GUI Efficiency Comparison Between Windows and Mac

Eric McCary and Jingyaun Zhang

The University of Alabama, Computer Science Department
Tuscaloosa, Alabama 35487-0290
eamccary@crimson.ua.edu

Abstract. In present times, it is not uncommon to have a desktop with two or more monitors. How these operating systems perform in a multiple monitor environment is an interesting topic. In this work, we will evaluate the efficiency of and compare how two popular operating systems, Windows and Mac OS, perform in large and multiple monitor environments. In particular, we will evaluate the performance of menu bars in both operating systems as they serve a near identical purpose and have the same functionality while providing their offerings differently. It is well-known that Mac OS uses a menu bar at the top of the screen (global) and Windows uses a menu bar attached to the top of its respective application (local). The conducted user study shows that the overall performance of Windows menu bar was better than that of the Mac menu bar implementation in the conducted tests.

Keywords: Graphical User Interface (GUI), Menu Bar, Title Bar, Operating System, Locality.

1 Introduction

Graphical User Interfaces (GUIs) have become an integral part of the average individual's everyday activities even in instances where one may not be aware of its composition or of its existence. Modern operating systems normally come packaged with a graphical user interface, and these GUIs have continuously evolved over the years, and in most cases have extensively enhanced the user experience with every major release. For example, Windows and Mac have GUIs that are tightly integrated into their OS kernels and have made significant changes on nearly all of their versions. [1]

Window's and Mac's GUI implementation both support the use of large and multiple monitor environments. This functionality was eventually added due to the changing customer desire and the cheaper cost of display equipment. Support for these multiple monitor configurations was added to Windows in its Windows 98 OS, and Mac implemented this as a standard feature in 1987. [2]

A significant occurrence in GUI efficiency is the convergence of stylistic elements belonging to Mac OS X and Windows GUIs. This fact brings to our attention the amount of similarity found in these two operating systems, and upon observation beyond the "look and feel", there are more resemblances and likeness of functionality. With so little difference between core functionality, it is useful to understand the

S. Yamamoto (Ed.): HIMI/HCII 2013, Part I, LNCS 8016, pp. 97–106, 2013.

differing means of utilizing the functionality offered by the OS GUIs and how efficient and user-friendly these methods are. More specifically, how efficiently these two GUI styles will handle themselves in environments with large or multiple display screens.

Since these two operating systems implement their menu bars differently, it is useful to measure the disparity in efficiency among them in single and multiple display scenarios. This paper will evaluate the graphical implementation of the menu bar in the GUIs used by Microsoft Windows and Mac OS X.

2 Initial Analysis

An excellent tool to measure efficiency of interface design is Fitts' Law. Fitts' Law, as defined in [3] as a mathematical model of fine motor control which predicts how long it takes to move from one position to another as a function of the distance to and size of the target area. Although Fitts' model was originally formulated to project how quickly a human could point at a physical button, we can use the same set of rules to determine how quickly we can acquire specific points or objects on a screen. The law also states that the bigger buttons are the faster they can be accessed and should be used for more important functions. This law has been previously defined as:

$$\text{Time} = a + b \log2 (2D / W + 1) \quad (1)$$

Mathematically interpreted, Fitts' Law is a linear regression model for a 2-dimensional space. [4] We will define D as the approximate distance of the impending movement and W is the width of the target. *a (intercept)* and *b (slope)* are empirical constants determined through linear regression which are device dependent.

The authors in [5] break the law down into two simple concepts:

- The farther away a target is, the longer it takes to acquire it with the mouse.
- The smaller a target is, the longer it takes to acquire it with the mouse.

For our purposes, Fitts' law will be used as a quantitative method of modeling user performance in rapid, aimed movements, where one's appendage starts at rest at a specific start position, and moves to rest within a target area while controlling a pointing device.

It is important to note the edges of a screen which ends the visible and tangible viewing area. These sections are thought of to be infinitely wide or tall because it is not possible to scroll or move past these areas. For example, it is not possible to scroll left past the leftmost boundary of a display. So we describe any object on the outermost borders as being infinitely wide or tall.

The authors in [6] explain how bigger targets are easier to click, and edge-adjacent targets are effectively infinitely big. This makes us aware of the fact that in order to improve user accuracy or acquisition speed, we can either make the desired object larger, or closer. To further examine this, we can inspect the equation that represents Fitts' Law with inputs from two hypothetical scenarios. These scenarios will demonstrate attributes that an infinitely large menu bar and a menu bar which has excellent

proximity to its application would possess. If we maintain the values of a and b (which means the start and end locations lie on the same vector on a 2D plane, or in this case, the display screen), we can determine the disparity in *Time* which the equations will yield.

$$\text{M1: } Time = a + b \log2 [\ 2(1000) / \infty + 1] \quad (2)$$

$$\text{M2: } Time = a + b \log2 [\ 2(0.001) / 100 + 1] \quad (3)$$

M1 represents the equation with inputs from an infinitely large menu bar, while *M2* represents a menu bar which is attached to its application. We can see that the solutions to these equations will be very small in either case, and in reality, the value of D in M2 will likely be a very small number, making both *Time* values very close to zero.

There are many of such truths of interface that extend beyond what the user is used to or favors, and good interface design requires close attention to these. So familiarity has a lot to do with comfort and ease of use and users frequently prefer the familiar to the more usable.

Fitts' Law can be utilized to scientifically estimate the results of the experiment to be detailed later in the paper. This process can provide users with a definitive value which delivers the calculated efficiency of acquiring the menu bar in each of the GUIs being tested.

3 Effect of Screen Sizes

Screen size further complicates the efficiency in a GUI when dealing with differing methodologies of menu bar placement. This is important to consider as the technological trends seem to be incorporating larger and larger screens into the majority of environments where computers are used with exception to mobile devices.

Mac OS X is installed on a limited set of hardware. The smallest of these is an 11-inch MacBook air. Mac OS X works well in this environment, as the menu bar of applications is detached and resides on the top of the screen. This type of configuration maintains a shorter travel distance to the buttons belonging to the running program. This is achieved due to the small screen size of this specific laptop. So in this situation there is less work to be done by the user in order to access application controls. On the other end of the spectrum, in a situation where OS X is being displayed on a large screen or even multiple screens, this operating system's menu bar is sometimes located a great distance from the focused application.

As in both OS GUIs, a single instance of application control objects being located on a single screen culminates into a concept of a main screen. This basically means that objects such as the Menu Bar can only be located on this singular screen. The fact that the menu bar can only reside on this screen may create more efficiency difficulties. Considering that normally when one uses multiple screens to expand their workspace, this individual would like to move application and documents into windows other than the main screen. Moving back and forth between these screens may create more work for the user.

Windows software can be installed on a wide variety of devices with a wide range of screen sizes. This operating system also makes use of the main screen concept

which will keep its taskbar on a single window when utilizing multiple monitors. However, Windows operates somewhat differently from Mac OS X with its menu bars for an application and allows for menu bars to be attached to their respective applications.

When dealing with single monitors of various sizes, Windows GUI configuration allows for universal locality of an application's control objects, including the menu bar. In this case, regardless of the screen size, the applications in use will still be relatively near to their respective controls and functional buttons.

We will be measuring efficiency of the GUI in part by the size and distance of objects that have an important relationship with each other. Many of the comparisons and tests will be evaluated with this premise in mind.

4 Application/Document Menu Bar

In an operating system, a menu bar is a horizontal strip that contains lists of available menus for a certain program. Windows and Mac OS X have different implementations of their respective menu bars which each of their dedicated followers seem to enjoy equally as much. In this section we will discuss each of the implementations.

Mac describes their menu bar as the semi-transparent bar that spans the entire width of the desktop at the top of the screen [7]. Their implementation is unique, in that is not attached to the application or document that it belongs to. For all of the applications running, there is a single menu bar that maintains the current application's menu options.

Varying Mac applications may have menu bars built onto them. Often these are open-source projects or java applications whose prime targets are not necessarily always Mac users. In addition to application specific menu options, Mac's global menu bar contains the Apple Menu, and it is always present, no matter which application you are using. It gives you quick access to a few essential system functions which are options Windows includes on its taskbar (which is comparable to Mac's Dock).

The placement of Mac's menu bar is important to consider. The top of the screen location is "infinitely tall". The acquisition of Mac's menu bar, according to Fitts' Law, takes advantage of its positioning and infinitely tall size. Also, multiple applications may be open simultaneously must also be addressed. These matters may occupy the current focus which will reassign the contents of the menu bar to the application currently in focus. When utilizing Fitts' Law to calculate the acquisition time of the menu bar, one would have to add the extra time to bring an application into focus which would be an inconvenience. Also, there may be multiple monitors in use. This will create in increase in total acquisition time as the time to navigate to the "main screen" must be added into the equation.

Windows menu bar is attached to whichever application it belongs to. This means that there can be multiple menu bars active simultaneously. This menu, much like Mac OS X menu bar menu, will be an interface to important functions offered by the application itself.

In Windows, there are no special cases when calculating the acquisition time for the menu bar. The reason for this is that the menu bar is attached to each application. This means no extra factors added in (screen navigation, focus...) as the application

interface is uniform in Windows and there are no shared modules which may inconvenience the user. Also, the menu bar itself and its' buttons will be much smaller than those belonging to a Mac menu bar. Application control buttons and menus on Windows are of a fixed size based on the application while on a Mac they are “infinitely tall”.

According to [3] Bruce Tognazzini, human computer interaction professional, claims the Mac OS menu bars can be accessed up to five times faster due to Fitts' law because the menu bar lies on a screen edge. The time to acquire a target is a function of the distance to and size of the target, with this knowledge Apple claims their pull-down menu acquisition can be approximately five times faster than Windows menu acquisition.

One can argue that any GUI offers the same “top of the screen” location if a window is simply maximized. Also, if one is utilizing an application and would like to close another, they would have to select the desired application to focus in Mac OS GUI while in Windows buttons grab the focus when the mouse hovers over it, and then close the application. Fitts’ Law does not account for the time that this extra step consumes. The effectiveness of this technique is also reduced on larger screens or with low mouse acceleration curves, especially due to the time required to travel back to a target in the window after using the menu. [8]

It is also important to consider the menu bars proximity to the application, and that its positioning is based on logically sound reasoning. Another important fact to consider is that most operating system environments place the menus within the application window, and that is a familiar location.

5 Experiments

Experiments were conducted to test both of these GUI’s efficiency. Each experiment yielded the time and distance it takes for a user to acquire several different control objects offered by the operating systems GUI.

Thirty participants were recruited to create the data necessary for this study. Twelve of the thirty expressed being familiar or expert users of Mac OS, while the others were primarily Windows users. The process was officially approved by the IRB at The University of Alabama and actions were consented to by the participants.

Each of the tests is comprised of a series of clicks with a pointing device which will each prompt the next step in the test. Upon the end of the testing session, the application will display the timing and distance results to the user. This test will record the time and distance traveled per task, and the deviation from the “best” possible route to complete the task (which will be a straight line).

5.1 Intuitive Impressions

The experiment is expected to gather data which will undoubtedly determine which of these two GUIs is more efficient in completing specific and common tasks. Initial impressions were based on most of the general population being more familiar with the Windows GUI environment, and environments fashioned like it. Since Windows’

menu bar is based on locality to application that owns it, and excels in certain situations where the display is larger than normal or when multiple displays which span a large area are in use.

While Windows' menu bar has the locality advantage in all situations other than full screen mode, Mac's menu bar is infinitely tall and users have the ability to simply throw the pointing device to the corner of the screen and acquire their menu.

Taking these details into consideration, the initial assumption is that Windows GUI would be the better option when comparing distance traveled to acquire the menu bar, while Mac would dominate the acquisition time category. This premise takes into account the differing screen sizes in use today, and the skill level and ability of the average user.

5.2 Experiment Detail

A program was written that simulates the GUI environments (Mac and Windows), and will allow for the completion of identical tasks while recording the time and distance of each of the tasks. This program was built with java on Windows 7 and Mac OS X Lion.

To record data, the program will present the user with a window which simulates an application. A red square will serve as the starting point for the current task. The initial view of the application is the same in each of the tests, while the tasks in each test are in the same order and routine. Figure 1 displays one of the testing environments for a task. For this specific widow size there will be a total of seven separate tasks (represented by each of the red squares) which will generate new data about the user and the GUI which is simulated as a housing for this application in the specific task.

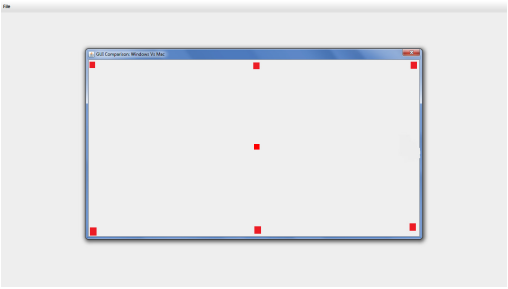

Fig. 1. Task start screen (Medium Window)

Once the starting point is clicked, the data from the actions which follow immediately will be recorded which should culminate with the user acquiring the File Menu in the simulated environment. The final data consists of distance per task, time per task, and each point the pointing device traverses over during the task session.

This data is recorded in three separate sessions where the participant will repeatedly acquire the menu bar in environments with different windows sizes. The largest windows size has nine points, the medium sized window has seven points, and the smallest window is host to a single point in the center. Each of these "start points" (red squares) will be acquired at the beginning of each task which upon completion (the user acquires the File Menu), will yield the desired result data.

6 Results

The data did not completely support the initial assumption that Mac's infinitely large menu bar would prove to increase acquisition rate. In fact, acquiring Mac's menu bar proved to be slightly slower on average in all but four of the tasks. These tasks were acquiring the menu bar from the bottom two corners of the medium sized window, and the top and bottom middle of the maximized window. Figure 2 shows the average time it took to complete a task during the tests below. This data is important to consider while misleading. Fast times could equate to the user being comfortable with the GUI (as suspected of the Windows users), or the ease of use and efficiency of the GUI.

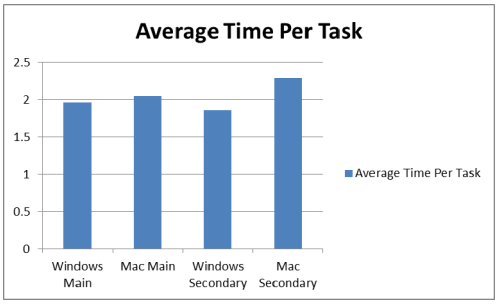

Fig. 2. Average Time per Task

When observing distance efficiency, it is obvious that Windows will likely require the user to move the pointing device the least distance with an acceptable amount of error. This premise was proven true as Windows was more efficient in all tasks. This data is represented in Figure 3. As is made understandable by the figure, Mac tasks took nearly twice as long as the Windows tasks. This is obviously directly affected by the environments locations of the features on the application which is determines the locality.

There is a very large disparity in this category with the advantage belonging to Windows. When comparing the distance traveled per amount of time, Mac more than doubles Windows in this category. This is partially a product of Mac's static menu bar position even though it is infinitely large. Windows menu bar is closer, but must move and resize itself as the window resizes and moves.

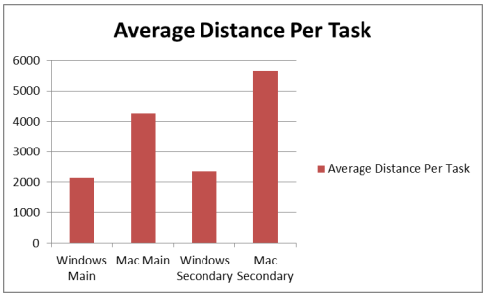

Fig. 3. Average Distance per Task

7 Faults and Error

To better understand the role of context in error inclusion and its adequacy, the deviation along the x-axis of the path that the participant took en route to completing each task was analyzed. This was accomplished by locating the start and end points of the task and every point directly between the two. In addition, the distance the participant's path is from each x-value had to be calculated. This data is very important to the experiment. While a normal amount of deviation is expected, if deviation is constantly above a certain threshold it will lead to results which are more defined by the ability and actions of the participants rather than the placement of objects and accessibility of the GUI.

This data is collected in units of pixels and the total deviation is the sum of the deviation of all points on the line.

Windows Deviation along x-axis:

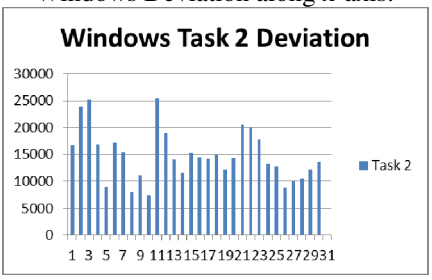

Fig. 4. Windows Task 2 Deviation

Mac Deviation along x-axis:

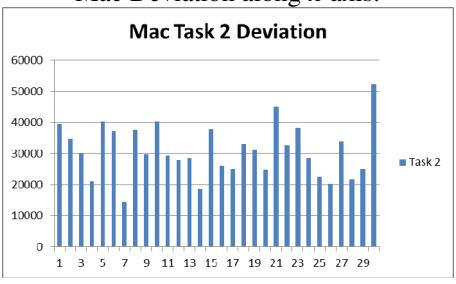

Fig. 5. Mac Task 2 Deviation

The numbers here are much more sporadic as they depend mainly on the participants' ability and focus. Each participant was encouraged to complete the tasks as they would in their most comfortable environment. It is understandable that the amount of deviation is greater on tasks executed in the Mac environment as the travel distance to complete the tasks is so much greater which creates more of an opportunity for error. With this method of error calculation, travel distance as well a precision

plays a large role in the participants overall route efficiency. Taking all of these factors into account will help us to come to the best conclusion.

The deviation recorded for each task is displayed as a summation of the deviation at each point which the user traversed over during task completion. Understanding that the Mac tasks on average are several times more than that of the Windows tasks, we can calculate the exact ratio of time per unit of distance. This will make the deviation data more relevant and useful. For example, the distance ratio of the Mac to Windows tasks on the main screen is approximately 3.321467. This means if we divide each of the tasks in the Mac deviation by this amount, will have the amount of deviation per an equal amount of distance which makes the two categories comparable.

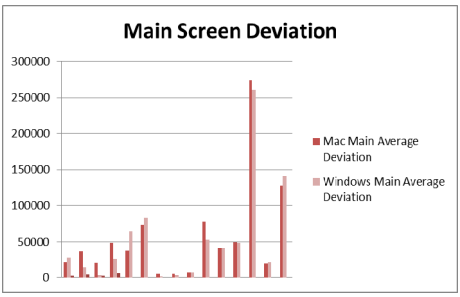

Fig. 6. Windows Deviation

Here, the Windows tasks have incurred more error in 5 of the 15 tasks. This can be attributed solely to the actions of the user, as the paths to complete tasks are all equally attainable.

The tasks set in the Windows GUI were completed slightly faster than the identical task in the Mac GUI. In terms of distance, the average distance per task in the Mac GUI is nearly double that in the Windows environment. When taking these facts into account, we see that the distance covered per unit of time is much more efficient in the Mac environment as the participants covered distance significantly faster here.

The experiment also calculated the deviation that the participants added in error to their routes along with the total time. In observing these details, we see that tasks completed in the simulated Mac environment were completed much more efficiently from a distance per unit of time standpoint due to the distance of the tasks being more the double the length of the Windows tasks while the deviation is relatively the same.

8 Conclusion

When determining the overall GUI efficiency, we must understand that the speed with which the participants completed the tasks reveals little about the efficiency of the environment whether it is of a small screen size or multi-monitor format. We must first weigh the value of the distance of the elements present on the interface as well as the time participants took to complete the tasks as the first line of assessment, then

weigh the deviation results and other factors exposed in this study. This will render our final result.

The results from this study lead us to conclude that Windows menu bar presents a shorter travel distance in identical situations than Macs' menu bar. Although In certain situations, each GUI environment has its advantages. For example, Mac's GUI excels in scenarios where full screen mode is in use. The results yielded in these situations displayed that the time per task in a Mac environment is less, while the distances are relatively equivalent. In situations where small or overly large windows sizes are necessary, Windows excels. The amount of deviation introduced into the experiments makes it difficult to reach a clear cut victor but according to the overall results, Windows menu bar implementation is more efficient when completing tasks such as or similar to the ones in the experiment. This would include most of the everyday operations that are completed using the bar. These results stand with regard to the minimal amount of difference in values recorded.

This paper has evaluated and discussed the differences and similarities between the menu bars utilized by Microsoft Windows and Mac OS X. These aspects were evaluated while operating in multiscreen as well as large and small screen environments. The experiment introduced for the user study described in this paper, was used to help solidify and prove that Windows menu bar implementation provided for a shorter travel distance and acquisition time of the elements present in each of the operating systems simulated GUI environments.

References

1. Dale. E.: Operating Systems Uncovered: The Inside Scoops Are Revealed (March 2012)
2. Dunn D., Mikes. N.: Multiple Monitor Computing. 9X Media Inc., http://www.9xmedia.com/PDFs/9X_Media_multiple_monitor_whitepaper.pdf
3. Tognazzini, B.: First Principles of Interaction Design, http://www.asktog.com/basics/firstPrinciples.html#fittsLaw
4. MacKenzie, I.S.: Fitts' Law as a Performance Model in Human-Computer Interaction. Ph.D. Thesis, http://www.yorku.ca/mack/phd.html
5. Harris, J.: Giving You Fitts, http://blogs.msdn.com/b/jensenh/archive/2006/08/22/711808.aspx (August 22, 2006)
6. Atwood, J.: Fitts' Law and Infinite Width, http://www.codinghorror.com/blog/2006/08/fitts-law-and-infinite-width.html (August 9, 2006)
7. Apple Inc.: http://www.apple.com
8. Leigh D.: "Mac" menubar as default, http://lists.kde.org/?l=kde-look&m=95705988431395&w=2 (April 30, 2000)