Energy-Efficient Graphical User Interface Design

Keith S. Vallerio, Lin Zhong and Niraj K. Jha
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544, USA
{vallerio,lzhong,jha}@ee.princeton.edu

***Abstract*— Mobile computing systems, such as cell phones and personal digital assistants (PDAs), have dramatically increased in sophistication. At the same time, the desire of consumers for portability limits battery size. As a result, many researchers have targeted hardware and software energy optimization. However, most of these techniques focus on compute-intensive applications, rather than interactive applications, which are dominant in mobile computing systems. These systems frequently use graphical user interfaces (GUIs) to handle human-computer interaction. This paper is the first to explore how GUIs can be designed to improve system energy efficiency. We investigate how GUI design approaches should be changed to meet tight energy budgets and demonstrate that energy-efficient GUI (E^2GUI) design techniques can significantly improve energy efficiency.**

I. Introduction

Modern mobile computing systems have become an integral part of the lives of millions of users. The increased computational capabilities of the latest microprocessors have enabled PDAs, cell phones and laptops to provide many valuable services to users. However, reducing energy consumption is still a major design challenge. As a result, much research has been directed toward developing energy-efficient systems from hardware, software, and system perspectives [1]–[3].

Although energy reduction techniques have been successful for compute-intensive systems, many applications for modern mobile computing systems involve a significant amount of user interaction, which are frequently handled by GUIs. A recent work has characterized the energy consumption of different GUI components and noted the potential impact of GUIs on energy efficiency [4]. Therefore, our work investigates the energy efficiency problem from a fourth perspective, the user's perspective. Specifically, it presents techniques for increasing the energy efficiency of mobile computing systems via E^2GUI design. We make the following contributions.

- We demonstrate the impact of GUI design on system energy efficiency.
- We propose and evaluate E^2GUI design techniques.
- We provide specific suggestions to GUI designers.

The paper is organized as follows. In Section II, we offer background information. In Section III, we present techniques for improving system energy efficiency. After that, we present our experimental setup in Section IV and experimental results in Section V. Section VI presents our conclusions.

II. Background

Conventional low power software techniques usually reduce the energy/power consumption only during system busy time. By neglecting wait time, they are limited in their ability to improve overall energy efficiency for interactive systems since these systems spend most of their time and energy waiting for user inputs [5]. During this wait time, the display must be on. Displays are known to be the largest single power consumer in mobile systems [6]–[8]. A potentially more effective approach for reducing system energy is to improve user productivity.

Acknowledgments: This work was supported by DARPA under contract no. DAAB07-02-C-P302.

Psychological studies [9] have shown that reading speed depends on GUI layout and conciseness as well as visibility. The cognitive process is governed by the Hick-Hyman Law [11], [12]. The insight from this law is that to accelerate the human cognitive process, a GUI should present as few choices as possible. The split menu [13] is an example. The motor speed of human users is governed by Fitts Law [14]. This relationship is commonly presented as Equation (1):

$$T = c_1 + c_2 \cdot log_2(\frac{D}{W} + 1) \quad (1)$$

In Equation (1), T is the time required to complete a task (i.e., moving from point A to point B), D is the distance between A and B, W is the width of the target and c_1 and c_2 are experimentally-determined constants. Based on this law, a GUI should utilize as much screen area as possible for widgets to be hit. Widgets that are supposed to be hit sequentially should be placed near each other. Much effort has been spent in exploiting this law to improve user speed [15]–[18]. We will present new energy-efficiency techniques based on this law.

III. E^2GUI design techniques

This section presents a framework and specific techniques for designing E^2GUIs for mobile computing systems.

A. Overview

Developing a framework for designing E^2GUIs for mobile systems requires an understanding of how these GUIs are used. Most mobile computing systems utilize relatively small displays in order to achieve higher portability. Althought GUI design for these small displays requires a different approach from traditional ones, this fact is not well-recognized. Many GUIs on handheld computers appear to be miniature versions of their desktop counterparts. These GUIs contain power-hungry widgets and are often inefficient for mobile computer user interaction. Thus, mobile computing system energy efficiency can be improved by better utilizing the limited screen real estate and selecting more energy-efficient widgets. Increasing user productivity provides a second method for improving energy efficiency

B. Optimization techniques

E^2GUI design is divided into power reduction techniques, performance enhancing techniques, and facilitators. Power reduction techniques reduce the power consumed by the display. Performance enhancing techniques reduce energy by improving user interaction speed. Facilitators do not reduce energy directly. Instead, they enable other techniques to be used more effectively. A summary of E^2GUI design techniques is presented in Table I. The technique names are listed in Column 1 and the types are listed in Columns 2–4.

1) Power reduction: The displays used for mobile computing systems affect the energy consumption of four types of hardware resources:

- the display by presenting pixel information,
- storage resources for recording screen data in an allocated memory called frame-buffer,
- CPU cycles for screen data generation, and

TABLE I
E^2GUI DESIGN TECHNIQUES

Technique	Pow.	Per.	Fac.
Low-energy colors	x		
Reduce change	x		
Hot keys		x	
User input cache		x	
Content placement		x	
Paged display			x
Quick buttons			x

- communication resources by moving data through busses between the CPU, memory, and display.

A detailed discussion of GUI's impact on energy efficiency is presented in [4].

The first power reduction technique, *low-energy color schemes*, reduces display energy by using colors and color patterns that take less power. This technique reduces the power consumption of the first two hardware resources. For example, thin film transistor (TFT) LCDs, which are typically used in PDAs, consume the least power when white and most when grey. Thus, GUI power consumption can be reduced by selecting energy-efficient colors. Of course, it is important to take visibility (color scheme, contrast ratio, and luminance) into account. Yellow may require less power than black, but a yellow on white display is much more difficult for a user to read than a black on white display. In addition to selecting energy-efficient colors, fine patterns and textures should be avoided, since they increase power consumption by increasing switching activity.

The second power reduction technique, *reduced screen changes*, reduces energy by reducing switching activity as well as the computation required for screen data generation. Many of the opportunities for reducing extraneous screen changes result from GUIs being designed for aesthetic purposes rather than energy efficiency. This can be seen in the following examples.

- User-perceived responsiveness is different from real responsiveness. Using a progress bar may make a user feel that the computer is more responsive, but it actually slows down the system and increases energy consumption.
- Animations give users a natural feeling for the screen changes (i.e., animations for pop-up menu windows, closing windows, and minimizing windows). However, these animations add little or no functionality, yet they increase the power consumption of all four hardware resources (display, memory, CPU, and communication).
- Many natural-to-use GUI widgets, such as scrollbars, were designed based on desktop GUIs and ported to handheld GUIs. However, scrollbars are very energy-inefficient since they consume power via all four hardware resources, while causing frequent screen updates [4].

2) Performance enhancement: Performance enhancement techniques focus on improving user productivity. Performance enhancements from better GUI design are more significant for interactive software since user input usually requires orders of magnitude more time than computation.

Hot keys enable traditional user inputs (save, open, etc.) via multi-key gestures (i.e., control-s, control-o, etc.). This technique is common on desktops but is not frequently used in mobile computing systems, such as PDAs, since the use of a stylus for input decreases the advantages of multi-key gestures. This technique becomes viable through the use of the quick buttons facilitator (see *Facilitators*) since the quick buttons enable the operation to be performed in a single gesture.

User input caches store the most recent (or most frequent) inputs by users. The small size of PDAs makes sizable text inputs tedious and time-consuming. By caching previous inputs, the input time can be greatly reduced. This method is used in the calculator benchmark described later on. The input cache entries can be based on most recent use, most frequent use, or most common use. This technique can benefit from paged displays and quick buttons.

Content placement reduces the user interaction time for frequent inputs by strategically laying out GUI content. This technique is based on Fitts Law. Frequently-pressed buttons should be as large as possible and near each other. An example is use of the whole screen to control page scrolling instead of small buttons on one side. This method is used in the Text-reader benchmark described in Section IV-B.1. A natural consequence of this design rule is that tasks that require multiple interactions should be simplified into ones requiring fewer interactions whenever possible. This method is used in the personnel viewer benchmark described in Section IV-B.2.

3) Facilitators: Facilitators are a pair of techniques that enable or enhance the effects of the other techniques. The first facilitator, *paged display*, enables increased user interface functionality by increasing the effective display size. It is similar to tabbed panels. Spreadsheet programs currently utilize this technique by dividing sessions into worksheets. This technique can be utilized to include a page of buttons designed to enhance user interaction speed. This enables buttons to be larger, which can reduce power consumption, as illustrated by the results presented in Section V-B.

The second facilitator, *quick buttons*, uses available hardware buttons to increase user interaction capabilities for the current application. Holding down a hardware button can act as a <SHIFT> or <CTRL> key, which can facilitate the use of key combinations that are traditionally impractical for handhelds. (Note that one straightforward use for quick buttons is to enable rapid switching between paged displays.)

IV. EXPERIMENTS

In this section, we describe the experimental setup and benchmarks. The experiments were performed using the original and an energy-efficient version of three GUIs. There was one set of experiments for each GUI.

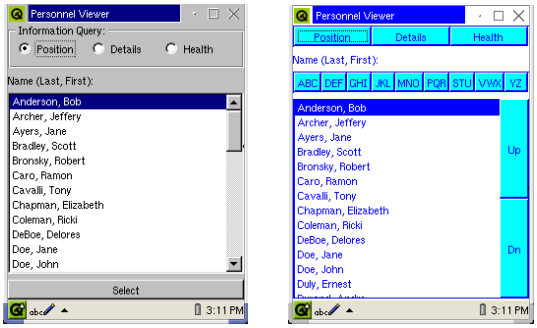
A. Experimental setup

This section describes the subjects, experimental procedure and hardware devices used for the experiments.

1) Subjects: A total of 16 subjects participated in the three sets of experiments. The group was composed of 10 male and six female electrical engineering and computer science graduate students. For each GUI, a different set of six subjects was selected. Two of the subjects participated in a second set of experiments.

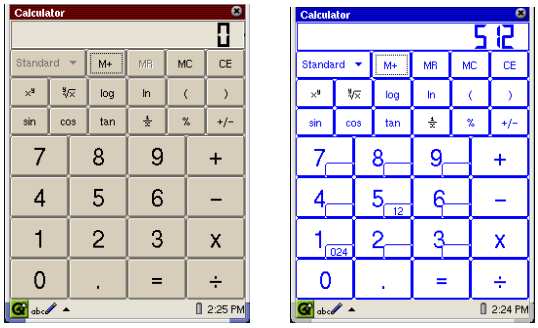
2) Experimental procedure: For each benchmark, an experiment set consisted of a training session followed by a series of experiments (trials). Initially, one version of the GUI was used to demonstrate the functionality of the application to the subject. (Half of the experiments started with the original GUI and half started with the E^2GUI to make the comparison fair.) Once the subject felt comfortable performing the training tasks, the experiments were performed using different data. This procedure was then repeated for the other version of the GUI. The calculator benchmark was an exception to this procedure since the measurements for the original and energy-efficient versions of the GUI were interleaved for the four experiments.

3) Hardware: The PDA used in our experiments was a Sharp Zaurus SL-5500 [19] running Embedix Linux and Qtopia. It is equipped with a 206MHz StrongARM SA-1110 CPU, 64MB SDRAM and 16MB FLASH ROM. It has a 3.5" 240x320 reflective TFT LCD with 16-bit color. The experiments were performed in a well-lit office. Hence, the front light was turned off. The GUI benchmarks were based on the Qt/Embedded GUI platform.



(a) Original (b) Energy-efficient

Fig. 1. Personnel viewer example



(a) Original (b) Energy-efficient

Fig. 2. Calculator example

Power measurements were made using an Agilent 34401A digital multimeter connected to a PC running Windows. Power was determined by measuring current through a 0.1Ω resistor connected in series with the PDA.

B. Benchmarks

Three GUI benchmarks were used to examine and verify different features of the E^2GUI design techniques. Here, we describe their functionality, implementation, and the tasks to be performed.

1) Text-reader with screen buttons: The first benchmark was based on the text editor included in Qtopia-1.5.0 [20]. The original version of the text-reader looks and functions the same way as a standard text editor, but we were only interested in the reading and browsing capabilities. To create an energy-efficient text-reader, we made the entire screen act as a pair of up and down buttons, called *screen buttons*. Thus, a user can browse by touching the screen anywhere in its top half and scroll down by touching it anywhere in its bottom half.

For this benchmark, two experiments were performed on each of the original and energy-efficient implementations. In each experiment, the subjects were given a line of text to look up within a short story.

2) Personnel viewer with multiple techniques: The second benchmark was based on a personnel viewer application. The personnel viewer allows a user to scroll through a list of names and display information about selected individuals. Radio buttons at the top allow a user to select between position, affiliation, and health queries. In this example, a straightforward implementation is compared with an energy-efficient implementation. Fig. 1 depicts the main screen for the two implementations. The original GUI requires two button presses to display information of the selected individual: the radio button at the top to select the data to be displayed and the pushbutton on the bottom to process the selection. The energy-efficient version replaces this two-button combination with a single set of pushbuttons on the top. The energy-efficient version also replaces the scrollbar with up and down buttons and alphabetic index tabs. Finally, the energy-efficient version has a low-power color scheme.

For this benchmark, two experiments were performed on each of the original and energy-efficient implementations. In each experiment, the subjects were given a name to look up and a query selection to make.

3) Calculator with input caching: The third benchmark was based on the calculator included in Qtopia-1.5.0 [20]. The original version of the calculator looks and functions in the same way as a real calculator. To create an energy-efficient calculator, we implemented an input cache button on each of the digit buttons. The input cache buttons store the most recent user input that begins with that digit. Fig. 2 depicts the two implementations of the calculator. It can be seen from the figure that a small portion of each digit button is designated as an input cache button in the energy-efficient version. For example, since the last value entered was 512, the input cache button on the 5 button displays 12 to indicate that pressing that button will enter 512. The energy-efficient calculator also makes use of a low-power color scheme.

A set of four experiments were performed on each of the original and energy-efficient implementations. In each experiment, the subjects calculated the product of a series of numbers, which had a repetitive pattern to make them easier to remember. The input vectors are listed below:

1) $11 \times 11 \times 21 \times 21 \times \cdots \times 91 \times 91$
2) $192 \times 192 \times 292 \times 292 \times \cdots \times 992 \times 992$
3) $1927 \times 1927 \times 2927 \times 2927 \times \cdots \times 9927 \times 9927$
4) $19273 \times 19273 \times 29273 \times 29273 \times \cdots \times 99273 \times 99273$

V. Experimental Results

In this section, we present the results of the experiments on the three benchmarks.

A. General results

The average improvement in performance for the text-reader was 23.7% and the average system energy saving was 26.9%. The improvements in energy and performance for this GUI come from replacing the scrollbar with the screen buttons. Screen buttons are energy-efficient and convenient since they are large and do not require users to move their eyes away from the text.

The average improvement in performance for the personnel viewer benchmark was 34.6% and the average system energy saving was 45.2%. These values are better than the results for the text-reader and demonstrate that complex GUIs can benefit from utilizing multiple energy-efficient design techniques.

Further analysis of these benchmarks revealed that scrollbars are highly inefficient. The power curves of the original GUIs (using the scrollbars) had more peak power spikes since tracking the stylus requires continuous computation and screen updates. Furthermore, several subjects reported that it was more difficult for them to use scrollbars (with the stylus) for browsing. They preferred using the buttons for both GUIs. Scrollbars were invented for desktop GUIs and these experiments demonstrate that they are not as well suited to handheld GUIs.

B. Calculator with input caching

Fig. 3 depicts task energy improvements of the E^2GUI using a trend line over all four trials for each subject. The X axis indicates the trial number and the Y axis indicates the percent improvement of the E^2GUI over the original. Si, $1 \leq i \leq 6$, denotes the subject number. The S4 line, corresponding to the fourth subject, is the result of a user who was slower to get used to the new GUI.

Fig. 3 demonstrates the effects of adding a user input cache. Similar to cache theory for microprocessor caches, it is evident from the figure that the effectiveness of the

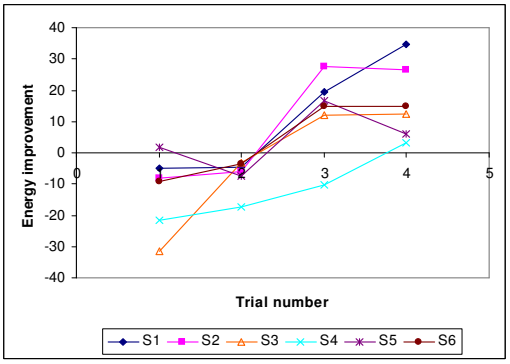


Fig. 3. Calculator experimental results

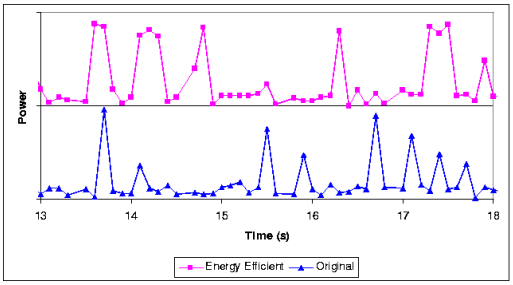


Fig. 4. Excerpts from the power data for Calculator experiments

user input cache depends on cache hits saving more time than the penalties they introduce (due to smaller digit buttons). Thus, the original is better for smaller inputs and the E^2GUI is better for larger inputs. These results are consistent with Fitts Law and cache theory.

For trial 3, the average performance improvement was 14.2% and the average system energy saving was 13.4%. For trial 4, these values were 19.3% and 16.4%, respectively. Note that there is energy reduction only when the inputs are sufficiently long. Thus, for applications such as text messaging, the cache should not include short words. The same rule applies to the auto-completion mechanism that is widely used in a virtual keyboard.

In comparing the time improvements (not depicted) with the results in Fig. 3, it was found that average power increased slightly for the E^2GUI. This is the result of using input caches, which reduce the size of buttons. Further experiments revealed that smaller buttons are more difficult to hit and require the user to spend more time in contact with the buttons. The PDA tracks the user inputs longer, thus consuming more energy. Fig. 4 presents this phenomenon graphically using two excerpts for one subject. In this figure, the X axis indicates time and the Y axis power. The wider peaks in the new GUI (on top) demonstrate this effect. Note that overall system energy still went down.

VI. Conclusions

The examples presented in this work utilize a variety of techniques focused on two common themes, reducing display power and increasing user interaction speed. Although energy-efficient color schemes reduce power, it is more important to avoid scrollbars and GUIs that require stylus tracking. Small buttons should also be avoided since their small size causes the user's stylus to linger longer on the touch-screen, requiring more computation energy.

The most effective way to improve energy efficiency is improving the latency caused by interfacing with humans. The simplest method for accomplishing this is to use fewer buttons and make them as large as possible. The subjects greatly benefited from using the index tabs in the personnel viewer application, indicating that designing GUIs to allow the user to capitalize on context-sensitive information improves performance. Finally, user input caches are effective if they can save many keystrokes, but should not be included otherwise. For PDAs, with a limited screen space, adding a cache button consumes part of the screen and reduces GUI efficacy if there are not enough cache hits.

Based on these observations, we make the following recommendations to GUI designers for systems with tight energy budgets.

- Reducing user interaction time should be the primary goal.
- Avoid vestiges of desktop systems (i.e., scrollbars, extraneous animations, etc).
- Make full use of available screen space (i.e., button size and placement).

In conclusion, the results demonstrate that proper GUI design can improve system energy efficiency without sacrificing application performance, ease of use or aesthetics. Our results indicate that E^2GUI design techniques can reduce system energy consumption by up to 45% while increasing performance by up to 35%. These results indicate that such techniques can contribute to prolonging the battery lifetime of mobile computing systems.

References

[1] J. M. Rabaey and M. Pedram, Eds., *Low Power Design Methodologies*. Kluwer Academic Publishers, 1995.
[2] J. R. Lorch and A. J. Smith, "Software strategies for portable computer energy management," *IEEE Personal Communications Magazine*, vol. 5, no. 3, pp. 60–73, June 1998.
[3] J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications," in *Proc. Symp. Operating Systems Principles*, Dec. 1999, pp. 48–63.
[4] L. Zhong and N. K. Jha, "Graphical user interface energy characterization for handheld computers," in *Proc. Int. Conf. Compilers, Architecture, and Synthesis for Embedded Systems*, Oct. 2003, pp. 232–242.
[5] ——, "Dynamic power optimization of interactive systems," in *Proc. Int. Conf. VLSI Design*, Jan. 2004, pp. 1041–1047.
[6] J. Lorch, "A complete picture of the energy consumption of a portable computer," Master's thesis, University of California at Berkeley, 1995.
[7] T. L. Cignetti, K. Komarov, and C. S. Ellis, "Energy estimation tools for the Palm," in *Proc. ACM Int. Workshop Modeling, Analysis and Simulation of Wireless & Mobile Systems*, Aug. 2000, pp. 96–103.
[8] J. Flinn, K. I. Farkas, and J. Anderson, "Power and energy characterization of the Itsy pocket computer (version 1.5)," Compaq Western Research Laboratory, Tech. Rep. Technical Note TN-56, Feb. 2000.
[9] R. P. Carver, *Reading Rate: A Review of Research and Theory*. San Diego, CA: Academic Press, Inc., 1990.
[10] P. Buser and M. Imbert, *Vision*. the MIT Press, 1992, Translated by R. H. Kay.
[11] W. E. Hick, "On the rate of gain of information," *Quarterly J. Experimental Psychology*, no. 4, pp. 11–36, 1952.
[12] R. Hyman, "Stimulus information as a determinant of reaction time," *J. Experimental Psychology*, no. 45, pp. 188–196, 1953.
[13] A. Sears and B. Schneiderman, "Split menus: Effectively using selection frequency to organize menus," *ACM Trans. Computer-Human Interaction*, vol. 1, no. 1, pp. 27–51, Mar. 1994.
[14] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *J. Experimental Psychology*, vol. 47, no. 6, pp. 381–391, June 1954.
[15] S. Zhai, S. Conversy, M. Beaudouin-Lafon, and Y. Guiard, "Human on-line response to target expansion," in *Proc. Conf. Human Factors in Computing Systems*, Apr. 2003, pp. 177–184.
[16] B. B. Bederson, "Fisheye menus," in *Proc. Symp. User Interface Software & Technology*, Nov. 2000, pp. 217–225.
[17] J. Accot and S. Zhai, "More than dotting the i's — foundations for crossing-based interfaces," in *Proc. Conf. Human Factors in Computing Systems*, Apr. 2002, pp. 73–80.
[18] I. S. MacKenzie and S. X. Zhang, "The design and evaluation of a high-performance soft keyboard," in *Proc. Conf. Human Factors in Computing Systems*, May 1999, pp. 25–31.
[19] Sharp Zaurus SL-5500 PDA, *http://www.myzaurus.com/*.
[20] Qtopia, *http://www.trolltech.com/products/qtopia/*.