Mufasa Client Class

Merlijn Wajer

October 6, 2009

1 Introduction

This is the official Mufasa Documentation. The main purpose of this document is to provide a clear view on Mufasa's architecture.

2 What is Mufasa?

Mufasa is a project that aims to create two things, a GUI to create scripts, and the Mufasa Macro Library. The Mufasa Macro Library (MML) will provide one with a way to control the mouse and keyboard, open files, open web pages, and capturing and analyzing images. The GUI will use MML for most of it's features. Each script thread will also use a Client class.

2.1 Simple feature overview

Mufasa is:

- Object Oriented. This means the code is generally more readable, and easier to maintain.
- Free Software, as in, Free.[1]
- MOAR

3 Important Classes in MML

3.1 The Client Class

The Client Class is the main Class, and is created to be able to run seperately from the User Interface, thus being thread safe. The Client class is mainly designed to be a container for other classes.

[1] License here

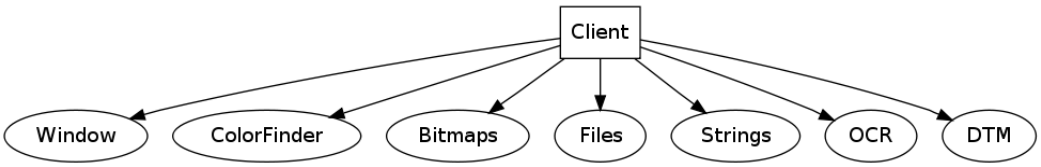

Figure 1: Classes that the Client contains.

3.2 The Window Class

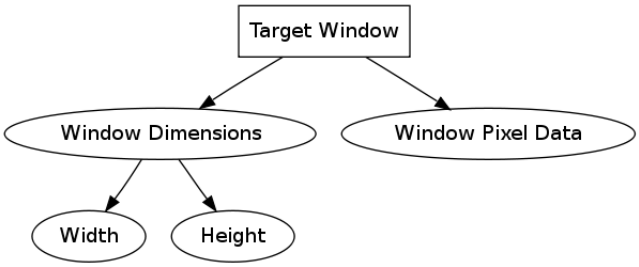

Figure 2: The structure of the Window class

The window class manages the core functionality for retreiving Window data, such as the actual pixels, position, or dimensions.

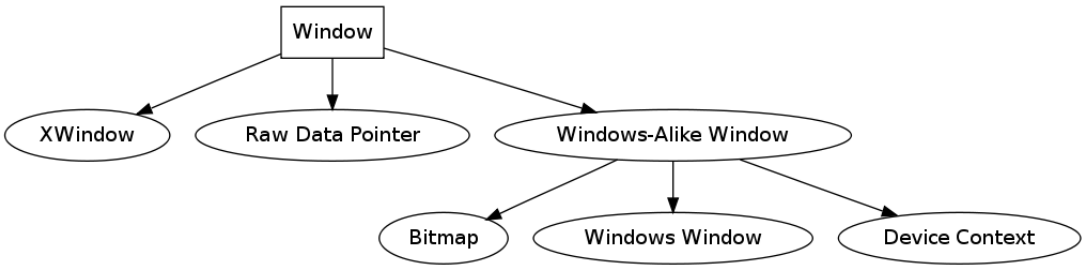

Figure 3: Different types of Windows. Note that XWindow is only for non-Windows systems.

Figure 3 shows the three different Window Types supported by Mufasa.
Quick overview of functions:

- ReturnData
- FreeReturnedData
- GetDimensions
- SetTargetWindow
- SetTargetIntArray
- SetTargetXWindow

- GetPixel

Together, these functions form the core of the window management, except for input.

3.3 The Input Class

The Input Class is the class that takes care of all the input.

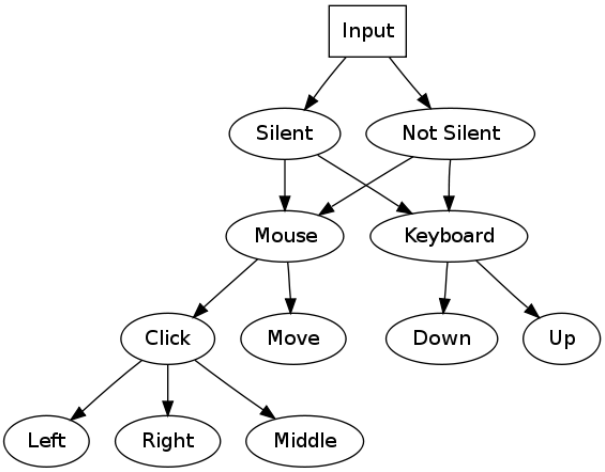

Figure 4: Input Functionality.

As one can see in Figure 4, MML aims to support both Silent and non Silent Input. Since the Input heavily differs per operating system, the Input class should have a general way of sending keys, possibly at the expense of losing some functionality.

3.4 The Color Convertions Include

This .inc file contains pascal code to quickly convert one colour type to another. It also adds support for comparing colours. The reason this is not a class, is because constantly dereferencing a class to call a single function doesn't do the speed of a program any good. There also wasn't really a need for a class, since none of these functions need to be initialized in any way.

3.5 The Colour Class

The colour class is a Class that does all the colour identfying and locating work. (FindColor, for example) The colour class uses the Convertions include for several of it's functions.

A FindColor-derivative function in Mufasa exists generally out of the following steps:

- Retrieve Client Data.
- Loop over the data, possibly with a special algorithm.

- Check the current pixel data against another colour, possibly with tolerance.
- Free the Client Data.
- Return found point(s).

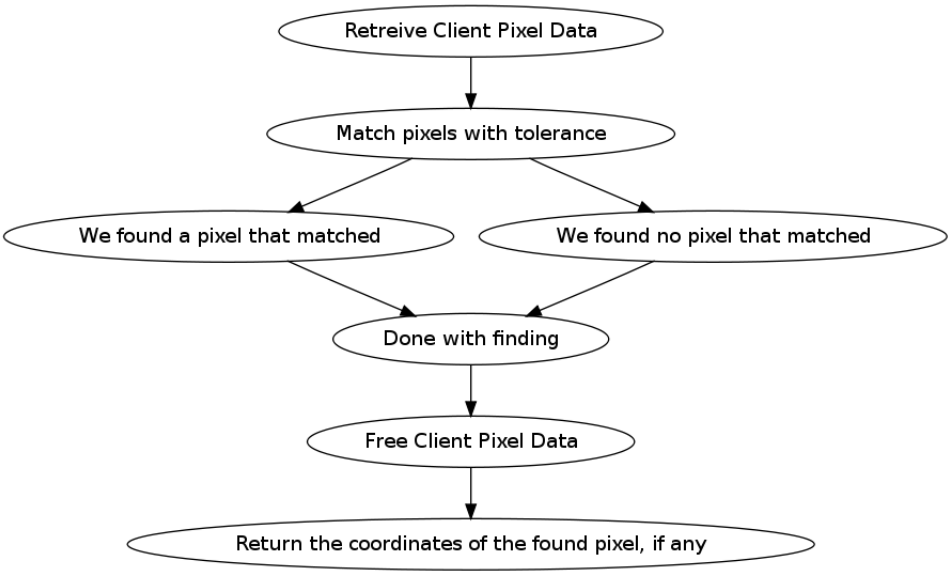

Figure 5: A basic find colour.

3.6 Notes on the previously mentioned classes

At this point it is unsure if the Input class will also take care of Sockets and Files. Most likely these will get their own class.

4 More On The Core Classes

The previously mentioned MML classes are considered to be the absolute core of the library. (Although one could argue that even the Colour class isn't part of the core classes.)

With these classes most functions that Mufasa will contain can be created. if you can make FindColor, you can make FindColorsSpiralTolerance, they don't really differ a lot. The same goes for DTM's, OCR and Bitmaps. Mouse and keyboard functions will be done with the Input class.

The MML contains more classes, and they will mainly utilize the previous mentioned classes. It is essential to understand the Classes architecture to fully understand Mufasa. Before work on other classes will be done, the core classes must be finished and stable.

A good rule of thumb is the following: any units that make extensive use of Compiler Directives, are considered a core unit.