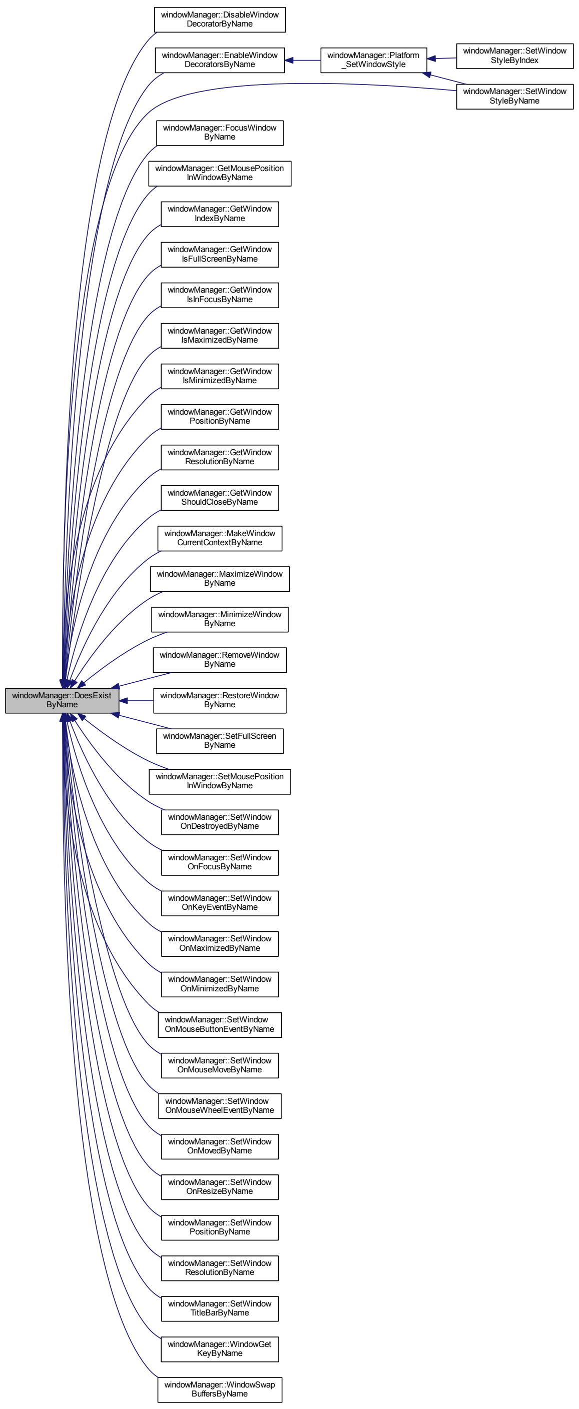
Referenced by DisableWindowDecoratorByName(), EnableWindowDecoratorsByName(), FocusWindowByName(), GetMousePositionInWindowByName(), GetWindowIndexByName(), GetWindowIsFullScreenByName(), Get↩WindowIsInFocusByName(), GetWindowIsMaximizedByName(), GetWindowIsMinimizedByName(), GetWindow↩PositionByName(), GetWindowResolutionByName(), GetWindowShouldCloseByName(), MakeWindowCurrent↩ContextByName(), MaximizeWindowByName(), MinimizeWindowByName(), RemoveWindowByName(), Restore↩WindowByName(), SetFullScreenByName(), SetMousePositionInWindowByName(), SetWindowOnDestroyedBy↩Name(), SetWindowOnFocusByName(), SetWindowOnKeyEventByName(), SetWindowOnMaximizedByName(), SetWindowOnMinimizedByName(), SetWindowOnMouseButtonEventByName(), SetWindowOnMouseMoveBy↩Name(), SetWindowOnMouseWheelEventByName(), SetWindowOnMovedByName(), SetWindowOnResizeBy↩Name(), SetWindowPositionByName(), SetWindowResolutionByName(), SetWindowStyleByName(), SetWindow↩TitleBarByName(), WindowGetKeyByName(), and WindowSwapBuffersByName().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( IsValid( windowName ) )
            {
                for ( auto window : instance->windowList )
                {
                    if( !strcmp( window->name, windowName ) )
                    {
                        return true;
                    }
                }
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::INVALID_WINDOW_NAME );
            return false;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED);
        return false;
    }
```

Here is the call graph for this function:


windowManager::DoesExist ByName
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::IsValid
TinyWindow_PrintErrorMessage

Here is the caller graph for this function:


windowManager::DisableWindowDecoratorByName
windowManager::EnableWindowDecoratorsByName
windowManager::Platform_SetWindowStyle
windowManager::SetWindowStyleByIndex
windowManager::SetWindowStyleByName
windowManager::FocusWindowByName
windowManager::GetMousePositionInWindowByName
windowManager::GetWindowIndexByName
windowManager::GetWindowIsFullScreenByName
windowManager::GetWindowIsInFocusByName
windowManager::GetWindowIsMaximizedByName
windowManager::GetWindowIsMinimizedByName
windowManager::GetWindowPositionByName
windowManager::GetWindowResolutionByName
windowManager::GetWindowShouldCloseByName
windowManager::MakeWindowCurrentContextByName
windowManager::MaximizeWindowByName
windowManager::MinimizeWindowByName
windowManager::RemoveWindowByName
windowManager::DoesExistByName
windowManager::RestoreWindowByName
windowManager::SetFullScreenByName
windowManager::SetMousePositionInWindowByName
windowManager::SetWindowOnDestroyedByName
windowManager::SetWindowOnFocusByName
windowManager::SetWindowOnKeyEventByName
windowManager::SetWindowOnMaximizedByName
windowManager::SetWindowOnMinimizedByName
windowManager::SetWindowOnMouseButtonEventByName
windowManager::SetWindowOnMouseMoveByName
windowManager::SetWindowOnMouseWheelEventByName
windowManager::SetWindowOnMovedByName
windowManager::SetWindowOnResizeByName
windowManager::SetWindowPositionByName
windowManager::SetWindowResolutionByName
windowManager::SetWindowTitleBarByName
windowManager::WindowGetKeyByName
windowManager::WindowSwapBuffersByName

4.2.3.6 static bool windowManager::EnableWindowDecoratorsByIndex (unsigned int *windowIndex,* unsigned int *decorators*) [inline], [static]

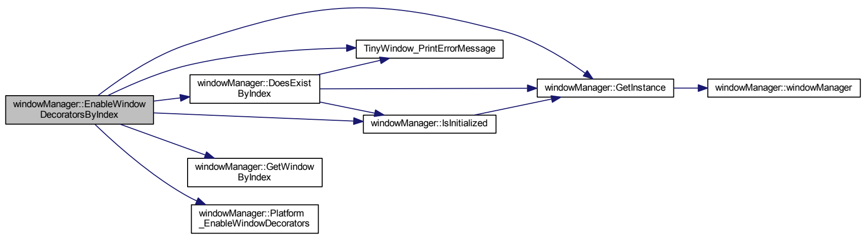
Enable windows decorators by index

Definition at line 1762 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get↩
WindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_EnableWindowDecorators(), TinyWindow_Print↩
ErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if (GetInstance()->IsInitialized())
        {
            if (DoesExistByIndex(windowIndex))
            {
                window_t* window = GetWindowByIndex(windowIndex);
                Platform_EnableWindowDecorators(window, decorators);
                return true;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED);
        return false;
    }
```

Here is the call graph for this function:


windowManager::EnableWindow DecoratorsByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _EnableWindowDecorators

4.2.3.7 static bool windowManager::EnableWindowDecoratorsByName (const char ∗ *windowName*, unsigned int *decorators*) [inline], [static]

Enable window decorators by name

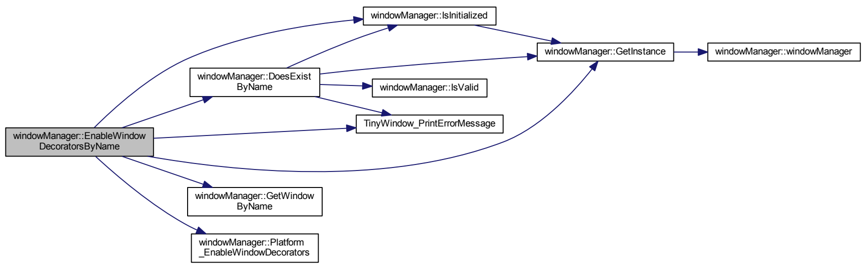
Definition at line 1743 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get↩
WindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_EnableWindowDecorators(), TinyWindow_Print↩
ErrorMessage(), and WINDOW_NOT_FOUND.

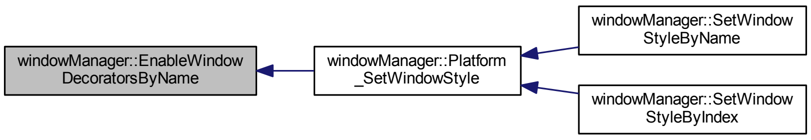
Referenced by Platform_SetWindowStyle().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                window_t* window = GetWindowByName(windowName);
                Platform_EnableWindowDecorators(window, decorators);
                return true;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::EnableWindow DecoratorsByName
windowManager::IsInitialized
windowManager::DoesExist ByName
windowManager::GetInstance
windowManager::windowManager
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName
windowManager::Platform _EnableWindowDecorators

Here is the caller graph for this function:


windowManager::EnableWindow DecoratorsByName
windowManager::Platform _SetWindowStyle
windowManager::SetWindow StyleByName
windowManager::SetWindow StyleByIndex

4.2.3.8 static bool windowManager::FocusWindowByIndex (unsigned int *windowIndex,* bool *newState*) `[inline]`, `[static]`

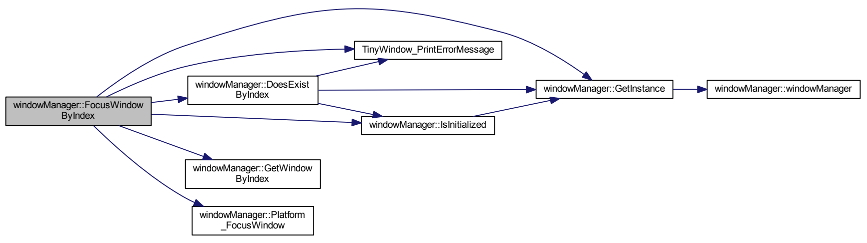
Set the window to be in focus by index

Definition at line 1493 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get←WindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_FocusWindow(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if (GetInstance()->IsInitialized())
        {
            if (DoesExistByIndex(windowIndex))
            {
                window_t* window = GetWindowByIndex(windowIndex);
                Platform_FocusWindow(window, newState);
                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED);
        return false;
    }
```

Here is the call graph for this function:


windowManager::FocusWindow ByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _FocusWindow

4.2.3.9 static bool windowManager::FocusWindowByName (const char ∗ *windowName,* bool *newState*) [inline], [static]

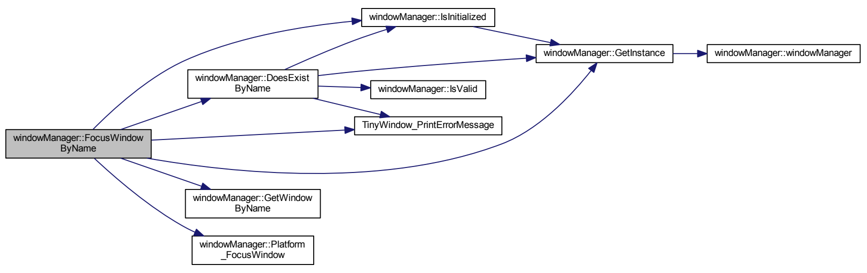
Set the window to be in focus by name

Definition at line 1474 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get←WindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_FocusWindow(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                window_t* window = GetWindowByName(windowName);
                Platform_FocusWindow(window, newState);
                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::FocusWindow ByName
windowManager::GetWindow ByName
windowManager::Platform _FocusWindow

4.2.3.10 static windowManager∗ windowManager::GetInstance (void) [inline], [static], [private]

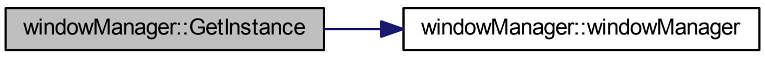
Definition at line 2432 of file TinyWindow.h.

References instance, and windowManager().

Referenced by AddWindow(), DisableWindowDecoratorByIndex(), DisableWindowDecoratorByName(), DoesExistByIndex(), DoesExistByName(), EnableWindowDecoratorsByIndex(), EnableWindowDecoratorsByName(), FocusWindowByIndex(), FocusWindowByName(), GetMousePositionInScreen(), GetMousePositionInWindowByIndex(), GetMousePositionInWindowByName(), GetNumWindows(), GetScreenResolution(), GetWindowIndexByName(), GetWindowIsFullScreenByIndex(), GetWindowIsFullScreenByName(), GetWindowIsInFocusByIndex(), GetWindowIsInFocusByName(), GetWindowIsMaximizedByIndex(), GetWindowIsMaximizedByName(), GetWindowIsMinimizedByIndex(), GetWindowIsMinimizedByName(), GetWindowNameByIndex(), GetWindowPositionByIndex(), GetWindowPositionByName(), GetWindowResolutionByIndex(), GetWindowResolutionByName(), GetWindowShouldCloseByIndex(), GetWindowShouldCloseByName(), Initialize(), IsInitialized(), MakeWindowCurrentContextByIndex(), MakeWindowCurrentContextByName(), MaximizeWindowByIndex(), MaximizeWindowByName(), MinimizeWindowByIndex(), MinimizeWindowByName(), PollForEvents(), RemoveWindowByIndex(), RemoveWindowByName(), RestoreWindowByIndex(), RestoreWindowByName(), SetFullScreenByIndex(), SetFullScreenByName(), SetMousePositionInScreen(), SetMousePositionInWindowByIndex(), SetMousePositionInWindowByName(), SetWindowOnDestroyedByIndex(), SetWindowOnDestroyedByName(), SetWindowOnFocusByIndex(), SetWindowOnFocusByName(), SetWindowOnKeyEventByIndex(), SetWindowOnKeyEventByName(), SetWindowOnMaximizedByIndex(), SetWindowOnMaximizedByName(), SetWindowOnMinimizedByIndex(), SetWindowOnMinimizedByName(), SetWindowOnMouseButtonEventByIndex(), SetWindowOnMouseButtonEventByName(), SetWindowOnMouseMoveByIndex(), SetWindowOnMouseMoveByName(), SetWindowOnMouseWheelEventByIndex(), SetWindowOnMouseWheelEventByName(), SetWindowOnMovedByIndex(), SetWindowOnMovedByName(), SetWindowOnResizeByIndex(), SetWindowOnResizeByName(), SetWindowPositionByIndex(), SetWindowPositionByName(), SetWindowResolutionByIndex(), SetWindowResolutionByName(), SetWindowStyleByIndex(), SetWindowStyleByName(), SetWindowTitleBarByIndex(), SetWindowTitleBarByName(), ShutDown(), WaitForEvents(), WindowGetKeyByIndex(), WindowGetKeyByName(), WindowSwapBuffersByIndex(), and WindowSwapBuffersByName().

```
{
    if ( windowManager::instance == nullptr )
    {
        windowManager::instance = new windowManager();
        return windowManager::instance;
    }

    else
    {
        return windowManager::instance;
    }
}
```

Here is the call graph for this function:



4.2.3.11 static bool windowManager::GetMousePositionInScreen (unsigned int & *x,* unsigned int & *y*) [inline], [static]

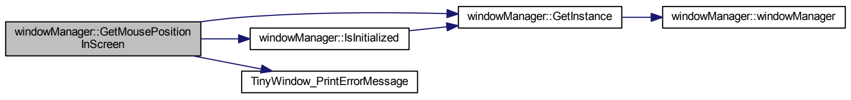
Return the mouse position in screen co-ordinates

Definition at line 438 of file TinyWindow.h.

References GetInstance(), instance, IsInitialized(), NOT_INITIALIZED, screenMousePosition, and TinyWindow_↩ PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            x = instance->screenMousePosition[0];
            y = instance->screenMousePosition[1];
            return true;
        }

        TinyWindow_PrintErrorMessage(
    tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::GetMousePosition InScreen
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage

4.2.3.12 static unsigned int∗ windowManager::GetMousePositionInScreen (void) [inline], [static]

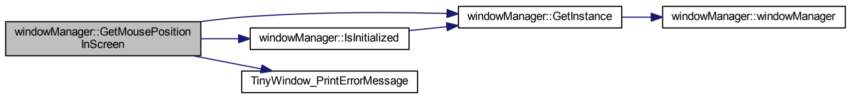
Return the mouse position in screen co-ordinates

Definition at line 453 of file TinyWindow.h.

References GetInstance(), instance, IsInitialized(), NOT_INITIALIZED, screenMousePosition, and TinyWindow_↩ PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            return instance->screenMousePosition;
        }

        TinyWindow_PrintErrorMessage(
    tinyWindowError_t::NOT_INITIALIZED );
        return nullptr;
    }
```

Here is the call graph for this function:


windowManager::GetMousePosition InScreen
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage

4.2.3.13 static bool windowManager::GetMousePositionInWindowByIndex (unsigned int *windowIndex,* unsigned int & *x,* unsigned int & *y*) [inline], [static]

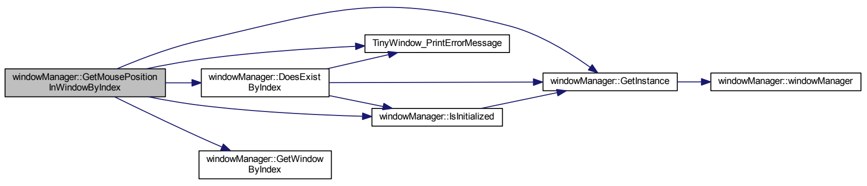
Return the mouse position relative to the given window's co-ordinates by setting X and Y

Definition at line 812 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), windowManager::window_t::mousePosition, NOT_INITIALIZED, TINYWINDOW_ERROR, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                x = GetWindowByIndex( windowIndex )->mousePosition[ 0 ];
                y = GetWindowByIndex( windowIndex )->mousePosition[ 1 ];
                return true;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return (bool)tinyWindowError_t::TINYWINDOW_ERROR;
    }
```

Here is the call graph for this function:


windowManager::GetMousePosition InWindowByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.14 static unsigned int∗ windowManager::GetMousePositionInWindowByIndex (unsigned int *windowIndex*) [inline], [static]

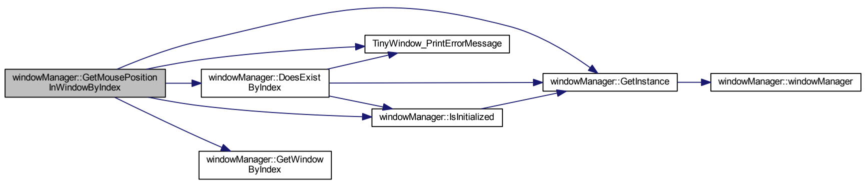
Return the mouse Position relative to the given window's co-ordinates as an array

Definition at line 849 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), windowManager::window_t::mousePosition, NOT_INITIALIZED, and TinyWindow_PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                return GetWindowByIndex( windowIndex )->mousePosition;
            }
            return nullptr;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return nullptr;
    }
```

Here is the call graph for this function:


windowManager::GetMousePosition InWindowByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.15 static bool windowManager::GetMousePositionInWindowByName (const char ∗ *windowName,* unsigned int & *x,* unsigned int & *y*) [inline], [static]

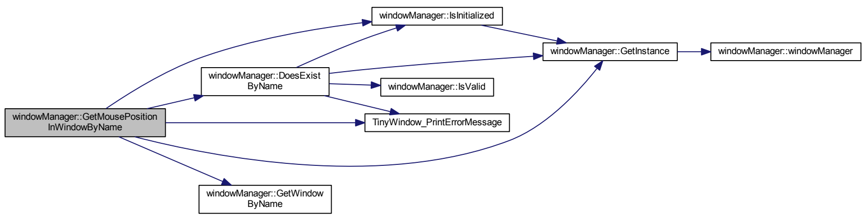
Return the mouse Position relative to the given window's co-ordinates by setting X and Y

Definition at line 793 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), windowManager::window_↩ t::mousePosition, NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                x = GetWindowByName( windowName )->mousePosition[ 0 ];
                y = GetWindowByName( windowName )->mousePosition[ 1 ];
                return true;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::GetMousePosition InWindowByName
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.16 static unsigned int∗ windowManager::GetMousePositionInWindowByName (const char ∗ *windowName*) [inline], [static]

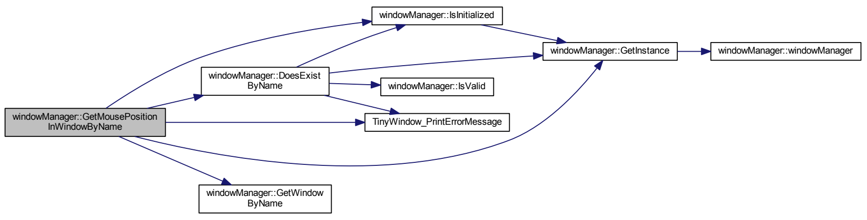
Return the mouse Position relative to the given window's co-ordinates as an array

Definition at line 832 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), windowManager::window_↩ t::mousePosition, NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                return GetWindowByName( windowName )->
      mousePosition;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return nullptr;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return nullptr;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetMousePosition InWindowByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.17 static int windowManager::GetNumWindows (void) [inline], [static]

Return the total amount of windows the manager has

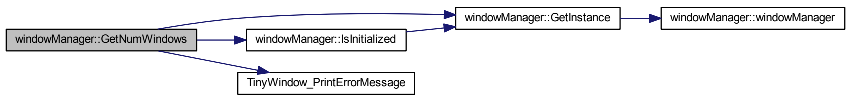
Definition at line 424 of file TinyWindow.h.

References GetInstance(), IsInitialized(), NOT_INITIALIZED, TINYWINDOW_ERROR, and TinyWindow_Print↩ ErrorMessage().

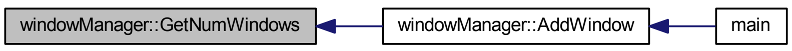
Referenced by AddWindow().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            return instance->windowList.size();
        }

        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return (int)tinyWindowError_t::TINYWINDOW_ERROR;
    }
```

Here is the call graph for this function:


windowManager::GetNumWindows
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage

Here is the caller graph for this function:



4.2.3.18 static unsigned int* windowManager::GetScreenResolution (void) [inline], [static]

Return the Resolution of the current screen

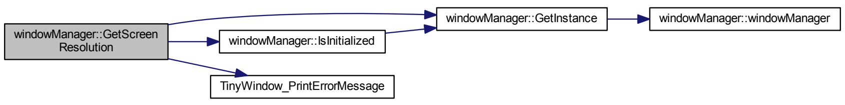
Definition at line 492 of file TinyWindow.h.

References GetInstance(), instance, IsInitialized(), NOT_INITIALIZED, screenResolution, and TinyWindow_Print←ErrorMessage().

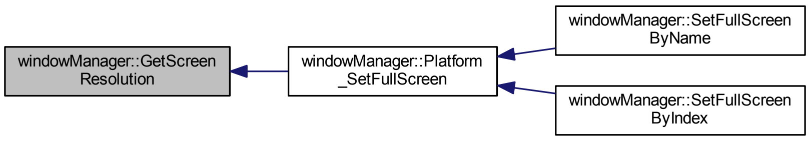
Referenced by Platform_SetFullScreen().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
#if defined( _WIN32 ) || defined( _WIN64 )
            RECT screen;
            HWND desktop = GetDesktopWindow();
            GetWindowRect( desktop, &screen );

            instance->screenResolution[0] = screen.right;
            instance->screenResolution[1] = screen.bottom;
            return instance->screenResolution;

#elif defined(__linux__)
            instance->screenResolution[0] = WidthOfScreen(XDefaultScreenOfDisplay(
  instance->currentDisplay));
            instance->screenResolution[1] = HeightOfScreen(XDefaultScreenOfDisplay(
  instance->currentDisplay));

            return instance->screenResolution;
#endif
        }

        TinyWindow_PrintErrorMessage(
  tinyWindowError_t::NOT_INITIALIZED );
        return nullptr;
    }
```

Here is the call graph for this function:


windowManager::GetScreen Resolution
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage

Here is the caller graph for this function:


windowManager::GetScreen Resolution
windowManager::Platform _SetFullScreen
windowManager::SetFullScreen ByName
windowManager::SetFullScreen ByIndex

4.2.3.19 static bool windowManager::GetScreenResolution (unsigned int & *width,* unsigned int & *Height*) [inline], [static]

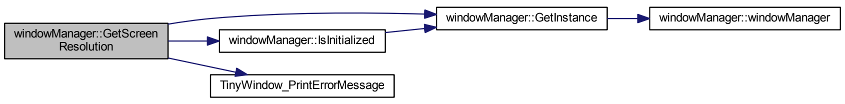
Return the Resolution of the current screen

Definition at line 519 of file TinyWindow.h.

References GetInstance(), IsInitialized(), NOT_INITIALIZED, and TinyWindow_PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
#if defined( _WIN32 ) || defined( _WIN64 )
            RECT screen;
            HWND desktop = GetDesktopWindow();
            GetWindowRect( desktop, &screen );
            width = screen.right;
            Height = screen.bottom;
#elif defined(__linux__)
            width = WidthOfScreen(XDefaultScreenOfDisplay(instance->currentDisplay));
            Height = HeightOfScreen(XDefaultScreenOfDisplay(instance->currentDisplay));

            instance->screenResolution[0] = width;
            instance->screenResolution[1] = Height;
#endif
            return true;
        }

        TinyWindow_PrintErrorMessage(
    tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::GetScreen Resolution
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage

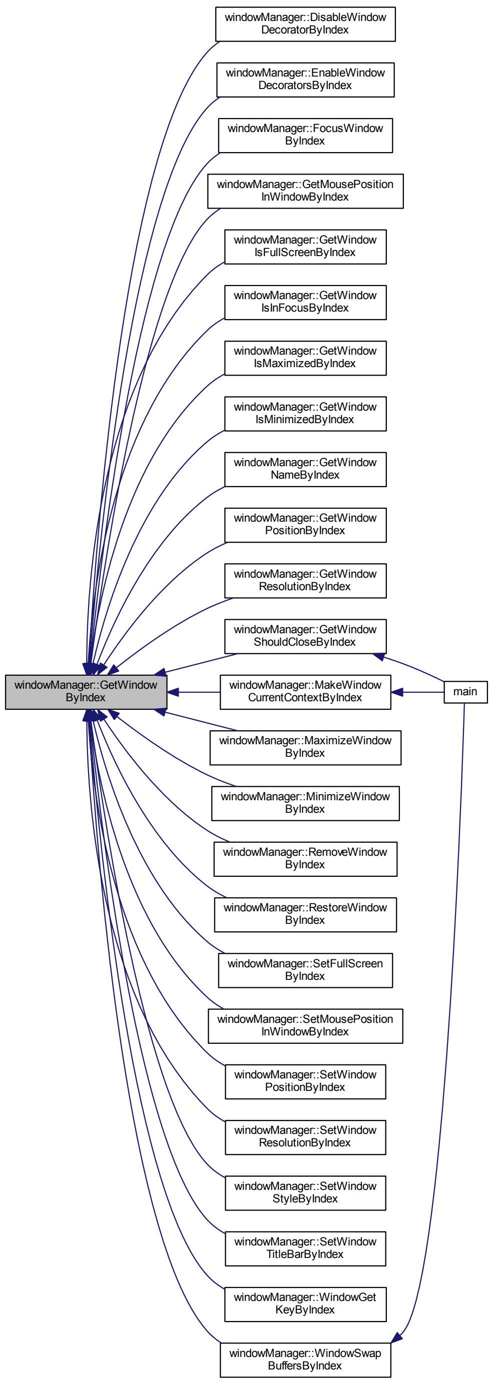
4.2.3.20 static window_t∗ windowManager::GetWindowByIndex (unsigned int *windowIndex*) [inline], [static], [private]

Definition at line 3104 of file TinyWindow.h.

Referenced by DisableWindowDecoratorByIndex(), EnableWindowDecoratorsByIndex(), FocusWindowByIndex(), GetMousePositionInWindowByIndex(), GetWindowIsFullScreenByIndex(), GetWindowIsInFocusByIndex(), Get←WindowIsMaximizedByIndex(), GetWindowIsMinimizedByIndex(), GetWindowNameByIndex(), GetWindow←PositionByIndex(), GetWindowResolutionByIndex(), GetWindowShouldCloseByIndex(), MakeWindowCurrent←ContextByIndex(), MaximizeWindowByIndex(), MinimizeWindowByIndex(), RemoveWindowByIndex(), Restore←WindowByIndex(), SetFullScreenByIndex(), SetMousePositionInWindowByIndex(), SetWindowPositionByIndex(), SetWindowResolutionByIndex(), SetWindowStyleByIndex(), SetWindowTitleBarByIndex(), WindowGetKeyBy←Index(), and WindowSwapBuffersByIndex().

```
{
    if ( windowIndex <= instance->windowList.size() - 1 )
    {
        return instance->windowList[windowIndex];
    }
    return nullptr;
}
```

Here is the caller graph for this function:


windowManager::DisableWindow DecoratorByIndex
windowManager::EnableWindow DecoratorsByIndex
windowManager::FocusWindow ByIndex
windowManager::GetMousePosition InWindowByIndex
windowManager::GetWindow IsFullScreenByIndex
windowManager::GetWindow IsInFocusByIndex
windowManager::GetWindow IsMaximizedByIndex
windowManager::GetWindow IsMinimizedByIndex
windowManager::GetWindow NameByIndex
windowManager::GetWindow PositionByIndex
windowManager::GetWindow ResolutionByIndex
windowManager::GetWindow ShouldCloseByIndex
windowManager::GetWindow ByIndex
windowManager::MakeWindow CurrentContextByIndex
main
windowManager::MaximizeWindow ByIndex
windowManager::MinimizeWindow ByIndex
windowManager::RemoveWindow ByIndex
windowManager::RestoreWindow ByIndex
windowManager::SetFullScreen ByIndex
windowManager::SetMousePosition InWindowByIndex
windowManager::SetWindow PositionByIndex
windowManager::SetWindow ResolutionByIndex
windowManager::SetWindow StyleByIndex
windowManager::SetWindow TitleBarByIndex
windowManager::WindowGet KeyByIndex
windowManager::WindowSwap BuffersByIndex

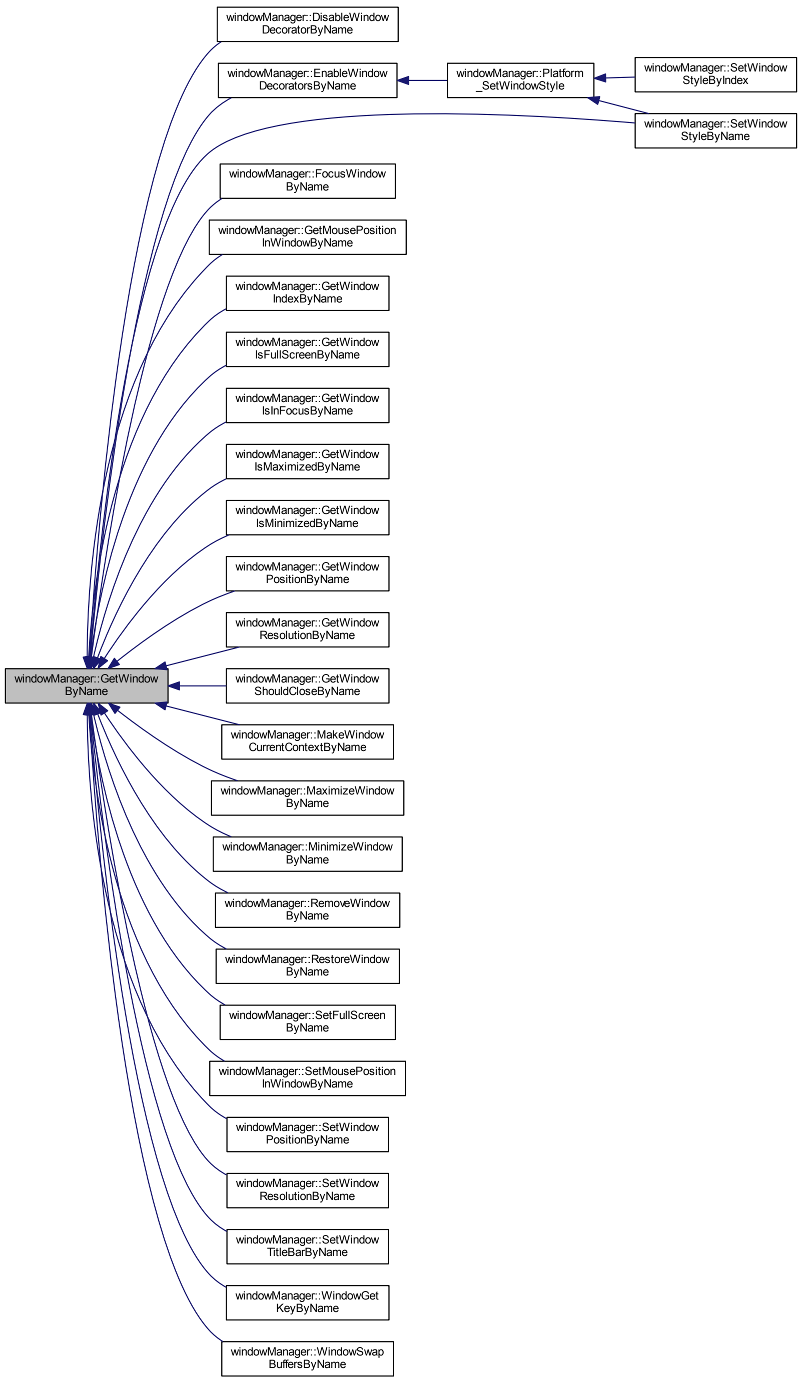
4.2.3.21 static window_t∗ windowManager::GetWindowByName (const char ∗ *windowName*) [inline], [static], [private]

Definition at line 3090 of file TinyWindow.h.

Referenced by DisableWindowDecoratorByName(), EnableWindowDecoratorsByName(), FocusWindowByName(), GetMousePositionInWindowByName(), GetWindowIndexByName(), GetWindowIsFullScreenByName(), Get↩WindowIsInFocusByName(), GetWindowIsMaximizedByName(), GetWindowIsMinimizedByName(), GetWindow↩PositionByName(), GetWindowResolutionByName(), GetWindowShouldCloseByName(), MakeWindowCurrent↩ContextByName(), MaximizeWindowByName(), MinimizeWindowByName(), RemoveWindowByName(), Restore↩WindowByName(), SetFullScreenByName(), SetMousePositionInWindowByName(), SetWindowPositionByName(), SetWindowResolutionByName(), SetWindowStyleByName(), SetWindowTitleBarByName(), WindowGetKeyBy↩Name(), and WindowSwapBuffersByName().

```
{
        for( auto window : instance->windowList )
        {
            if ( !strcmp( window->name, windowName ) )
            {
                return window;
            }
        }

        return nullptr;
}
```

Here is the caller graph for this function:


windowManager::DisableWindow DecoratorByName
windowManager::EnableWindow DecoratorsByName
windowManager::Platform _SetWindowStyle
windowManager::SetWindow StyleByIndex
windowManager::SetWindow StyleByName
windowManager::FocusWindow ByName
windowManager::GetMousePosition InWindowByName
windowManager::GetWindow IndexByName
windowManager::GetWindow IsFullScreenByName
windowManager::GetWindow IsInFocusByName
windowManager::GetWindow IsMaximizedByName
windowManager::GetWindow IsMinimizedByName
windowManager::GetWindow PositionByName
windowManager::GetWindow ResolutionByName
windowManager::GetWindow ByName
windowManager::GetWindow ShouldCloseByName
windowManager::MakeWindow CurrentContextByName
windowManager::MaximizeWindow ByName
windowManager::MinimizeWindow ByName
windowManager::RemoveWindow ByName
windowManager::RestoreWindow ByName
windowManager::SetFullScreen ByName
windowManager::SetMousePosition InWindowByName
windowManager::SetWindow PositionByName
windowManager::SetWindow ResolutionByName
windowManager::SetWindow TitleBarByName
windowManager::WindowGet KeyByName
windowManager::WindowSwap BuffersByName

4.2.3.22 static unsigned int windowManager::GetWindowIndexByName (const char ∗ *windowName*) [inline], [static]

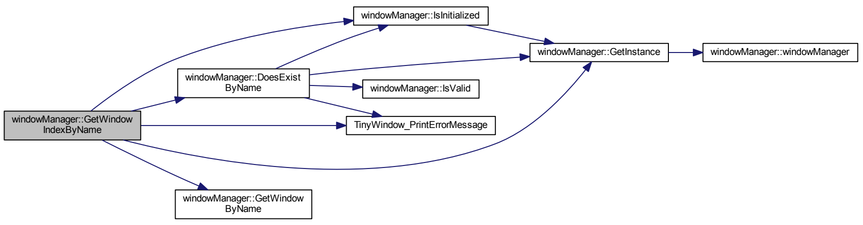
Get window index by name

Definition at line 1309 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), windowManager::window_t::iD, Is↩ Initialized(), NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                return GetWindowByName( windowName )->iD;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow IndexByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.23 static bool windowManager::GetWindowIsFullScreenByIndex (unsigned int *windowIndex*) [inline], [static]

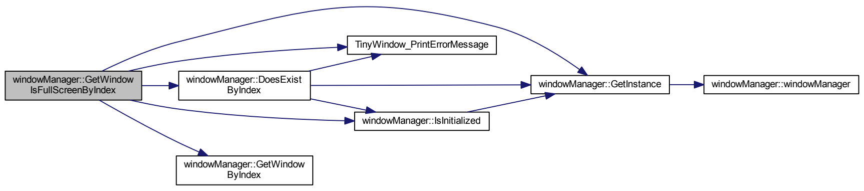
Return whether the given window is in full screen mode

Definition at line 1083 of file TinyWindow.h.

References windowManager::window_t::currentState, DoesExistByIndex(), FULLSCREEN, GetInstance(), Get↩ WindowByIndex(), IsInitialized(), NOT_INITIALIZED, and TinyWindow_PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                return (GetWindowByIndex(windowIndex)->currentState ==
tinyWindowState_t::FULLSCREEN);
            }

            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED);
        return false;
    }
```

Here is the call graph for this function:


windowManager::GetWindow IsFullScreenByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.24 static bool windowManager::GetWindowIsFullScreenByName (const char ∗ *windowName*) [inline], [static]

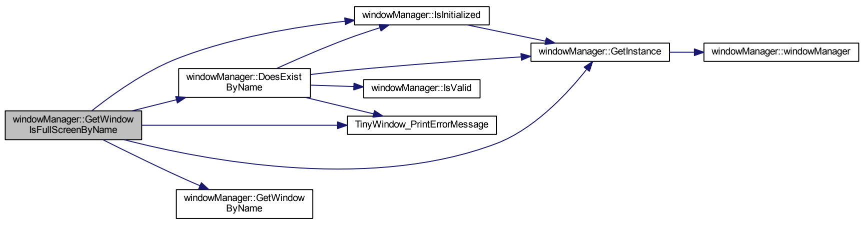
Return whether the given window is in full screen mode

Definition at line 1066 of file TinyWindow.h.

References windowManager::window_t::currentState, DoesExistByName(), FULLSCREEN, GetInstance(), Get↩
WindowByName(), IsInitialized(), NOT_INITIALIZED, and TinyWindow_PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                return ( GetWindowByName( windowName )->currentState ==
tinyWindowState_t::FULLSCREEN );
            }

            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED);
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow IsFullScreenByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.25 static bool windowManager::GetWindowIsInFocusByIndex (unsigned int *windowIndex*) [inline], [static]

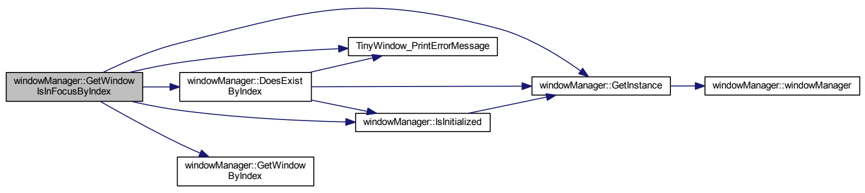
Get whether the window is in focus by index

Definition at line 1455 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), windowManager::window_t::inFocus, Is←
Initialized(), NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                return GetWindowByIndex( windowIndex )->inFocus;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }

        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::GetWindow IsInFocusByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.26 static bool windowManager::GetWindowIsInFocusByName (const char ∗ *windowName*) [inline], [static]

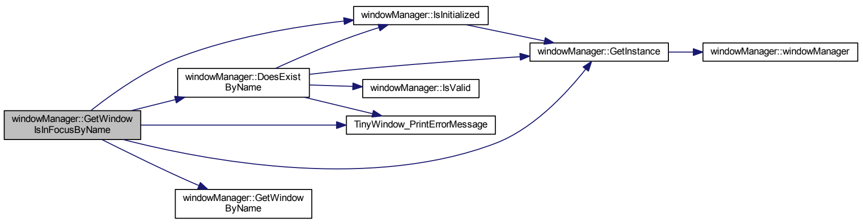
Get whether the window is in focus by name

Definition at line 1438 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), windowManager::window_t::inFocus, Is←
Initialized(), NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                return GetWindowByName( windowName )->inFocus;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow IsInFocusByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.27 static bool windowManager::GetWindowIsMaximizedByIndex (unsigned int *windowIndex*) [inline], [static]

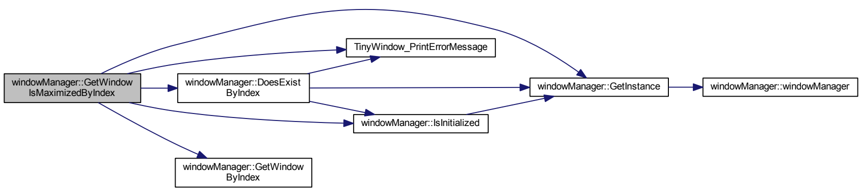
Return whether the given window is currently maximized

Definition at line 1235 of file TinyWindow.h.

References windowManager::window_t::currentState, DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), MAXIMIZED, NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                return (GetWindowByIndex(windowIndex)->currentState ==
tinyWindowState_t::MAXIMIZED);
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


TinyWindow_PrintErrorMessage
windowManager::GetWindow IsMaximizedByIndex
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.28 static bool windowManager::GetWindowIsMaximizedByName (const char ∗ *windowName*) [inline], [static]

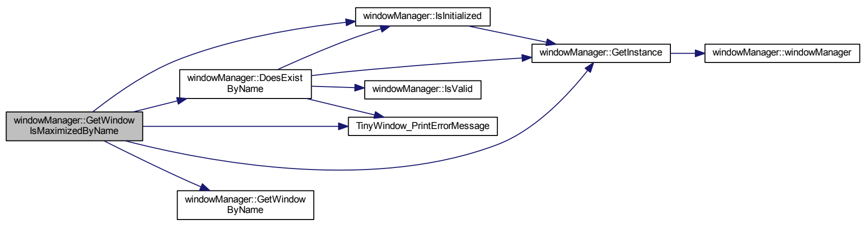
Return whether the current window is currently maximized

Definition at line 1217 of file TinyWindow.h.

References windowManager::window_t::currentState, DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), MAXIMIZED, NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                return (GetWindowByName(windowName)->currentState == tinyWindowState_t::MAXIMIZED);
            }

            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow IsMaximizedByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.29 static bool windowManager::GetWindowIsMinimizedByIndex (unsigned int *windowIndex*) [inline], [static]

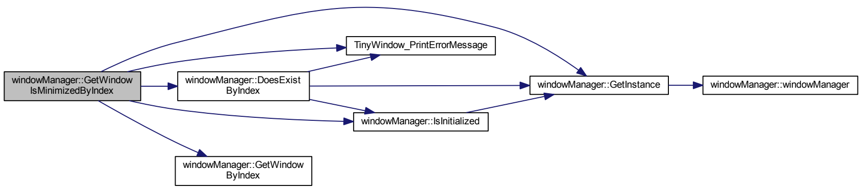
Returns whether the given window is minimized

Definition at line 1161 of file TinyWindow.h.

References windowManager::window_t::currentState, DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), MINIMIZED, NOT_INITIALIZED, TINYWINDOW_ERROR, and TinyWindow_PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                return (GetWindowByIndex(windowIndex)->currentState == tinyWindowState_t::MINIMIZED);
            }
            return (bool)tinyWindowError_t::TINYWINDOW_ERROR;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return (bool)tinyWindowError_t::TINYWINDOW_ERROR;
    }
```

Here is the call graph for this function:


windowManager::GetWindow IsMinimizedByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.30 static bool windowManager::GetWindowIsMinimizedByName (const char ∗ *windowName*) [inline], [static]

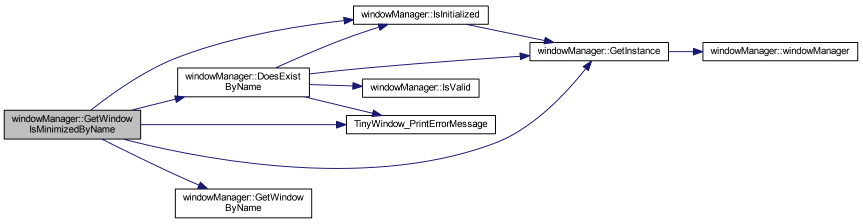
Returns whether the given window is minimized

Definition at line 1145 of file TinyWindow.h.

References windowManager::window_t::currentState, DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), MINIMIZED, NOT_INITIALIZED, TINYWINDOW_ERROR, and TinyWindow_PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                return (GetWindowByName(windowName)->currentState ==
tinyWindowState_t::MINIMIZED);
            }
            return (bool)tinyWindowError_t::TINYWINDOW_ERROR;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return (bool)tinyWindowError_t::TINYWINDOW_ERROR;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow IsMinimizedByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.31 static const char∗ windowManager::GetWindowNameByIndex (unsigned int *windowIndex*) [inline], [static]

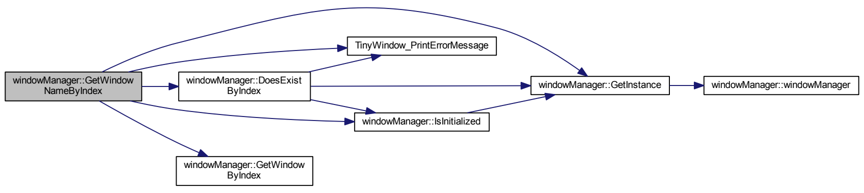
Get window name by index

Definition at line 1292 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), windowManager::window_t::name, NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                return GetWindowByIndex( windowIndex )->name;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return nullptr;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return nullptr;
    }
```

Here is the call graph for this function:


windowManager::GetWindowNameByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExistByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindowByIndex

4.2.3.32 static bool windowManager::GetWindowPositionByIndex (unsigned int *windowIndex*, unsigned int & *x*, unsigned int & *y*) [inline], [static]

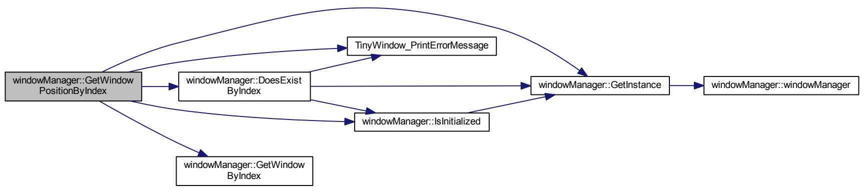
Return the Position of the given window relative to screen co-ordinates by setting X and Y

Definition at line 689 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), NOT_INITIALIZED, windowManager::window_t::position, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                x = GetWindowByIndex( windowIndex )->position[ 0 ];
                y = GetWindowByIndex( windowIndex )->position[ 1 ];
                return true;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::GetWindow PositionByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.33 static unsigned int∗ windowManager::GetWindowPositionByIndex (unsigned int *windowIndex*) [inline], [static]

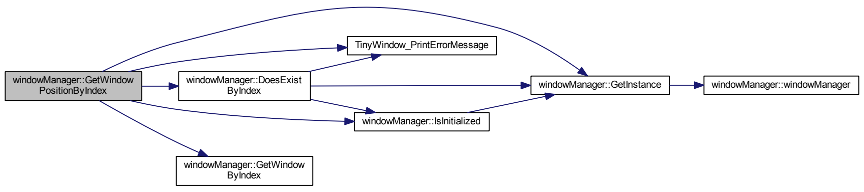
Return the Position of the given window relative to screen co-ordinates as an array

Definition at line 727 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), NOT_INITIALIZED, window←Manager::window_t::position, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                return GetWindowByIndex( windowIndex )->position;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return nullptr;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return nullptr;
    }
```

Here is the call graph for this function:


windowManager::GetWindow PositionByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.34 static bool windowManager::GetWindowPositionByName (const char ∗ *windowName,* unsigned int & *x,* unsigned int & *y*) [inline], [static]

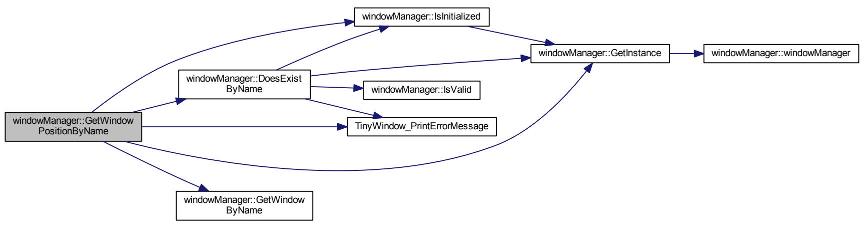
Return the Position of the given window relative to screen co-ordinates by setting X and Y

Definition at line 670 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), NOT_INITIALIZED, windowManager::window_t::position, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
  {
      if ( GetInstance()->IsInitialized() )
      {
          if ( DoesExistByName( windowName ) )
          {
              x = GetWindowByName( windowName )->position[ 0 ];
              y = GetWindowByName( windowName )->position[ 1 ];
              return true;
          }
          TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
          return false;
      }
      TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
      return false;
  }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow PositionByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.35 static unsigned int∗ windowManager::GetWindowPositionByName (const char ∗ *windowName*) [inline], [static]

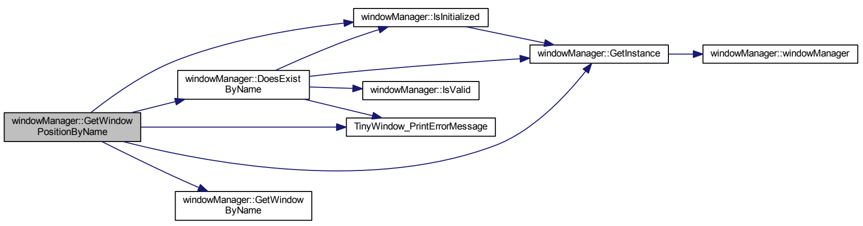
Return the Position of the given window relative to screen co-ordinates as an array

Definition at line 709 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), NOT_INITIALIZED, windowManager::window_t::position, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
  {
      if ( GetInstance()->IsInitialized() )
      {
          if ( DoesExistByName( windowName ) )
          {
              return GetWindowByName( windowName )->position;
          }
          TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
          return nullptr;
      }

      TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
      return nullptr;
  }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow PositionByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.36 static bool windowManager::GetWindowResolutionByIndex (unsigned int *windowIndex,* unsigned int & *width,* unsigned int & *height*) [inline], [static]

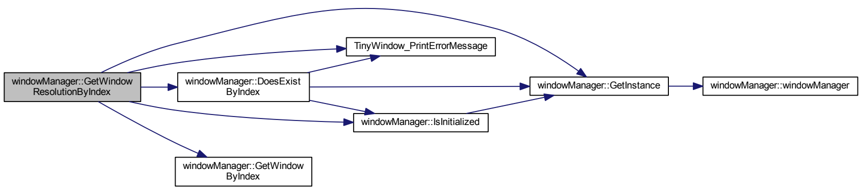
Return the Resolution of the given window by setting width and height

Definition at line 565 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), NOT_INITIALIZED, window↩ Manager::window_t::resolution, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                width = GetWindowByIndex( windowIndex )->resolution[ 0 ];
                height = GetWindowByIndex( windowIndex )->resolution[ 1 ];

                return true;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }

        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


TinyWindow_PrintErrorMessage
windowManager::GetWindow ResolutionByIndex
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.37 static unsigned int∗ windowManager::GetWindowResolutionByIndex (unsigned int *windowIndex*) [inline], [static]

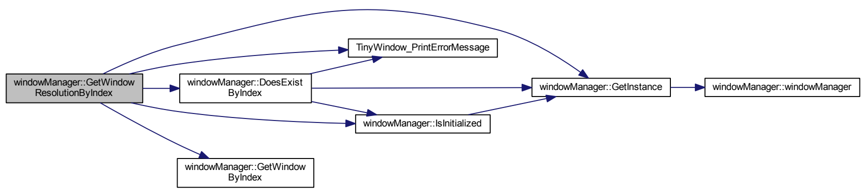
Return the Resolution of the Given Window as an array of doubles

Definition at line 605 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), NOT_INITIALIZED, window←Manager::window_t::resolution, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                return GetWindowByIndex( windowIndex )->
resolution;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return nullptr;
        }

        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return nullptr;
    }
```

Here is the call graph for this function:


windowManager::GetWindow ResolutionByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.38 static bool windowManager::GetWindowResolutionByName (const char ∗ *windowName,* unsigned int & *width,* unsigned int & *height*) [inline], [static]

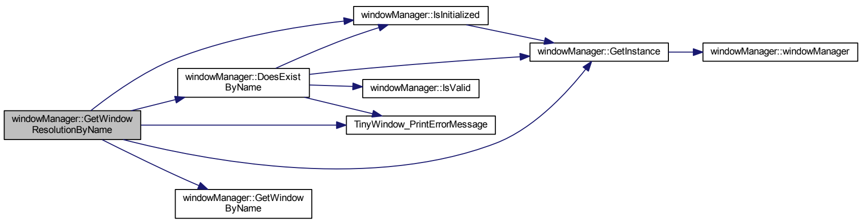
Return the Resolution of the given window by setting width and height

Definition at line 546 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), NOT_INITIALIZED, windowManager::window_t::resolution, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                width = GetWindowByName( windowName )->resolution[ 0 ];
                height = GetWindowByName( windowName )->
resolution[ 1 ];
                return false;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow ResolutionByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.39 static unsigned int∗ windowManager::GetWindowResolutionByName (const char ∗ *windowName*) [inline], [static]

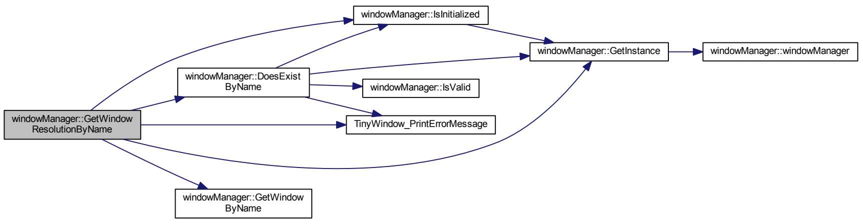
Return the Resolution of the given Window as an array of doubles

Definition at line 587 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), NOT_INITIALIZED, windowManager::window_t::resolution, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                return GetWindowByName( windowName )->resolution;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return nullptr;
        }

        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return nullptr;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow ResolutionByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.40 static bool windowManager::GetWindowShouldCloseByIndex (unsigned int *windowIndex*) [inline], [static]

Return whether the given window should be closing

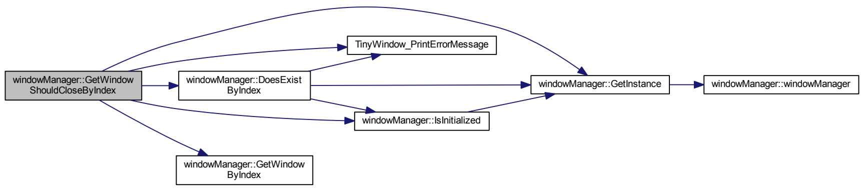
Definition at line 965 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), NOT_INITIALIZED, window↵ Manager::window_t::shouldClose, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

Referenced by main().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                return GetWindowByIndex( windowIndex )->
shouldClose;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }

        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::GetWindow ShouldCloseByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex

Here is the caller graph for this function:

windowManager::GetWindow ShouldCloseByIndex
main

4.2.3.41 static bool windowManager::GetWindowShouldCloseByName (const char ∗ *windowName*) [inline], [static]

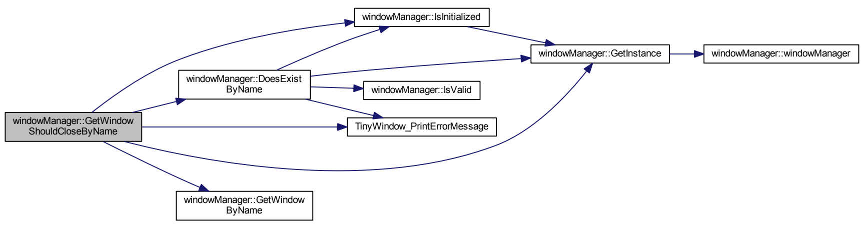
Return whether the given window should be closing

Definition at line 947 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), NOT_INITIALIZED, windowManager::window_t::shouldClose, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                return GetWindowByName( windowName )->shouldClose;
            }
            TinyWindow_PrintErrorMessage(
  tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }

        TinyWindow_PrintErrorMessage(
  tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetWindow ShouldCloseByName
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName

4.2.3.42 static bool windowManager::Initialize (void) [inline], [static]

Initialize the window manager

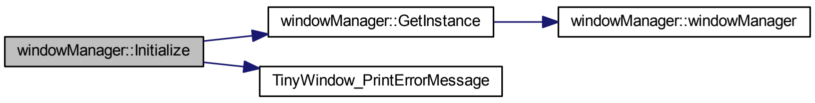
Definition at line 1552 of file TinyWindow.h.

References GetInstance(), instance, isInitialized, screenResolution, TinyWindow_PrintErrorMessage(), and WIN↩ DOWS_CANNOT_INITIALIZE.

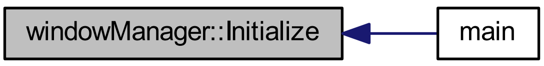
Referenced by main().

```
    {
        GetInstance()->isInitialized = false;
#if defined( _WIN32 ) || defined( _WIN64 )
        CreateTerminal();
        RECT desktop;

        HWND desktopHandle = GetDesktopWindow();

        if (desktopHandle)
        {
            GetWindowRect(desktopHandle, &desktop);

            instance->screenResolution[0] = desktop.right;
            instance->screenResolution[1] = desktop.bottom;
            instance->isInitialized = true;
            return true;
        }

        TinyWindow_PrintErrorMessage(
    tinyWindowError_t::WINDOWS_CANNOT_INITIALIZE);
        return false;
#elif defined(__linux__)
        instance->currentDisplay = XOpenDisplay(0);

        if (!instance->currentDisplay)
        {
            TinyWindow_PrintErrorMessage(
    tinyWindowError_t::LINUX_CANNOT_CONNECT_X_SERVER);
            return false;
        }

        instance->screenResolution[0] = WidthOfScreen(
            XScreenOfDisplay(instance->currentDisplay,
            DefaultScreen(instance->currentDisplay)));

        instance->screenResolution[1] = HeightOfScreen(
            XScreenOfDisplay(instance->currentDisplay,
            DefaultScreen(instance->currentDisplay)));

        instance->isInitialized = true;
        return true;
#endif
    }
```

Here is the call graph for this function:


windowManager::Initialize
windowManager::GetInstance
windowManager::windowManager
TinyWindow_PrintErrorMessage

Here is the caller graph for this function:



4.2.3.43 static bool windowManager::IsInitialized (void) [inline], [static]

Return whether the window manager has been initialized

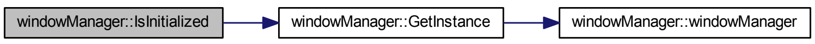
Definition at line 1598 of file TinyWindow.h.

References GetInstance(), and isInitialized.

Referenced by AddWindow(), DisableWindowDecoratorByIndex(), DisableWindowDecoratorByName(), Does←↩ExistByIndex(), DoesExistByName(), EnableWindowDecoratorsByIndex(), EnableWindowDecoratorsByName(), FocusWindowByIndex(), FocusWindowByName(), GetMousePositionInScreen(), GetMousePositionInWindow←↩ByIndex(), GetMousePositionInWindowByName(), GetNumWindows(), GetScreenResolution(), GetWindow←↩IndexByName(), GetWindowIsFullScreenByIndex(), GetWindowIsFullScreenByName(), GetWindowIsInFocus←↩ByIndex(), GetWindowIsInFocusByName(), GetWindowIsMaximizedByIndex(), GetWindowIsMaximizedByName(), GetWindowIsMinimizedByIndex(), GetWindowIsMinimizedByName(), GetWindowNameByIndex(), GetWindow←↩PositionByIndex(), GetWindowPositionByName(), GetWindowResolutionByIndex(), GetWindowResolutionBy←↩Name(), GetWindowShouldCloseByIndex(), GetWindowShouldCloseByName(), MakeWindowCurrentContext←↩ByIndex(), MakeWindowCurrentContextByName(), MaximizeWindowByIndex(), MaximizeWindowByName(), MinimizeWindowByIndex(), MinimizeWindowByName(), PollForEvents(), RemoveWindowByIndex(), Remove←↩WindowByName(), RestoreWindowByIndex(), RestoreWindowByName(), SetFullScreenByIndex(), SetFull←↩ScreenByName(), SetMousePositionInScreen(), SetMousePositionInWindowByIndex(), SetMousePositionIn←↩WindowByName(), SetWindowOnDestroyedByIndex(), SetWindowOnDestroyedByName(), SetWindowOnFocus←↩ByIndex(), SetWindowOnFocusByName(), SetWindowOnKeyEventByIndex(), SetWindowOnKeyEventByName(), SetWindowOnMaximizedByIndex(), SetWindowOnMaximizedByName(), SetWindowOnMinimizedByIndex(), Set←↩WindowOnMinimizedByName(), SetWindowOnMouseButtonEventByIndex(), SetWindowOnMouseButtonEvent←↩ByName(), SetWindowOnMouseMoveByIndex(), SetWindowOnMouseMoveByName(), SetWindowOnMouse←↩WheelEventByIndex(), SetWindowOnMouseWheelEventByName(), SetWindowOnMovedByIndex(), SetWindow←↩OnMovedByName(), SetWindowOnResizeByIndex(), SetWindowOnResizeByName(), SetWindowPositionBy←↩Index(), SetWindowPositionByName(), SetWindowResolutionByIndex(), SetWindowResolutionByName(), Set←↩WindowStyleByIndex(), SetWindowStyleByName(), SetWindowTitleBarByIndex(), SetWindowTitleBarByName(), ShutDown(), WaitForEvents(), WindowGetKeyByIndex(), WindowGetKeyByName(), WindowSwapBuffersByIndex(), and WindowSwapBuffersByName().

```
{
    return GetInstance()->isInitialized;
}
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager

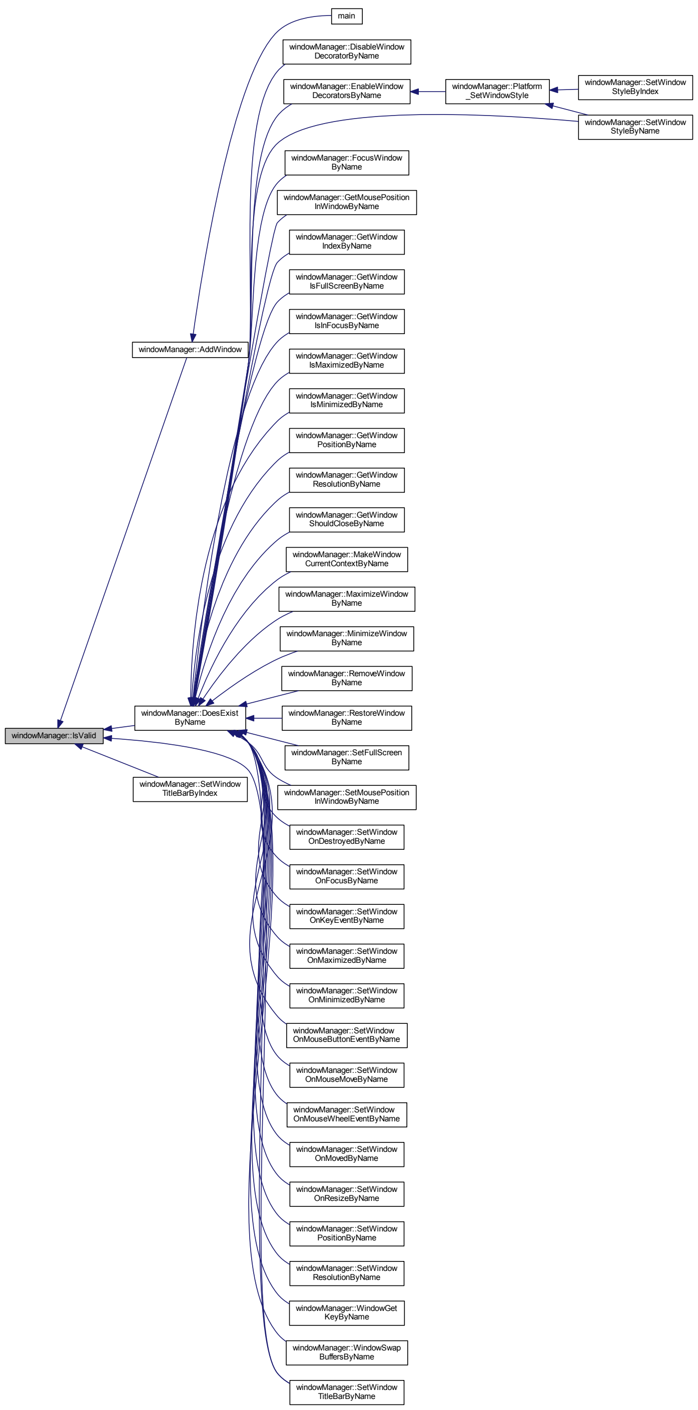
4.2.3.44 static bool windowManager::IsValid (const char ∗ *stringParameter*) [inline], [static], [private]

Definition at line 2421 of file TinyWindow.h.

Referenced by AddWindow(), DoesExistByName(), SetWindowTitleBarByIndex(), and SetWindowTitleBarBy←↩Name().

```
{
    return ( stringParameter != nullptr );
}
```

Here is the caller graph for this function:


main
windowManager::DisableWindow DecoratorByName
windowManager::EnableWindow DecoratorsByName
windowManager::Platform _SetWindowStyle
windowManager::SetWindow StyleByIndex
windowManager::SetWindow StyleByName
windowManager::FocusWindow ByName
windowManager::GetMousePosition InWindowByName
windowManager::GetWindow IndexByName
windowManager::GetWindow IsFullScreenByName
windowManager::GetWindow IsInFocusByName
windowManager::AddWindow
windowManager::GetWindow IsMaximizedByName
windowManager::GetWindow IsMinimizedByName
windowManager::GetWindow PositionByName
windowManager::GetWindow ResolutionByName
windowManager::GetWindow ShouldCloseByName
windowManager::MakeWindow CurrentContextByName
windowManager::MaximizeWindow ByName
windowManager::MinimizeWindow ByName
windowManager::RemoveWindow ByName
windowManager::DoesExist ByName
windowManager::RestoreWindow ByName
windowManager::IsValid
windowManager::SetFullScreen ByName
windowManager::SetWindow TitleBarByIndex
windowManager::SetMousePosition InWindowByName
windowManager::SetWindow OnDestroyedByName
windowManager::SetWindow OnFocusByName
windowManager::SetWindow OnKeyEventByName
windowManager::SetWindow OnMaximizedByName
windowManager::SetWindow OnMinimizedByName
windowManager::SetWindow OnMouseButtonEventByName
windowManager::SetWindow OnMouseMoveByName
windowManager::SetWindow OnMouseWheelEventByName
windowManager::SetWindow OnMovedByName
windowManager::SetWindow OnResizeByName
windowManager::SetWindow PositionByName
windowManager::SetWindow ResolutionByName
windowManager::WindowGet KeyByName
windowManager::WindowSwap BuffersByName
windowManager::SetWindow TitleBarByName

4.2.3.45 static bool windowManager::MakeWindowCurrentContextByIndex (unsigned int *windowIndex*) `[inline]`, `[static]`

Make the given window be the current OpenGL Context to be drawn to

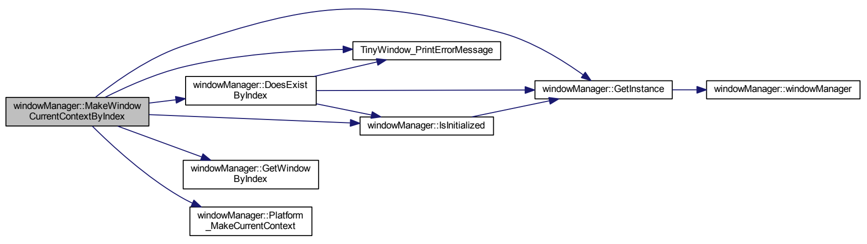
Definition at line 1045 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get↩WindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_MakeCurrentContext(), TinyWindow_PrintError↩Message(), and WINDOW_NOT_FOUND.

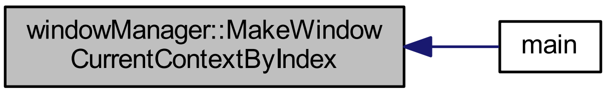
Referenced by main().

```
    {
        if (GetInstance()->IsInitialized())
        {
            if (DoesExistByIndex(windowIndex))
            {
                window_t* window = GetWindowByIndex(windowIndex);
                Platform_MakeCurrentContext(window);

                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::MakeWindow CurrentContextByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _MakeCurrentContext

Here is the caller graph for this function:



4.2.3.46 static bool windowManager::MakeWindowCurrentContextByName (const char ∗ *windowName*) [inline], [static]

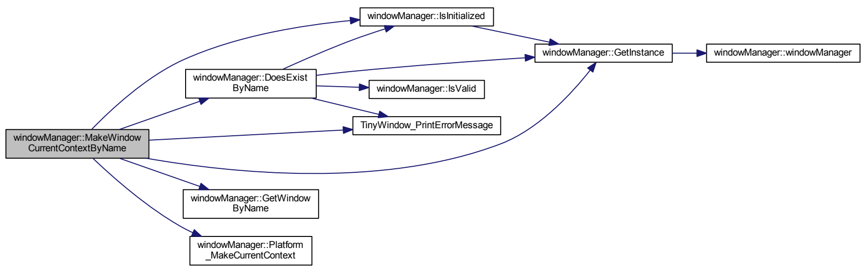
Make the given window be the current OpenGL Context to be drawn to

Definition at line 1025 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get↩WindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_MakeCurrentContext(), TinyWindow_PrintError↩Message(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                window_t* window = GetWindowByName(windowName);
                Platform_MakeCurrentContext(window);

                return true;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::MakeWindow CurrentContextByName
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName
windowManager::Platform _MakeCurrentContext

4.2.3.47 static bool windowManager::MaximizeWindowByIndex (unsigned int *windowIndex,* bool *newState*) [inline], [static]

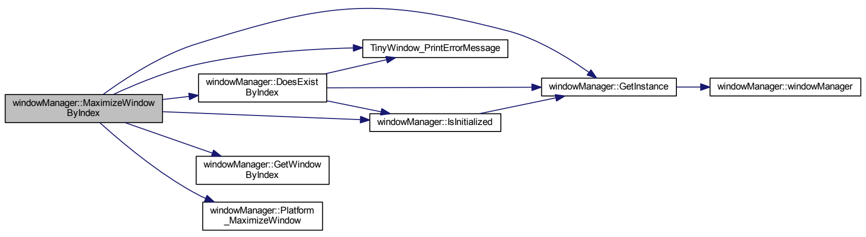
Toggle the maximization state of the current window

Definition at line 1272 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get←WindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_MaximizeWindow(), TinyWindow_PrintError←Message(), and WINDOW_NOT_FOUND.

```
    {
        if (GetInstance()->IsInitialized())
        {
            if (DoesExistByIndex(windowIndex))
            {
                window_t* window = GetWindowByIndex(windowIndex);
                Platform_MaximizeWindow(window, newState);
                return true;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED);
        return false;
    }
```

Here is the call graph for this function:


windowManager::MaximizeWindow ByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _MaximizeWindow

4.2.3.48 static bool windowManager::MaximizeWindowByName (const char ∗ *windowName,* bool *newState*) [inline], [static]

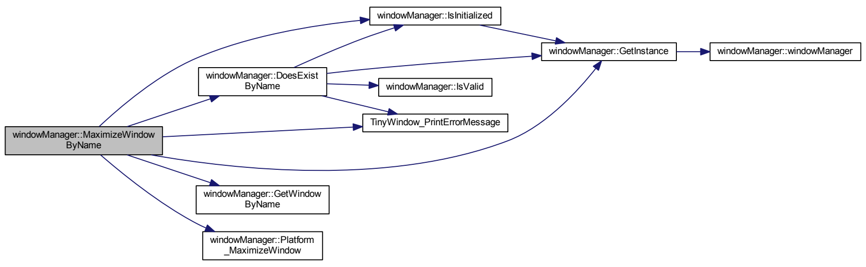
Toggle the maximization state of the current window

Definition at line 1253 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get←WindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_MaximizeWindow(), TinyWindow_PrintError←Message(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                window_t* window = GetWindowByName(windowName);
                Platform_MaximizeWindow(window, newState);
                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::MaximizeWindow ByName
windowManager::GetWindow ByName
windowManager::Platform _MaximizeWindow

4.2.3.49 static bool windowManager::MinimizeWindowByIndex (unsigned int *windowIndex,* bool *newState*) [inline], [static]

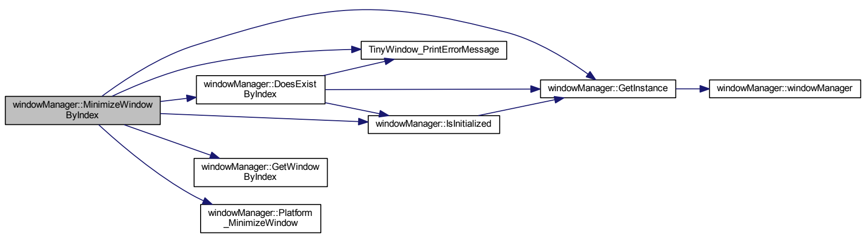
Toggle the minimization state of the window

Definition at line 1197 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get←
WindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_MinimizeWindow(), TinyWindow_PrintError←
Message(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                window_t* window = GetWindowByIndex(windowIndex);
                Platform_MinimizeWindow(window, newState);
                return true;
            }
            TinyWindow_PrintErrorMessage(
  tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
  tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::MinimizeWindow ByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _MinimizeWindow

4.2.3.50 static bool windowManager::MinimizeWindowByName (const char ∗ *windowName,* bool *newState*) [inline], [static]

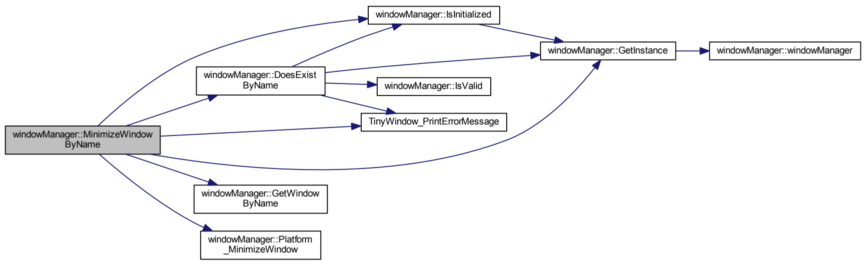
Toggle the minimization state of the given window

Definition at line 1178 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get←
WindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_MinimizeWindow(), TinyWindow_PrintError←
Message(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                window_t* window = GetWindowByName(windowName);
                Platform_MinimizeWindow(window, newState);
                return true;
            }
            TinyWindow_PrintErrorMessage(
 tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
 tinyWindowError_t::NOT_INITIALIZED);
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::MinimizeWindow ByName
windowManager::GetWindow ByName
windowManager::Platform _MinimizeWindow

4.2.3.51 static void windowManager::Platform_DisableWindowDecorators (window_t ∗ *window,* unsigned int *decorators*) [inline], [static], [private]

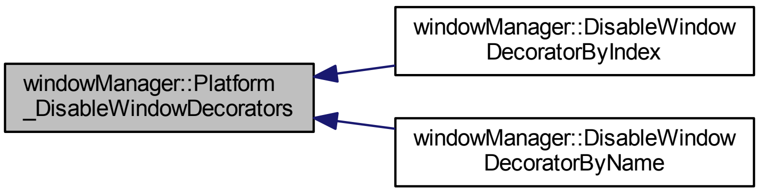
Definition at line 2890 of file TinyWindow.h.

References windowManager::window_t::currentWindowStyle, DECORATOR_BORDER, DECORATOR_CLOSE←BUTTON, DECORATOR_ICON, DECORATOR_MAXIMIZEBUTTON, DECORATOR_MINIMIZEBUTTON, DEC←ORATOR_SIZEABLEBORDER, and DECORATOR_TITLEBAR.

Referenced by DisableWindowDecoratorByIndex(), and DisableWindowDecoratorByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        if (decorators & DECORATOR_BORDER)
        {
            window->currentWindowStyle &= ~WS_BORDER;
        }

        if (decorators & DECORATOR_TITLEBAR)
        {
            window->currentWindowStyle &= ~WS_MAXIMIZEBOX;
        }

        if (decorators & DECORATOR_ICON)
        {
            window->currentWindowStyle &= ~WS_ICONIC;
        }

        if (decorators & DECORATOR_CLOSEBUTTON)
        {
            window->currentWindowStyle &= ~WS_SYSMENU;
        }

        if (decorators & DECORATOR_MINIMIZEBUTTON)
        {
            window->currentWindowStyle &= ~WS_MINIMIZEBOX;
        }

        if (decorators & DECORATOR_MAXIMIZEBUTTON)
        {
            window->currentWindowStyle &= ~WS_MAXIMIZEBOX;
        }

        if (decorators & DECORATOR_SIZEABLEBORDER)
        {
            window->currentWindowStyle &= ~WS_SIZEBOX;
        }

        SetWindowLongPtr(window->windowHandle, GWL_STYLE,
            window->currentWindowStyle | WS_VISIBLE);
#elif defined(__linux__)
        if (decorators & DECORATOR_CLOSEBUTTON)
        {
            //I hate doing this but it is necessary to keep functionality going.
            bool minimizeEnabled = false;
            bool maximizeEnabled = false;

            if (decorators & DECORATOR_MAXIMIZEBUTTON)
            {
                maximizeEnabled = true;
            }

            if (decorators & DECORATOR_MINIMIZEBUTTON)
            {
                minimizeEnabled = true;
            }

            window->currentWindowStyle &= ~LINUX_DECORATOR_CLOSE;

            if (maximizeEnabled)
            {
                window->currentWindowStyle |= LINUX_DECORATOR_MAXIMIZE;
            }

            if (minimizeEnabled)
            {
                window->currentWindowStyle |= LINUX_DECORATOR_MINIMIZE;
            }

            window->decorators = 1;
        }

        if (decorators & DECORATOR_MINIMIZEBUTTON)
        {
            window->currentWindowStyle &= ~LINUX_DECORATOR_MINIMIZE;
            window->decorators = 1;
        }

        if (decorators & DECORATOR_MAXIMIZEBUTTON)
        {
            bool minimizeEnabled = false;

            if (decorators & DECORATOR_MINIMIZEBUTTON)
            {
                minimizeEnabled = true;
            }

            window->currentWindowStyle &= ~LINUX_DECORATOR_MAXIMIZE;

            if (minimizeEnabled)
            {
                window->currentWindowStyle |= LINUX_DECORATOR_MINIMIZE;
            }

            window->decorators = 1;
        }

        if (decorators & DECORATOR_ICON)
        {
            //Linux ( at least cinnamon ) does not have icons in the window. only in the taskb ar icon
        }

        //just need to set it to 1 to enable all decorators that include title bar
        if (decorators & DECORATOR_TITLEBAR)
        {
            window->decorators = LINUX_DECORATOR_BORDER;
        }

        if (decorators & DECORATOR_BORDER)
        {
            window->decorators = 0;
        }

        if (decorators & DECORATOR_SIZEABLEBORDER)
        {
            window->decorators = 0;
        }

        long hints[5] = { LINUX_FUNCTION | LINUX_DECORATOR, window->
      currentWindowStyle, window->decorators, 0, 0 };

        XChangeProperty(instance->currentDisplay, window->windowHandle, window->AtomHints, XA_ATOM,
       32,
            PropModeReplace, (unsigned char*)hints, 5);

        XMapWindow(instance->currentDisplay, window->windowHandle);
#endif
    }
```

Here is the caller graph for this function:


windowManager::Platform_DisableWindowDecorators
windowManager::DisableWindowDecoratorByIndex
windowManager::DisableWindowDecoratorByName

4.2.3.52 static void windowManager::Platform_EnableWindowDecorators (window_t ∗ *window,* unsigned int *decorators*) [inline], [static], [private]

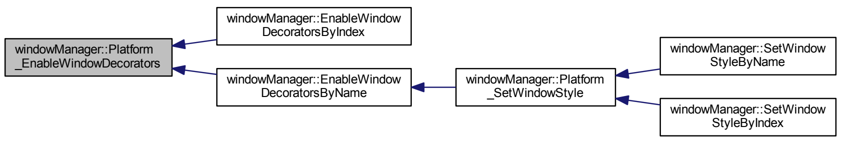
Definition at line 2799 of file TinyWindow.h.

References windowManager::window_t::currentWindowStyle, DECORATOR_BORDER, DECORATOR_CLOSE←BUTTON, DECORATOR_ICON, DECORATOR_MAXIMIZEBUTTON, DECORATOR_MINIMIZEBUTTON, DEC←ORATOR_SIZEABLEBORDER, and DECORATOR_TITLEBAR.

Referenced by EnableWindowDecoratorsByIndex(), and EnableWindowDecoratorsByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        window->currentWindowStyle = WS_VISIBLE | WS_CLIPSIBLINGS;

        if (decorators & DECORATOR_BORDER)
        {
            window->currentWindowStyle |= WS_BORDER;
        }

        if (decorators & DECORATOR_TITLEBAR)
        {
            window->currentWindowStyle |= WS_CAPTION;
        }

        if (decorators & DECORATOR_ICON)
        {
            window->currentWindowStyle |= WS_ICONIC;
        }

        if (decorators & DECORATOR_CLOSEBUTTON)
        {
            window->currentWindowStyle |= WS_SYSMENU;
        }

        if (decorators & DECORATOR_MINIMIZEBUTTON)
        {
            window->currentWindowStyle |= WS_MINIMIZEBOX | WS_SYSMENU;
        }

        if (decorators & DECORATOR_MAXIMIZEBUTTON)
        {
            window->currentWindowStyle |= WS_MAXIMIZEBOX | WS_SYSMENU;
        }

        if (decorators & DECORATOR_SIZEABLEBORDER)
        {
            window->currentWindowStyle |= WS_SIZEBOX;
        }

        SetWindowLongPtr(window->windowHandle, GWL_STYLE,
            window->currentWindowStyle);
#elif defined(__linux__)
        if (decorators & DECORATOR_CLOSEBUTTON)
        {
            window->currentWindowStyle |= LINUX_DECORATOR_CLOSE;
            window->decorators = 1;
        }

        if (decorators & DECORATOR_MINIMIZEBUTTON)
        {
            window->currentWindowStyle |= LINUX_DECORATOR_MINIMIZE;
            window->decorators = 1;
        }

        if (decorators & DECORATOR_MAXIMIZEBUTTON)
        {
            window->currentWindowStyle |= LINUX_DECORATOR_MAXIMIZE;
            window->decorators = 1;
        }

        if (decorators & DECORATOR_ICON)
        {
            //Linux ( at least cinnamon ) does not have icons in the window. only in the task bar icon
        }

        //just need to set it to 1 to enable all decorators that include title bar
        if (decorators & DECORATOR_TITLEBAR)
        {
            window->decorators = 1;
        }

        if (decorators & DECORATOR_BORDER)
        {
            window->decorators = 1;
        }

        if (decorators & DECORATOR_SIZEABLEBORDER)
        {
            window->decorators = 1;
        }

        long hints[5] = { LINUX_FUNCTION | LINUX_DECORATOR, window->
  currentWindowStyle, window->decorators, 0, 0 };

        XChangeProperty(instance->currentDisplay, window->windowHandle, window->AtomHints, XA_ATOM,
   32,
            PropModeReplace, (unsigned char*)hints, 5);

        XMapWindow(instance->currentDisplay, window->windowHandle);
#endif
    }
```

Here is the caller graph for this function:


windowManager::Platform _EnableWindowDecorators
windowManager::EnableWindow DecoratorsByIndex
windowManager::EnableWindow DecoratorsByName
windowManager::Platform _SetWindowStyle
windowManager::SetWindow StyleByName
windowManager::SetWindow StyleByIndex

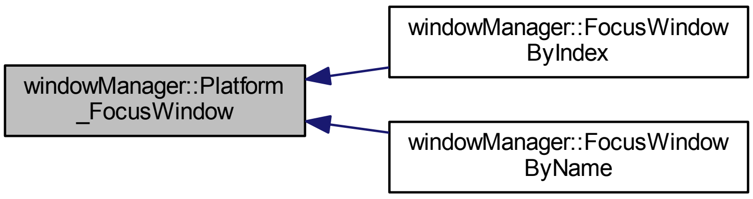
4.2.3.53 static void windowManager::Platform_FocusWindow (window_t ∗ *window*, bool *newState*) [inline], [static], [private]

Definition at line 2686 of file TinyWindow.h.

Referenced by FocusWindowByIndex(), and FocusWindowByName().

```
    {
        if (newState)
        {
#if defined( _WIN32 ) || defined( _WIN64 )
            SetFocus(window->windowHandle);
#elif defined(__linux__)
            XMapWindow(instance->currentDisplay, window->windowHandle);
#endif
        }

        else
        {
#if defined(_WIN32) || defined(_WIN64)
            SetFocus(nullptr);
#elif defined(__linux__)
            XUnmapWindow(instance->currentDisplay, window->windowHandle);
#endif
        }
    }
```

Here is the caller graph for this function:



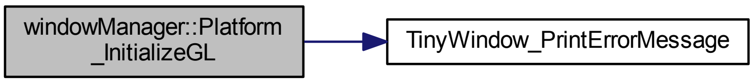
4.2.3.54 static bool windowManager::Platform_InitializeGL (window_t ∗ *window*) [inline], [static], [private]

Definition at line 2455 of file TinyWindow.h.

References windowManager::window_t::contextCreated, INVALID_CONTEXT, and TinyWindow_PrintError←Message().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        window->deviceContextHandle = GetDC(window->windowHandle);
        InitializePixelFormat(window);
        window->glRenderingContextHandle = wglCreateContext(window->deviceContextHandle);
        wglMakeCurrent(window->deviceContextHandle, window->glRenderingContextHandle);

        window->contextCreated = (window->glRenderingContextHandle != nullptr);

        if (window->contextCreated)
        {
            return true;
        }

        TinyWindow_PrintErrorMessage(
    tinyWindowError_t::INVALID_CONTEXT);
        return false;
#elif defined(__linux__)
        if (!window->context)
        {
            window->context = glXCreateContext(
                instance->currentDisplay,
                window->visualInfo,
                0,
                true);

            if (window->context)
            {
                glXMakeCurrent(instance->currentDisplay,
                    window->windowHandle,
                    window->context);

                XWindowAttributes l_Attributes;

                XGetWindowAttributes(instance->currentDisplay,
                    window->windowHandle, &l_Attributes);
                window->position[0] = l_Attributes.x;
                window->position[1] = l_Attributes.y;

                window->contextCreated = true;
                return true;
            }
            return false;
        }

        else
        {
            TinyWindow_PrintErrorMessage(
    tinyWindowError_t::EXISTING_CONTEXT);
            return false;
        }
        return false;
#endif
    }
```

Here is the call graph for this function:



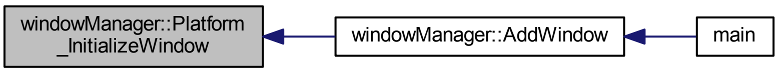
4.2.3.55 static void windowManager::Platform_InitializeWindow (**window_t** ∗ *window*) [inline], [static], [private]

Definition at line 2446 of file TinyWindow.h.

Referenced by AddWindow().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        Windows_InitializeWindow( window );
#elif defined(__linux__)
        Linux_InitializeWindow( window );
#endif
    }
```

Here is the caller graph for this function:



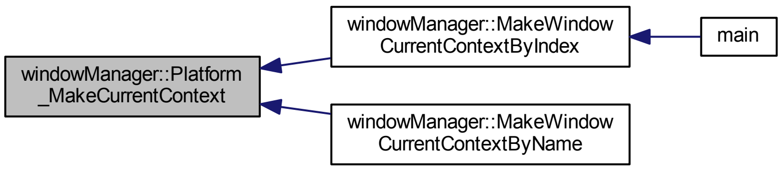
4.2.3.56 static void windowManager::Platform_MakeCurrentContext (window_t ∗ *window*) [inline], [static], [private]

Definition at line 2567 of file TinyWindow.h.

Referenced by MakeWindowCurrentContextByIndex(), and MakeWindowCurrentContextByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        wglMakeCurrent(window->deviceContextHandle,
            window->glRenderingContextHandle);
#elif defined(__linux__)
        glXMakeCurrent(instance->currentDisplay, window->windowHandle,
            window->context);
#endif
    }
```

Here is the caller graph for this function:


windowManager::Platform_MakeCurrentContext
windowManager::MakeWindowCurrentContextByIndex
windowManager::MakeWindowCurrentContextByName
main

4.2.3.57 static void windowManager::Platform_MaximizeWindow (window_t ∗ *window,* bool *newState*) [inline], [static], [private]

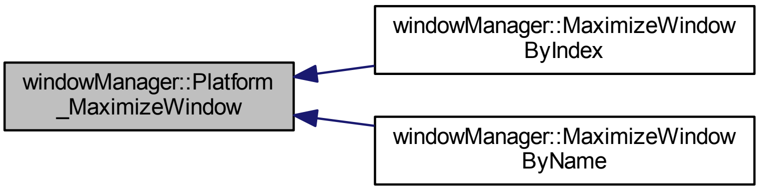
Definition at line 2628 of file TinyWindow.h.

References windowManager::window_t::currentState, MAXIMIZED, and NORMAL.

Referenced by MaximizeWindowByIndex(), and MaximizeWindowByName().

```
    {
        if (newState)
        {
            window->currentState = tinyWindowState_t::MAXIMIZED;
#if defined( _WIN32 ) || defined( _WIN64 )
            ShowWindow(window->windowHandle, SW_MAXIMIZE);
#elif defined(__linux__)
            XEvent currentEvent;
            memset(&currentEvent, 0, sizeof(currentEvent));

            currentEvent.xany.type = ClientMessage;
            currentEvent.xclient.message_type = window->AtomState;
            currentEvent.xclient.format = 32;
            currentEvent.xclient.window = window->windowHandle;
            currentEvent.xclient.data.l[0] = (window->currentState ==
      tinyWindowState_t::MAXIMIZED);
            currentEvent.xclient.data.l[1] = window->AtomMaxVert;
            currentEvent.xclient.data.l[2] = window->AtomMaxHorz;

            XSendEvent(instance->currentDisplay,
                XDefaultRootWindow(instance->currentDisplay),
                0, SubstructureNotifyMask, &currentEvent);
#endif
        }

        else
        {
            window->currentState = tinyWindowState_t::NORMAL;
#if defined( _WIN32 ) || defined( _WIN64 )
            ShowWindow(window->windowHandle, SW_RESTORE);
#elif defined(__linux__)
            XEvent currentEvent;
            memset(&currentEvent, 0, sizeof(currentEvent));

            currentEvent.xany.type = ClientMessage;
            currentEvent.xclient.message_type = window->AtomState;
            currentEvent.xclient.format = 32;
            currentEvent.xclient.window = window->windowHandle;
            currentEvent.xclient.data.l[0] = (window->currentState ==
      tinyWindowState_t::MAXIMIZED);
            currentEvent.xclient.data.l[1] = window->AtomMaxVert;
            currentEvent.xclient.data.l[2] = window->AtomMaxHorz;

            XSendEvent(instance->currentDisplay,
                XDefaultRootWindow(instance->currentDisplay),
                0, SubstructureNotifyMask, &currentEvent);
#endif
        }
    }
```

Here is the caller graph for this function:


windowManager::Platform_MaximizeWindow
windowManager::MaximizeWindowByIndex
windowManager::MaximizeWindowByName

4.2.3.58 static void windowManager::Platform_MinimizeWindow (**window_t** ∗ *window,* bool *newState*) [inline], [static], [private]

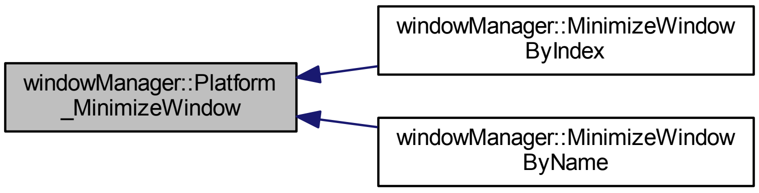
Definition at line 2603 of file TinyWindow.h.

References windowManager::window_t::currentState, MINIMIZED, and NORMAL.

Referenced by MinimizeWindowByIndex(), and MinimizeWindowByName().

```
    {
        if (newState)
        {
            window->currentState = tinyWindowState_t::MINIMIZED;

#if defined( _WIN32 ) || defined( _WIN64 )
            ShowWindow(window->windowHandle, SW_MINIMIZE);
#elif defined(__linux__)
            XIconifyWindow(instance->currentDisplay,
                window->windowHandle, 0);
#endif
        }

        else
        {
            window->currentState = tinyWindowState_t::NORMAL;
#if defined( _WIN32 ) || defined( _WIN64 )
            ShowWindow(window->windowHandle, SW_RESTORE);
#elif defined(__linux__)
            XMapWindow(instance->currentDisplay, window->windowHandle);
#endif
        }
    }
```

Here is the caller graph for this function:



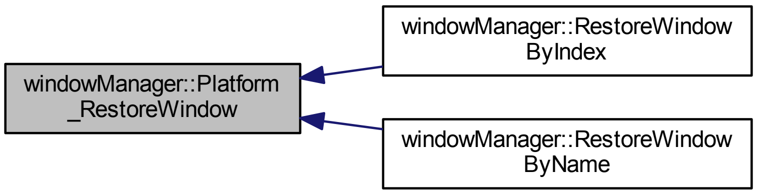
4.2.3.59 static void windowManager::Platform_RestoreWindow (window_t ∗ *window*) [inline], [static], [private]

Definition at line 2707 of file TinyWindow.h.

Referenced by RestoreWindowByIndex(), and RestoreWindowByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        ShowWindow(window->windowHandle, SW_RESTORE);
#elif defined(__linux__)
        XMapWindow(instance->currentDisplay, window->windowHandle);
#endif
    }
```

Here is the caller graph for this function:


windowManager::Platform_RestoreWindow
windowManager::RestoreWindowByIndex
windowManager::RestoreWindowByName

4.2.3.60 static void windowManager::Platform_SetFullScreen (window_t ∗ *window*) [inline], [static], [private]

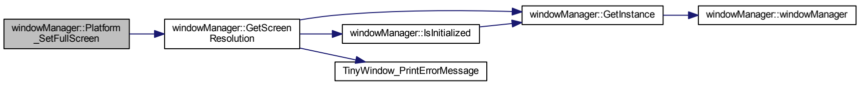
Definition at line 2578 of file TinyWindow.h.

References GetScreenResolution().

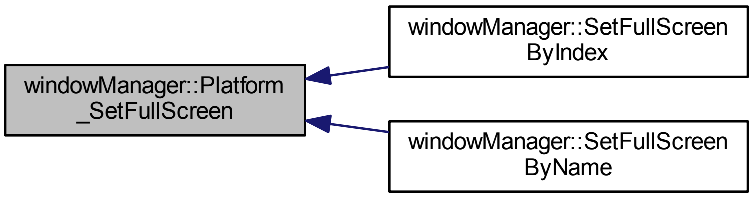
Referenced by SetFullScreenByIndex(), and SetFullScreenByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        SetWindowLongPtr(window->windowHandle, GWL_STYLE,
            WS_SYSMENU | WS_POPUP | WS_CLIPCHILDREN | WS_CLIPSIBLINGS | WS_VISIBLE);

        MoveWindow(window->windowHandle, 0, 0, windowManager::GetScreenResolution()[0],
            windowManager::GetScreenResolution()[1], true);
#elif defined(__linux__)
        XEvent currentEvent;
        memset(&currentEvent, 0, sizeof(currentEvent));

        currentEvent.xany.type = ClientMessage;
        currentEvent.xclient.message_type = window->AtomState;
        currentEvent.xclient.format = 32;
        currentEvent.xclient.window = window->windowHandle;
        currentEvent.xclient.data.l[0] = window->currentState == tinyWindowState_t::FULLSCREEN;
        currentEvent.xclient.data.l[1] = window->AtomFullScreen;

        XSendEvent(instance->currentDisplay,
            XDefaultRootWindow(instance->currentDisplay),
            0, SubstructureNotifyMask, &currentEvent);
#endif
    }
```

Here is the call graph for this function:


windowManager::Platform_SetFullScreen
windowManager::GetScreenResolution
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage

Here is the caller graph for this function:



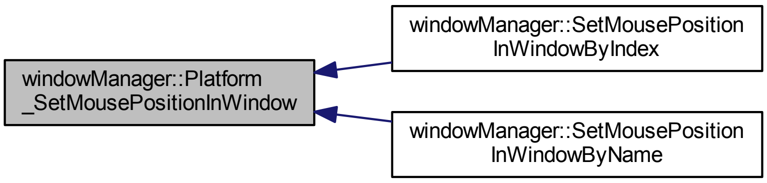
4.2.3.61 static void windowManager::Platform_SetMousePositionInWindow (window_t * *window,* unsigned int *x,* unsigned int *y*) [inline], [static], [private]

Definition at line 2540 of file TinyWindow.h.

Referenced by SetMousePositionInWindowByIndex(), and SetMousePositionInWindowByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        POINT mousePoint;
        mousePoint.x = x;
        mousePoint.y = y;
        ScreenToClient(window->windowHandle, &mousePoint);
        SetCursorPos(mousePoint.x, mousePoint.y);
#elif defined(__linux__)
        XWarpPointer(
            windowManager::GetDisplay(),
            window->windowHandle, window->windowHandle,
            window->position[0], window->position[1],
            window->resolution[0], window->resolution[1],
            x, y);
#endif
    }
```

Here is the caller graph for this function:


windowManager::Platform
_SetMousePositionInWindow
windowManager::SetMousePosition
InWindowByIndex
windowManager::SetMousePosition
InWindowByName

4.2.3.62 static void windowManager::Platform_SetWindowPosition (window_t * *window,* unsigned int *x,* unsigned int *y*) [inline], [static], [private]

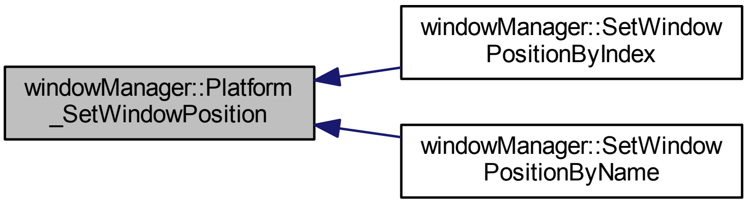
Definition at line 2522 of file TinyWindow.h.

References windowManager::window_t::resolution.

Referenced by SetWindowPositionByIndex(), and SetWindowPositionByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        SetWindowPos(window->windowHandle, HWND_TOP, x, y,
            window->resolution[0], window->resolution[1],
            SWP_SHOWWINDOW | SWP_NOSIZE);
#elif defined(__linux__)
        XWindowChanges windowChanges;

        windowChanges.x = x;
        windowChanges.y = y;

        XConfigureWindow(
            instance->currentDisplay,
            window->windowHandle, CWX | CWY, &windowChanges);
#endif
    }
```

Here is the caller graph for this function:


windowManager::Platform_SetWindowPosition
windowManager::SetWindowPositionByIndex
windowManager::SetWindowPositionByName

4.2.3.63 static void windowManager::Platform_SetWindowResolution (window_t * *window*) [inline], [static], [private]

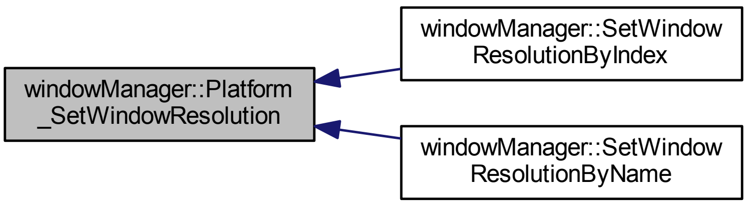
Definition at line 2509 of file TinyWindow.h.

References windowManager::window_t::position, and windowManager::window_t::resolution.

Referenced by SetWindowResolutionByIndex(), and SetWindowResolutionByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        SetWindowPos(window->windowHandle, HWND_TOP,
            window->position[0], window->position[1],
            window->resolution[0], window->resolution[1],
            SWP_SHOWWINDOW | SWP_NOMOVE);
#elif defined(__linux__)
        XResizeWindow(instance->currentDisplay,
            window->windowHandle, window->resolution[0], window->resolution[1]);
#endif
    }
```

Here is the caller graph for this function:


windowManager::Platform _SetWindowResolution
windowManager::SetWindow ResolutionByIndex
windowManager::SetWindow ResolutionByName

4.2.3.64 static void windowManager::Platform_SetWindowStyle (**window_t** ∗ *window*, **tinyWindowStyle_t** *windowStyle*) [inline], [static], [private]

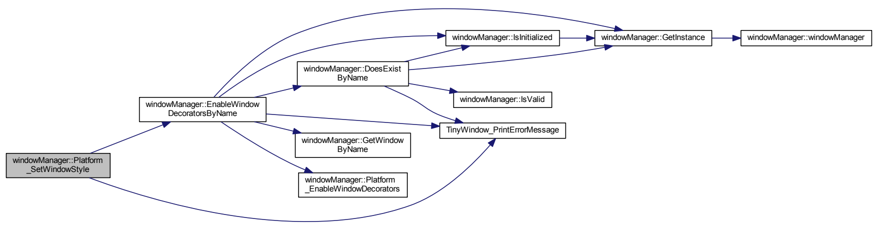
Definition at line 2716 of file TinyWindow.h.

References BARE, DECORATOR_BORDER, DECORATOR_CLOSEBUTTON, DECORATOR_MAXIMIZEBUT↩ TON, DECORATOR_MINIMIZEBUTTON, DECORATOR_TITLEBAR, DEFAULT, EnableWindowDecoratorsBy↩ Name(), INVALID_WINDOWSTYLE, windowManager::window_t::name, POPUP, and TinyWindow_PrintError↩ Message().

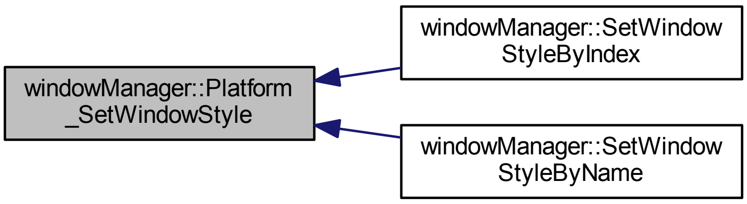
Referenced by SetWindowStyleByIndex(), and SetWindowStyleByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        switch (windowStyle)
        {
        case tinyWindowStyle_t::DEFAULT:
        {
            EnableWindowDecoratorsByName(window->name,
 DECORATOR_TITLEBAR | DECORATOR_BORDER |
                DECORATOR_CLOSEBUTTON |
 DECORATOR_MINIMIZEBUTTON | DECORATOR_MAXIMIZEBUTTON);
            break;
        }

        case tinyWindowStyle_t::POPUP:
        {
            EnableWindowDecoratorsByName(window->name, 0);
            break;
        }

        case tinyWindowStyle_t::BARE:
        {
            EnableWindowDecoratorsByName(window->name,
 DECORATOR_TITLEBAR | DECORATOR_BORDER);
            break;
        }

        default:
        {
            TinyWindow_PrintErrorMessage(
 tinyWindowError_t::INVALID_WINDOWSTYLE);
            break;
        }
        }

#elif defined(__linux__)
        switch (windowStyle)
        {
            case tinyWindowStyle_t::DEFAULT:
        {
            window->decorators = (1L << 2);
            window->currentWindowStyle = LINUX_DECORATOR_MOVE | LINUX_DECORATOR_CLOSE |
                LINUX_DECORATOR_MAXIMIZE | LINUX_DECORATOR_MINIMIZE;
            long Hints[5] = { LINUX_FUNCTION | LINUX_DECORATOR, window->
currentWindowStyle, window->decorators, 0, 0 };

            XChangeProperty(instance->currentDisplay, window->windowHandle, window->AtomHints,
XA_ATOM, 32, PropModeReplace,
                (unsigned char*)Hints, 5);

            XMapWindow(instance->currentDisplay, window->windowHandle);
            break;
        }

            case tinyWindowStyle_t::BARE:
        {
            window->decorators = (1L << 2);
            window->currentWindowStyle = (1L << 2);
            long Hints[5] = { LINUX_FUNCTION | LINUX_DECORATOR, window->
currentWindowStyle, window->decorators, 0, 0 };

            XChangeProperty(instance->currentDisplay, window->windowHandle, window->AtomHints,
XA_ATOM, 32, PropModeReplace,
                (unsigned char*)Hints, 5);

            XMapWindow(instance->currentDisplay, window->windowHandle);
            break;
        }

            case tinyWindowStyle_t::POPUP:
        {
            window->decorators = 0;
            window->currentWindowStyle = (1L << 2);
            long Hints[5] = { LINUX_FUNCTION | LINUX_DECORATOR, window->
currentWindowStyle, window->decorators, 0, 0 };

            XChangeProperty(instance->currentDisplay, window->windowHandle, window->AtomHints,
XA_ATOM, 32, PropModeReplace,
                (unsigned char*)Hints, 5);

            XMapWindow(instance->currentDisplay, window->windowHandle);
            break;
        }

        default:
        {
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_WINDOWSTYLE);
            break;
        }
        }
#endif
    }
```

Here is the call graph for this function:


windowManager::Platform _SetWindowStyle
windowManager::EnableWindow DecoratorsByName
windowManager::DoesExist ByName
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName
windowManager::Platform _EnableWindowDecorators

Here is the caller graph for this function:


windowManager::Platform_SetWindowStyle
windowManager::SetWindowStyleByIndex
windowManager::SetWindowStyleByName

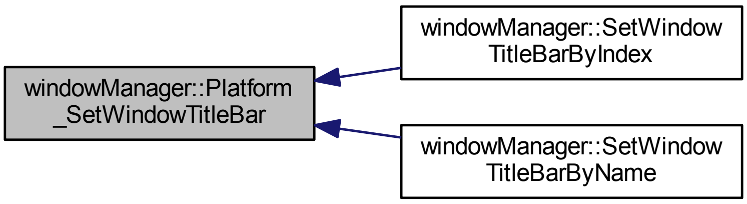
4.2.3.65 static void windowManager::Platform_SetWindowTitleBar (window_t ∗ *window,* const char ∗ *newTitle*) [inline], [static], [private]

Definition at line 2677 of file TinyWindow.h.

Referenced by SetWindowTitleBarByIndex(), and SetWindowTitleBarByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        SetWindowText(window->windowHandle, newTitle);
#elif defined(__linux__)
        XStoreName(instance->currentDisplay, window->windowHandle, newTitle);
#endif
    }
```

Here is the caller graph for this function:


windowManager::Platform_SetWindowTitleBar
windowManager::SetWindowTitleBarByIndex
windowManager::SetWindowTitleBarByName

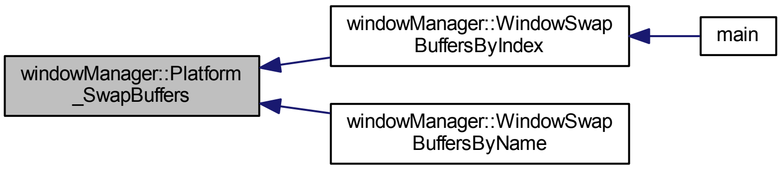
4.2.3.66 static void windowManager::Platform_SwapBuffers (window_t ∗ *window*) [inline], [static], [private]

Definition at line 2558 of file TinyWindow.h.

Referenced by WindowSwapBuffersByIndex(), and WindowSwapBuffersByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        SwapBuffers(window->deviceContextHandle);
#elif defined(__linux__)
        glXSwapBuffers(instance->currentDisplay, window->windowHandle);
#endif
    }
```

Here is the caller graph for this function:


windowManager::Platform _SwapBuffers
windowManager::WindowSwap BuffersByIndex
main
windowManager::WindowSwap BuffersByName

4.2.3.67 static void windowManager::PollForEvents (void) [inline], [static]

Ask the window manager to poll for events

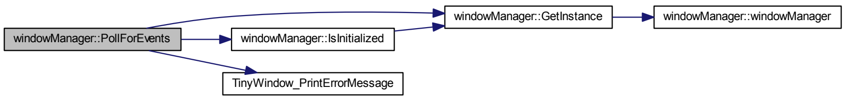
Definition at line 1606 of file TinyWindow.h.

References GetInstance(), instance, IsInitialized(), NOT_INITIALIZED, and TinyWindow_PrintErrorMessage().

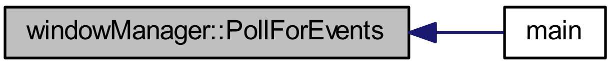
Referenced by main().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
#if defined( _WIN32 ) || defined( _WIN64 )
            //only process events if there are any to process
            if (PeekMessage(&instance->message, 0, 0, 0, PM_REMOVE))
            {
                TranslateMessage(&instance->message);
                DispatchMessage(&instance->message);
            }
#elif defined(__linux__)
            //if there are any events to process
            if (XEventsQueued(instance->currentDisplay, QueuedAfterReading))
            {
                XNextEvent(instance->currentDisplay, &instance->currentEvent);

                XEvent currentEvent = instance->currentEvent;

                Linux_ProcessEvents(currentEvent);
            }
#endif
        }

        else
        {
            TinyWindow_PrintErrorMessage(
  tinyWindowError_t::NOT_INITIALIZED );
        }
    }
```

Here is the call graph for this function:


windowManager::PollForEvents
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage

Here is the caller graph for this function:



4.2.3.68 static bool windowManager::RemoveWindowByIndex (unsigned int *windowIndex*) `[inline], [static]`

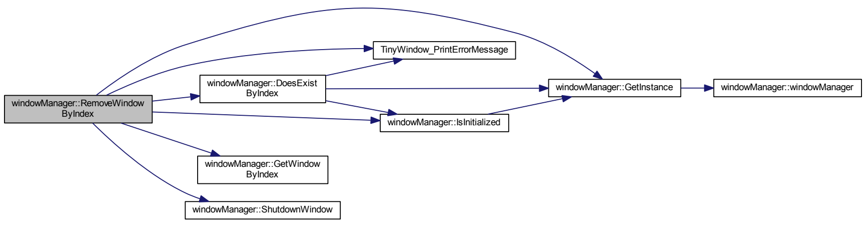
Remove window from the manager by index

Definition at line 1685 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), NOT_INITIALIZED, ShutdownWindow(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                ShutdownWindow( GetWindowByIndex( windowIndex ) );
                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::RemoveWindow ByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::ShutdownWindow

4.2.3.69 static bool windowManager::RemoveWindowByName (const char ∗ *windowName*) [inline], [static]

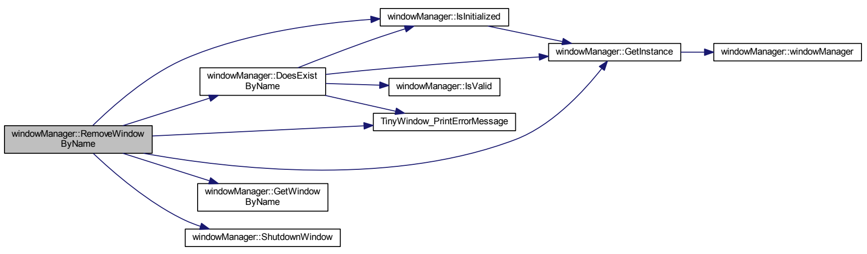
Remove window from the manager by name

Definition at line 1667 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), NOT_INITIALIZED, ShutdownWindow(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByName( windowName ) )
           {
               ShutdownWindow( GetWindowByName( windowName ) );
               return true;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::RemoveWindow ByName
windowManager::GetWindow ByName
windowManager::ShutdownWindow

4.2.3.70 static bool windowManager::RestoreWindowByIndex (unsigned int *windowIndex*) [inline], [static]

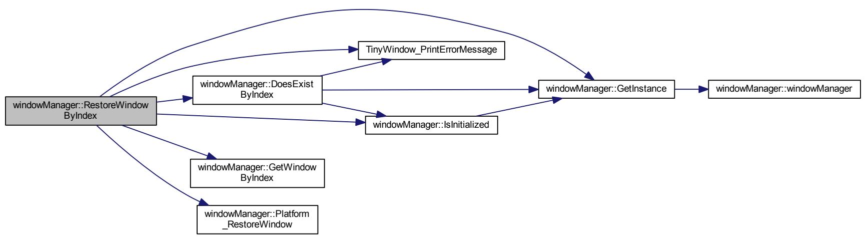
Restore the window by index

Definition at line 1532 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get←WindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_RestoreWindow(), TinyWindow_PrintError←Message(), and WINDOW_NOT_FOUND.

```
   {
       if (GetInstance()->IsInitialized())
       {
           if (DoesExistByIndex(windowIndex))
           {
               window_t* window = GetWindowByIndex(windowIndex);
               Platform_RestoreWindow(window);
               return true;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED);
       return false;
   }
```

Here is the call graph for this function:


windowManager::RestoreWindow ByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _RestoreWindow

4.2.3.71 static bool windowManager::RestoreWindowByName (const char ∗ *windowName*) [inline], [static]

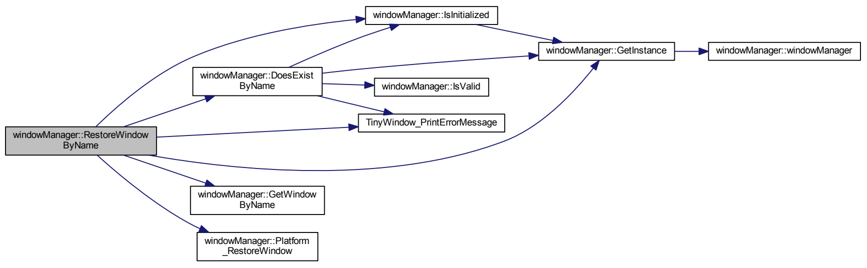
Restore the window by name

Definition at line 1513 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_RestoreWindow(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                window_t* window = GetWindowByName(windowName);
                Platform_RestoreWindow(window);
                return true;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::RestoreWindow ByName
windowManager::GetWindow ByName
windowManager::Platform _RestoreWindow

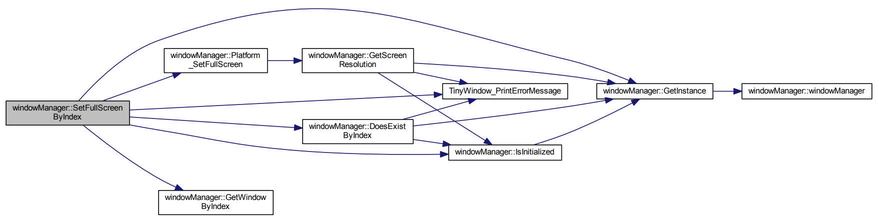
4.2.3.72 static bool windowManager::SetFullScreenByIndex (unsigned int *windowIndex,* bool *newState*) [inline], [static]

Definition at line 1123 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, windowManager::window_t::currentState, Does↩ExistByIndex(), FULLSCREEN, GetInstance(), GetWindowByIndex(), IsInitialized(), NORMAL, NOT_INITIALIZED, Platform_SetFullScreen(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByIndex( windowIndex ) )
           {
               window_t* window = GetWindowByIndex(windowIndex);
               window->currentState = (newState == true) ?
tinyWindowState_t::FULLSCREEN :
tinyWindowState_t::NORMAL;

               Platform_SetFullScreen(window);
               return true;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::SetFullScreen ByIndex
windowManager::Platform _SetFullScreen
windowManager::GetScreen Resolution
TinyWindow_PrintErrorMessage
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByIndex
windowManager::IsInitialized
windowManager::GetWindow ByIndex

4.2.3.73 static bool windowManager::SetFullScreenByName (const char ∗ *windowName,* bool *newState*) [inline], [static]

Toggle the given window's full screen mode

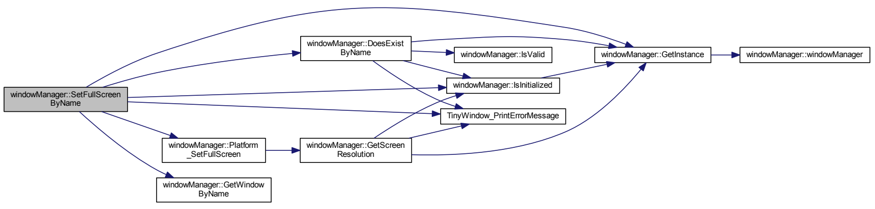
Definition at line 1101 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, windowManager::window_t::currentState, Does↩ExistByName(), FULLSCREEN, GetInstance(), GetWindowByName(), IsInitialized(), NORMAL, NOT_INITIALIZED, Platform_SetFullScreen(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByName( windowName ) )
           {
               window_t* window = GetWindowByName(windowName);

               window->currentState = (newState == true) ?
```

```
  tinyWindowState_t::FULLSCREEN :
  tinyWindowState_t::NORMAL;

                Platform_SetFullScreen(window);
                return true;
            }
            TinyWindow_PrintErrorMessage(
  tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
  tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetFullScreen ByName
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage
windowManager::Platform _SetFullScreen
windowManager::GetScreen Resolution
windowManager::GetWindow ByName

4.2.3.74 static bool windowManager::SetMousePositionInScreen (unsigned int *x*, unsigned int *y*) [inline], [static]

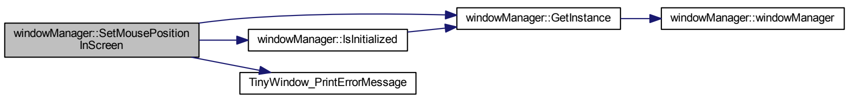
Set the position of the mouse cursor relative to screen co-ordinates

Definition at line 467 of file TinyWindow.h.

References GetInstance(), instance, IsInitialized(), NOT_INITIALIZED, screenMousePosition, and TinyWindow_↩ PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            instance->screenMousePosition[0] = x;
            instance->screenMousePosition[1] = y;

#if defined( _WIN32 ) || defined( _WIN64 )
            SetCursorPos(x, y);
#elif defined(__linux__)
            XWarpPointer(instance->currentDisplay, None,
                XDefaultRootWindow(instance->currentDisplay), 0, 0,
                GetScreenResolution()[0],
                GetScreenResolution()[1],
                x, y);
#endif
            return true;
        }
        TinyWindow_PrintErrorMessage(
  tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetMousePosition InScreen
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage

4.2.3.75 static bool windowManager::SetMousePositionInWindowByIndex (unsigned int *windowIndex,* unsigned int *x,* unsigned int *y*) [inline], [static]

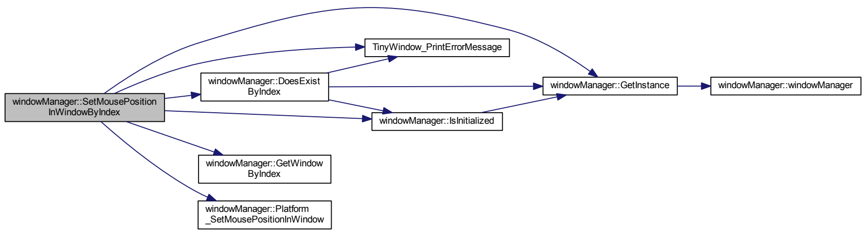
Set the mouse Position of the given window's co-ordinates

Definition at line 888 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get←WindowByIndex(), IsInitialized(), windowManager::window_t::mousePosition, NOT_INITIALIZED, Platform_Set←MousePositionInWindow(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                GetWindowByIndex( windowIndex )->mousePosition[ 0 ] = x;
                GetWindowByIndex( windowIndex )->mousePosition[ 1 ] = y;
                window_t* window = GetWindowByIndex(windowIndex);

                Platform_SetMousePositionInWindow(window, x, y);
                return true;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetMousePosition InWindowByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _SetMousePositionInWindow

4.2.3.76 static bool windowManager::SetMousePositionInWindowByName (const char ∗ *windowName,* unsigned int *x,* unsigned int *y*) [inline], [static]

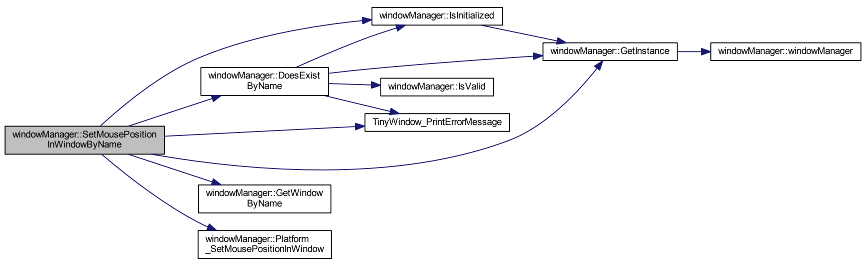
Set the mouse Position of the given window's co-ordinates

Definition at line 866 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get←WindowByName(), IsInitialized(), windowManager::window_t::mousePosition, NOT_INITIALIZED, Platform_← SetMousePositionInWindow(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                GetWindowByName( windowName )->mousePosition[ 0 ] = x;
                GetWindowByName( windowName )->mousePosition[ 1 ] = y;
                window_t* window = GetWindowByName(windowName);
                Platform_SetMousePositionInWindow(window, x, y);
                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }

        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::SetMousePosition InWindowByName
windowManager::GetWindow ByName
windowManager::Platform _SetMousePositionInWindow

4.2.3.77 static bool windowManager::SetWindowIconByIndex (void) [inline], [static]

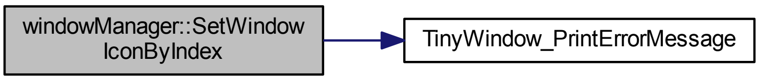
Set the window icon by index (currently not functional)

Definition at line 1408 of file TinyWindow.h.

References FUNCTION_NOT_IMPLEMENTED, and TinyWindow_PrintErrorMessage().

```
    {
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::FUNCTION_NOT_IMPLEMENTED);
        return false;
        /*if ( GetInstance()->IsInitialized() )
        {
            if (IsValid(icon))
            {
                if (DoesExistByIndex(windowIndex))
                {
#if defined( _WIN32 ) || defined( _WIN64 )
                    Windows_SetWindowIcon(GetWindowByIndex(windowIndex), icon, width, height);
#elif defined(__linux__)
                    Linux_SetWindowIcon(GetWindowByIndex(windowIndex), icon, width, height);
#endif
                    return true;
                }
                TinyWindow_PrintErrorMessage(tinyWindowError_t::TINYWINDOW_ERROR_WINDOW_NOT_FOUND);
                return false;
            }
            TinyWindow_PrintErrorMessage(tinyWindowError_t::TINYWINDOW_ERROR_INVALID_ICON_PATH);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::TINYWINDOW_ERROR_NOT_INITIALIZED );
        return false;*/
    }
```

Here is the call graph for this function:


windowManager::SetWindow IconByIndex
TinyWindow_PrintErrorMessage

4.2.3.78 static bool windowManager::SetWindowIconByName (void) [inline], [static]

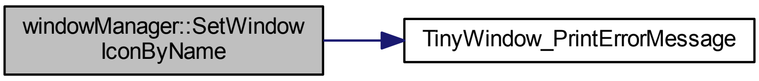
Set the window icon by name (currently not functional)

Definition at line 1377 of file TinyWindow.h.

References FUNCTION_NOT_IMPLEMENTED, and TinyWindow_PrintErrorMessage().

```
    {
        TinyWindow_PrintErrorMessage(
    tinyWindowError_t::FUNCTION_NOT_IMPLEMENTED);
        return false;
    /*  if ( GetInstance()->IsInitialized() )
        {
            if (IsValid(icon))
            {
                if (DoesExistByName(windowName))
                {
#if defined( _WIN32 ) || defined( _WIN64 )
                    //Windows_SetWindowIcon(GetWindowByName(windowName), icon, width, height);
#elif defined(__linux__)
                    //Linux_SetWindowIcon();//GetWindowByName(windowName), icon, width, height);
#endif
                    return true;
                }
                TinyWindow_PrintErrorMessage(tinyWindowError_t::TINYWINDOW_ERROR_WINDOW_NOT_FOUND);
                return false;
            }

            TinyWindow_PrintErrorMessage(tinyWindowError_t::TINYWINDOW_ERROR_INVALID_ICON_PATH);
            return false;
        }

        TinyWindow_PrintErrorMessage( tinyWindowError_t::TINYWINDOW_ERROR_NOT_INITIALIZED );
        return false;*/
    }
```

Here is the call graph for this function:


windowManager::SetWindow IconByName
TinyWindow_PrintErrorMessage

4.2.3.79 static bool windowManager::SetWindowOnDestroyedByIndex (unsigned int *windowIndex,* std::function< void(void)> *onDestroyed*) [inline], [static]

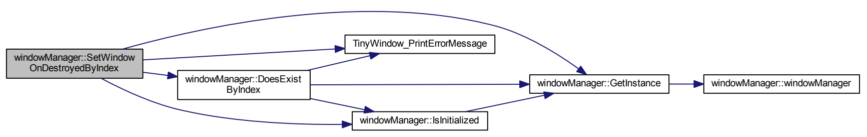
Set the window on destroyed event callback by index

Definition at line 1986 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↵
Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByIndex( windowIndex ) )
           {
               if (onDestroyed != nullptr)
               {
                   GetWindowByIndex(windowIndex)->
destroyedEvent = onDestroyed;
                   return true;
               }
               TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_CALLBACK);
               return false;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::SetWindow OnDestroyedByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.80 static bool windowManager::SetWindowOnDestroyedByName (const char ∗ *windowName,* std::function< void(void)> *onDestroyed*) [inline], [static]

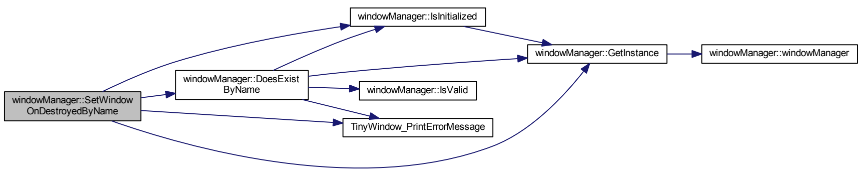
Set the window on destroyed event callback by name

Definition at line 1963 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↵
Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByName( windowName ) )
           {
               if (onDestroyed != nullptr)
               {
                   GetWindowByName(windowName)->destroyedEvent = onDestroyed;
                   return true;
               }
               TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_CALLBACK);
               return false;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::SetWindow OnDestroyedByName
windowManager::DoesExist ByName
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::IsValid
TinyWindow_PrintErrorMessage

4.2.3.81 static bool windowManager::SetWindowOnFocusByIndex (unsigned int *windowIndex,* std::function< void(bool)> *onFocus*) [inline], [static]

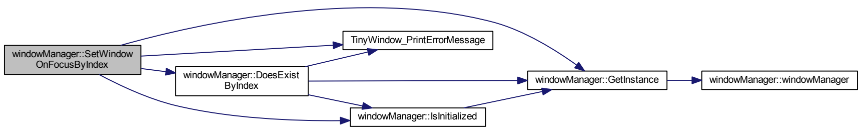
Set the window on focus event callback by index

Definition at line 2127 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByIndex( windowIndex ) )
           {
               if (onFocus != nullptr)
               {
                   GetWindowByIndex(windowIndex)->focusEvent = onFocus;
                   return true;
               }
               TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_CALLBACK);
               return false;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::SetWindow OnFocusByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.82 static bool windowManager::SetWindowOnFocusByName (const char ∗ *windowName,* std::function< void(bool)> *onFocus*) [inline], [static]

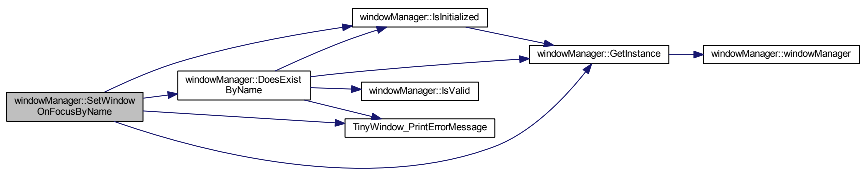
Set the window on focus event callback by name

Definition at line 2104 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny←Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByName( windowName ) )
           {
               if (onFocus != nullptr)
               {
                   GetWindowByName(windowName)->focusEvent = onFocus;
                   return true;
               }
               TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_CALLBACK);
               return false;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::SetWindow OnFocusByName
windowManager::DoesExist ByName
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::IsValid
TinyWindow_PrintErrorMessage

4.2.3.83 static bool windowManager::SetWindowOnKeyEventByIndex (unsigned int *windowIndex,* std::function< void(unsigned int, tinyWindowKeyState_t)> *onKey*) [inline], [static]

Set the window on key event callback by index

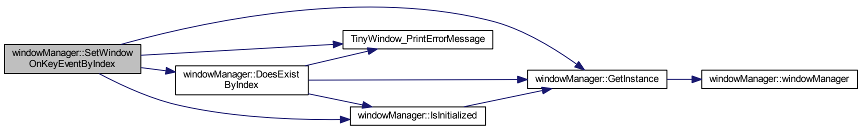
Definition at line 1845 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny←Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByIndex( windowIndex ) )
           {
               if (onKey != nullptr)
               {
                   GetWindowByIndex(windowIndex)->keyEvent = onKey;
                   return true;
```

```
                }
                TinyWindow_PrintErrorMessage( tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetWindow OnKeyEventByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.84 static bool windowManager::SetWindowOnKeyEventByName (const char ∗ *windowName,* std::function< void(unsigned int, tinyWindowKeyState_t)> *onKey*) [inline], [static]

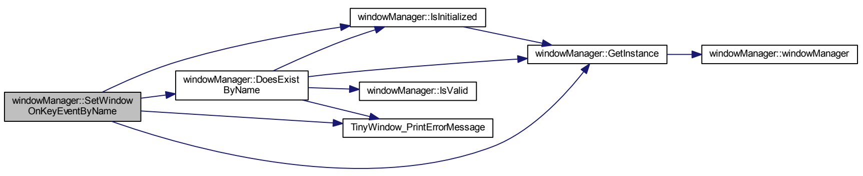
Set the window on key event callback by name

Definition at line 1821 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny←Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                if (onKey != nullptr)
                {
                    GetWindowByName(windowName)->keyEvent = onKey;
                    return true;
                }
                TinyWindow_PrintErrorMessage( tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }

            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::SetWindow OnKeyEventByName
TinyWindow_PrintErrorMessage

4.2.3.85 static bool windowManager::SetWindowOnMaximizedByIndex (unsigned int *windowIndex,* std::function< void(void)> *onMaximized*) [inline], [static]

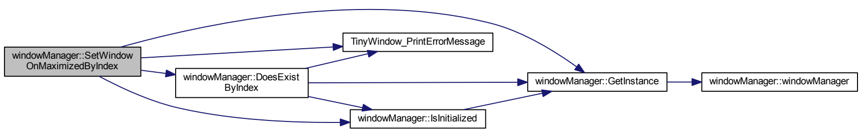
Set the window on maximized event callback by index

Definition at line 2033 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny←Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                if (onMaximized != nullptr)
                {
                    GetWindowByIndex(windowIndex)->
    maximizedEvent = onMaximized;
                    return true;
                }
                TinyWindow_PrintErrorMessage(
    tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage(
    tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
    tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetWindow OnMaximizedByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.86 static bool windowManager::SetWindowOnMaximizedByName (const char ∗ *windowName,* std::function< void(void)> *onMaximized*) [inline], [static]

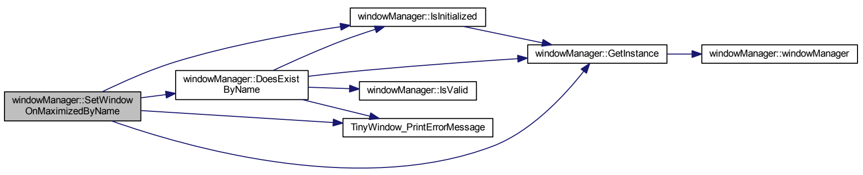
Set the window on maximized event callback by name

Definition at line 2010 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny←Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                if (onMaximized != nullptr)
                {
                    GetWindowByName(windowName)->maximizedEvent = onMaximized;
                    return true;
                }
                TinyWindow_PrintErrorMessage( tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetWindow OnMaximizedByName
windowManager::DoesExist ByName
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::IsValid
TinyWindow_PrintErrorMessage

4.2.3.87 static bool windowManager::SetWindowOnMinimizedByIndex (unsigned int *windowIndex,* std::function< void(void)> *onMinimized*) [inline], [static]

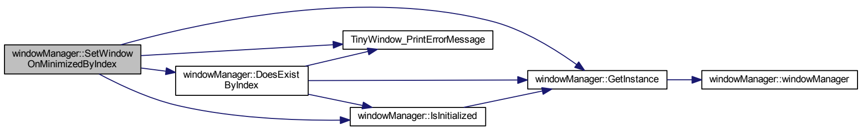
Set the window on minimized event callback by index

Definition at line 2080 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                if (onMinimized != nullptr)
                {
                    GetWindowByIndex(windowIndex)-> minimizedEvent = onMinimized;
                    return true;
                }
                TinyWindow_PrintErrorMessage( tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetWindow OnMinimizedByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.88 static bool windowManager::SetWindowOnMinimizedByName (const char ∗ *windowName,* std::function< void(void)> *onMinimized*) [inline], [static]

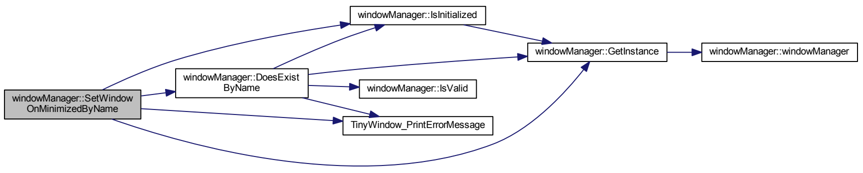
Set the window on minimized event callback by name

Definition at line 2057 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByName( windowName ) )
           {
               if (onMinimized != nullptr)
               {
                   GetWindowByName(windowName)->minimizedEvent = onMinimized;
                   return true;
               }
               TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_CALLBACK);
               return false;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::SetWindow OnMinimizedByName
TinyWindow_PrintErrorMessage

4.2.3.89 static bool windowManager::SetWindowOnMouseButtonEventByIndex (unsigned int *windowIndex,* std::function< void(tinyWindowMouseButton_t, tinyWindowButtonState_t)> *onMouseButton*) [inline], [static]

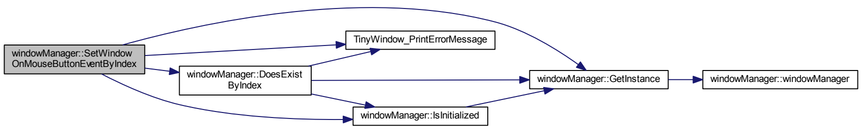
Set the window on mouse button event callback by index

Definition at line 1892 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                if (onMouseButton != nullptr)
                {
                    GetWindowByIndex(windowIndex)->mouseButtonEvent = onMouseButton;
                    return true;
                }
                TinyWindow_PrintErrorMessage( tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetWindow OnMouseButtonEventByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.90 static bool windowManager::SetWindowOnMouseButtonEventByName (const char ∗ *windowName,* std::function< void(tinyWindowMouseButton_t, tinyWindowButtonState_t)> *onMouseButton*) [inline], [static]

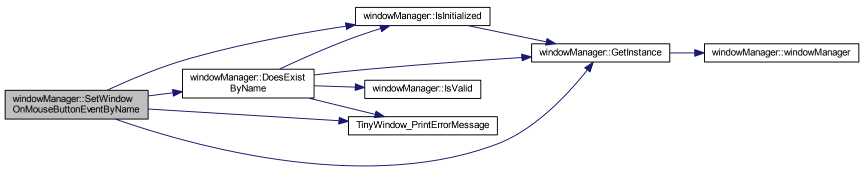
Set the window on mouse button event callback by name

Definition at line 1869 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                if (onMouseButton != nullptr)
                {
                    GetWindowByName(windowName)->mouseButtonEvent = onMouseButton;
                    return true;
                }
                TinyWindow_PrintErrorMessage( tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::SetWindow OnMouseButtonEventByName
TinyWindow_PrintErrorMessage

4.2.3.91 static bool windowManager::SetWindowOnMouseMoveByIndex (unsigned int *windowIndex,* std::function< void(unsigned int, unsigned int, unsigned int, unsigned int)> *onMouseMove*) [inline], [static]

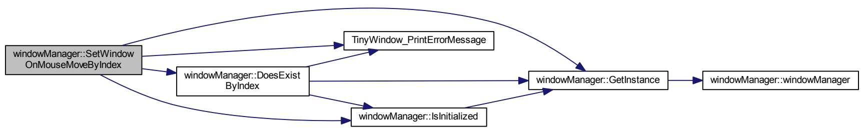
Set the window on mouse move event callback by index

Definition at line 2269 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                if (onMouseMove != nullptr)
                {
                    GetWindowByIndex(windowIndex)->mouseMoveEvent = onMouseMove;
                    return true;
                }
                TinyWindow_PrintErrorMessage(tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage(tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


TinyWindow_PrintErrorMessage
windowManager::SetWindow OnMouseMoveByIndex
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.92 static bool windowManager::SetWindowOnMouseMoveByName (const char ∗ *windowName,* std::function< void(unsigned int, unsigned int, unsigned int, unsigned int)> *onMouseMove*) [inline], [static]

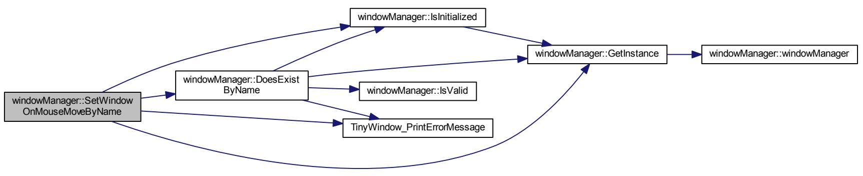
Set the window on mouse move event callback by name

Definition at line 2246 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByName( windowName ) )
           {
               if (onMouseMove != nullptr)
               {
                   GetWindowByName(windowName)->mouseMoveEvent = onMouseMove;
                   return true;
               }
               TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_CALLBACK);
               return false;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::SetWindow OnMouseMoveByName
windowManager::DoesExist ByName
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::IsValid
TinyWindow_PrintErrorMessage

4.2.3.93 static bool windowManager::SetWindowOnMouseWheelEventByIndex (unsigned int *windowIndex,* std::function< void(tinyWindowMouseScroll_t)> *onMouseWheel*) [inline], [static]

Set the window on mouse wheel event callback by index

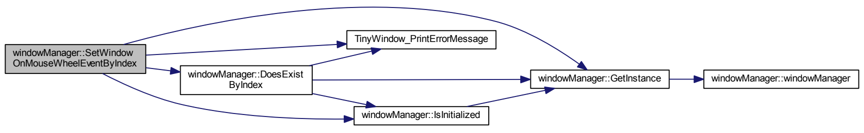
Definition at line 1939 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByIndex( windowIndex ) )
           {
               if (onMouseWheel != nullptr)
               {
                   GetWindowByIndex(windowIndex)->
mouseWheelEvent = onMouseWheel;
```

```
                    return true;
                }
                TinyWindow_PrintErrorMessage( tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetWindow OnMouseWheelEventByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.94 static bool windowManager::SetWindowOnMouseWheelEventByName (const char ∗ *windowName,* std::function< void(tinyWindowMouseScroll_t)> *onMouseWheel*) [inline], [static]

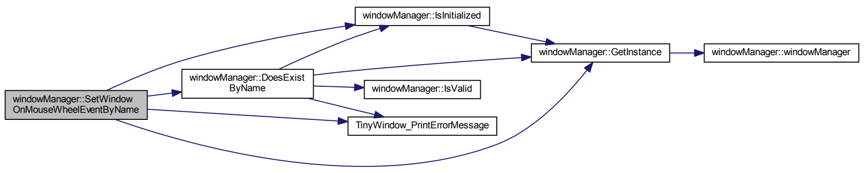
Set the window on mouse wheel event callback by name

Definition at line 1916 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                if (onMouseWheel != nullptr)
                {
                    GetWindowByName(windowName)->mouseWheelEvent = onMouseWheel;
                    return true;
                }
                TinyWindow_PrintErrorMessage( tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetWindow OnMouseWheelEventByName
windowManager::IsInitialized
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::GetInstance
windowManager::windowManager

4.2.3.95 static bool windowManager::SetWindowOnMovedByIndex (unsigned int *windowIndex,* std::function< void(unsigned int, unsigned int)> *onMoved*) [inline], [static]

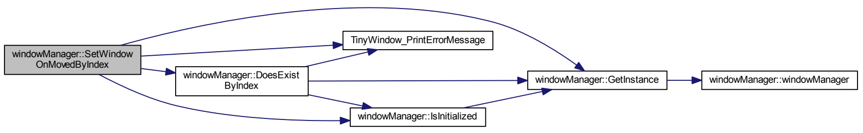
Set the window on moved event callback by index

Definition at line 2175 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
{
    if ( GetInstance()->IsInitialized() )
    {
        if ( DoesExistByIndex( windowIndex ) )
        {
            if (onMoved != nullptr)
            {
                GetWindowByIndex(windowIndex)->movedEvent = onMoved;
                return true;
            }
            TinyWindow_PrintErrorMessage( tinyWindowError_t::INVALID_CALLBACK);
            return false;
        }
        TinyWindow_PrintErrorMessage( tinyWindowError_t::WINDOW_NOT_FOUND);
        return false;
    }
    TinyWindow_PrintErrorMessage( tinyWindowError_t::NOT_INITIALIZED );
    return false;
}
```

Here is the call graph for this function:


windowManager::SetWindow OnMovedByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.96 static bool windowManager::SetWindowOnMovedByName (const char ∗ *windowName,* std::function< void(unsigned int, unsigned int)> *onMoved*) [inline], [static]

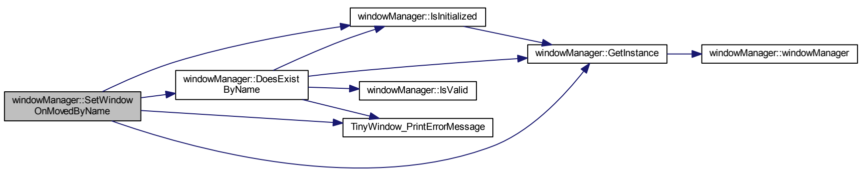
Set the window on moved event callback by name

Definition at line 2151 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
{
    if ( GetInstance()->IsInitialized() )
    {
        if ( DoesExistByName( windowName ) )
        {
            if (onMoved != nullptr)
            {
                GetWindowByName(windowName)->movedEvent = onMoved;
                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }

            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::SetWindow OnMovedByName
TinyWindow_PrintErrorMessage

4.2.3.97 static bool windowManager::SetWindowOnResizeByIndex (unsigned int *windowIndex,* std::function< void(unsigned int, unsigned int)> *onResize*) [inline], [static]

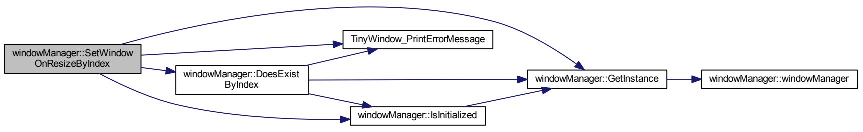
Set the window on resized event callback by index

Definition at line 2222 of file TinyWindow.h.

References DoesExistByIndex(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                if (onResize != nullptr)
                {
                    GetWindowByIndex(windowIndex)->resizeEvent = onResize;
                    return true;
                }
                TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


TinyWindow_PrintErrorMessage
windowManager::SetWindow OnResizeByIndex
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized

4.2.3.98 static bool windowManager::SetWindowOnResizeByName (const char ∗ *windowName,* std::function< void(unsigned int, unsigned int)> *onResize*) [inline], [static]

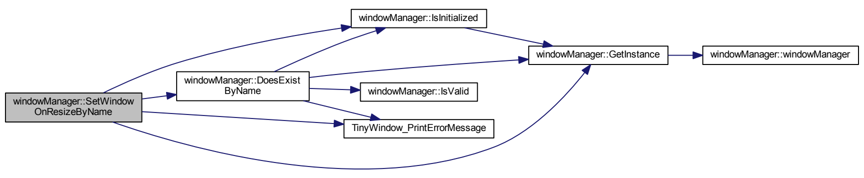
Set the window on resized event callback by name

Definition at line 2199 of file TinyWindow.h.

References DoesExistByName(), GetInstance(), INVALID_CALLBACK, IsInitialized(), NOT_INITIALIZED, Tiny↩ Window_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                if (onResize != nullptr)
                {
                    GetWindowByName(windowName)->resizeEvent = onResize;
                    return true;
                }
                TinyWindow_PrintErrorMessage(
     tinyWindowError_t::INVALID_CALLBACK);
                return false;
            }
            TinyWindow_PrintErrorMessage(
     tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
     tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
windowManager::SetWindow OnResizeByName
TinyWindow_PrintErrorMessage

4.2.3.99 static bool windowManager::SetWindowPositionByIndex (unsigned int *windowIndex,* unsigned int *x,* unsigned int *y*) [inline], [static]

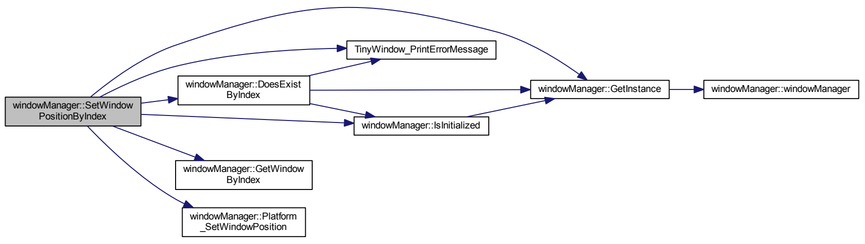
Set the position of the given window relative to screen co-ordinates

Definition at line 769 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get↩ WindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_SetWindowPosition(), windowManager::window_t↩ ::position, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
  {
      if ( GetInstance()->IsInitialized() )
      {
          if ( DoesExistByIndex( windowIndex ) )
          {
              GetWindowByIndex( windowIndex )->position[ 0 ] = x;
              GetWindowByIndex( windowIndex )->position[ 1 ] = y;
              window_t* window = GetWindowByIndex(windowIndex);
              Platform_SetWindowPosition(window, x, y);
              return true;
          }

          TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
          return false;
      }

      TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
      return false;
  }
```

Here is the call graph for this function:



windowManager::SetWindow PositionByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _SetWindowPosition

4.2.3.100 static bool windowManager::SetWindowPositionByName (const char ∗ *windowName,* unsigned int *x,* unsigned int *y*) [inline], [static]

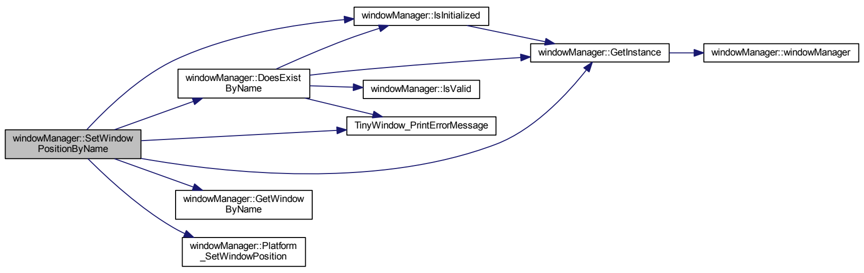
Set the Position of the given window relative to screen co-ordinates

Definition at line 745 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get↩ WindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_SetWindowPosition(), windowManager::window_t↩ ::position, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
  {
      if ( GetInstance()->IsInitialized() )
      {
          if ( DoesExistByName( windowName ) )
          {
              GetWindowByName( windowName )->position[ 0 ] = x;
              GetWindowByName( windowName )->position[ 1 ] = y;
              window_t* window = GetWindowByName(windowName);

              Platform_SetWindowPosition(window, x, y);
              TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
              return true;
          }
          TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND );
          return false;
      }

      TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
      return false;
  }
```

Here is the call graph for this function:


windowManager::SetWindow PositionByName
windowManager::IsInitialized
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::GetInstance
windowManager::windowManager
windowManager::GetWindow ByName
windowManager::Platform _SetWindowPosition

4.2.3.101 static bool windowManager::SetWindowResolutionByIndex (unsigned int *windowIndex,* unsigned int *width,* unsigned int *height*) [inline], [static]

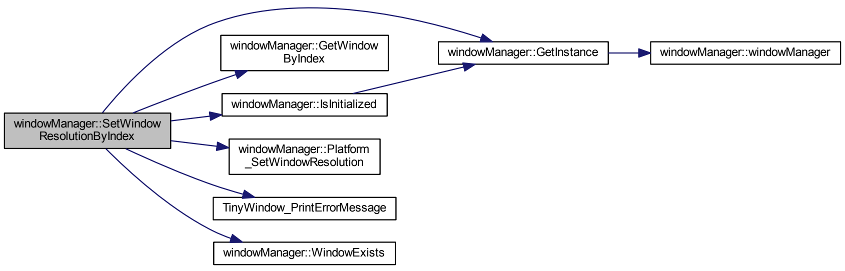
Set the Size/Resolution of the given window

Definition at line 647 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, GetInstance(), GetWindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_SetWindowResolution(), windowManager::window_t::resolution, TinyWindow_PrintErrorMessage(), WINDOW_NOT_FOUND, and WindowExists().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( WindowExists( windowIndex ) )
            {
                GetWindowByIndex( windowIndex )->resolution[ 0 ] = width;
                GetWindowByIndex( windowIndex )->resolution[ 1 ] = height;
                window_t* window = GetWindowByIndex(windowIndex);

                Platform_SetWindowResolution(window);
                return true;
            }
            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetWindow ResolutionByIndex
windowManager::GetWindow ByIndex
windowManager::IsInitialized
windowManager::Platform _SetWindowResolution
TinyWindow_PrintErrorMessage
windowManager::WindowExists
windowManager::GetInstance
windowManager::windowManager

4.2.3.102 static bool windowManager::SetWindowResolutionByName (const char ∗ *windowName,* unsigned int *width,* unsigned int *height*) [inline], [static]

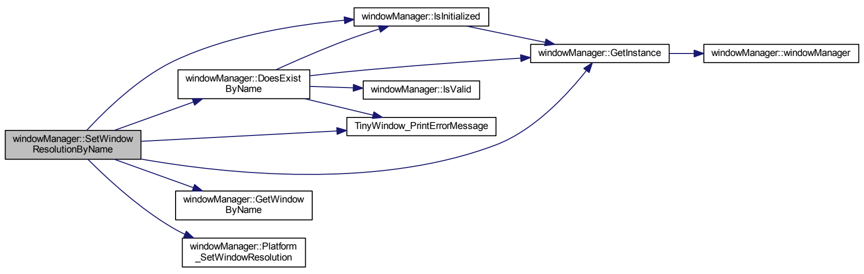
Set the Size/Resolution of the given window

Definition at line 624 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get↩ WindowByName(), INVALID_CONTEXT, IsInitialized(), Platform_SetWindowResolution(), windowManager↩ ::window_t::resolution, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
  {
      if ( GetInstance()->IsInitialized() )
      {
          if ( DoesExistByName( windowName ) )
          {
              GetWindowByName( windowName )->resolution[ 0 ] = width;
              GetWindowByName( windowName )->resolution[ 1 ] = height;
              window_t* window = GetWindowByName(windowName);

              Platform_SetWindowResolution(window);
              return true;
          }
          TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
          return false;
      }

      TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_CONTEXT );
      return false;
  }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::SetWindow ResolutionByName
windowManager::GetWindow ByName
windowManager::Platform _SetWindowResolution

4.2.3.103 static bool windowManager::SetWindowStyleByIndex (unsigned int *windowIndex,* tinyWindowStyle_t *windowStyle*) [inline], [static]

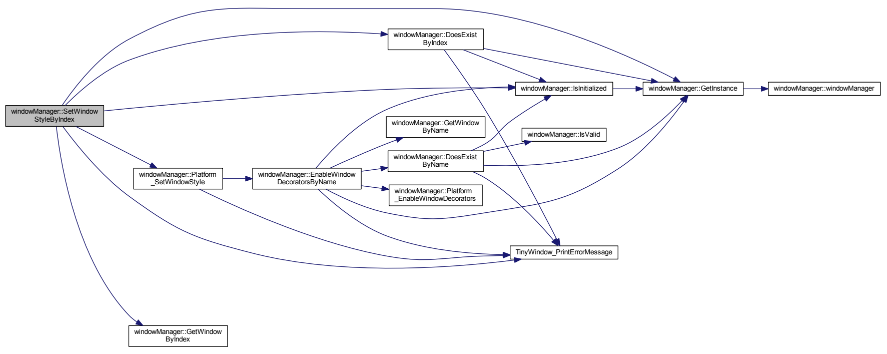
Set the window style preset by index

Definition at line 1723 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get↩ WindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_SetWindowStyle(), TinyWindow_PrintError↩ Message(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByIndex( windowIndex ) )
           {
               window_t* window = GetWindowByIndex(windowIndex);
               Platform_SetWindowStyle(window, windowStyle);
               return true;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::SetWindowStyleByIndex
windowManager::DoesExistByIndex
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::GetWindowByName
windowManager::IsValid
windowManager::Platform_SetWindowStyle
windowManager::EnableWindowDecoratorsByName
windowManager::DoesExistByName
windowManager::Platform_EnableWindowDecorators
TinyWindow_PrintErrorMessage
windowManager::GetWindowByIndex

4.2.3.104 static bool windowManager::SetWindowStyleByName (const char ∗ *windowName*, tinyWindowStyle_t *windowStyle*) [inline], [static]

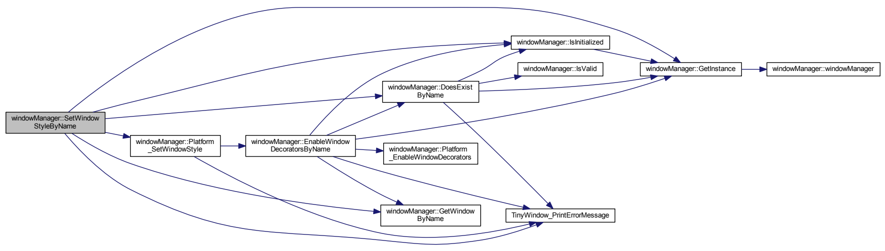
Set the window style preset by name

Definition at line 1704 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get↩ WindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_SetWindowStyle(), TinyWindow_PrintError↩ Message(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if ( DoesExistByName( windowName ) )
           {
               window_t* window = GetWindowByName(windowName);
               Platform_SetWindowStyle(window, windowStyle);
               return true;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::SetWindow StyleByName
windowManager::Platform _SetWindowStyle
windowManager::EnableWindow DecoratorsByName
windowManager::DoesExist ByName
windowManager::IsInitialized
windowManager::IsValid
windowManager::GetInstance
windowManager::windowManager
windowManager::Platform _EnableWindowDecorators
windowManager::GetWindow ByName
TinyWindow_PrintErrorMessage

4.2.3.105 static bool windowManager::SetWindowTitleBarByIndex (unsigned int *windowIndex,* const char ∗ *newName*) `[inline], [static]`

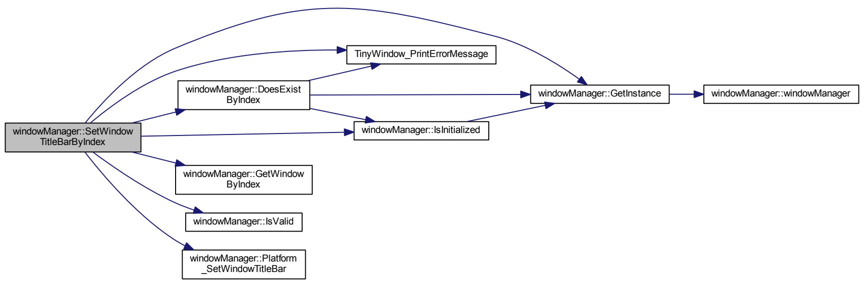
Set the window title bar by index

Definition at line 1351 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get←WindowByIndex(), INVALID_WINDOW_NAME, IsInitialized(), IsValid(), NOT_INITIALIZED, Platform_SetWindow←TitleBar(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if (IsValid(newName))
            {
                if (DoesExistByIndex(windowIndex))
                {
                    window_t* window = GetWindowByIndex(windowIndex);
                    Platform_SetWindowTitleBar(window, newName);
                    return true;
                }
                TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
                return false;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::INVALID_WINDOW_NAME);
            return false;
        }

        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::SetWindow TitleBarByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::IsValid
windowManager::Platform _SetWindowTitleBar

4.2.3.106 static bool windowManager::SetWindowTitleBarByName (const char ∗ *windowName,* const char ∗ *newTitle*) [inline], [static]

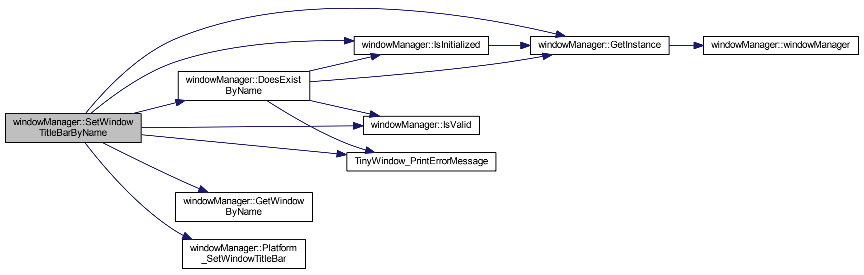
Set the window title bar by name

Definition at line 1327 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get←WindowByName(), INVALID_WINDOW_NAME, IsInitialized(), IsValid(), NOT_INITIALIZED, Platform_SetWindow←TitleBar(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
   {
       if ( GetInstance()->IsInitialized() )
       {
           if (IsValid(newTitle))
           {
               if (DoesExistByName(windowName))
               {
                   window_t* window = GetWindowByName(windowName);
                   Platform_SetWindowTitleBar(window, newTitle);
                   return true;
               }
               TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
               return false;
           }
           TinyWindow_PrintErrorMessage(
tinyWindowError_t::INVALID_WINDOW_NAME);
           return false;
       }
       TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
       return false;
   }
```

Here is the call graph for this function:


windowManager::SetWindow TitleBarByName
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::GetWindow ByName
windowManager::Platform _SetWindowTitleBar

4.2.3.107 static void windowManager::ShutDown (void) [inline], [static]

Use this to shutdown the window manager when your program is finished

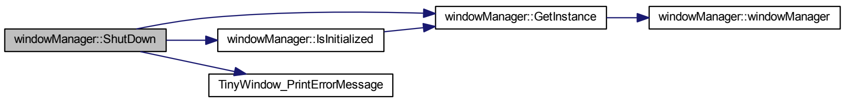
Definition at line 368 of file TinyWindow.h.

References GetInstance(), instance, IsInitialized(), isInitialized, NOT_INITIALIZED, and TinyWindow_PrintError←Message().

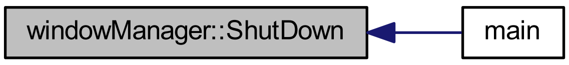
Referenced by main().

```
    {
        if (GetInstance()->IsInitialized())
        {
            for (auto CurrentWindow : instance->windowList)
            {
                delete CurrentWindow;
            }

#if defined( CURRENT_OS_LINUX )
            XCloseDisplay(instance->currentDisplay);
#endif

            instance->windowList.clear();
            instance->isInitialized = false;
            delete instance;

        }
        TinyWindow_PrintErrorMessage(
    tinyWindowError_t::NOT_INITIALIZED);
    }
```

Here is the call graph for this function:


windowManager::ShutDown
windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
TinyWindow_PrintErrorMessage

Here is the caller graph for this function:



4.2.3.108 static void windowManager::ShutdownWindow (window_t * *window*) [inline], [static], [private]

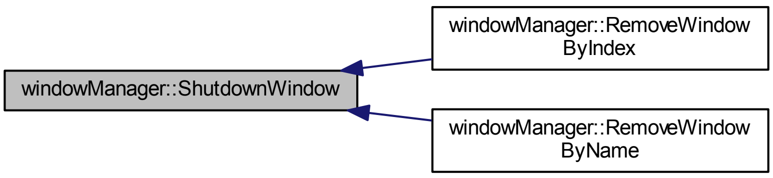
Definition at line 3017 of file TinyWindow.h.

References windowManager::window_t::name.

Referenced by RemoveWindowByIndex(), and RemoveWindowByName().

```
    {
#if defined( _WIN32 ) || defined( _WIN64 )
        if (window->glRenderingContextHandle)
        {
            wglMakeCurrent(nullptr, nullptr);
            wglDeleteContext(window->glRenderingContextHandle);
        }

        if (window->paletteHandle)
        {
            DeleteObject(window->paletteHandle);
        }
        ReleaseDC(window->windowHandle, window->deviceContextHandle);
        UnregisterClass(window->name, window->instanceHandle);

        FreeModule(window->instanceHandle);

        window->deviceContextHandle = nullptr;
        window->windowHandle = nullptr;
        window->glRenderingContextHandle = nullptr;
#elif defined(__linux__)
        if (window->currentState == tinyWindowState_t::FULLSCREEN)
        {
            RestoreWindowByName(window->name);
        }

        glXDestroyContext(instance->currentDisplay, window->context);
        XUnmapWindow(instance->currentDisplay, window->windowHandle);
        XDestroyWindow(instance->currentDisplay, window->windowHandle);
        window->windowHandle = 0;
        window->context = 0;
#endif
    }
```

Here is the caller graph for this function:


windowManager::RemoveWindow ByIndex
windowManager::ShutdownWindow
windowManager::RemoveWindow ByName

4.2.3.109 static void windowManager::WaitForEvents (void) [inline], [static]

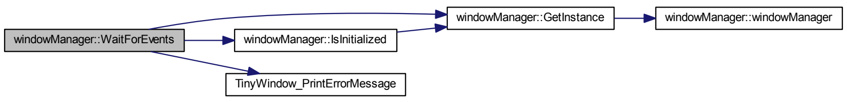
Ask the window manager to wait for events

Definition at line 1639 of file TinyWindow.h.

References GetInstance(), instance, IsInitialized(), NOT_INITIALIZED, and TinyWindow_PrintErrorMessage().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
#if defined( _WIN32 ) || defined( _WIN64 )
            //process even if there aren't any to process
            GetMessage(&instance->message, 0, 0, 0);
            TranslateMessage(&instance->message);
            DispatchMessage(&instance->message);
#elif defined(__linux__)
            //even if there aren't any events to process
            XNextEvent(instance->currentDisplay, &instance->currentEvent);

            XEvent currentEvent = instance->currentEvent;

            Linux_ProcessEvents(currentEvent);
#endif
        }

        else
        {
            TinyWindow_PrintErrorMessage(
    tinyWindowError_t::NOT_INITIALIZED );
        }
    }
```

Here is the call graph for this function:


windowManager::WaitForEvents
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
TinyWindow_PrintErrorMessage

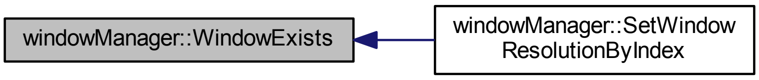
4.2.3.110 static bool windowManager::WindowExists (unsigned int *windowIndex*) [inline], [static], [private]

Definition at line 2426 of file TinyWindow.h.

Referenced by SetWindowResolutionByIndex().

```
{
    return ( windowIndex <= instance->windowList.size() - 1 );
}
```

Here is the caller graph for this function:


windowManager::WindowExists
windowManager::SetWindow ResolutionByIndex

4.2.3.111 static tinyWindowKeyState_t windowManager::WindowGetKeyByIndex (unsigned int *windowIndex,* unsigned int *key*) [inline], [static]

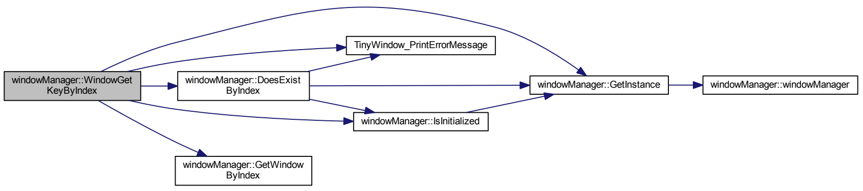
Returns the current state of the given key relative to the given window

Definition at line 929 of file TinyWindow.h.

References BAD, DoesExistByIndex(), GetInstance(), GetWindowByIndex(), IsInitialized(), windowManager::window_t::keys, NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
{
    if ( GetInstance()->IsInitialized() )
    {
        if ( DoesExistByIndex( windowIndex ) )
        {
            return GetWindowByIndex( windowIndex )->keys[ key ];
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
        return tinyWindowKeyState_t::BAD;
    }
    TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
    return tinyWindowKeyState_t::BAD;
}
```

Here is the call graph for this function:


windowManager::WindowGetKeyByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExistByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindowByIndex

4.2.3.112 static tinyWindowKeyState_t windowManager::WindowGetKeyByName (const char ∗ *windowName,* unsigned int *key*) [inline], [static]

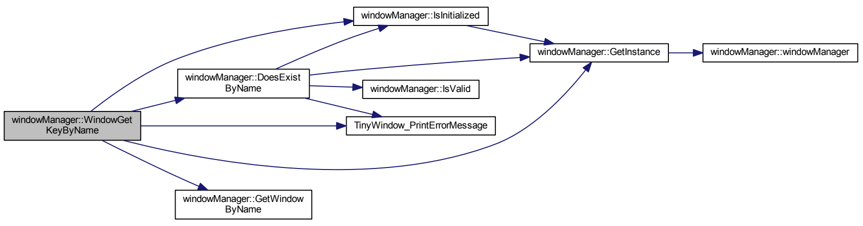
Returns the current state of the given key relative to the given window

Definition at line 911 of file TinyWindow.h.

References BAD, DoesExistByName(), GetInstance(), GetWindowByName(), IsInitialized(), windowManager::window_t::keys, NOT_INITIALIZED, TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                return GetWindowByName( windowName )->keys[ key ];
            }

            TinyWindow_PrintErrorMessage(
tinyWindowError_t::WINDOW_NOT_FOUND);
            return tinyWindowKeyState_t::BAD;
        }
        TinyWindow_PrintErrorMessage(
tinyWindowError_t::NOT_INITIALIZED );
        return tinyWindowKeyState_t::BAD;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExistByName
windowManager::IsValid
windowManager::WindowGetKeyByName
TinyWindow_PrintErrorMessage
windowManager::GetWindowByName

4.2.3.113 static bool windowManager::WindowSwapBuffersByIndex (unsigned int *windowIndex*) [inline], [static]

Swap the draw buffers of the given window

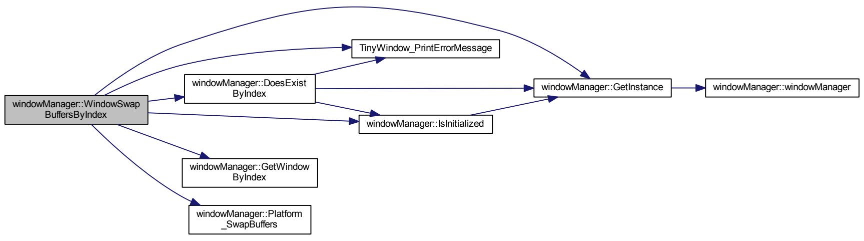
Definition at line 1005 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByIndex(), GetInstance(), Get←WindowByIndex(), IsInitialized(), NOT_INITIALIZED, Platform_SwapBuffers(), TinyWindow_PrintErrorMessage(), and WINDOW_NOT_FOUND.

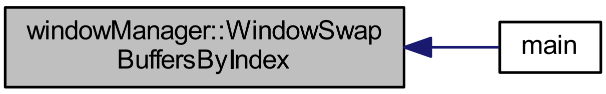
Referenced by main().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByIndex( windowIndex ) )
            {
                window_t* window = GetWindowByIndex(windowIndex);
                Platform_SwapBuffers(window);
                return true;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::WINDOW_NOT_FOUND);
            return false;
        }
        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::WindowSwap BuffersByIndex
TinyWindow_PrintErrorMessage
windowManager::DoesExist ByIndex
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::GetWindow ByIndex
windowManager::Platform _SwapBuffers

Here is the caller graph for this function:



4.2.3.114 static bool windowManager::WindowSwapBuffersByName (const char ∗ *windowName*) [inline], [static]

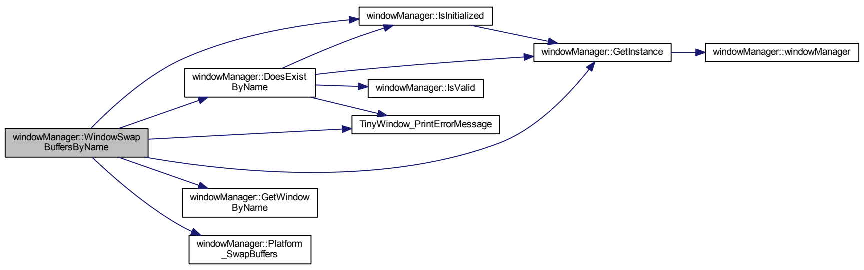
Swap the draw buffers of the given window

Definition at line 984 of file TinyWindow.h.

References windowManager::window_t::AtomDesktopGeometry, DoesExistByName(), GetInstance(), Get↩ WindowByName(), IsInitialized(), NOT_INITIALIZED, Platform_SwapBuffers(), and TinyWindow_PrintError↩ Message().

```
    {
        if ( GetInstance()->IsInitialized() )
        {
            if ( DoesExistByName( windowName ) )
            {
                window_t* window = GetWindowByName(windowName);
                Platform_SwapBuffers(window);

                return true;
            }
            TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED);
            return false;
        }

        TinyWindow_PrintErrorMessage(
      tinyWindowError_t::NOT_INITIALIZED );
        return false;
    }
```

Here is the call graph for this function:


windowManager::IsInitialized
windowManager::GetInstance
windowManager::windowManager
windowManager::DoesExist ByName
windowManager::IsValid
TinyWindow_PrintErrorMessage
windowManager::WindowSwap BuffersByName
windowManager::GetWindow ByName
windowManager::Platform _SwapBuffers

4.2.4 Member Data Documentation

4.2.4.1 windowManager ∗ windowManager::instance = nullptr [static], [private]

Definition at line 2414 of file TinyWindow.h.

Referenced by AddWindow(), GetInstance(), GetMousePositionInScreen(), GetScreenResolution(), Initialize(), PollForEvents(), SetMousePositionInScreen(), ShutDown(), and WaitForEvents().

4.2.4.2 bool windowManager::isInitialized [private]

Definition at line 2419 of file TinyWindow.h.

Referenced by Initialize(), IsInitialized(), and ShutDown().

4.2.4.3 unsigned int windowManager::screenMousePosition[2] [private]

Definition at line 2417 of file TinyWindow.h.

Referenced by GetMousePositionInScreen(), and SetMousePositionInScreen().

4.2.4.4 unsigned int windowManager::screenResolution[2] [private]

Definition at line 2416 of file TinyWindow.h.

Referenced by GetScreenResolution(), and Initialize().

4.2.4.5 std::vector< window_t* > windowManager::windowList [private]

Definition at line 2413 of file TinyWindow.h.

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

- Include/TinyWindow.h

Chapter 5

File Documentation

5.1 Example/CMakeFiles/3.4.2/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID ""
- #define ARCHITECTURE_ID ""
- #define DEC(n)
- #define HEX(n)

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" "" "]"
- char const ∗ info_platform = "INFO" ":" "platform[" "" "]"
- char const ∗ info_arch = "INFO" ":" "arch[" "" "]"
- const char ∗ info_language_dialect_default

5.1.1 Macro Definition Documentation

5.1.1.1 #define ARCHITECTURE_ID ""

Definition at line 411 of file CMakeCXXCompilerId.cpp.

5.1.1.2 #define COMPILER_ID ""

Definition at line 248 of file CMakeCXXCompilerId.cpp.

5.1.1.3 #define DEC(*n*)

Value:

```
('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 415 of file CMakeCXXCompilerId.cpp.

5.1.1.4 #define HEX(*n*)

Value:

```
('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 426 of file CMakeCXXCompilerId.cpp.

5.1.1.5 #define PLATFORM_ID ""

Definition at line 361 of file CMakeCXXCompilerId.cpp.

5.1.1.6 #define STRINGIFY(*X*) STRINGIFY_HELPER(X)

Definition at line 265 of file CMakeCXXCompilerId.cpp.

5.1.1.7 #define STRINGIFY_HELPER(*X*) #X

Definition at line 264 of file CMakeCXXCompilerId.cpp.

5.1.2 Function Documentation

5.1.2.1 int main (int *argc,* char ∗ *argv[ ]*)

Definition at line 494 of file CMakeCXXCompilerId.cpp.

References info_compiler, info_language_dialect_default, and info_platform.

```
{
  int require = 0;
  require += info_compiler[argc];
  require += info_platform[argc];
#ifdef COMPILER_VERSION_MAJOR
  require += info_version[argc];
#endif
#ifdef SIMULATE_ID
  require += info_simulate[argc];
#endif
#ifdef SIMULATE_VERSION_MAJOR
  require += info_simulate_version[argc];
#endif
  require += info_language_dialect_default[argc];
  (void)argv;
  return require;
}
```

5.1.3 Variable Documentation

5.1.3.1 char const∗ info_arch = "INFO" ":" "arch[" "" "]"

Definition at line 477 of file CMakeCXXCompilerId.cpp.

5.1.3.2 char const∗ info_compiler = "INFO" ":" "compiler[" "" "]"

Definition at line 255 of file CMakeCXXCompilerId.cpp.

Referenced by main().

5.1.3.3 const char∗ info_language_dialect_default

Initial value:

```
= "INFO" ":" "dialect_default["



  "98"
"]"
```

Definition at line 482 of file CMakeCXXCompilerId.cpp.

Referenced by main().

5.1.3.4 char const∗ info_platform = "INFO" ":" "platform[" "" "]"

Definition at line 476 of file CMakeCXXCompilerId.cpp.

Referenced by main().

5.2 Example/CMakeFiles/feature_tests.cxx File Reference

Functions

- int main (int argc, char ∗∗argv)

Variables

- const char features [ ]

5.2.1 Function Documentation

5.2.1.1 int main (int *argc*, char ∗∗ *argv*)

Definition at line 377 of file feature_tests.cxx.

References features.

```
377 { (void)argv; return features[argc]; }
```

5.2.2 Variable Documentation

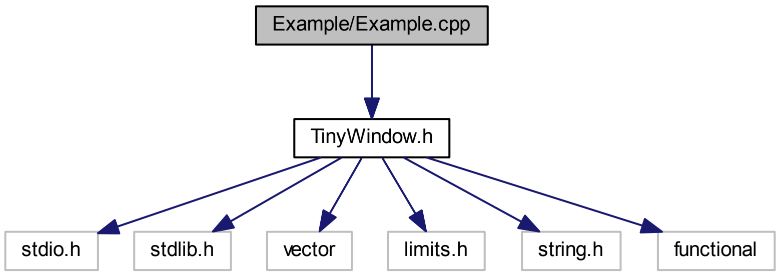
5.2.2.1 const char features[ ]

Definition at line 2 of file feature_tests.cxx.

Referenced by main().

5.3 Example/Example.cpp File Reference

```
#include "TinyWindow.h"
```
Include dependency graph for Example.cpp:



Functions

- void handleKeyPresses (unsigned int key, tinyWindowKeyState_t keyState)
- int main ()

5.3.1 Function Documentation

5.3.1.1 void handleKeyPresses (unsigned int *key*, tinyWindowKeyState_t *keyState*)

Definition at line 3 of file Example.cpp.

References DOWN.

```
{
    if(keyState == tinyWindowKeyState_t::DOWN)
    {
        printf("%c \t", key);
    }
}
```

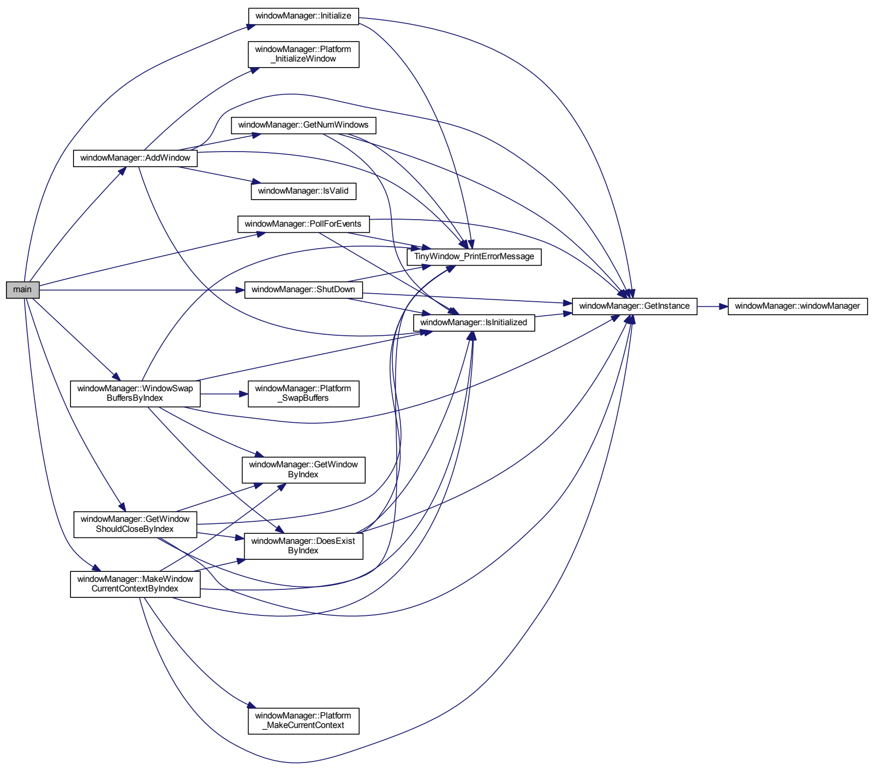
5.3.1.2 int main ( )

Definition at line 11 of file Example.cpp.

References windowManager::AddWindow(), windowManager::GetWindowShouldCloseByIndex(), window↩Manager::Initialize(), windowManager::MakeWindowCurrentContextByIndex(), windowManager::PollForEvents(), windowManager::ShutDown(), and windowManager::WindowSwapBuffersByIndex().

```
{
    windowManager::Initialize();

    windowManager::AddWindow("Example");
    windowManager::SetWindowOnKeyEventByName("Example",
   handleKeyPresses);

    glClearColor(0.25f, 0.25f, 0.25f, 1.0f);
    while (!windowManager::GetWindowShouldCloseByIndex(0))
    {
        windowManager::PollForEvents();// or WaitForEvents

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        windowManager::MakeWindowCurrentContextByIndex(0);
        windowManager::WindowSwapBuffersByIndex(0);
    }

    windowManager::ShutDown();
    return 0;
}
```

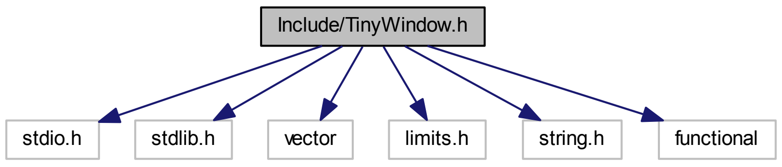
Here is the call graph for this function:


windowManager::Initialize
windowManager::Platform_InitializeWindow
windowManager::GetNumWindows
windowManager::AddWindow
windowManager::IsValid
windowManager::PollForEvents
TinyWindow_PrintErrorMessage
main
windowManager::ShutDown
windowManager::GetInstance
windowManager::windowManager
windowManager::IsInitialized
windowManager::WindowSwapBuffersByIndex
windowManager::Platform_SwapBuffers
windowManager::GetWindowByIndex
windowManager::GetWindowShouldCloseByIndex
windowManager::DoesExistByIndex
windowManager::MakeWindowCurrentContextByIndex
windowManager::Platform_MakeCurrentContext

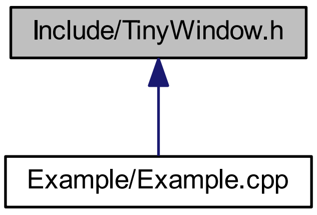
5.4 Include/TinyWindow.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <limits.h>
#include <string.h>
#include <functional>
```

Include dependency graph for TinyWindow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class windowManager
- struct windowManager::window_t

Enumerations

- enum tinyWindowKeyState_t { tinyWindowKeyState_t::UP = 0, tinyWindowKeyState_t::DOWN, tinyWindow↩
  KeyState_t::BAD = -1 }
- enum tinyWindowKey_t {
  KEY_ERROR = -1, KEY_FIRST = 256 + 1, KEY_F1, KEY_F2,
  KEY_F3, KEY_F4, KEY_F5, KEY_F6,
  KEY_F7, KEY_F8, KEY_F9, KEY_F10,
  KEY_F11, KEY_F12, KEY_CAPSLOCK, KEY_LEFTSHIFT,
  KEY_RIGHTSHIFT, KEY_LEFTCONTROL, KEY_RIGHTCONTROL, KEY_LEFTWINDOW,
  KEY_RIGHTWINDOW, KEY_LEFTALT, KEY_RIGHTALT, KEY_ENTER,
  KEY_PRINTSCREEN, KEY_SCROLLLOCK, KEY_NUMLOCK, KEY_PAUSE,
  KEY_INSERT, KEY_HOME, KEY_END, KEY_PAGEUP,
  KEY_PAGEDOWN, KEY_ARROW_DOWN, KEY_ARROW_UP, KEY_ARROW_LEFT,
  KEY_ARROW_RIGHT, KEY_KEYPAD_DIVIDE, KEY_KEYPAD_MULTIPLY, KEY_KEYPAD_SUBTRACT,
  KEY_KEYPAD_ADD, KEY_KEYPAD_ENTER, KEY_KEYPAD_PERIOD, KEY_KEYPAD_0,
  KEY_KEYPAD_1, KEY_KEYPAD_2, KEY_KEYPAD_3, KEY_KEYPAD_4,
  KEY_KEYPAD_5, KEY_KEYPAD_6, KEY_KEYPAD_7, KEY_KEYPAD_8,
  KEY_KEYPAD_9, KEY_BACKSPACE, KEY_TAB, KEY_DELETE,
  KEY_ESCAPE, KEY_LAST = KEY_ESCAPE }
- enum tinyWindowButtonState_t { tinyWindowButtonState_t::UP = 0, tinyWindowButtonState_t::DOWN }
- enum tinyWindowMouseButton_t { tinyWindowMouseButton_t::LEFT = 0, tinyWindowMouseButton_t::RIG↩
  HT, tinyWindowMouseButton_t::MIDDLE, tinyWindowMouseButton_t::LAST }
- enum tinyWindowMouseScroll_t { tinyWindowMouseScroll_t::DOWN = 0, tinyWindowMouseScroll_t::UP }
- enum tinyWindowStyle_t { tinyWindowStyle_t::BARE = 1, tinyWindowStyle_t::DEFAULT, tinyWindowStyle↩
  _t::POPUP }
- enum tinyWindowState_t { tinyWindowState_t::NORMAL = 0, tinyWindowState_t::MAXIMIZED, tinyWindow↩
  State_t::MINIMIZED, tinyWindowState_t::FULLSCREEN }
- enum tinyWindowDecorator_t {
  DECORATOR_TITLEBAR = 0x01, DECORATOR_ICON = 0x02, DECORATOR_BORDER = 0x04, DECO↩
  RATOR_MINIMIZEBUTTON = 0x08,
  DECORATOR_MAXIMIZEBUTTON = 0x010, DECORATOR_CLOSEBUTTON = 0x20, DECORATOR_SI↩
  ZEABLEBORDER = 0x40 }

- enum tinyWindowError_t {
  tinyWindowError_t::TINYWINDOW_ERROR = -1, tinyWindowError_t::INVALID_WINDOW_NAME = 0, tiny↩
  WindowError_t::INVALID_ICON_PATH, tinyWindowError_t::INVALID_WINDOW_INDEX,
  tinyWindowError_t::INVALID_WINDOW_STATE, tinyWindowError_t::INVALID_RESOLUTION, tiny↩
  WindowError_t::INVALID_CONTEXT, tinyWindowError_t::EXISTING_CONTEXT,
  tinyWindowError_t::NOT_INITIALIZED, tinyWindowError_t::ALREADY_INITIALIZED, tinyWindowError_t::I↩
  NVALID_TITLEBAR, tinyWindowError_t::INVALID_CALLBACK,
  tinyWindowError_t::WINDOW_NOT_FOUND, tinyWindowError_t::INVALID_WINDOWSTYLE, tinyWindow↩
  Error_t::FUNCTION_NOT_IMPLEMENTED, tinyWindowError_t::LINUX_CANNOT_CONNECT_X_SERVER,
  tinyWindowError_t::LINUX_INVALID_VISUALINFO, tinyWindowError_t::LINUX_CANNOT_CREATE_WIN↩
  DOW, tinyWindowError_t::LINUX_FUNCTION_NOT_IMPLEMENTED, tinyWindowError_t::WINDOWS_C↩
  ANNOT_CREATE_WINDOW,
  tinyWindowError_t::WINDOWS_CANNOT_INITIALIZE, tinyWindowError_t::WINDOWS_FUNCTION_NOT↩
  _IMPLEMENTED }

Functions

- static void TinyWindow_PrintErrorMessage (const tinyWindowError_t errorNumber)

Variables

- _WIN32 _WIN64 const int DEFAULT_WINDOW_WIDTH = 1280
- const int DEFAULT_WINDOW_HEIGHT = 720
- const int LINUX_FUNCTION = 1
- const int LINUX_DECORATOR = 2

5.4.1 Enumeration Type Documentation

5.4.1.1 enum **tinyWindowButtonState_t** [strong]

Enumerator

UP The mouse button is currently up

DOWN The mouse button is currently down

Definition at line 106 of file TinyWindow.h.

```
{
    UP = 0,
    DOWN
};
```

5.4.1.2 enum tinyWindowDecorator_t

Enumerator

DECORATOR_TITLEBAR The title bar decoration of the window

DECORATOR_ICON The icon decoration of the window

DECORATOR_BORDER The border decoration of the window

DECORATOR_MINIMIZEBUTTON The minimize button decoration of the window

DECORATOR_MAXIMIZEBUTTON The maximize button decoration pf the window

DECORATOR_CLOSEBUTTON The close button decoration of the window

DECORATOR_SIZEABLEBORDER The sizable border decoration of the window

Definition at line 141 of file TinyWindow.h.

```
{
    DECORATOR_TITLEBAR = 0x01,
    DECORATOR_ICON = 0x02,
    DECORATOR_BORDER = 0x04,
    DECORATOR_MINIMIZEBUTTON = 0x08,
    DECORATOR_MAXIMIZEBUTTON = 0x010,
    DECORATOR_CLOSEBUTTON = 0x20,
    DECORATOR_SIZEABLEBORDER = 0x40,
};
```

5.4.1.3 enum tinyWindowError_t [strong]

Enumerator

TINYWINDOW_ERROR

INVALID_WINDOW_NAME If an invalid window name was given

INVALID_ICON_PATH If an invalid icon path was given

INVALID_WINDOW_INDEX If an invalid window index was given

INVALID_WINDOW_STATE If an invalid window state was given

INVALID_RESOLUTION If an invalid window resolution was given

INVALID_CONTEXT If the OpenGL context for the window is invalid

EXISTING_CONTEXT If the window already has an OpenGL context

NOT_INITIALIZED If the window is being used without being initialized

ALREADY_INITIALIZED If the window was already initialized

INVALID_TITLEBAR If the Title-bar text given was invalid

INVALID_CALLBACK If the given event callback was invalid

WINDOW_NOT_FOUND If the window was not found in the window manager

INVALID_WINDOWSTYLE If the window style gives is invalid

FUNCTION_NOT_IMPLEMENTED If the function has not yet been implemented in the current version of the API

LINUX_CANNOT_CONNECT_X_SERVER Linux: if cannot connect to an X11 server

LINUX_INVALID_VISUALINFO Linux: if visual information given was invalid

LINUX_CANNOT_CREATE_WINDOW Linux: when X11 fails to create a new window

LINUX_FUNCTION_NOT_IMPLEMENTED Linux: when the function has not yet been implemented on the Linux in the current version of the API

WINDOWS_CANNOT_CREATE_WINDOW Windows: when Win32 cannot create a window

WINDOWS_CANNOT_INITIALIZE Windows: when Win32 cannot initialize

WINDOWS_FUNCTION_NOT_IMPLEMENTED Windows: when a function has yet to be implemented on the Windows platform in the current version of the API

Definition at line 152 of file TinyWindow.h.

```
{
    TINYWINDOW_ERROR = -1,
    INVALID_WINDOW_NAME = 0,
    INVALID_ICON_PATH,
    INVALID_WINDOW_INDEX,
    INVALID_WINDOW_STATE,
    INVALID_RESOLUTION,
    INVALID_CONTEXT,
    EXISTING_CONTEXT,
    NOT_INITIALIZED,
    ALREADY_INITIALIZED,
    INVALID_TITLEBAR,
    INVALID_CALLBACK,
    WINDOW_NOT_FOUND,
    INVALID_WINDOWSTYLE,
    FUNCTION_NOT_IMPLEMENTED,
    LINUX_CANNOT_CONNECT_X_SERVER,
    LINUX_INVALID_VISUALINFO,
    LINUX_CANNOT_CREATE_WINDOW,
    LINUX_FUNCTION_NOT_IMPLEMENTED,
    WINDOWS_CANNOT_CREATE_WINDOW,
    WINDOWS_CANNOT_INITIALIZE,
    WINDOWS_FUNCTION_NOT_IMPLEMENTED,
};
```

5.4.1.4 enum tinyWindowKey_t

Enumerator

KEY_ERROR The key pressed is considered invalid

KEY_FIRST The fist key that is not a char

KEY_F1 The F1 key

KEY_F2 The F2 key

KEY_F3 The F3 key

KEY_F4 The F4 key

KEY_F5 The F5 key

KEY_F6 The F6 key

KEY_F7 The F7 key

KEY_F8 The F8 key

KEY_F9 The F9 key

KEY_F10 The F10 key

KEY_F11 The F11 key

KEY_F12 The F12 key

KEY_CAPSLOCK The CapsLock key

KEY_LEFTSHIFT The left Shift key

KEY_RIGHTSHIFT The right Shift key

KEY_LEFTCONTROL The left Control key

KEY_RIGHTCONTROL The right Control key

KEY_LEFTWINDOW The left Window key

KEY_RIGHTWINDOW The right Window key

KEY_LEFTALT The left Alternate key

KEY_RIGHTALT The right Alternate key

KEY_ENTER The Enter/Return key

KEY_PRINTSCREEN The PrintScreen key

KEY_SCROLLLOCK The ScrollLock key

KEY_NUMLOCK The NumLock key

KEY_PAUSE The pause/break key

KEY_INSERT The insert key

KEY_HOME The Home key

KEY_END The End key

KEY_PAGEUP The PageUp key

KEY_PAGEDOWN The PageDown key

KEY_ARROW_DOWN The ArrowDown key

KEY_ARROW_UP The ArrowUp key

KEY_ARROW_LEFT The ArrowLeft key

KEY_ARROW_RIGHT The ArrowRight key

KEY_KEYPAD_DIVIDE The KeyPad Divide key

KEY_KEYPAD_MULTIPLY The Keypad Multiply key

KEY_KEYPAD_SUBTRACT The Keypad Subtract key

KEY_KEYPAD_ADD The Keypad Add key

KEY_KEYPAD_ENTER The Keypad Enter key

KEY_KEYPAD_PERIOD The Keypad Period/Decimal key

KEY_KEYPAD_0 The Keypad 0 key

KEY_KEYPAD_1 The Keypad 1 key

KEY_KEYPAD_2 The Keypad 2 key

KEY_KEYPAD_3 The Keypad 3 key

KEY_KEYPAD_4 The Keypad 4 key

KEY_KEYPAD_5 The Keypad 5 key

KEY_KEYPAD_6 The Keypad 6 key

KEY_KEYPAD_7 The Keypad 7 key

KEY_KEYPAD_8 The keypad 8 key

KEY_KEYPAD_9 The Keypad 9 key

KEY_BACKSPACE The Backspace key

KEY_TAB The Tab key

KEY_DELETE The Delete key

KEY_ESCAPE The Escape key

KEY_LAST The last key to be supported

Definition at line 44 of file TinyWindow.h.

```
{
    KEY_ERROR = -1,
    KEY_FIRST = 256 + 1,
    KEY_F1,
    KEY_F2,
    KEY_F3,
    KEY_F4,
    KEY_F5,
    KEY_F6,
    KEY_F7,
    KEY_F8,
    KEY_F9,
    KEY_F10,
    KEY_F11,
    KEY_F12,
    KEY_CAPSLOCK,
    KEY_LEFTSHIFT,
    KEY_RIGHTSHIFT,
    KEY_LEFTCONTROL,
    KEY_RIGHTCONTROL,
    KEY_LEFTWINDOW,
    KEY_RIGHTWINDOW,
    KEY_LEFTALT,
    KEY_RIGHTALT,
    KEY_ENTER,
    KEY_PRINTSCREEN,
    KEY_SCROLLLOCK,
    KEY_NUMLOCK,
    KEY_PAUSE,
    KEY_INSERT,
    KEY_HOME,
    KEY_END,
    KEY_PAGEUP,
    KEY_PAGEDOWN,
    KEY_ARROW_DOWN,
    KEY_ARROW_UP,
    KEY_ARROW_LEFT,
    KEY_ARROW_RIGHT,
    KEY_KEYPAD_DIVIDE,
    KEY_KEYPAD_MULTIPLY,
    KEY_KEYPAD_SUBTRACT,
    KEY_KEYPAD_ADD,
    KEY_KEYPAD_ENTER,
    KEY_KEYPAD_PERIOD,
    KEY_KEYPAD_0,
    KEY_KEYPAD_1,
    KEY_KEYPAD_2,
    KEY_KEYPAD_3,
    KEY_KEYPAD_4,
    KEY_KEYPAD_5,
    KEY_KEYPAD_6,
    KEY_KEYPAD_7,
    KEY_KEYPAD_8,
    KEY_KEYPAD_9,
    KEY_BACKSPACE,
    KEY_TAB,
    KEY_DELETE,
    KEY_ESCAPE,
    KEY_LAST = KEY_ESCAPE,
};
```

5.4.1.5 enum tinyWindowKeyState_t [strong]

Enumerator

UP The key is currently up

DOWN The key is currently down

BAD If get key state fails (could not name it ERROR)

Definition at line 37 of file TinyWindow.h.

```
{
    UP = 0,
    DOWN,
    BAD = -1,
};
```

5.4.1.6 enum tinyWindowMouseButton_t [strong]

Enumerator

LEFT	The left mouse button
RIGHT	The right mouse button
MIDDLE	The middle mouse button / ScrollWheel
LAST	The last mouse button to be supported

Definition at line 112 of file TinyWindow.h.

```
{
    LEFT = 0,
    RIGHT,
    MIDDLE,
    LAST,
};
```

5.4.1.7 enum tinyWindowMouseScroll_t [strong]

Enumerator

DOWN	The mouse wheel up
UP	The mouse wheel down

Definition at line 120 of file TinyWindow.h.

```
{
    DOWN = 0,
    UP
};
```

5.4.1.8 enum tinyWindowState_t [strong]

Enumerator

NORMAL	The window is in its default state
MAXIMIZED	The window is currently maximized
MINIMIZED	The window is currently minimized
FULLSCREEN	The window is currently full screen

Definition at line 133 of file TinyWindow.h.

```
{
    NORMAL = 0,
    MAXIMIZED,
    MINIMIZED,
    FULLSCREEN,
};
```

5.4.1.9 enum tinyWindowStyle_t [strong]

Enumerator

BARE The window has no decorators but the window border and title bar

DEFAULT The default window style for the respective platform

POPUP The window has no decorators

Definition at line 126 of file TinyWindow.h.

```
{
    BARE = 1,
    DEFAULT,
    POPUP,
};
```

5.4.2 Function Documentation

5.4.2.1 static void TinyWindow_PrintErrorMessage (const tinyWindowError_t *errorNumber*) [static]

Print out the error associated with the given error number

Definition at line 210 of file TinyWindow.h.

References ALREADY_INITIALIZED, EXISTING_CONTEXT, FUNCTION_NOT_IMPLEMENTED, INVALID_CALLBACK, INVALID_CONTEXT, INVALID_ICON_PATH, INVALID_RESOLUTION, INVALID_TITLEBAR, INVALID_WINDOW_INDEX, INVALID_WINDOW_NAME, INVALID_WINDOW_STATE, INVALID_WINDOWSTYLE, LINUX_CANNOT_CONNECT_X_SERVER, LINUX_CANNOT_CREATE_WINDOW, LINUX_FUNCTION_NOT_IMPLEMENTED, LINUX_INVALID_VISUALINFO, NOT_INITIALIZED, WINDOW_NOT_FOUND, WINDOWS_CANNOT_CREATE_WINDOW, and WINDOWS_FUNCTION_NOT_IMPLEMENTED.

Referenced by windowManager::AddWindow(), windowManager::DisableWindowDecoratorByIndex(), windowManager::DisableWindowDecoratorByName(), windowManager::DoesExistByIndex(), windowManager::DoesExistByName(), windowManager::EnableWindowDecoratorsByIndex(), windowManager::EnableWindowDecoratorsByName(), windowManager::FocusWindowByIndex(), windowManager::FocusWindowByName(), windowManager::GetMousePositionInScreen(), windowManager::GetMousePositionInWindowByIndex(), windowManager::GetMousePositionInWindowByName(), windowManager::GetNumWindows(), windowManager::GetScreenResolution(), windowManager::GetWindowIndexByName(), windowManager::GetWindowIsFullScreenByIndex(), windowManager::GetWindowIsFullScreenByName(), windowManager::GetWindowIsInFocusByIndex(), windowManager::GetWindowIsInFocusByName(), windowManager::GetWindowIsMaximizedByIndex(), windowManager::GetWindowIsMaximizedByName(), windowManager::GetWindowIsMinimizedByIndex(), windowManager::GetWindowIsMinimizedByName(), windowManager::GetWindowNameByIndex(), windowManager::GetWindowPositionByIndex(), windowManager::GetWindowPositionByName(), windowManager::GetWindowResolutionByIndex(), windowManager::GetWindowResolutionByName(), windowManager::GetWindowShouldCloseByIndex(), windowManager::GetWindowShouldCloseByName(), windowManager::Initialize(), windowManager::MakeWindowCurrentContextByIndex(), windowManager::MakeWindowCurrentContextByName(), windowManager::MaximizeWindowByIndex(), windowManager::MaximizeWindowByName(), windowManager::MinimizeWindowByIndex(), windowManager::MinimizeWindowByName(), windowManager::Platform_InitializeGL(), windowManager::Platform_SetWindowStyle(), windowManager::PollForEvents(), windowManager::RemoveWindowByIndex(), windowManager::RemoveWindowByName(), windowManager::RestoreWindowByIndex(), windowManager::RestoreWindowByName(), windowManager::SetFullScreenByIndex(), windowManager::SetFullScreenByName(), windowManager::SetMousePositionInScreen(), windowManager::SetMousePositionInWindowByIndex(), windowManager::SetMousePositionInWindowByName(), windowManager::SetWindowIconByIndex(), windowManager::SetWindowIconByName(), windowManager::SetWindowOnDestroyedByIndex(), windowManager::SetWindowOnDestroyedByName(), windowManager::SetWindowOnFocusByIndex(), windowManager::SetWindowOnFocusByName(), windowManager::SetWindowOnKeyEventByIndex(), windowManager::SetWindowOnKeyEventByName(), windowManager::SetWindowOnMaximizedByIndex(), windowManager::SetWindowOnMaximizedByName(), windowManager::SetWindowOnMinimizedByIndex(), windowManager::SetWindowOnMinimizedByName(), windowManager::SetWindowOnMouseButtonEventByIndex(), windowManager::SetWindowOnMouseButtonEventByName(), windowManager::SetWindowOnMouseMoveByIndex(), windowManager::SetWindowOnMouseMoveByName(), windowManager::SetWindowOnMouseWheelEventByIndex(), windowManager::SetWindowOnMouseWheelEventByName(), windowManager::SetWindowOnMovedByIndex(), windowManager::SetWindowOnMovedByName(), windowManager::SetWindowOnResizeByIndex(), windowManager::SetWindowOnResizeByName(), windowManager::SetWindowPositionByIndex(), windowManager::SetWindowPositionByName(), windowManager::SetWindowResolutionByIndex(), windowManager::SetWindowResolutionByName(), windowManager::SetWindowStyleByIndex(), windowManager::SetWindowStyleByName(), windowManager::SetWindowTitleBarByIndex(), windowManager::SetWindowTitleBarByName(), windowManager::ShutDown(), windowManager::WaitForEvents(), windowManager::WindowGetKeyByIndex(), windowManager::WindowGetKeyByName(), windowManager::WindowSwapBuffersByIndex(), and windowManager::WindowSwapBuffersByName().

```
{
    switch ( errorNumber )
    {
        case tinyWindowError_t::INVALID_WINDOW_NAME:
        {
            printf( "Error: invalid window name \n" );
            break;
        }

        case tinyWindowError_t::INVALID_ICON_PATH:
        {
            printf( "Error: invalid icon path \n" );
            break;
        }

        case tinyWindowError_t::INVALID_WINDOW_INDEX:
        {
            printf( "Error: invalid window index \n" );
            break;
        }

        case tinyWindowError_t::INVALID_WINDOW_STATE:
        {
            printf( "Error: invalid window state \n" );
            break;
        }

        case tinyWindowError_t::INVALID_RESOLUTION:
        {
            printf( "Error: invalid resolution \n" );
            break;
        }

        case tinyWindowError_t::INVALID_CONTEXT:
        {
            printf( "Error: Failed to create OpenGL context \n" );
            break;
        }

        case tinyWindowError_t::EXISTING_CONTEXT:
        {
            printf( "Error: context already created \n" );
            break;
        }

        case tinyWindowError_t::NOT_INITIALIZED:
        {
            printf( "Error: Window manager not initialized \n" );
            break;
        }

        case tinyWindowError_t::ALREADY_INITIALIZED:
        {
            printf( "Error: window has already been initialized \n" );
            break;
        }

        case tinyWindowError_t::INVALID_TITLEBAR:
        {
            printf( "Error: invalid title bar name ( cannot be null or nullptr ) \n" );
            break;
        }

        case tinyWindowError_t::INVALID_CALLBACK:
        {
            printf( "Error: invalid event callback given \n" );
            break;
        }

        case tinyWindowError_t::WINDOW_NOT_FOUND:
        {
            printf( "Error: window was not found \n" );
            break;
        }

        case tinyWindowError_t::INVALID_WINDOWSTYLE:
        {
            printf( "Error: invalid window style given \n" );
            break;
        }

        case tinyWindowError_t::FUNCTION_NOT_IMPLEMENTED:
        {
            printf( "Error: I'm sorry but this function has not been implemented yet :( \n" );
            break;
        }

        case tinyWindowError_t::LINUX_CANNOT_CONNECT_X_SERVER:
        {
            printf( "Error: cannot connect to X server \n" );
            break;
        }

        case tinyWindowError_t::LINUX_INVALID_VISUALINFO:
        {
            printf( "Error: Invalid visual information given \n" );
            break;
        }

        case tinyWindowError_t::LINUX_CANNOT_CREATE_WINDOW:
        {
            printf( "Error: failed to create window \n" );
            break;
        }

        case tinyWindowError_t::LINUX_FUNCTION_NOT_IMPLEMENTED:
        {
            printf( "Error: function not implemented on linux platform yet. sorry :( \n" );
            break;
        }

        case tinyWindowError_t::WINDOWS_CANNOT_CREATE_WINDOW:
        {
            printf( "Error: failed to create window \n" );
            break;
        }

        case tinyWindowError_t::WINDOWS_FUNCTION_NOT_IMPLEMENTED:
        {
            printf( "Error: function not implemented on Windows platform yet. sorry ;( \n" );
            break;
        }

        default:
        {
            printf( "Error: unspecified Error \n" );
            break;
        }
    }
}
```

5.4.3 Variable Documentation

5.4.3.1 const int DEFAULT_WINDOW_HEIGHT = 720

Definition at line 35 of file TinyWindow.h.

5.4.3.2 _WIN32 _WIN64 const int DEFAULT_WINDOW_WIDTH = 1280

Definition at line 34 of file TinyWindow.h.

5.4.3.3 const int LINUX_DECORATOR = 2

Definition at line 179 of file TinyWindow.h.

5.4.3.4 const int LINUX_FUNCTION = 1

Definition at line 178 of file TinyWindow.h.

5.5 README.md File Reference

Index