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

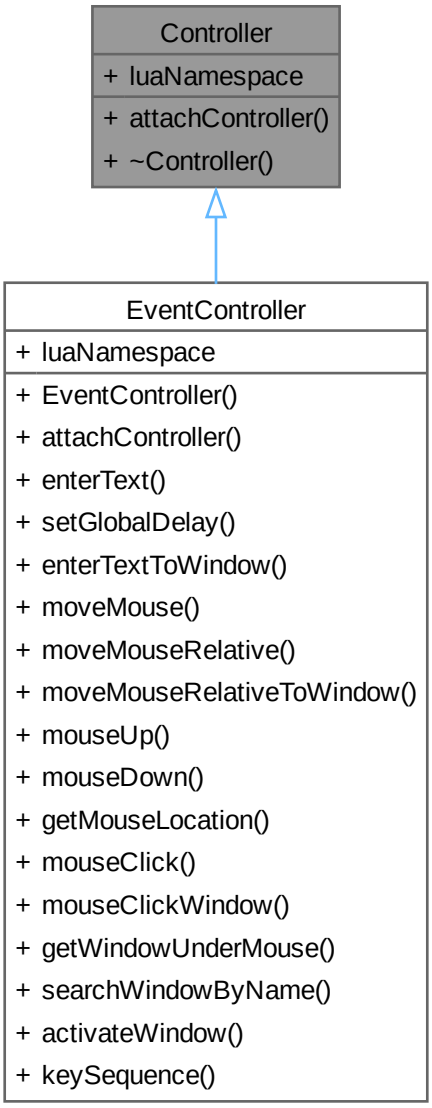
Here is a list of all documented files with brief descriptions:

Chapter 5

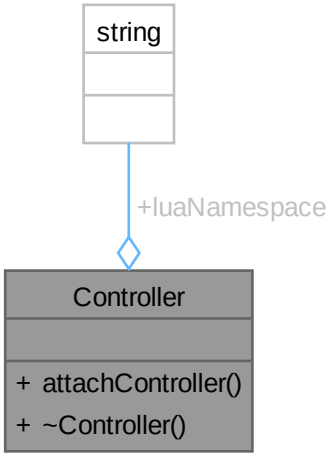
Class Documentation

5.1 Controller Class Reference

Inheritance diagram for Controller:



Collaboration diagram for Controller:



Public Member Functions

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

Public Attributes

- std::string **luaNamespace**

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

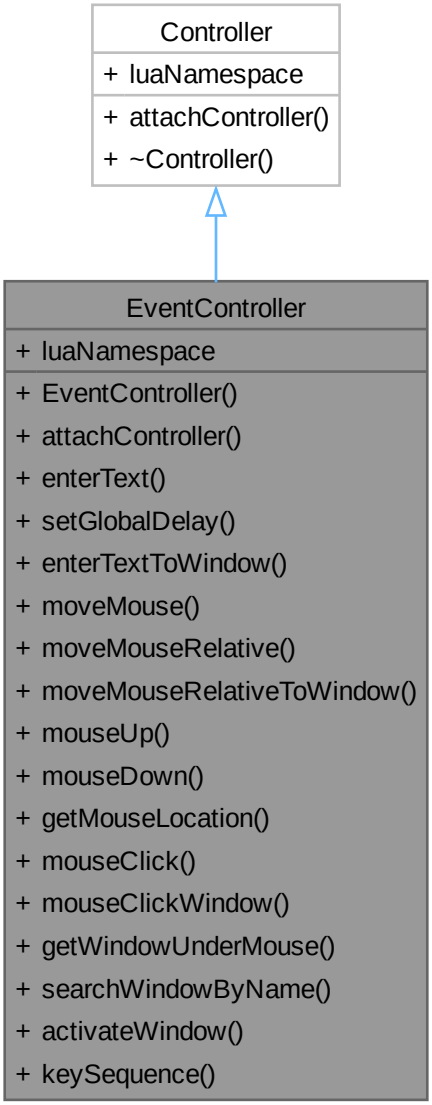
- src/Controller.hpp

5.2 EventController Class Reference

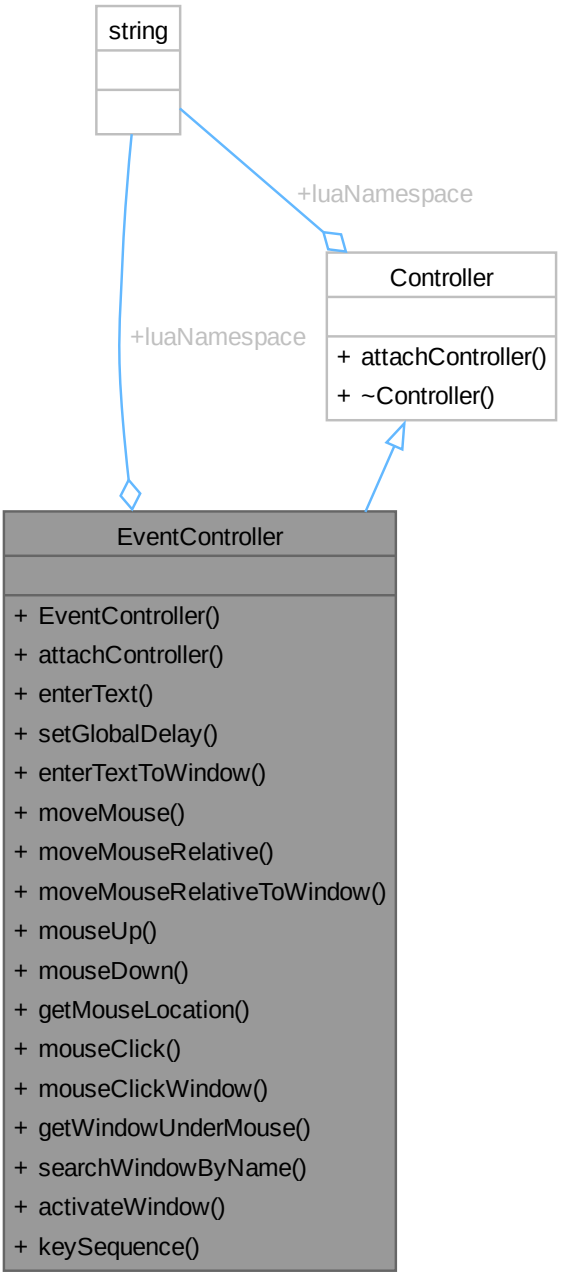
Class for handling X11 events.

`#include <EventController.hpp>`

Inheritance diagram for EventController:



Collaboration diagram for EventController:



Public Member Functions

- void attachController (sol::state &lua)
- void **enterText** (std::string text)
- void **setGlobalDelay** (int newDelay)
- void **enterTextToWindow** (std::string text, 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 ()
- Window **searchWindowByName** (std::string name)
- void **activateWindow** (Window window)
- 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]
```

Reimplemented from Controller.

5.2.2.2 getMouseLocation()

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

Get the current mouse location.

Returns

A tuple of X and Y coordinates.

5.2.2.3 getWindowUnderMouse()

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

Get the window the mouse is currently over

Returns

Selected window

5.2.2.4 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.

5.2.2.5 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.

5.2.2.6 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.

5.2.2.7 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.

5.2.2.8 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.

5.2.2.9 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.

5.2.2.10 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

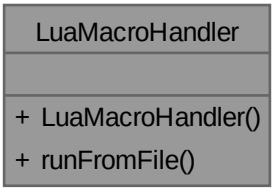
window	the target window.
x	the target X coordinate on the screen in pixels.
y	the target Y coordinate on the screen in pixels.

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

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

5.3 LuaMacroHandler Class Reference

Collaboration diagram for LuaMacroHandler:



Public Member Functions

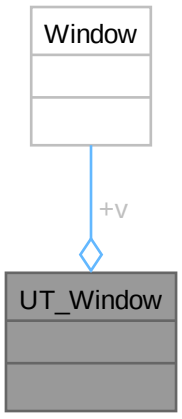
- void **runFromFile** (std::string name)

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>
00004 class Controller {
00005 public:
00006   std::string luaNamespace;
00007   virtual void attachController(sol::state &lua) {};
00008   virtual ~Controller(){};
00009 };
```

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 delay;
00021
00022 public:
00023   std::string luaNamespace = "event";
00024   EventController();
00025   void attachController(sol::state &lua);
00026   void enterText(std::string text);
00027   void setGlobalDelay(int newDelay);
00028   void enterTextToWindow(std::string text, 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   Window searchWindowByName(std::string name);
00102   void activateWindow(Window window);
00103   void keySequence(std::string sequence);
00104 };
```

6.3 LuaMacroHandler.hpp

```
00001 #pragma once
00002 #include "Controller.hpp"
00003 #include "EventController.hpp"
00004 class LuaMacroHandler {
00005   sol::state lua;
00006   std::vector<Controller *>
00007       controllers; // TODO smartpointery/destruktor, bo beda
00008                    // leaki w uj jak tego nie zwolnimy, a sie samo nie zwolni
00009   void attachAllControllers();
00010
00011 public:
00012   LuaMacroHandler();
00013   void runFromFile(std::string name);
00014 };
```

Index