Tiling Window Manager for MINIX

Luis Pardo Antigua

Carleton University

Table of Contents

Section 1 – Introduction

1.1 – Context

At the start of the winter term (2018) the only type of operating systems I had ever used were OS that support desktop environments (DE). MINIX is mainly thought of as a command-line based operating system, so when I was first introduced to MINIX this term, I found it cumbersome to use.

Somewhere in between the command line and the desktop environment, there is the window manager.

For this project I made a tiling window manager for MINIX 3. I named it Luis's Window Manager or LWM for short.

This is a screenshot of MINIX running LWM:

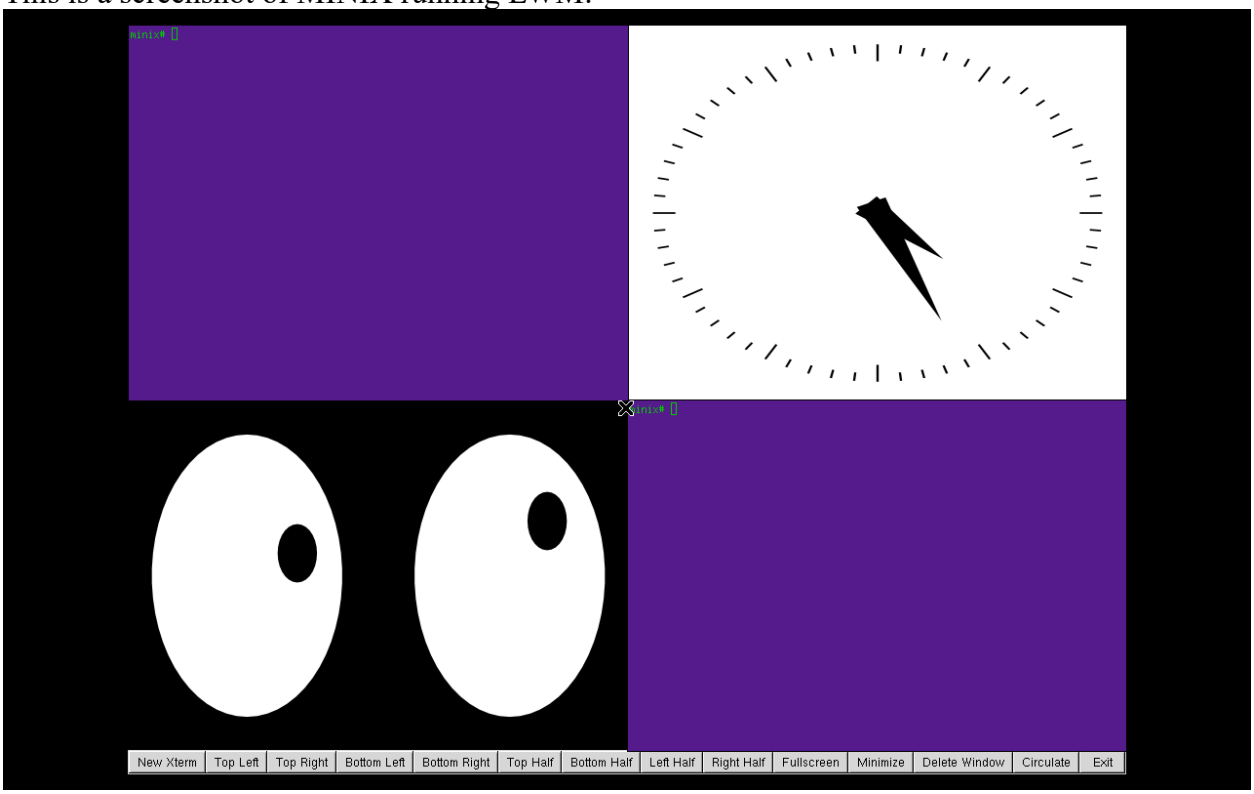

Figure 1: Use of LWM to tile four different windows into each corner of the screen

1.2 – Problem Statement

Without using the X Window System (X11 or simply X), a MINIX user only has access to one terminal by default. If a user wants to open more than one terminal without using X11, then methods such as using SSH to connect to MINIX from the host computer must be used (assuming MINIX is running as a guest virtual machine).

However, if the user wants to have more than one window or terminal natively running in MINIX, then the use of X11 is required.

To start running an X server, one must use the command “startx”.

To configure the programs that will be executed by default when startx is called, the file “/etc/X11/xinit/xinitrc” must be configured. Programs such as xterm (a terminal emulator) or xclock (a clock) can be setup to launch at X’s start-up.

Multiple windows can be opened without the need of a window manager, but to do this, inconvenient methods such as manually configuring xinitrc are needed.

Once the user has configured xinitrc, if further changes are required, then the connection to the X server must be closed, xinitrc reconfigured and startx called again.

Alternatively, if the user would like to change the size or position of a currently running xterm then a binary package such as “xtermcontrol-3.3” must be installed and long commands such as “--geometry=WIDTHxHEIGHT+XOFF+YOFF” must be used.

To avoid all of this hassle, window managers can be used.

1.3 – Result

Unbeknownst to me until I started working on this project, MINIX comes with a very good window manager installed by default, Tom's Window Manager (TWM).

For this project I learned how to write my own window manager. In one hand, the window manager I created lacks some of the features TWM offers, but on the other hand, it provides features TWM doesn't support, such as window tiling, circulating between windows and a more aesthetically pleasing colour scheme for default xterm windows.

1.4 – Outline

The rest of this report is structured as follows:

Section 2, briefly explains some basic concepts needed to understand this project. The topics in this section include X11, Xlib and widget toolkits.

Section 3, describes in detail how LWM works and goes over a couple of cases that illustrate why it might be better to use MINIX with a window manager than without one.

Section 4, here I evaluate the results of my work and reflect on the efficiency of using LWM.

Section 5, conclusion and summary of the project. Highlight achievements of the project and explain how the project aligns with the course.

Section 2 – Background Information

This report refers to topics such as X11, Xlib (lower level library to write X11-related programs) and the GTK+ widget toolkit (higher level library for X11).

X11 was developed at Massachusetts Institute of Technology (MIT) and initially released in 1987. It provides a basic framework for user interfaces and it is common in UNIX-like operating systems.

Xlib is a library that contains functions to allow a client to interact with an X server. It has been around since the 1985. Compared to widget toolkits, Xlib can be thought of as a low-level library. XCB, released in 2001, is another library meant to replace Xlib, which in some ways is better and even lower-level than Xlib. From reading the guide “How X Window Managers Work, And How to Write One” by Chuan Ji, I got the impression it is better for a beginner like myself to write their first window manager using Xlib rather than the more complex XCB.

Nowadays most applications do not use Xlib directly, but instead they use higher level libraries known as widget toolkits.

Popular examples of such widget toolkits are GTK+ and Qt.

In this project I use a combination of GTK+ 1.2 and Xlib. GTK+ is used to create the menu bar, and Xlib is used to handle window operations such as tiling.

The current version of GTK+ is GTK+ 3, however MINIX only natively supports GTK+ 1.2, so that is the version I used for this project.

Section 3 – Result

In this section I go over what each button in the menu bar does and give examples of why it can be easier to use MINIX with a window manager than without.

Figure 2: LWM menu bar

The "New Xterm" button allows the user to create a new X terminal emulator. The function called when the aforementioned button is clicked, has a hard-coded command that creates a new xterm with a pre-set geometry with origin at position (0,0) in the screen, that is the top left corner, and colours meant to resemble those of Ubuntu's terminal.

When any of the buttons in charge of configuring windows is pressed, a ConfigureRequest event is sent to the server. The server in turn sends the request to the window manager, and ultimately the window manager decides whether to grant, change or ignore the request to configure the window. Figure 3 illustrates this process.

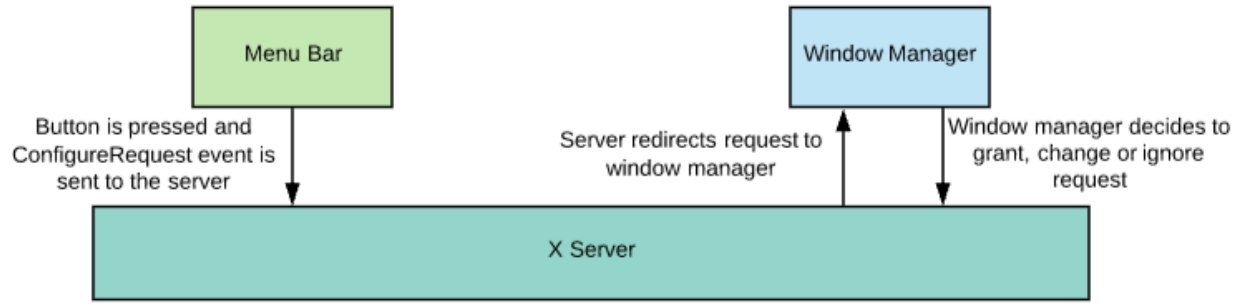

Figure 3: Handling of a ConfigureRequest event

For the buttons responsible for tiling windows as well as the "Minimize" and "Delete Window" buttons, there is a 1 second delay from the time the user clicks on the button and the time the desired window is tiled, minimized or deleted. This delay is meant to give the user enough time to move the mouse pointer over to the window on which the desired configuration will be executed. The way the window manager knows on what window to perform a given operation is through the Xlib function XQueryPointer(…) which returns the window on which the cursor is currently on. In lwm.h there is a macro defined as SLEEP_TIMER that is currently set to 1. If the user finds that 1 second is not enough time to move the cursor to the desired position, the timer can be set to a different value such as 2 seconds. However, after testing the program myself, which I realize might not be the best idea, I found that 1 second is just fine. If I had more time to work on this project, I would make it so that the window manager listens for ButtonPress events in an infinite event loop, so that instead of using a timer, the user can click on the window to be modified.

The tiling buttons allow the user to tile windows in quarters (each corner of the screen), halves (each half of the screen: top, bottom, left and right) or full-screen.

The “Minimize” button changes the height and width of the window specified by the user. In the file lwm.h there is a macro “MIN_SIZE” that is set to 50, meaning that by default when a window is minimized, its height and width are both changed to 50 pixels.

The button “Circulate” allows the user to cycle through windows. This is useful when windows overlap each other. When the button is pressed the function XCirculateSubwindows(…) is called and the lowest child of the root window is raised. So, if there is a window that cannot be seen because another window is on top of it, the circulate button brings the obscured window back on the screen. Figure 4 is meant to help illustrate how the window stack works.

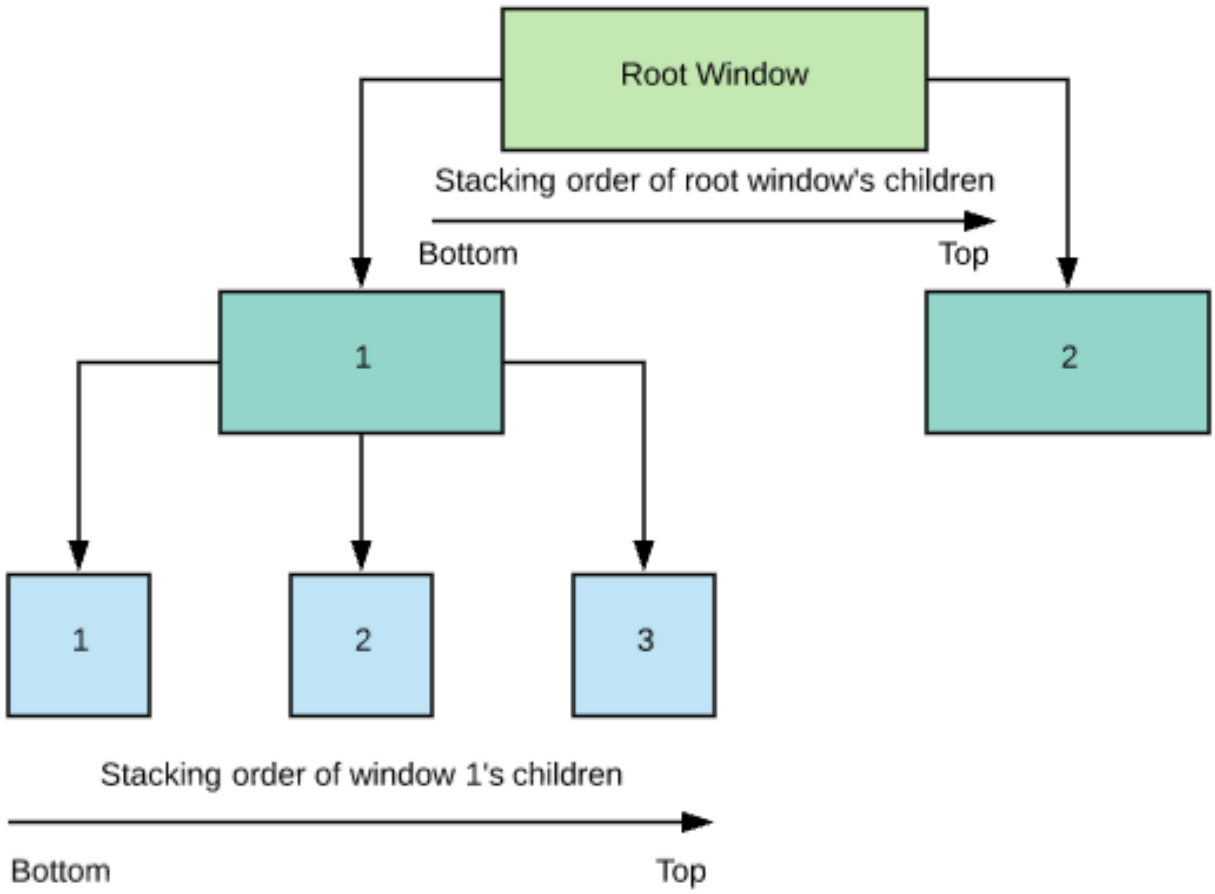

Figure 4: Window stack layout

The “Delete Window” button destroys a window and all of its sub-windows, effectively removing a window from the screen for good. If the main window, that is the xterm created by default during X’s start-up is deleted, then the connection to X is lost and the entire session ends, sending the user back to MINIX’s command line (Figure 5).

Figure 5: MINIX's command line

The "Exit" button closes LWM's connection to the X server, effectively closing down the window manager, but not ending the current X session, meaning the user is not returned to MINIX's command line.

The menu bar, created with GTK+, is actually a window with multiple children windows in it. It is made up of one "window" and one "menu_box" widget, as well as a separate widget for each button.

The menu itself is created with GTK+ functions and the functions that handle window operations such as tiling and deletion come from Xlib.

Figure 1 illustrates a scenario where the user has two terminals open and tiled to convenient positions in the screen. Figure 6 on the hand, represents a scenario where the user has no easy method to resize the windows on the screen nor can they have a second terminal open. If they called xterm in one terminal to create a second terminal, the terminal in which xterm was called would be unable to receive any new commands until the user pressed "Ctrl+C" and killed the second terminal, this dilemma is illustrated in Figure 7. With a window manager such as LWM the user does not have to suffer through the nightmares from either Figure 6 or Figure 7, instead they can easily manage windows like illustrated in Figure 1.

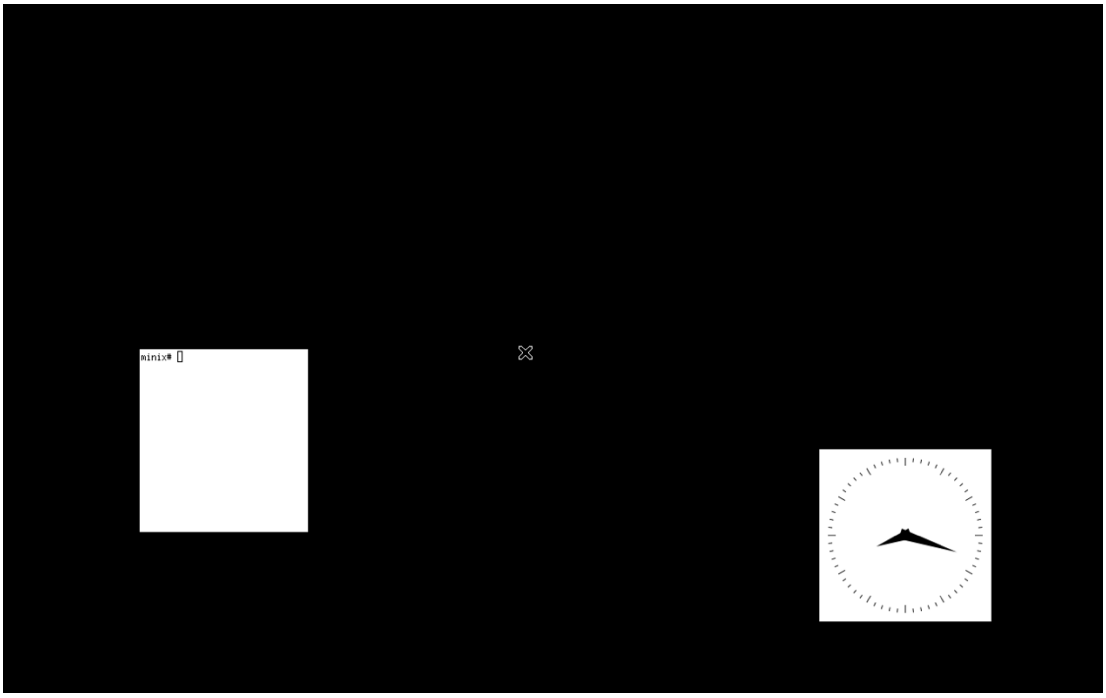

Figure 6: MINIX running xterm and xclock without a window manager

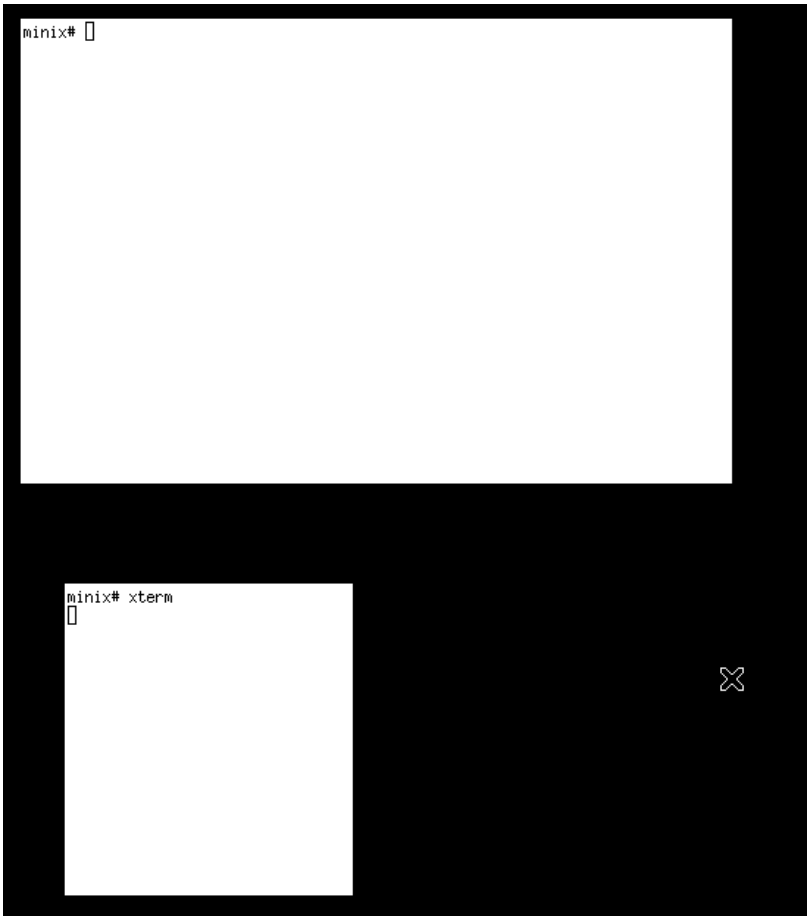

Figure 7: Using one xterm to create another xterm

Section 4 - Evaluation and Quality Assurance

Using LWM it takes about 1 second to tile, minimize or delete a window. If SLEEP_TIMER is changed to a different number such as 2 seconds, then it would take about 2 seconds per operation.

Without a window manager if the user would like to resize an xterm window they would have to type in a command specifying the x and y coordinates of the top left corner of the window as well as the width and height of the window in terms of columns and rows.

If they would like to resize a different window such as xclock they would have to close the current xclock probably using “kill xclock_pid” and then make a new xclock window with the desired geometry or they would have to close the X session, configure xinitrc and call startx again.

It is safe to assume that typing long commands with numbers that sometimes need to be calculated, finding the pid of programs to kill or re-configuring xinitrc should take more than 1 second.

Comparing the time of performing aforementioned operations on windows using LWM vs not using a window manager, the result is roughly 1 second per operation vs more than 1 second per operation. This makes using LWM a winner over not using a window manager.

The efficiency of LWM could be improved, if there was not a fixed time delay (in this case 1 second) between the time the user clicks on a button and the time the window is actually modified. To reduce this delay, I could have implemented an event loop to listen for ButtonPress events. This way the user could click on a button such as “Top Left” and then click on the window to be tiled to the top left of the screen. If the user manages to click on the window in less than 1 second, then the operation would be faster than what it currently is with SLEEP_TIMER set to 1.

I had one user test LWM. The user had to be explained about the 1 second delay. The delay was not an intuitive concept seen in the other operating systems the user is accustomed to, and so he was not able to figure it out without being told. The user suggested to change the timer to 2 seconds instead of 1 second.

Section 5 – Conclusion

5.1 – Summary

I used Xlib and GTK+ to create a tiling window manager for MINIX. LWM provides the user with options to create new terminals, minimize, delete, circulate, move and resize windows to predetermined positions in the screen (quarters, halves and full-screen), all with the push of a single button and with efficiency of roughly 1 second per operation.

5.2 – Relevance

In the course we used MINIX at the start of the term. It was challenging at first to use MINIX, but after doing this project I feel very familiar not only working with MINIX but other UNIX-like operating systems such as Ubuntu. When doing the exercises for the course, I often had to dig through MINIX's file system to find specific files that were needed for specific reasons, for example, finding the right header to get a program to compile. Likewise, when I started the project, I had to dig through MINIX to become acquainted with many of the system files related to X. This is how I eventually learned that xinitrc is directly responsible for launching applications during X's start-up after calling startx.

5.3 – Future Work

Over the summer I might work on a window manager for Linux using a modern widget toolkit such as GTK+ 3 or Qt and upload it to my GitHub account as part of my portfolio.

Section 6 – Acronyms

i. Desktop environment…….DE
ii. Luis's Window Manager……LWM
iii. Massachusetts Institute of Technology…..MIT
iv. Operating system……...OS
v. Tom's Window Manager……...TWM
vi. Window manager……..WM
vii. X terminal emulator……..xterm
viii. X Window System……X11 or simply X

Section 7 - References

Below is a window manager I studied:

1. https://github.com/TimothyLottes/MinWM

This is a guide I read:

1. https://seasonofcode.com/posts/how-x-window-managers-work-and-how-to-write-one-part-i.html

These are books I read and found very useful:

1. XLIB Programming Manual, Rel. 5, Third Edition by Adrian Nye
   Available online through Carleton's library:
   http://proquest.safaribooksonline.com.proxy.library.carleton.ca/9780596806187
2. Xlib – C Language X Interface: X Consortium Standard by James Gettys and Robert W. Scheifler
3. GTK v1.2 Tutorial by Tony Gale