optimacro

Generated by Doxygen 1.12.0

Chapter 1

optimacro

Aplikacja do automatyzacji zadań i tworzenia makr na Linuksie (X11)

CELE: • Możliwość tworzenia skryptów oraz skrótów klawiszowych do automatyzacji zdarzeń w LINUKSIE • Ma na celu optymalizacje zdarzeń w linuxie / automatyzacje

Przykłady: • Przykłady np. ruszanie myszka żeby cie nie wylogowało • Zautomatyzowane wykonywanie powtarzalnych/żmudnych zadań • Planowanie wykonania zadań

Aplikacja: • UI do zarzadzania tym • Tworzeni kodu z bloków sracz tak żeby • Apliakcja na smarftona by odpalać skrypty przez telefon

Technologie: • Elektron (klient) • C++/Go/Python (serwer) • Serwer – klient

Skrypt który pobiera pogodę i daje tapetę np.

Możliwość Wysokopoziomowych i niskopoziomowych backend

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

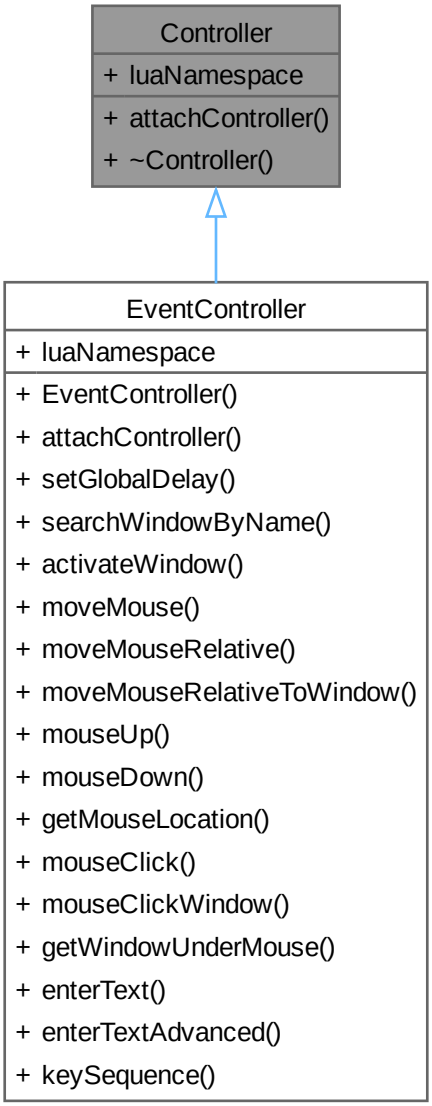
Class Documentation

5.1 Controller Class Reference

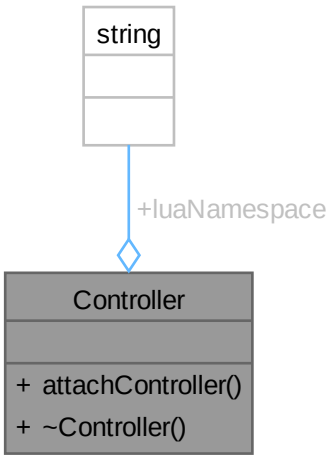
Base class for controllers.

`#include <Controller.hpp>`

Inheritance diagram for Controller:



Collaboration diagram for Controller:



Public Member Functions

- virtual void attachController (sol::state &lua)

Public Attributes

- std::string luaNamespace

5.1.1 Detailed Description

Base class for controllers.

5.1.2 Member Function Documentation

5.1.2.1 attachController()

```
virtual void Controller::attachController (
            sol::state & lua)  [inline], [virtual]
```

Method for attaching controller to lua state.

Parameters

lua	Lua state to attach to.

Reimplemented in EventController.

5.1.3 Member Data Documentation

5.1.3.1 luaNamespace

```
std::string Controller::luaNamespace
```

Name of lua namespace to use. This will be used as the base class name in lua scripts

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

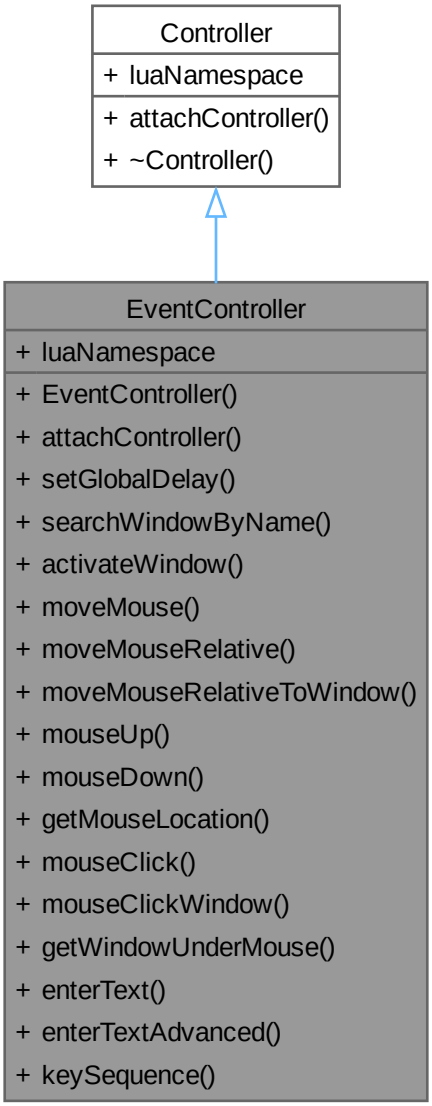
- src/Controller.hpp

5.2 EventController Class Reference

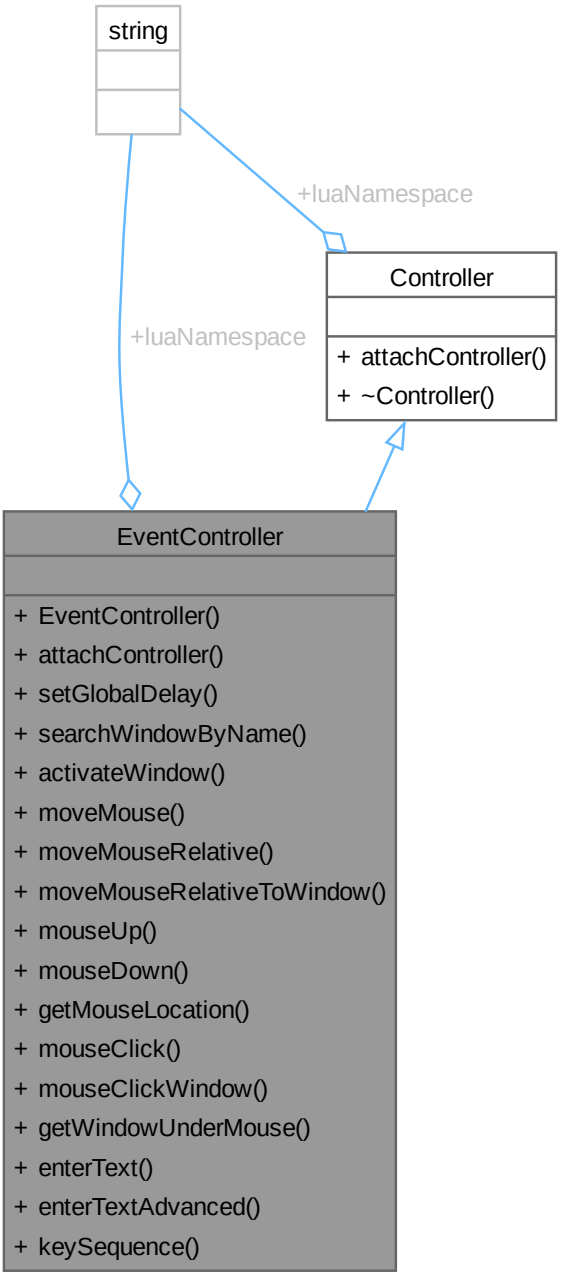
Class for handling X11 events.

```
#include <EventController.hpp>
```

Inheritance diagram for EventController:



Collaboration diagram for EventController:



Public Member Functions

- void attachController (sol::state &lua)
- void **setGlobalDelay** (int newDelay)
- Window **searchWindowByName** (std::string name)
- void **activateWindow** (Window window)
- void moveMouse (int x, int y)

- void moveMouseRelative (int x, int y)
- void moveMouseRelativeToWindow (Window window, int x, int y)
- void mouseUp (Window window, int button)
- void mouseDown (Window window, int button)
- std::tuple< int, int > getMouseLocation ()
- void mouseClick (int button)
- void mouseClickWindow (Window window, int button)
- Window getWindowUnderMouse ()
- void enterText (std::string text)
- void enterTextAdvanced (std::string text, Window window, uint delay)
- void keySequence (std::string sequence)

Public Member Functions inherited from Controller

Public Attributes

- std::string **luaNamespace** = "event"

Public Attributes inherited from Controller

- std::string luaNamespace

5.2.1 Detailed Description

Class for handling X11 events.

This class provides functions for sending events to X11. The class is exposed in lua as "event" namespace. Bindings are defined in attachController method.

5.2.2 Member Function Documentation

5.2.2.1 attachController()

```
void EventController::attachController (
            sol::state & lua)  [virtual]
```

Method for attaching controller to lua state.

Parameters

lua	Lua state to attach to.

Reimplemented from Controller.

5.2.2.2 enterText()

```
void EventController::enterText (
            std::string text)
```

Type a string to the current window.

Parameters

text	The string to type, like "Hello world!"

5.2.2.3 enterTextAdvanced()

```
void EventController::enterTextAdvanced (
            std::string text,
            Window window,
            uint delay)
```

Type a string to the specified window with specified delay.

Parameters

text	The string to type, like "Hello world!"
window	The window you want to send keystrokes to or CURRENTWINDOW
delay	The delay between keystrokes in microseconds

5.2.2.4 getMouseLocation()

```
std::tuple< int, int > EventController::getMouseLocation ()
```

Get the current mouse location.

Returns

A tuple of X and Y coordinates.

5.2.2.5 getWindowUnderMouse()

```
Window EventController::getWindowUnderMouse ()
```

Get the window the mouse is currently over

Returns

Selected window

5.2.2.6 keySequence()

```
void EventController::keySequence (
            std::string sequence)
```

Send a keysequence to the current window.

This allows you to send keysequences by symbol name. Any combination of X11 KeySym names separated by '+' are valid. Single KeySym names are valid, too.

Examples: "l" "semicolon" "alt+Return" "Alt_L+Tab"

Parameters

sequence	The string keysequence to send.

5.2.2.7 mouseClick()

```
void EventController::mouseClick (
            int button)
```

Send a click for a specific mouse button at the current mouse location to the current window.

Parameters

button	The mouse button. Generally, 1 is left, 2 is middle, 3 is right, 4 is wheel up, 5 is wheel down.

5.2.2.8 mouseClickWindow()

```
void EventController::mouseClickWindow (
            Window window,
            int button)
```

Send a click for a specific mouse button at the current mouse location to a specific window.

Parameters

window	The window you want to send the event
button	The mouse button. Generally, 1 is left, 2 is middle, 3 is right, 4 is wheel up, 5 is wheel down.

5.2.2.9 mouseDown()

```
void EventController::mouseDown (
            Window window,
            int button)
```

Send a mouse press (aka mouse down) for a given button at the current mouse location.

Parameters

window	The window you want to send the event to
button	The mouse button. Generally, 1 is left, 2 is middle, 3 is right, 4 is wheel up, 5 is wheel down.

5.2.2.10 mouseUp()

```
void EventController::mouseUp (
            Window window,
            int button)
```

Send a mouse release (aka mouse up) for a given button at the current mouse location.

Parameters

window	The window you want to send the event to
button	The mouse button. Generally, 1 is left, 2 is middle, 3 is right, 4 is wheel up, 5 is wheel down.

5.2.2.11 moveMouse()

```
void EventController::moveMouse (
            int x,
            int y)
```

Move the mouse to a specific location.

Parameters

x	the target X coordinate on the screen in pixels.
y	the target Y coordinate on the screen in pixels.

5.2.2.12 moveMouseRelative()

```
void EventController::moveMouseRelative (
            int x,
            int y)
```

Move the mouse relative to it's current position.

Parameters

x	the distance in pixels to move on the X axis.
y	the distance in pixels to move on the Y axis.

5.2.2.13 moveMouseRelativeToWindow()

```
void EventController::moveMouseRelativeToWindow (
            Window window,
            int x,
            int y)
```

Move the mouse to a specific location relative to the top-left corner of a window.

Parameters

window	the target window.
x	the target X coordinate on the screen in pixels.
y	the target Y coordinate on the screen in pixels.

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

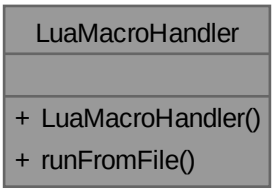
- src/EventController.hpp
- src/EventController.cpp

5.3 LuaMacroHandler Class Reference

Class for handling lua macros.

`#include <LuaMacroHandler.hpp>`

Collaboration diagram for LuaMacroHandler:



Public Member Functions

- void runFromFile (std::string name)

5.3.1 Detailed Description

Class for handling lua macros.

This class provides functions for running lua macros. It is also responsible for creating and attaching controllers for various functionality.

5.3.2 Member Function Documentation

5.3.2.1 runFromFile()

```
void LuaMacroHandler::runFromFile (
            std::string name)
```

Runs lua macro from file.

Parameters

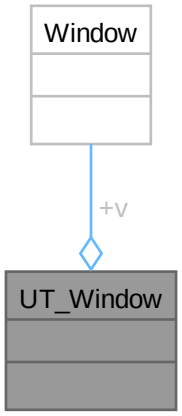
name	Name of file to run.

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

- src/LuaMacroHandler.hpp
- src/LuaMacroHandler.cpp

5.4 UT_Window Struct Reference

Collaboration diagram for UT_Window:



Public Attributes

- Window **v**

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

- src/EventController.hpp

Chapter 6

File Documentation

6.1 Controller.hpp

```
00001 #pragma once
00002
00003 #include <sol/sol.hpp>
00007 class Controller {
00008 public:
00013   std::string luaNamespace;
00019   virtual void attachController(sol::state &lua) {};
00020   virtual ~Controller(){};
00021 };
```

6.2 EventController.hpp

```
00001 #pragma once
00002 #include "Controller.hpp"
00003 #include <X11/X.h>
00004 #include <string>
00005 #include <tuple>
00006 #include <xdo.h>
00007
00008 struct UT_Window {
00009   Window v;
00010 };
00018 class EventController : public Controller {
00019   xdo_t *instance;
00020   int defaultDelay;
00021
00022 public:
00023   std::string luaNamespace = "event";
00024   EventController();
00025   void attachController(sol::state &lua);
00026   void setGlobalDelay(int newDelay);
00027   Window searchWindowByName(std::string name);
00028   void activateWindow(Window window);
00029   // MOUSE EVENTS
00036   void moveMouse(int x, int y);
00043   void moveMouseRelative(int x, int y);
00052   void moveMouseRelativeToWindow(Window window, int x, int y);
00061   void mouseUp(Window window, int button);
00070   void mouseDown(Window window, int button);
00076   std::tuple<int, int> getMouseLocation();
00084   void mouseClick(int button);
00093   void mouseClickWindow(Window window, int button);
00099   Window getWindowUnderMouse();
00101   // KEYBOARD EVENTS
00102
00109   void enterText(std::string text);
00117   void enterTextAdvanced(std::string text, Window window, uint delay);
00133   void keySequence(std::string sequence);
00135 };
```

6.3 LuaMacroHandler.hpp

```
00001 #pragma once
00002 #include "Controller.hpp"
00003 #include "EventController.hpp"
00004
00012 class LuaMacroHandler {
00013   sol::state lua;
00014   std::vector<Controller *>
00015       controllers; // TODO smartpointery/destruktor, bo beda
00016                    // leaki w uj jak tego nie zwolnimy, a sie samo nie zwolni
00017   void attachAllControllers();
00018
00019 public:
00020   LuaMacroHandler();
00026   void runFromFile(std::string name);
00027 };
```

Index