EXHIBIT B

Subject: Paper and Submission

Date: Saturday, September 11, 2004 2:24:22 PM ET

From: Amy Karlson

To: 'John SanGiovanni', Bederson, Ben

Hi,
Attached is what I intend to submit. I'm not sure anyone has the stomach to read it another time, but I will do so. Just wanted to send it FYI.

John – Sorry for not responding yesterday – I too was at the HCIL retreat. Thanks for you comments on the paper. Your extra information about the Shneiderman reference was just what I needed to find a paper that presents results from a study showing drag to select to be more accurate than tap to select on touchscreen devices.

Ben – Do we have a way for John to upload a 30M file?

Thanks,
Amy

Two Designs for One-Handed Thumb Use on Small Devices: AppLens and LaunchTile

ABSTRACT

We have designed two interfaces to support one-handed thumb use for PDAs and cell phones. Both use Scalable User Interface (ScUI) techniques to support multiple devices with different resolutions and aspect ratios. The designs use variations of zooming interface techniques to provide multiple views of application data: AppLens uses tabular fisheye to access to nine applications, while LaunchTile uses pure zoom to access thirty-six applications. Two sets of thumb gestures represent different philosophies for one-handed interaction. We conducted two studies to evaluate our designs. The first study explored whether users could learn and execute the AppLens gesture set with only minimal training. Participants performed more accurately and efficiently using semantic gestures for directional navigation than abstract gestures for object interaction. A second study gathered user reactions to each interface, as well as comparative preferences. With minimal exposure to each design, most users favored the tabular fisheye interface.

Author Keywords

One-handed, mobile devices, gestures, application notification, Piccolo, thumb navigation, Zoomable User Interfaces (ZUIs).

ACM Classification Keywords

H.5.2. User Interfaces: Input devices and strategies

INTRODUCTION

The current generation of mobile computing hardware features a variety of devices for interaction with the system software. Many of the devices, typically classified as “smartphones”, feature a numeric keypad or miniature thumb keyboard for input together with a ruggedized screen for display output. These devices have the advantage of single-handed interaction, but most smartphones lack a touch-sensitive display, limiting the options for interaction design to simple directional navigation. Another design approach, typically classified as a “Personal Digital Assistant” (PDA) features a touch-sensitive display surface designed primarily to be used with an included stylus. This design offers greater software design flexibility, but many of the small targets designed for a stylus are too small for fingertip actuation, making one-handed use difficult or impossible.

Our goal is to create a new single-handed interaction system for both smartphone and PDA devices. In addition to providing support for varied input hardware, our designs were also informed by Scalable User Interface (ScUI) techniques [1,3]. This design philosophy allows the system to adapt to different screen resolutions, and supports either portrait or landscape device rotation. Such an architecture could allow developers to create individual applications that target a wider variety of screen resolutions. The display size range of this design study was defined to target screens from 2” to 5”, measured diagonally, with resolutions ranging from 176x220 to 800x600. This is accomplished by using the University of Maryland’s Piccolo.NET development toolkit for zooming and scalable user interfaces [4,16].

The secondary goal of this project is to create a new software development framework that is informed by recent trends in mobile device software development. Most current PDA software interfaces are designed for focused interaction with a single task or application, with limited consideration or display real estate allocated for notifications (instant messages, SMS/MMS, email, appointment reminders, voicemail) or monitoring of ambient information streams (RSS feeds, web services, stocks, weather, sports scores). In our proposed design, each application has a dynamic information launch tile in the place of a static launch icon. This feature provides the user the ability to glance at the interface and get quick access to current information with little or no device interaction.

The primary focus of this paper addresses the shell and notification area. We approach these goals by describing two designs: AppLens (characterized by zoom+fisheye) and LaunchTile (characterized by zoom+pan). The two approaches employ variations of zooming interface techniques [5] to overview several notification tiles, each roughly the size of a postage stamp. AppLens uses a tabular fisheye approach to provide integrated access to and notification for 9 applications. LaunchTile uses pure zooming within a landscape of 36 applications to accomplish the same goals.

Fisheye and pure zoomable techniques both offer promise, but there are no clear guidelines as to when each

approach should be used. So our approach in this work has been to design and build the best interfaces we could with roughly the same functionality, and then compare and contrast the results. In this way, we hope to understand which design direction makes the most sense for this domain and to learn something about the trade-offs between the two approaches.

For device interaction when using a touch-sensitive screen, both designs utilize a basic gestural system for navigation within the notification & application zoom canvas. While this paper does not address single-handed text input techniques, we included a modular InputTile concept in order to support a variety of one-handed input systems, including single and multitap alphanumeric keypad input, as well as miniature thumb keyboards and unistroke input systems executed with a thumb (e.g., Graffiti[6], Quikwriting[15]).

RELATED WORK

Gestures have proven a popular interaction alternative when hardware alone fails to effectively support user tasks, typical of many nontraditional devices, from mobile computers to wall-sized displays [9]. Gestures can be very efficient, combining both command and operand in a single motion, and is space-conserving, reducing the need for software buttons and menus. However, the invisible nature of gestures can make them hard to remember, and recognition errors can negatively impact user satisfaction [13]. Recent research efforts pairing gestures with PDA-sized devices, have emphasized gestures based on changes in device position or orientation [10,19,25]. However, our work more closely resembles the onscreen gestures that have played a prominent role in stylus-based mobile computing. Such efforts can be roughly categorized as application-specific (e.g. Power Browser [7]) or general purpose (e.g., Quikwriting [15]).

Our thumb-as-stylus designs support usage scenarios in which only one hand is available for device operation, such as talking on the phone or carrying a briefcase. Although existing stylus-based gesture systems do not preclude the use of the thumb, we are not aware of any systems that have been specifically designed for the limited precision and extent of the thumb. One handed device interaction has typically focused on text entry techniques, but beyond one-handed text entry on a numeric keypad, most one-handed text entry systems, including those specifically designed for thumb-based entry, require specialized hardware, such as accelerometers [23,27] or customized keyboards [14]. Our one-handed designs target existing hardware platforms and address the general task of system navigation and interaction, in a manner that is both device and application independent. The Jackito PDA [12] supports thumb-only interaction, but presumes two handed use and is not gesture oriented.

THE ZOOM+DISTORT APPROACH: APPLENS

AppLens provides notification and one-handed access to nine applications, and is strongly motivated by our earlier work on DateLens, a tabular fisheye calendar [3]. We refer to AppLens as a “shell” application for its role in organizing and managing access to other applications

Generalized Data Access Using Tabular Fisheyes

Spence and Apperley [8] introduced the “bifocal display” as one of the first examples of fisheye distortion applied to computer interfaces. Furnas extended the bifocal display to include cognitive perceptual strategies and introduced a set of analytical functions to automatically compute generalized fisheye effects [8]. Since then, fisheye distortion has been applied with mixed success across a variety of domains, including graphs [22], networks [24], spreadsheets [18], and documents [11,21].

Bederson et al. [3] drew upon prior work in fisheye calendars and tables in developing DateLens, a space-conserving calendar for personal digital assistants (PDAs), which performed favorably for long-term scheduling and planning tasks when compared to a traditional calendar implementation. The strength of DateLens lay partly in the pairing of distortion and scalability, which allowed the details of a single day to be viewed in the context of up to several months of appointment data, and partly in the design of effective representations for the variety of cell sizes and aspect ratios resulting from tabular distortion. One drawback of DateLens, however, was that it required two hands, with many small widgets requiring a stylus to use. The current designs extend the principles of DateLens to include one-handed thumb access and generalize the design for application across a variety of domains.

We developed a general framework that provides a grid, tabular layout, and default views for cell contents at a variety of sizes and aspect ratios. We also developed a general API to make it as easy as possible for applications to be built within this framework: developers replace a small number of cell views with representations that are meaningful within the target domain.

AppLens Notification

The AppLens shell has been implemented within our generalized tabular fisheye framework, using a 3x3 grid, and assigning one of nine applications to each cell. The support for tabular layout includes notification at 3 zoom levels: *Notification*, *Context* and *Full*.

Notification distributes the available screen real estate equally among the 9 application tiles (Figure 1a). One tile remains reserved for settings, which can be used to configure the selection of applications which occupy the other 8 notification tiles. Generally, tiles at *Notification* size display high level static and/or dynamic application-specific notification information.

Context zoom allocates roughly half the available display area to a single tile, compressing the remaining tiles according to a tabular fisheye distortion technique [3,18] (Figure 1c). A tile at *Context* size typically appears much like a fully functional application, but selectively displays features to accommodate display constraints, and is not interactive. Tiles on the periphery of a *Context* zoom, or *peripheral* tiles, may be rendered quite differently depending on their position relative to the detail tile, which dictates whether the peripheral tile is a square, a wide-flat rectangle, or a narrow-tall rectangle. To reduce visual overload at *Context* size peripheral tiles are displayed at 40% transparency. The contents of distorted peripheral tiles are not themselves distorted, but rather change representation to provide the most meaning in the space available.

The third and final *Full* zoom level expands a tile to a fully interactive application that occupies 100% of the display (Figure 2b).

Gesture-Based Cursor Navigation

Existing application designs for PDAs are often inappropriate for one-handed use due to their reliance on screen regions that typically cannot be reached while maintaining control of the device (e.g., accessing the start menu in the upper left hand corner of a display while holding the device in the right hand), and the use of standard widgets that are too small for reliable thumb use (e.g., radio buttons, checkboxes, onscreen keyboards). LaunchTile solves this problem by redesigning application access and interaction such that all targets are both large and within thumb reach. In support of more traditional designs, AppLens instead uses an object cursor, depicted as a dynamically sized rectangular orange border that users move from one on-screen object to another via command gestures issued anywhere on the PDA screen. Although PDAs do not traditionally support cursors as desktops do, our cursor identifies the on-screen object that is the current interaction target. However, cursors are not new to PDA interface design: the WebThumb [28] web browser includes a similar notion of cursor, but which is controlled via directional hardware, and others [26] have explored device tilting to manipulate PDA cursors.

Neither the cursor nor gestures interfere with the most common stylus interactions of tap and tap+hold. Although gestures do overlap stylus drag commands, dragging is rarely used in applications and could be distinguished from gestures by explicitly setting a gesture input mode.

We established a core set of commands that would allow users to navigate applications using only the input cursor. Our command language supports directional navigation (UP, DOWN, LEFT, RIGHT) as well as two widget interaction commands: one equivalent to a stylus tap (ACTIVATE), and the other which negates activation (CANCEL), equivalent to tapping the stylus outside the target widget. We also include the convenience commands FORWARD and BACKWARD, equivalent to TAB and ALT-TAB on Windows PCs.

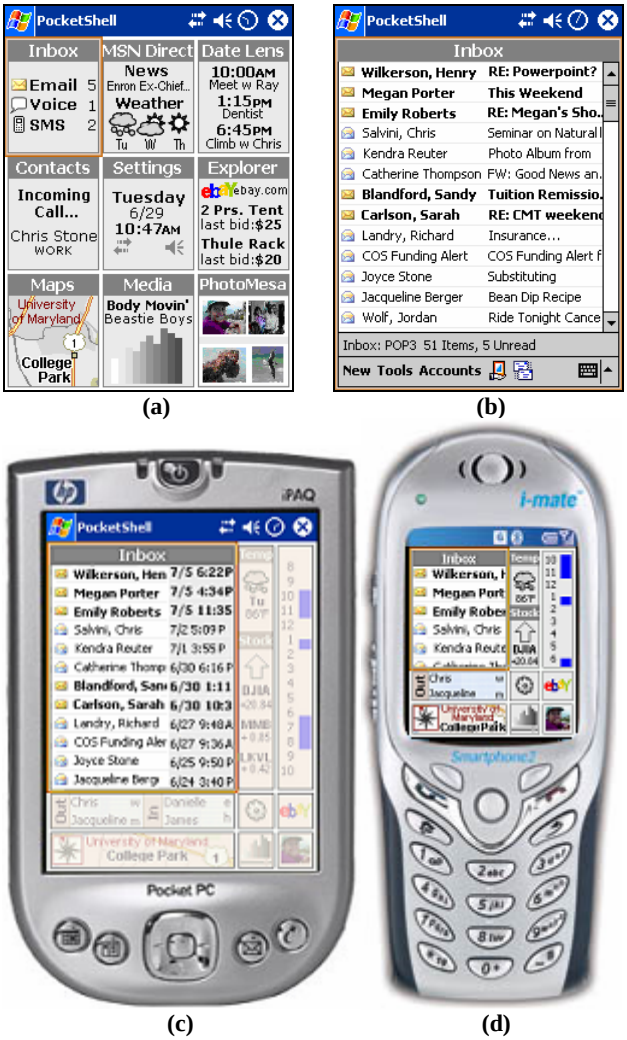

Figure 1. AppLens Zoom Levels: (a) Notification, (b) Full, (c,d) Context.

Command Gestures

Motivated by the limited mobility of a person's thumb, we based the gesture set on the portion of the screen a typical user would be easily able to access with their thumb using one hand. This resulted in a design that used just the lower right portion of the screen as shown in Figure 2 (the mapping would be reflected for left handed use). By dividing the pie-shaped area into four concentric tracks, and subdividing each track radially into sectors, we envisioned a hierarchical input mechanism whereby commands were defined by the order of sectors crossed along a path from the innermost track to the outermost track. Varying the restrictions on the number of sectors the path visits within a single track (without repeats)

yields between 21 and 801 unique paths. A mid-complexity path rule that allows for zero or one adjacent sectors to be visited within a single track before advancing to the next track yields 259 paths, and is illustrated with a partial calculation in Figure 3. A redesign of the map increased the sector sizes and strategically mapped sectors to the traditional keypad (Figure 4) to support both PDA and smartphone.

Figure 2. Screen area accessible with one hand .

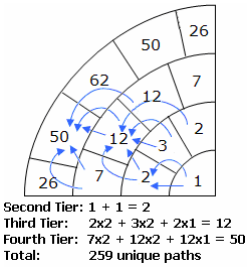

Figure 3. Gesture space calculation.

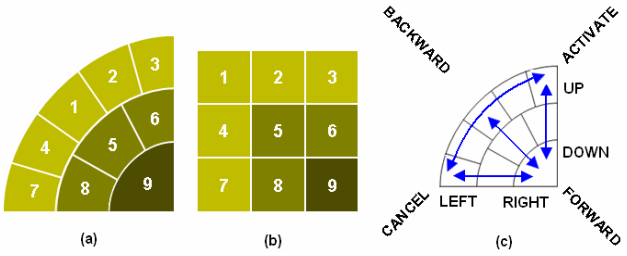

Figure 4. The final gesture map. Sectors in (a) correspond to keypad numbers in (b), and relate to gestures as in (c).

Mapping Commands to Gestures

A drawback of the hierarchical input design is its unidirectional nature, which prevents intuitive mappings for commands such as right and down, since all commands start in the lowest, rightmost part of the screen. We also were concerned that it would be too difficult to learn and use such a complicated design – especially with a thumb. We instead developed a simpler set of gestures to maximize memorability and robustness of execution. We assigned semantic mappings for the directional commands UP, DOWN, LEFT and RIGHT. Two gestures defined by pivoting the thumb from bottom to top and top to bottom we assigned to ACTIVATE and CANCEL respectively, both to reinforce their opposing relationship as well as for ergonomic ease in issuing common commands. Finally, we assigned the upper-left to lower-right diagonal to FORWARD due to its relative forward nature, and by similar reasoning, the reverse gesture to BACKWARD. Each gesture is uniquely defined by a slope and direction, or vector, which allows gestures to be robust and highly personalizable: users can issue gestures of any length (beyond an activation threshold of 20 pixels) anywhere on the screen surface.

The eight vector gestures can be superimposed on the original sector design, each defined by the sectors in which the gesture starts and ends. (Figure 4c). Using the numeric key that corresponds to the sector in which the gesture ends yields a mapping of 3-ACTIVATE, 7-CANCEL, 9-FORWARD, and 6-BACKWARD. Since smartphones also have a joystick which can be used for UP, DOWN, LEFT and RIGHT, there is no overlap among commands.

Using Command Gestures within AppLens

AppLens zoom levels can be changed using ACTIVATE and CANCEL gesture commands. The ACTIVATE command gesture propagates the equivalent of a stylus tap to the current input target, hence its effects are target-specific. The ACTIVATE gesture zooms in, animating the layout first from *Notification* to *Context* zoom, and then to *Full* zoom. Once at *Full* zoom, the input cursor will transition to the objects *within* the application, at which point the command gestures will affect the current target, not the application itself. In this way the AppLens architecture is hierarchical. The CANCEL command acts to negate the effects of the ACTIVATE command. At *Full* zoom, the effects of the CANCEL command depend on the location of the cursor and the state of its target. If the current target is an internal target in an activated state, the CANCEL command deactivates the target. If, however, the target is in an inactive state, the CANCEL command causes the application tile to animate from *Full* zoom to *Context* Zoom, and if issued again, to *Notification* zoom.

THE ZOOM+PAN APPROACH: LAUNCHTILE

Our second design, LaunchTile proposes another way to interact with a grid of notification tiles. The primary shell of the LaunchTile design is an interactive zoomspace consisting of 36 notification tiles, divided into 9 zones of 4 tiles each. As a central design element, LaunchTile uses a large blue onscreen button (called Blue) to provide a consistent point of reference for zooming & panning navigation, and to unify the shell and applications with a consistent visual metaphor. Onscreen tiles, menus, and functions maintain a consistent relative position to Blue.

Home Screen

The Home screen of LaunchTile divides the screen area into 4 equally-sized notification tiles (Figure 5a). The Home screen represents the 4 center-most tiles of the 36-tile zoomspace. To see other tiles, the user pans around the space, and the system snaps to each 4-tile group, called a Zone. In this paper, the zones are described with a numerical designator from 1 to 9, starting in the upper left, and going across. This maps to the numbers on a conventional telephone keypad. Zone 5 is the center zone, which defines the home screen, and shows the 4 highest priority notification tiles, as configured by the user.

Figure 5. Three LaunchTile zoom levels: (a,b) Home, (c) World View, (d) Application

Panning Techniques

To support different input hardware and various styles of interaction, there are several ways to pan around the zoomspace. If the device has a multidirectional control joystick, the user can push it in the direction of the targeted zone. This design permits the user to quickly glance at the 4 highest priority tiles with no interaction, and the 16 tiles in zones 2, 4, 6, and 8 are a single tap away. As many directional pads do not support diagonal action, the 16 additional tiles in the corner zones 1, 3, 7, and 9 are two taps away. As with AppLens, the zoomspace does not wrap. There are also three other panning techniques. If the device has a touch-sensitive display, the user can use his or her thumb to drag the canvas around. When doing so, dragging action is "on rails", permitting the user to drag vertically and horizontally, but not diagonally. When panning, Blue remains stationary, and snaps the view to the center of each zone, represented by an empty circular space in the hub of 4 tiles in the zone. These virtual guides help keep the user from getting caught between zones.

Within each 4-tile zone, we've also designed an indicator widget to show the user's relative location within the zoomspace. The indicator has two components. First, directional arrows show where other zones are. If the user only sees indicators pointing up and right, they know they are in zone 7. Next to the directional arrows are blue dots that represent all zones not currently in view. The blue dots could also be used to indicate an alert or status change in a neighboring zone, though this feature was not implemented in our prototype. The fourth way to pan around the zoomspace is to press this directional indicator itself. An oversized hit target ensures that the user can easily hit this without using a stylus.

Zooming out to the World View

From any one of the nine 4-tile zones, the user also has the option to zoom out to view the entire 36-tile zoomspace (Figure 5c). Since all 36 tiles are visible at once, this view reduces each tile to a small launch icon or miniature visualization. From this view, the user can clearly see the absolute location of each tile. This view, called the World View, is a way to monitor all applications at once. In the World view, the display is divided into a grid of 9 hit targets, each mapping a 4-tile zone. Single-tapping a zone animates into that zone, and displays the zone's 4 notification tiles.

Zooming In to an Application

From any zone, the user taps any of the 4 notification tiles to launch the corresponding application. The point of view follows a zooming animation further into the zoomspace until the target application fills the entire display (Figure 5d). If the device has only a numeric keypad (no touchscreen), the user simply presses the numeric key in the corner that corresponds to the zone. Pressing 1 launches the upper left tile, 3 launches the upper right, 7 launches the lower left, 9 launches the lower right. This provides quick, single-tap access to each visible tile. This technique is inspired by ZoneZoom by Robbins et al. [20].

As the system zooms, Blue stays in view and retains its function as a central point of reference. Application menu commands are represented as onscreen buttons clustered around Blue, now positioned at the bottom of the display. Each menu button has its appropriate numeric label to further illustrate its mapping to a mobile phone keypad. This allows a smartphone user to access application commands by pressing the corresponding keypad number.

A visual indicator to the left of the zoomspace animates during interaction to reflect the user's current absolute zoom level within the LaunchTile environment.

Zoom Control

In every screen but Home, the user presses Blue to move deeper into the object hierarchy. A dedicated Back button, both onscreen and in hardware, takes the user back out. In the Home screen, Blue toggles between the two different notification views, Home (4 tiles are visible) and World View (all 36 tiles are visible). Once an application is launched, three dedicated software buttons along the top edge of the screen support inter- and intra- application navigation. A green Home button immediately zooms the view out to Home view. There is also a Back button on the upper right edge of the screen, and another global command key. The placeholder for this function in our prototype is an icon for voice command & control. On a non-touchscreen device, Back and Home commands are executed with dedicated physical buttons, such as those provided on a smartphone device.

Application-Level Interaction

Although the focus of these designs was notification and shell interaction, we continued our interaction philosophy throughout the depth of zoomspace, into the application level. In the LaunchTile prototype, we attempted to make application-level interaction consistent with navigation among the notification tiles. Within an application, several gestures are supported which are designed specifically for single-handed navigation and item selection. Previously, others have demonstrated that for direct-manipulation interfaces, a grounded tap-drag-select-release technique is more accurate than a tap-to-select [17]. As such, we made all LaunchTile tap-to-select targets at least 35x35 pixels. In those areas where limited display real estate necessitates smaller targets, the central Blue widget becomes a moveable toolglass which can be positioned over the desired object, email, or text. The large thumb-friendly drag target is offset below the selection area, to keep the user's thumb from occluding the target. Alternatively, the user can drag anywhere around Blue to move the application space. This technique can be used to pan around a map, scroll a list of emails, or to navigate to text outside the view area. Together, these two interaction techniques permit the user to address a large application zoomspace, yet bring focus to a target much smaller than a finger. A non-touchscreen user simply uses the multidirectional joystick to position the selection area on the screen. Keys 2 and 8 are used for page control up & down, and 4 and 6 can be either menu items or horizontal page control, as appropriate.

Once the targeted item is in the selection area, a tap on Blue (also supported through hardware via the 5 key) replaces the application menus with a context-sensitive menu, now clustered immediately around the selection area. Like the menu items, each key on a numeric keypad executes its function as relative to Blue. A second tap on Blue sends the user to the deepest level of the zoomspace, text input. At this level, a modular text input object called an InputTile appears around the selection area for alphanumeric input. If the user has a mini qwerty keyboard, they can alternatively use this for directly entering text. With this design, a double tap on Blue while text is selected will bypass the context menu and take the user directly into text edit mode.

IMPLEMENTATION

The AppLens and LaunchTile prototypes have been built using the University of Maryland's PocketPiccolo.NET development toolkit for Zoomable User Interfaces (ZUIs) [4,16]. Although the primary development and test platform has been the HP iPAQ PocketPC running Microsoft Windows Mobile 2003, both run unmodified on the iMate Smarphone II running Windows Mobile Smartphone 2003 (Figures 1d and 5b).

Although the core architecture and gesture recognition for each shell has been implemented, applications have been simulated with images. This has allowed us to put designs that are faithful to the look and feel of each shell in the hands of users for early feedback, but falls short of full interactivity. However, because LaunchTile design principles extend to the applications themselves, we have included email and mapping as two interactive examples within the LaunchTile prototype.

STUDIES

We conducted two studies to inform future research directions. The first study explored whether users could learn and perform the AppLens gesture set with only minimal training. The second was a usability study which gathered user reactions to each shell design, as well as their comparative preferences.

APPLENS GESTURE STUDY

A significant difference between the gesture designs of each shell is that all gestures in LaunchTile are direct manipulation, while all gestures in AppLens are remote-control commands. In LaunchTile, users physically drag objects with their thumbs, be it the zoomspace, a toolglass, a map, or a list of email headers. In AppLens, gestures only affect the current cursor target, rather than the objects over which the gesture is performed. The success of AppLens therefore depends as much on the utility of the tabular design as the ability for users to execute the gesture command set. Participants were tested on gesture execution performance and efficiency in navigating an information space.

Participants: 20 participants (12 male, 8 female) were recruited from the general population with the only selection criteria being that participants were right handed. The median participant age was 30, and while 12 of the participants considered themselves advanced computer users, only 6 regularly used a PDA.

Materials: The study was run on an HP iPAQ measuring 4.5 in x 2.8 in x 0.5 in with a 3.5 inch screen. We constructed a software test environment modeled after AppLens: a hierarchical tabular information space, with each level in the hierarchy represented by a 3x3 grid of equally-distributed cells, numbered 1-9 like a smarthphone keypad. *Context* zoom was eliminated from the test environment, so that activating a cell animated the display to the next level. Cell labels served as navigational landmarks, each labeled with a hierarchical address constructed by prepending the cell's local logical number (1-9) with its parent's label, separated by a dot. We reserved an area at the top of the screen for task instructions, and disabled tapping to restrict input to gestures alone. Gestures retained their meaning from AppLens: Navigational gestures LEFT, RIGHT, UP and DOWN controlled the movement of an orange rectangular cursor within a level; ACTIVATE zoomed in to the next level and CANCEL zoomed out to the previous level. Within the context of the text environment, FORWARD and BACKWARD were used for intra-cell selection, allowing participants to cycle a highlight either forward or backward through cell label digits. Highlighted digits could then be bolded or "activated" using the ACTIVATE gesture or un-bolded or "deactivated" using the CANCEL gesture. Participants were provided with a reference sheet that described the hierarchical information space and labeling scheme, as well as the eight gestures to be used for navigation and interaction.

Tasks: Participants performed two types of tasks. Gesture tasks involved performing a gesture given the associated command name. Navigation tasks required participants to navigate to a particular cell in the hierarchy and/or to activate or deactivate a cell label digit.

Measures: Application logs recorded task time, the gestures performed for each task, and whether the task was completed correctly. Participants were instructed to press a hardware button between tasks, which served to record task time and advance to the next task. Due to a bug in our logging software, task time was recorded at second rather than millisecond resolution. However, since our goal was to identify performance trends rather than comparison to another input method, one second resolution was sufficient. Participants rated their experience using a 9-point Likert scale: (1) frustrating – satisfying, (2) hard to learn –easy to learn, (3) hard to use – easy to use, (4) slow – fast, and (5) boring – fun.

Procedure: After reading a description of the test environment and gestures, participants performed 16 practice tasks similar in nature to those in the navigation task phase. Participants practiced for 5-10 minutes.

After the practice phase, participants performed gesture tasks: presented with a command name, participants performed the associated gesture and pressed a hardware button to advance to the next task. Command names were presented to participants in random order, four times for each of the eight commands. The gesture reference sheet was placed face down so that the administrator could record the number of times it was looked at.

The second test phase required participants to perform goal-directed tasks within the information space. These included navigation, activation, navigation+activation, and cancellation tasks. Participants then recorded their subjective ratings of the interaction experience.

Results

In the gesture phase of the study, participants correctly performed gestures an average of 87% of the time. However, looking at gesture type, we see that participants correctly performed directional gestures 93% of the time, but had more difficulty with diagonal gestures ACTIVATE and CANCEL at 88% and 85% respectively. BACKWARD and FORWARD had the worst success rate, at 70% and 64% of the time respectively. Time to perform gestures followed a similar trend. On average, participants required 2.4 seconds to perform each gesture: 1.5 - 1.7 seconds for directional gestures, 2.6 - 2.8 seconds for ACTIVATE and CANCEL gestures, and 3.6 – 3.7 seconds for BACKWARD and FORWARD gestures respectively. Neither measure correlated with computer or PDA experience. Although we tallied user peeks at a reference sheet, we assumed that acts of page flipping and answer searching have been reflected in the task time, and did not analyze this data further.

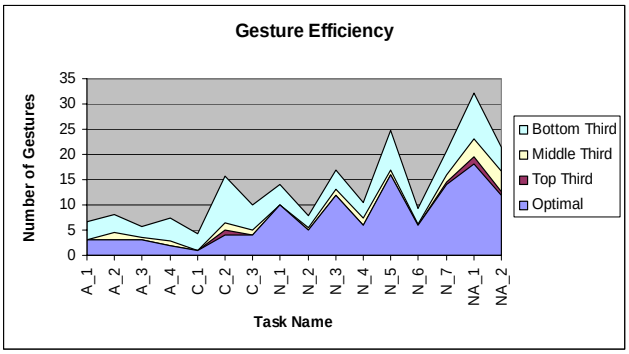

Figure 6. Average number of gestures per task compared with the optimum.

The second test phase evaluated accuracy and efficiency for goal directed navigation tasks. The average task success rate was 95%. While both navigation and activation+navigation tasks were performed with 98% accuracy, activation was close behind at 96%. Cancelation tasks, however, were only completed correctly 83% of the time. On average, participants performed 2.4 additional steps than the optimal number to complete a task. Breaking participants into three performance groups for each task, it is clear that one third of participants performed nearly optimally on all but a

few tasks (Figure 6). The next third of participants performed comparably to the first third, but the gaps were wider on the tasks the first third had trouble with. Thus, the most significant difficulties in performing the phase 2 tasks were experienced by only a third of the participants.

The subjective rating for each of our 5 satisfaction measures on a scale of 1-9 where 9 was positive all fell within a 1 point span of each other, between 5.9 (satisfying) and 6.75 (fun).

Analysis

Because the gesture phase of the study did not distinguish between errors of recall and execution, we could not classify the reasons for error. Analyzing the log files, it is safe to say that errors of both types occurred. The low error and speed measures for directional navigation support our hypothesis that the directional gestures are semantic in design, presumably contributing to better learnability, more reliable execution, and lower cognitive demand. The positive jump in execution speed for the other four gestures is unsurprising when we consider that the mappings between gesture and command are more abstract than the directional mappings, and likely require more cognitive effort to perform the mental transformation. This alone does not explain the associated increase in error rate, but insights from Long's work on gesture similarity [13] suggest that users may perceive the diagonal gestures as similar and therefore more difficult to learn.

The difference in performance data between ACTIVATE and CANCEL vs. BACKWARD and FORWARD may be attributed to the more physically challenging nature of latter two, but may also be due to disproportionate practice time received, considering a single navigation task provided more opportunities to issue ACTIVATE and CANCEL gestures than intra-cell activation tasks provided for FORWARD and BACKWARD. These intuitions also help explain the efficiency results – users were more successful and efficient in pure navigation tasks which contained a proportionally large number of directional gestures compared with intra-cell activation/cancellation tasks. The relative complexity of the gesture navigation environment may have confounded the results by inflating the efficiency measures, but we feel that it also made the results more relevant to the AppLens design.

APPLENS AND LAUNCHTILE USABILITY STUDY

The goal of our second study was to understand usability issues for new users of each shell design and to elicit general reactions and comparative preferences.

Participants: There were 10 participants (8 male, 2 female) from a local scientific research laboratory. 3 participants were in their 20s, 5 in their 30s, and 2 were 40 or older. While all participants considered themselves advanced computer users, 4 used PDAs regularly, and 4 had never used a PDA.

Measures: Participants provided subjective design-specific and comparative reactions to AppLens and LaunchTile through think aloud and questionnaires.

Materials: The shell prototypes were run on the same hardware used in the first study. A one-page document described the AppLens design and gestures set, followed by a list of eight associated tasks. A two-page document described the LaunchTile design and methods for navigation and interaction, followed by a list of 11 tasks.

Tasks: For each interface, participants performed tasks which were designed to exercise the full range of navigation and interaction features. For example, LaunchTile tasks included navigating to specific zones, finding specific applications, and opening and editing an email message. AppLens tasks included navigating to specific tiles, and answering questions about applications.

Procedure: Study participants were introduced to each design by reading its design document and performing each of the related tasks. During the tasks, the test administrator recorded think-aloud reactions and usability observations. After task completion, the administrator recorded answers to open-ended questions related to the usability of the interface, such as likes and dislikes, features that were easy or hard to use or learn, and comfort level. The same procedure was repeated for the second interface. The administrator balanced the order of the interfaces among participants. After interacting with both interfaces, participants were asked comparative preference questions.

Results

Perhaps due to the richness of the LaunchTile environment, reactions were complex as well as mixed. Because only one question focused on a specific interface design feature (AppLens context view), we regard commonality in participant responses indicative of the highest-impact interface characteristics. We report here on the strongest trends in opinion. Nearly half the participants reacted positively to the aesthetics of LaunchTile, and specifically to Blue. Six participants appreciated the amount of information available through the two zoom perspectives World and Home, and another six thought that zooming between those perspectives was one of the easiest aspects of the interface. Seven participants felt comfortable using one hand to perform the tasks, and eight felt they were able to effectively navigate within LaunchTile.

Surprisingly, a majority (7) of participants had difficulty with navigating by dragging within LaunchTile, commenting most often that it was unintuitive. Six participants struggled with the mulit-modal nature of

Blue, unsure about its role in different contexts, especially within applications. A related problem was that of differentiating between the roles of the Home, Back, and Blue buttons from within an application. The majority of participants were tentative and had difficulty performing tasks within the email application. Ultimately, participants were evenly divided (3 vs. 3) about whether they would choose to use LaunchTile for their own application management, with 4 participants abstaining.

Reactions to AppLens were more consistent. Half the participants commented that they liked the notification view and the ability to access all nine notification tiles within both *Notification* and *Context* views. Even though two participants found nothing redeeming about *Context* view, all others found the *Context* view useful at least some of the time. Seven participants enjoyed application navigation and found it easy to do, but participants performed the majority of navigation using tapping rather than gestures. Even so, participants were required to use gestures to zoom out from *Full* zoom, and 2 participants particularly liked the gestures. Five participants agreed that the gestures were the most difficult aspect of the interface, but disagreed on why, citing confusion over zoom-in vs. zoom-out gestures, difficulty performing the ACTIVATE gesture, difficulty navigating with gestures, and misrecognition of gestures. All participants found AppLens both easy to learn and effective for navigating among applications, all but one found one-handed use comfortable, and six out of seven participants stated they would prefer AppLens over their most familiar PDA operating system.

Comparing the two interfaces, most participants recognized the trade off between the number of applications that could be viewed at once and the amount of information conveyed for each, yet seven out of nine participants thought AppLens provided better at-a-glance value. Although nearly half the participants were reluctant to compare the speed of information access between the two interfaces due to the differing amounts of information available, seven out of nine participants thought AppLens supported faster data access. Additionally, AppLens was considered easier to use (7 out of 9), and 8 out of 9 would prefer AppLens for use on their own device. In response to our general question about the utility of one-handed use, 7 participants thought one-handed interaction would be useful at least some of the time, with 3 of those participants preferring one-handed to two-handed use in all situations. However 2 participants stated that they would never want to use a PDA with one hand, regardless of the interface design.

DISCUSSION

While AppLens appeared to be the preferred design, we must take care in assigning reasons for the preference. First, AppLens is a simpler design and a shallower prototype than LaunchTile. Its appeal may have been that users felt proficient and better able to manage 9 applications (versus 36) with minimal use. Its simplicity also made AppLens less prone to the performance limitations of the target hardware, which noticeably impacted LaunchTile zooming. However, more experience with the two designs might have tipped the scales in the other direction, as Bederson has pointed out that even complex interfaces have the potential to be highly satisfying after users have expended the effort to become expert users [2]. A vocal minority of expert PDA users who much preferred LaunchTile supported this possibility, citing the large number of applications and configurable layout as very attractive. Thus a different participant population may have offered different opinions. While we do not consider LaunchTile in this class of expert interface, clearly 10 minutes is not sufficient for users to become proficient with the variety of interaction techniques supported by the interface.

Bederson hypothesizes that user satisfaction is related to how well an interface supports “flow”, which correlates inversely to the degree to which an interface gets in the way of, or interrupts, user tasks[2]. Blue is an example of a LaunchTile feature that interrupts flow: it performs different functions in different contexts, requiring users to keep track of the system state to predict the outcome of tapping Blue. This type of functional overloading is a well-known design issue, but is commonly used regardless. For example, both the Microsoft and Apple desktop media players use a single button for both Play and Pause. Just as with LaunchTile, these designs compromise simpler mappings in favor of a visually simpler design. The difference, however, is that both media players change the button icon to reflect the current state (i.e., the function the button will perform when pressed) so that users don’t have to remember the state or deduce the state from less obvious cues. A similar adaptation for Blue may reduce or even eliminate user confusion in the LaunchTile design.

CONCLUSION

Based on our participants’ strong interest in one-handed PDA use, and generally positive reactions to their interaction experiences, we are convinced of the value of research in one-handed, notification-based designs. We have less evidence of the utility of design scalability. Although we have demonstrated the feasibility of transferring both interfaces to smartphones, we do not know whether the designs support smartphone usage scenarios. With respect to command gestures, we are encouraged by the modest yet positively skewed satisfaction ratings for gesture interaction as well as what we consider very reasonable performance for both directional gesture execution and navigation tasks. It’s clear, however, that the introduction of two additional

diagonal gestures degrades performance and confuses users. We will need to explore whether AppLens can be an effective interface without these additional commands, or whether a different mapping of commands to gestures or on-screen cues can make the full set of gestures as reliable and learnable as directional gestures seem to be. Finally, we anticipate that extended usage studies with wider populations will unearth more subtle usability issues. With refinement, we hope a single design will emerge to provide a consistent, flexible environment designed for single-handed use.

ACKNOWLEDGMENTS

Removed for blind review.

REFERENCES

1. Bederson, B. B. PhotoMesa: A Zoomable Image Browser using Quatnum Treemaps and Bubblemaps. *Proc. UIST*, ACM Press (2001), 71-80.
2. Bederson, B. B. Interfaces for staying in the flow. *Ubiquity 5*, 7 (2004).
3. Bederson, B. B., Clamage, A., Czerwinski, M. and Robertson, G. DateLens: A Fisheye Calendar Interface for PDAs. *ACM Trans. Comput.-Hum. Interact. 10*, 4 (2003).
4. Bederson, B. B., Grosjean, J. and Meyer, J. Toolkit Design for Interactive Structured Graphics. *IEEE Trans. Soft-Eng. 30*, 8 (2004), 535-546.
5. Bederson, B. B., Meyer, J. and Good, L. Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. *Proc. UIST* (2000), 171-180.
6. Blickenstorfer, C. H. Graffiti: Wow! *Pen Computing Magazine* (1995), 30-31.
7. Buyukkokten, O., Garcia-Molina, H., Paepcke, A. and Winograd, T. Power browser: efficient Web browsing for PDAs. *Proc. CHI*, ACM Press (2000), 430-437.
8. Furnas, G. W. Generalized fisheye views. *Proc. CHI*, ACM Press (1986), 16-23.
9. Guimbretiere, F., Stone, M. and Winograd, T. Fluid interaction with high-resolution wall-size displays. *Proc. UIST*, ACM Press (2001), 21-30.
10. Hinckley, K., Pierce, J., Sinclair, M. and Horvitz, E. Sensing techniques for mobile interaction. *Proc. UIST*, ACM Press (2000), 91-100.
11. Hornbæk, K. and Frøkjær, E. Reading of electronic documents: the usability of linear, fisheye, and overview+detail interfaces. *Proc. CHI*, ACM Press (2001), 293-300.
12. Jackito PDA. http://www.jackito-pda.com/.
13. Long, C. A. Visual similarity of pen gestures. *Proc. CHI*, ACM Press (2000), 360-367.
14. Lyons, K., Starner, T., Plaisted, D., Fusia, J., Lyons, A., Drew, A. and Looney, E. W. Twiddler typing: one-handed chording text entry for mobile phones. *Proc. CHI*, ACM Press (2004), 671-678.
15. Perlin, K. Quikwriting: continuous stylus-based text entry. *Proc. UIST*, ACM Press (1998), 215-216.
16. Piccolo.NET. http://www.cs.umd.edu/hcil/piccolo/.
17. Potter, R. L., Weldon, L. J. and Shneiderman, B. Improving the accuracy of touch screens: an experimental evaluation of three strategies. *Proc. CHI*, ACM Press (1988), 27--32.
18. Rao, R. and Card, S. K. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. *Proc. CHI*, ACM Press (1994), 318-322.
19. Rekimoto, J. Tilting operations for small screen interfaces. *Proc. UIST*, ACM Press (1996), 167-168.
20. Robbins, D. C., Cutrell, E., Sarin, R. and Horvitz, E. ZoneZoom: map navigation for smartphones with recursive view segmentation. *Proc. AVI*, ACM Press (2004), 231-234.
21. Robertson, G. G. and Mackinlay, J. D. The Document Lens. *Proc. UIST*, ACM Press (1993), 101-108.
22. Sarkar, M. and Brown, M. H. Graphical Fisheye Views of Graphs. *Proc. CHI*, ACM Press (1992), 83-91.
23. Sazawal, V., Want, R. and Borriello, G. The Unigesture Approach. *Proc. Mobile HCI*, Springer-Verlag (2002), 256-270.
24. Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S. and Roseman, M. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Trans. Comput.-Hum. Interact. 3*, 2 (1996), 162-188.
25. Strachan, S., Murray-Smith, R., Oakley, I. and Angesleva, J. Dynamic Primitives for Gestural Interaction. *Proc. Mobile HCI*, Springer Verlag (2004)
26. Weberg, L., Torbj, Brange, r. and Hansson, s. W. A piece of butter on the PDA display. *ext. abstracts CHI*, ACM Press (2001), 435-436.
27. Wigdor, D. and Balakrishnan, R. TiltText: using tilt for text input to mobile phones. *Proc. UIST*, ACM Press (2003), 81-90.
28. Wobbrock, J. O., Forlizzi, J., Hudson, S. E. and Myers, B. A. WebThumb: interaction techniques for small-screen browsers. *Proc. UIST*, ACM Press (2002), 205-208.

Presents and studies two scalable designs which support one-handed thumb use on PDAs and cell phones. Can provide users with a consistent, single-handed experience across devices.