Radial Marking Menu Performance Improvement and User Type Detection

Timothy E. Burke - tburke2@umbc.edu

Prepared for CMSC601 - Dr. Tim Finin

Reader - Dr. Amy Hurst

May 15, 2011

1 Abstract

Menus in graphical user interfaces have existed for over two decades and have largely been variations on the linear drop-down menu paradigm. In many operating systems and software packages, menus are unchanging and non-adaptive, forcing users to memorize menu command layouts in order to find desired commands. Research has been conducted to allow menu systems to become adaptive to single users over time or to dynamically detect a user's expertise level in order to reorganize or alter the display of menus to improve command selection and execution time. More recently, devices utilizing touch-screen, multi-touch, and pen-based input have gained popularity, yet the linear menu paradigm, which is not as well suited for those input modalities, has persisted. An alternate menu paradigm, radial marking menus, has emerged as one possible alternative to the linear menu for touch and pen-based input devices. The purpose of this study is to explore two previous methods for improving performance (command selection and execution time) of linear menus and adapting those methods to radial marking menus. Specifically, exploring one method for adapting to a single user over time, and another utilizing machine learning techniques to dynamically detect a user's level of expertise.

2 Introduction

Radial marking menus are a relatively underutilized menu paradigm in user interface design, and have been the subject of study for applicability to differ-

ent tasks. With the proliferation of mobile and portable devices that utilize touch and multi-touch screens, marking menus are poised to become more popular. Previous studies have largely focused on different layout modalities and the performance of those layouts as it relates to specific user tasks and for specific interface types. In those cases, the menus studied were static and unchanging; they did not adapt to any specific user, either over time or dynamically. Extending the adaptive capability of menu systems can lead to an improved usability experience for the user. Adaptability of radial marking menus is an underexplored area in user interface design, and is worthy of exploration.

Previous research in menu adaptability has explored observing a single user to determine, over time, which specific menu commands and the types of commands that are most used. Frequently used commands and types of commands are increased in size or moved to more easily accessible positions within linear menus. This adaptive paradigm could be modified for radial menu layouts, and studying the effectiveness of adaptive radial marking menus is a goal of this study. Additional previous research has explored dynamic detection of specific user types, novice vs. expert or experienced user, through machine learning predictive models - also with linear menus. A benefit of identifying to the software system the user type is menu contents, application behavior, and help systems can be adjusted according to the needs of differing user types. Studying how this method can be applied to radial menus is also a goal of this study.

Methods for this study would include implementing a test system or systems that present users with an interface that includes radial marking menus,

and recording how the user interacts with the menu systems over time. Applying frequency-based changes to the menu system presented to the user allows measuring for improvements in menu search and command execution time and accuracy, if any. Data recorded for studying adaptive menus can also be used as a basis for training a machine learning based classifier for identifying an individual user type, and the resulting training data can be compared for accuracy against the results of previous studies.

3 Related Work

Previous work has explored the single level radial marking menus, and while effective, has found that the number of options for the menu is generally limited to eight or less selectable options [7]. For the purpose of this survey, much of the work explored focuses on extending the range and scope of the radial marking menu in several ways. Included are methods for extending the marking menu paradigm hierarchically through different layout designs [1][3][5][7], leveraging multi-touch [5] and gestures[1], and by using pen-based input systems [6][8].

4 Background

Not all users interact with software systems in the same way or with the same level of proficiency [4]. Optimizing the usability performance of a software system for different user types provides an enhanced experience. Different models exist for modeling the user experience around the different user types,

novice and expert. While not an exhaustive description of potential models, some examples include modeling the user experience around learning over time how a single user executes commands and interacts with a software system and adapting accordingly [2], and another dynamically detects the proficiency of a user based upon predefined statistical models and machine learning algorithms to adjust the user interface of a system [4]. Both models present methodologies for enhancing the user experience through menu command execution, though neither specifically has been adapted for use in a system utilizing marking menus, a potential area for further research and is intended to be explored in the experiments in this study.

Over time, it is possible to detect and learn how an individual user interacts with a software system. If an individual repeatedly selects the same menu commands, a software menu system can gradually increase the size of frequently selected items to provide the user a larger target and decrease selection times [2]. Additionally, this model can also selectively reorganize menu items by placing frequently selected items at or close to the top of the menu, further decreasing potential selection times [2]. This method is limited as the behavior can be influenced by other users interacting with the system in ways that change the frequency of selected menu items.

Deviating from menu performance, the data collected as the user interacts with the software system, including the menu systems, can be used to identify user types. A classifier, or predictive statistical model can be created from training data depicting novice or expert users and, when used with machine learning algorithms, can correctly identify the user type with a high degree of accuracy [4]. This model is task independent as it models user behaviors

rather than specifically measuring execution times for specific tasks in a software system. For example, novice users take longer to locate most menu items, where expert-level users, regardless of their familiarity with a specific piece of software, are more likely to quickly select appropriate menus and commands based on a more general knowledge of how menus are organized [4]. With knowledge of the user type, a software system can appropriately adapt to different user types and provide to the user appropriate contextual information for different tasks within the system.

5 Technical Approach

As a technical consideration, it is necessary to construct a set of menu systems to accommodate this study. Multiple different interpretations on the radial marking menu paradigm exist, and styles of hierarchical radial marking menus include those with sub-menus arranged in concentric rings around the original level [3][7], sparsely arrange spoke-like sub-menus [7], fan and half-pie arranged menus [3][7], and stem and flower-like arranged menus [1]. This is by no means an exhaustive list of possible designs, rather a sampling of common or similar designs. For the purpose of this study, a simple, non-gesture-specific radial marking menu implementation will be constructed to incorporate elements from concentric ring styled menus and from the similar, stacked half-pie menu designs.

5.1 Technologies

Different systems of input are to be explored to explore potential differences in user interaction. Specifically, three different broad classes of devices are to be explored: small screen, medium-sized, and large-format input surfaces. Small screen devices, including touch devices like the iPod Touch, iPhone, Android phones, and pen/stylus based PDA devices, will be explored in two sub-categories, touch/multi-touch devices and pen-based devices. Medium sized devices will specifically focus on tablet sized devices such as Microsoft Windows tablet PCs, iPads, Android-based tablets, also subdivided along the touch/multi-touch vs. pen-based input divide. Lastly, large-format touch/multi-touch devices will utilized, specifically exploring Microsoft Surface, a large, table-like touch interface.

The software development process will vary somewhat from platform to platform, given the different device and software vendors. For Microsoft-based systems like Surface, and Windows Tablet Edition, the Microsoft .NET libraries will be leveraged, including the Surface Software Development Kit specifically for Microsoft Surface, with the programming language being C# (C-Sharp). For Apple iOS devices like the iPod Touch, iPhone, and iPad, the Apple iOS Software Development Kit and core foundation libraries will be used, with Objective-C as the programming language. Lastly, for Android-based systems, including the spectrum of Android phones and tablets, the Google Android Software Development Kit will be used, with the programming language being Java. All of the mentioned languages and development environments provide a rich set of tools for developing applications for their

respective platforms.

5.2 Test Application

A representative, general-purpose application will be built, incorporating the same feature set across all platforms. Specifically, a basic word processor will be constructed to incorporate a set of basic features that most users would be familiar with, such as cut-copy-paste, text formatting, creation of numbered or bulleted lists, font selection, find and replace for text, file saving, and printing. The application will incorporate an on-screen keyboard for text input, but will eschew the more traditional linear menu paradigm in favor of only using hierarchical radial marking menus. Some basic gestures will be incorporated into the test system, including a single tap to move the typing cursor, double-tapping to perform a word selection, touch-swipe-release to select ranges of text, and touch and hold to display a menu.

Given the purpose goal of this study is to study touch, multi-touch, and pen-based input systems, the test system will not incorporate the use of a mouse. Accordingly, test subjects will be given basic instruction on how to interact with the system at the beginning of each testing session, with some variation depending on the input system - small, medium, or large format display and for touch or pen-based input. Users will be given a specific set of tasks to complete with the system during the testing session. Possible tasks would include composing an email or letter, entering and formatting a contact list, transcribing a short document, or applying formatting to a plain-text document.

5.3 Data

As users interact with the test system, multiple points of data will be captured for analysis. Measurements for speed to selection, pointing accuracy, start and stop coordinates, and overall interaction time will be captured. Further, menu command and command group frequency will be recorded. All approaches will utilize a common methodology for capturing data in a platform-agnostic manner. Specifically, all data will be captured and output into a common Comma Separated Values (CSV) format, which can be easily imported into a number of commonly utilized statistics and modeling software packages.

6 Methodology

This study will be conducted in two separate phases. The first phase will build upon the work previously published by Cockburn, et. al [2] for adaptive menu systems. The second will build upon the work previously published by Hurst, et. al [4], for dynamic user-type detection. Both previous experiments work with linear menu systems, where this study will explore adapting those experiments to radial marking menus.

6.1 Adaptive Menus

This study, which was conducted on linear menu designs, studies a single user, over time, and progressively modifies the menus by altering the size of frequently utilized menu commands, or by reorganizing the menus by placing

frequently executed commands closer to the top of the menu. These methods have been shown to reduce the search to execution time, thereby making the system more efficient for the user.

Building on the previous work of Cockburn et al., this study will expand that experiment to hierarchical radial marking menus. Over the course of one or multiple testing sessions with individual test subjects, the layout of the radial marking menus will be altered in a similar fashion to the previously discussed method, with frequently executed commands increasing in size, and with frequently used command groups moved to more optimal positions within the menu hierarchy. By applying the adaptive menu paradigm to marking menus, we hypothesize that user performance of the menu system will be measurably improved. Validation of this hypothesis would manifest in decreased selection time for menu commands, and decreased overall time spent interacting with the menu system when executing menu-based commands.

6.2 Dynamic Detection

A classifier, or predictive statistical model can be created from training data depicting novice or expert users and, when used with machine learning algorithms, can correctly identify the user type with a high degree of accuracy [4]. This model is task independent as it models user behaviors rather than specifically measuring execution times for specific tasks in a software system. For example, novice users take longer to locate most menu items, where expert-level users, regardless of their familiarity with a specific piece of soft-

ware, are more likely to quickly select appropriate menus and commands based on a more general knowledge of how menus are organized [4]. With knowledge of the user type, a software system can appropriately adapt to different user types and provide to the user appropriate contextual information for different tasks within the system.

Again, this study seeks to build on the previous work in this area by applying the techniques described to radial marking menus. Initial data collection will involve multiple sessions with individual users, interacting with a non-adaptive radial marking menu system. For initial sessions, users with no prior knowledge of the menu systems will be tasked with using the system, completing the same or similar tasks repeatedly. As a user would have no prior experience with the software or menu system, data collected in that session, specifically early in the session, would be representative of novice user interaction patterns. Interaction data collected later in the initial session or in subsequent sessions would be considered expert or experienced data as the user would have previously interacted with the system and had some memory of the menu layout.

The identified sets of data would then be utilized to build a machine learning classifier that can be used to correctly identify a potential user as novice or expert. Additional user trials would be conducted to assess the accuracy of the classifier. Repeat users from previous trials would be considered as expert users, while new users with no prior experience with the system would be considered as novice. We hypothesize that, with sufficient training data, the classifier can achieve a statistically significant accuracy in detecting expertise level of a user. This hypothesis would be determined

valid if the classifier is able to reasonably approach the results of the previous study introducing this methodology by Hurst et al., upwards of 90% correct user type detection.

7 Future Work

Many possible areas of future work could be derived from this study and the work it builds upon. If the study is not successful at achieving results approaching that of the previous related work, an exploration to determine the cause of the failure would be conducted. Further, more detailed analysis of the data could be conducted to better understand the strengths and weaknesses of each of the different input systems explored, specifically understanding the strengths and weaknesses of touch/multi-touch vs. pen-based input. Additionally, performing further analysis to understand how users interact differently from one size format device to another is planned.

8 Conclusions

Much work has been done to study how static radial marking menus perform for differing tasks. Additionally, work has been conducted to improve the performance of menu systems in general. Those performance improvement studies have focused on the more traditional linear menu paradigm. This study seeks to explore the possibility of extending that prior work by applying the methods and findings used to enhance linear menus to radial marking menus. Specifically, applying the single user learning model to marking

menus, improving their performance by adjusting command size and placement, though in such a way that would not reduce the effectiveness of gestures learned by the user. Also, the dynamic detection algorithm discussed could be adapted to systems using radial marking menus and studied for user type detection effectiveness.

Represented in this study are previously explored, novel techniques that can be applied to alternate systems. For these prior experiments, and the work that builds upon them, the ultimate goal is to improve the usability of software systems for end-users. The experiments described in this study seek to improve the ability of software systems to better adapt to a wide range of users with differing skill levels.

9 References

1. Gilles Bailly, Eric Lecolinet, and Laurence Nigay, "Flower Menus: A New Type of Marking Menu with Large Menu Breadth, Within groups and Efficient Expert Mode Memorization," Proceedings of the working conference on Advanced visual interfaces, 2008.

2. Andy Cockburn, Carl Gutwin, and Saul Greenberg, "A Predictive Model of Menu Performance," ACM SIGCHI Proceedings, 2007.

3. Tobias Hesselmann, Stefan Floring, and Marwin Schmitt, "Stacked Half-Pie Menus: Navigating Nested Menus on Interactive Tabletops," Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, 2009.

4. Amy Hurst, Scott E. Hudson, and Jennifer Mankoff, "Dynamic Detection of Novice vs. Skilled Use Without a Task Model," ACM SIGCHI Proceedings, 2007.

5. G. Julian Lepinski, Tovi Grossman, George Fitzmaurice, "The Design and Evaluation of Multitouch Marking Menus," ACM SIGCHI Proceedings, 2010.

6. Karyn Moffatt and Joanna McGrenere, "Exploring Methods to Improve Pen-Based Menu Selection for Younger and Older Adults," ACM Transactions on Accessible Computing (TACCESS), 2009.

7. Krystian Samp and Stefan Decker, "Supporting menu design with radial layouts," Proceedings of the International Conference on Advanced Visual Interfaces, 2010.

8. Feng Tian, Lishuang Xu, Hongan Wang, Xiaolong Zhang, Yuanyuan Liu, Vidya Setlur, and Guozhong Dai, "Tilt Menu: Using the 3D Orientation Information of Pen Devices to Extend the Selection Capability of Pen-based User Interfaces," ACM SIGCHI Proceedings, 2008.