THE WHIM WINDOW MANAGER

Steve Redler IV, SCSE
SR Technology New Jersey U.S.A

George Peter Staplin

Abstract

Whim is a window manager written for the X11 graphical windowing system that has its primary logic and functionality coded in Tcl/Tk. It uses two newly developed c extensions to give Tcl/Tk the fine control needed to properly implement a window manager, with Whim being not just a proof of concept, but instead a full function, comfortable and intuitive interface to X11 applications. The technical design goals of Whim include: being deployed as a Starkit, and to optionally use the tile extension for enhanced window titlebar theming. Most importantly, Whim enables full customization at the scripting level, eliminating the need to work with the X11 c api directly.

1 Introduction

One of the many enjoyable aspects of using a Linux/BSD Operating System is having the ability to fully customize the look and feel of virtually every aspect of the OS. The window manager is the most visible application running on a computer workstation, and is also one of the more difficult pieces of software to write from scratch. Usually one would just pick a window manager that has enough customizable features to satisfy ones requirements. All window managers are compiled applications, typically written in the c language. No one has ever written a full feature window manager in a scripting language because scripting languages lack an interface to all the apis of the X Window System.

The idea to create a window manager where the core logic and functionality was coded in a scripting language is not new, and several attempts were made at developing a wm around Tcl/Tk. Unfortunately stock Tk only has a few interfaces into X Windows, so an extension must be written to add the missing interfaces. Whim uses a small extension called “pwm”, written by George Peter Staplin, that provides necessary Xlib and Tk C function access to the X Windows System. The majority of the Tcl/Tk code that comprises the bulk of Whim was also written by George Staplin. Steve Redler focused on external application integration, testing and design goals, mainly trying to keep the Tcl code as simple as possible, so that virtually anyone could understand how to make their own modifications.

2 Design Goals

There were many goals that were laid out for the design of Whim. All of them have been reached. They include:

Rely on a simple and minimalistic c extension.
Utilize stock Tcl/Tk or Tclkit
Be deployable as a cross platform Starkit
Run on handheld Linux base computers
Use Tcl for all the logic
Keep the logic as simple as possible
A cool name (how else would you pronounce wm?)

3 The pwm extension

The pwm extension contains around 100 functions that allow a Tcl/Tk script to interact with the X Windows System. Many are just simple front ends to Xlib functions. For example, pwm.destroy.window is a frontend to XdestroyWindow. Others include obvious functions like following:

pwm.create.window	pwm.move.window	pwm.send.event.message
pwm.resize.window	pwm.lower.window	pwm.raise.window
pwm.destroy.window	pwm.get.window.title	pwm.event.handler

Detailed explanation of each function and corresponding Xlib functions are outside the scope of this paper. Even though each function is relatively simple, the task of scripting them into a properly functioning window manager took great insight into the inner workings of the X Windows System. At version 494, Whim is a stable, functional and feature filled window manager. At this point, any further modifications can be done at the Tcl/Tk level. Whims Tcl core was designed specifically with global arrays to track state for simplicity instead of OO. Most of the events are X related, others are stock Tk. It was designed to be easy to tweak and extend.

4 Current Feature Set

Support for most ICCCM hints
Support for the important MWM hints
Support for shaped windows
Transient support
Scripting console
Tile support
Two focus models (click-to-focus and focus-follows-mouse)
Optional "Position Transients over Masters" behavior
Optional "Position Windows Under Pointer" behavior
Desktop applets (per-desktop)
Move window to another desktop via a titlebar menu
Multilayer applets make it possible to run multiple applets per-desktop
Auto start X if running from a console
Focus follows mouse
Click to focus
Move window to another desktop via a titlebar menu
Take screen shot command in titlebar menus
Innovative SRIV window resizing
Resize window via left click/drag on window border
Window titlebars and frames
Maximize/restore/roll-up buttons in titlebar
Interaction behavior configurable via the desktop menu
Each virtual desktop is a canvas
Key binding support
A configurable cascading right-click menu
Several IPC methods to interact with applets and applications
Each window container is a frame that can have other widgets embedded
It's easily customizable via Tcl and the Behavior menu
It's BSD licensed
It doesn't require any patches to Tcl/Tk

4.1 Whims Canvas Desktop

Whim supports multiple desktops. Each desktop is a canvas, so you can draw many different types of things, easily! You can extend your interface as you like! It is possible to create widgets that are integrated into your desktop, for macros, or common tasks. Some examples of desktop integration are in the external applets that have been created. They showcase how any type of textual or graphical data can be programmatically placed onto a desktop.

Whim uses a sorted list of window ids for each desktop. So if you create windows A B C, and the focus order is A B C, and C is destroyed, then B gets focus, and so on. The list is resorted as focus patterns change, so the focus order should be correct in all cases.

4.2 New Features

Whim has two new features: It will start X if you run it from a console. So now you can try Whim without messing up your existing window manager configuration. Also, it will copy out default configuration files into your home directory if they don't exist so you have something to start with. Also, If you run Whim from inside X WITH another window manager running, it will re-launch itself inside Xnest automatically, so you can try it from inside your existing X setup.

4.3 Interprocess Communication

Whim includes two pure Tcl IPC systems, arraysync and netcall. Both were developed at the same time, one by each author, with some similar and some different goals. Arraysync has been use more extensively in a number of applets and applications to provide a method to pass data and commands between themselves and Whim.

4.3.1 Arraysync Overview

Arraysync, as its name implies, allows multiple applications to share one or more Tcl global arrays via tcp socket. It can be configured for a traditional client-server arrangement, where one application, such as Whim, is the server, and all X Windows apps and Whim applets are the clients. Arrays can be shared between a client and server only, or amongst all clients and the server at once. A second mode, which operates in a peer to peer fashion can also be configured. Basically, the first instance of arraysync becomes the server, the following instances become clients. If the server is killed, the clients all race to become the new server, and since only one can win due to opening a server socket first, all the rest of the clients connect to the new server instead. In peer to peer mode, the server routes commands and replies between multiple clients, so that each client does not have to explicitly connect to each other.

Within Whim, arraysync is loaded in server mode. Client apps can connect to Whim, pass it commands, connect to shared arrays to receive notification of special events, like a theme change, or that Whim was restarted. Many more events and data points could be exposed in the future. A taskbar applet is an example of a client application that queries Whim for state information about each window so that it can update its interface. That interface is sent to Whim to be rendered, by sending the Tcl/Tk GUI code through a sequence of arraysync commands. The taskbar also shares the main window status global array with whim. By setting traces on this array, the taskbar applet can be alerted to any window event, such as a window being created or destroyed, minimized or maximized, etc.

5 Configuration

Whim comes with the ability to configure its behavior on the fly without editing configuration files. There is a Behavior desktop menu, and an interactive console.

At the moment the configuration is stored in several scripts that are evaluated and stored in ::env(HOME). ~/.whim_config is for user overrides, and custom scripts. You don't have to add anything to it to use Whim, but you might enjoy some of Whim more if you do customize it. ~/.whim_behavior is an automatically saved file that contains the current configuration. Whenever you exit or restart Whim the current configuration is dumped into that file. Starting with release 308.13 it now contains a version header for automatic upgrade compatibility.

5.1 Typical configurations

In order to start an applet on a particular desktop when Whim starts, you could add a line to ~/.whim_config using a pattern like this:

```
on startup start.applet taskbar .desktop0
```

To start an applet on every desktop you can use the for.every.desktop iterator in ~/.whim_config. For example:

```
for.every.desktop desktop {
 on startup start.applet taskbar $desktop
}
```

If you wanted to change the desktop mouse button bindings, in your ~/.whim_config or at the Whim console you can use set.config desktop_bindings, like so:

```
set.config desktop_bindings {<Double-ButtonPress-1> {nullexec xterm} \
  <ButtonPress-3> {display.desktop.menu %desktop %X %Y}
```

The command nullexec is a frontend to Tcl's exec that handles channels in a slightly different way that tends to work better with applications like mplayer, and others. The %desktop key is substituted with the $::current_desktop at the time that the set.config occurs. set.config desktop_bindings iterates and applies this to every desktop automatically. %X and %Y are for the mouse cursor position. This is a dynamic configuration option, so you can change it at runtime and all desktop bindings will be updated.

If you run Whim with Tile, you may want to change the Tile theme. Press the Console button on the desktop controls. Enter the command "set.config tile_theme step" (without quotation marks). There are many other themes available including:

* default
* clam
* blue
* plastik
* srivlg
* classic
* sriv

Note: the default theme is sriv (by Steve Redler IV).

Changing the menus is accomplished by editing ~/.whim_menus. It is a file that is sourced to initialize the ::whim_menus list. A simple example from George's machine is:

```
set ::whim_menus {
 xterm xterm
 Firefox ~/linux_runtime/firefox/firefox
 xmix xmix
 Plan plan
}
```

A more advanced menu system is also available. It was created by Andrew Black and is available on the Whim web site.

If you would like change the number of available desktops, then in your ~/.whim_config you will need to use set.config desktops. For example:

```
set.config desktops {Work WWW Games}
```

6 Conclusion

Whim, by virtue of its design, may be one of the most versatile window managers available. Although it has configuration options like most window managers, it has unsurpassed reconfigurability because its core logic can be easily modified on any computer that has a text editor available. Whim wasn't made to be just a proof of concept. It is an easy to use window manager thats comfortable to use all day.