EXHIBIT A

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

Amy K. Karlson, Benjamin B. Bederson
Human-Computer Interaction Lab
Computer Science Department
Univ. of Maryland, College Park, MD, 20742
{akk, bederson}@cs.umd.edu

John SanGiovanni
Microsoft Research
One Microsoft Way, Redmond, WA 98052
johnsang@microsoft.com

ABSTRACT
We present 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 nine applications, while LaunchTile uses pure zoom to access thirty-six applications. We introduce two sets of thumb gestures, each representing different philosophies for one-handed interaction. We conducted two studies to evaluate our designs. In the first study, we explored whether users could learn and execute the AppLens gesture set with minimal training. Participants performed more accurately and efficiently using gestures for directional navigation than using gestures for object interaction. In the second study, we gathered user reactions to each interface, as well as comparative preferences. With minimal exposure to each design, most users favored AppLens's tabular fisheye interface.

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

ACM Classification Keywords
H.5.2. User Interfaces: Input Devices and Strategies, Interaction Styles, Screen Design; D.2.2 User Interfaces; I.3.6 Interaction Techniques

INTRODUCTION
The current generation of mobile computing hardware features a variety of devices for interaction with the system software. One such class of devices, typically referred to as "smartphones", features a numeric keypad or miniature thumb keyboard for input together with a screen for display output. These devices allow for single-handed interaction, which provides users with the ability to place calls and access information when one hand is otherwise occupied. However, because smartphones lack a touch-sensitive display, user interaction is constrained to a discrete set of keys, and thus the options for interaction design are limited to keypad-mapped functions and 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 this work, we have focused on the problem navigating device applications, and have adopted two design strategies. The first leverages Scalable User Interface (ScUI) techniques [2,3] to allow the system to adapt to different screen resolutions as well as support both portrait and landscape device rotation. This architecture allows developers to create a single application that can target a wide variety of screen resolutions, and provides users with a consistent interface and interaction model across devices. We accomplish this using the University of Maryland's Piccolo.NET development toolkit for zoomable and scalable user interfaces [5,18].

Our second design strategy provides access to rich notification information from multiple applications. Most current PDA interfaces are designed for focused interaction with a single task or application, with limited consideration or display real estate allocated for notifications (e.g., email, appointment reminders) or monitoring of ambient information streams (e.g., stocks, sport scores). In our proposed designs, each application has a dynamic launch tile in the place of a static launch icon. This feature offers high-value at-a-glace information for several applications at once, as well as on-demand application launch when users desire more detailed information.

In the work presented here, however, we limit our discussion of scalability and notification in favor of emphasizing the design features relevant to one-handed

interaction. We proceed by describing two designs: AppLens (characterized by zoom+fisheye) and LaunchTile (characterized by zoom+pan). The two approaches employ variations of zooming interface techniques [1] 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 nine applications. LaunchTile uses pure zooming within a landscape of thirty-six 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. Our approach in this work, therefore, 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 gestural system for navigation within the application's zoomspace. While our designs do not directly address one-handed text entry, they are compatible with a variety of existing single-handed text input techniques, including single- and multi-tap alphanumeric keypad input, as well as miniature thumb keyboards and unistroke input systems executed with a thumb (e.g., Graffiti [6], Quikwriting [17]).

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 are 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 [15]. Recent research efforts pairing gestures with PDA-sized devices have emphasized gestures based on changes in device position or orientation [10,21,28]. However, our work more closely resembles the onscreen gestures that have played a prominent role in stylus-based mobile computing (e.g., Power Browser [7]).

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. The EdgeWrite [32] gesture-based stylus text entry system is particularly suited to mobile usage scenarios due to its use of physical barriers. EdgeWrite adaptation to one-handed mobile computing is compelling, but would require expanding the dedicated input area to accommodate thumb-resolution gestures.

One handed device interaction has typically focused on text entry techniques, but beyond a numeric keypad, most one-handed text entry systems require specialized hardware, such as accelerometers [25,30] or customized keyboards [16]. We instead address the more general task of system navigation and interaction, and restrict our designs to existing hardware.

Commercial products have also emerged with related design goals. The Jackito PDA [14] supports thumb-only interaction, but presumes two handed use and is not gesture oriented. Lastly, Apple's iPod is an elegant one-handed interaction solution for audio play and storage [13], but is currently not designed for the type of generalized application interaction we propose.

THE ZOOM+FISHEYE APPROACH: APPLENS

AppLens provides one-handed access to nine applications, and is strongly motivated by 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 [27] 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 [24], networks [26], spreadsheets [20], and documents [12,23].

Bederson et al. [3] drew upon that work in developing DateLens, a space-conserving calendar for PDAs, which was shown to perform favorably for long-term scheduling and planning tasks when compared to a traditional calendar implementation. One of the strengths of DateLens was 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. Also important was the design of effective representations for the variety of cell sizes and aspect ratios that resulted from tabular distortion. One drawback of DateLens, however, was that it required two-handed stylus interaction to actuate the small interface widgets. Our design extends the principles of DateLens to include one-handed thumb access and generalizes the approach for use across a variety of domains.

We developed a scalable 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 simple for applications to be built within this framework; developers need only to replace a small number of cell views with representations that are meaningful within the target domain.

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

AppLens Zoom Levels

The AppLens shell (Figure 1) 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 representations at 3 zoom levels: *Notification*, *Context* and *Full*.

Notification zoom distributes the available screen real estate equally among the 9 application tiles (Figure 1a). One tile (shown centered) 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 (Figure 1c) allocates roughly half the available display area to a single *focus* tile, compressing the remaining tiles according to a tabular fisheye distortion technique [3,20]. 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, called *peripheral* tiles, may be rendered quite differently depending on their position relative to the focus tile, which dictates whether the peripheral tile is a square, a wide-flat rectangle, or a narrow-tall rectangle. To reduce visual overload, 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 1b).

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, and on-screen keyboards). In support of traditional designs, AppLens uses an object cursor to identify the on-screen object that is the current interaction target. The cursor is depicted as a dynamically-sized rectangular orange border that users move from one on-screen object to another via command gestures. Cursors are not new to PDA interface design: the WebThumb [31] web browser includes a similar notion of cursor, but which is controlled via directional hardware, and others [29] 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 handheld 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 widget activation (CANCEL), equivalent to tapping the stylus outside the target widget. We also include the convenience commands FORWARD and BACKWARD, equivalent to TAB and SHIFT-TAB on Windows PCs.

Command Gestures

Our use of gestures is motivated by Hirotaka's observation that the button positions on cell phones require interaction using a low, unstable grip [11]. PDA joysticks face a similar drawback in that they are typically located along the lower perimeter of the device. AppLens avoids this problem since its gestures can be issued anywhere on the screen. Each AppLens 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 touch-sensitive surface. This flexibility lets users interact with our designs using the grasp that provides maximum stability in a mobile scenario.

Figure 2. Screen area accessible with one hand.

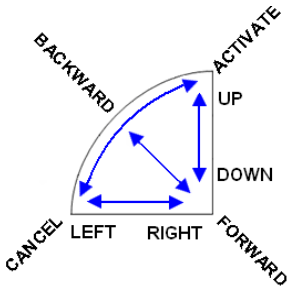

Figure 3. The gesture set.

We based the gesture set on the limited motion range of thumbs (Figure 2), with the goal of creating a gesture language that could be learned with minimal training. After informally experimenting with a variety of gestures, we developed a simple set of gestures with the aim of maximizing memorability and robustness of execution (Figure 3). We assigned the directional commands UP, DOWN, LEFT and RIGHT to spatial gestures that map directly to their function. We assigned ACTIVATE and CANCEL to the two gestures defined by pivoting the thumb from bottom to top and top to bottom respectively. We made these assignments 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.

The eight gesture commands can also be activated with a numeric keypad by mapping each to the key corresponding to the gesture endpoint: 1-BACKWARD, 3-ACTIVATE, 7-CANCEL, and 9-FORWARD. Since smartphones have a joystick that can be used for directional navigation, the keypad-to-command mapping is not necessary, but can be assigned as follows: 2-UP, 4-LEFT, 6-RIGHT and 8-DOWN.

Using Command Gestures within AppLens

Users navigate between AppLens zoom levels using ACTIVATE and CANCEL gestures. As a rule, the ACTIVATE gesture behaves as a stylus tap on the current cursor target, thus its effects are target-specific. Since application tiles are non-interactive at *Notification* and *Context* zoom levels, the ACTIVATE gesture simply zooms in, animating the layout first from *Notification* to *Context* zoom, and then to *Full* zoom. Once at *Full* zoom, the input cursor transitions to the objects *within* the application, at which point the command gestures affect the current target widget. The CANCEL command negates 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. The CANCEL command will first deactivate an active target. If the current target is not in an active state, CANCEL will cause the application tile to animate from *Full* zoom to *Context* zoom, and if issued again, to *Notification* zoom.

Figure 4. Three LaunchTile zoom levels: (a, b) Zone, (c) World, (d) Application

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 application tiles, divided into 9 zones of 4 tiles each (Figure 4c). The 36 tile configuration is an exploration of the maximum number of applications that can reasonably fit within the display space. Since the design is fundamentally scalable, however, it can display any number of tiles *up to* 36, and fewer may even be preferable. As a central design element, LaunchTile uses a large blue onscreen button, called Blue (Figure 4a), to unify the shell and applications with a consistent visual metaphor. Blue provides a consistent point of reference for zooming and panning navigation, with onscreen tiles, menus, and functions maintaining a consistent relative position to Blue. The LaunchTile zoomspace consists of 3 zoom levels: World (36 tiles, Figure 4c), Zone (4 tiles, Figures 4a and b) and Application (1 tile, Figure 4d).

Zone View

The Zone view of LaunchTile divides the screen area into 4 equally-sized application tiles (Figure 4a). To view other tiles, the user pans the zoomspace to neighboring 4-tile clusters, called *zones*. The zones are arranged as a 3x3 grid, each associated with a numerical designator from 1 to 9 as 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.

Panning Techniques

To support various input hardware and styles of interaction, there are several ways to pan the zoomspace within Zone view. If the device has a multidirectional control joystick, the user can push it in the direction of the targeted zone. From the Home screen, the 16 tiles in zones 2, 4, 6, and 8 are only 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. Alternatively, if the device has a touch-sensitive display, the user can use his or her thumb directly to drag the zoomspace. Dragging is performed "on rails", permitting the user to drag vertically and horizontally, but not diagonally. Although the zoomspace moves with the thumb during dragging, Blue remains centered and stationary. Because only one instance of Blue exists within Zone view, each zone is depicted with an empty center hub during dragging. Upon thumb release, the zoomspace animates to align Blue with the closest zone's empty hub. The visual and automated guidance ensures the user is never caught between zones.

Within each 4-tile zone, indicator widgets communicate the user's relative location within the zoomspace. The indicator has two components. First, directional arrows show where other zones are. If users only see indicators pointing up and right, they know they are in zone 7. Small blue dots featured with the arrows represent the location of 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 final way to pan the zoomspace is to tap the directional indicator itself. An oversized hit target ensures that the user can easily hit the indicator without using a stylus.

Zooming Out to the World View

From any one of the nine 4-tile Zone view zones, the user may tap Blue (or press the 5 key) to zoom out to view the entire 36-tile zoomspace (Figure 4c). Since all 36 tiles are visible at once, this view reduces each tile to a small icon. From this World view, the user can easily see the absolute location of each tile, as well as monitor all applications at once. In the World view, the display is divided into a grid of 9 hit targets, each mapping to a 4-tile zone. Single-tapping a zone animates to Zone view, displaying the zone's 4 notification tiles.

Zooming In to an Application

The user taps any of the 4 notification tiles within Zone view to launch the corresponding application. An animated zoom draws the zoomspace toward the user until the target application fills the entire display (Figure 4d). If the device has only a numeric keypad (no touchscreen), the user 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, and 9 launches the lower right. This technique provides quick, single-tap access to each visible tile, and was inspired by ZoneZoom of Robbins et al. [22].

As the system zooms, Blue stays in view and retains its function as a central point of reference (Figure 4d). Application menu commands are represented as on-screen buttons clustered around Blue, now positioned at the bottom of the display. Each menu button displays a numeric label, so that mobile phone users may activate each menu by pressing the corresponding number on the keypad. 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

Pressing Blue typically moves the user deeper into the object hierarchy, while a dedicated Back button moves the user up the hierarchy. In the Zone view however, Blue toggles between Zone view (4 tiles) and World view (36 tiles). Once an application is launched, three dedicated software buttons along the top edge of the screen support inter- and intra-application navigation (Figure 4d). A green Home button immediately zooms the view out to the Home screen. 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 and control. On a non-touchscreen device, Back and Home commands are executed with dedicated physical buttons, such as those provided on a smartphone.

Application-Level Interaction

Although the original focus of our designs was on the application "shell", we extended the LaunchTile interaction philosophy to the application level, where we sought to make interaction consistent with navigation among the application tiles. Several gestures have been designed to specifically support one-handed navigation and item selection within applications. Previously, others have demonstrated that for direct-manipulation interfaces, a grounded tap-drag-select-release technique is more accurate than a tap-to-select [19]. We therefore made all LaunchTile tap-to-select targets large enough for thumb activation. In cases when limited display real estate necessitates smaller targets, the central Blue widget serves as a moveable tool glass which can be positioned over the target object (e.g., email header, message text). The large thumb-friendly drag target is offset below the selection area to prevent the user's thumb from occluding the target. Alternatively, the user can drag the application

contents, such as to pan a map, scroll a list of email, or navigate to text outside the view area. Together, these two interaction techniques permit the user to address a large application content space, yet bring focus to a target much smaller than a finger. Alternatively, a non-touchscreen user uses the multidirectional joystick to position the selection area on the screen. Keys 2 and 8 are used for page control up and down, and 4 and 6 can be either menu items or horizontal page control, as appropriate.

Once the targeted item is in the tool glass selection area, a tap on Blue (or pressing the 5 key) activates the item, which may either draw the user further into the application (e.g., open an email), or may replace the application menus with a context-sensitive menu clustered immediately around Blue. As with the application menus, keys on a numeric keypad execute context menus according to their positions relative to Blue. At this point 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. Alternatively a mini qwerty keyboard may be used 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) [5,18]. Although the primary development and test platform has been the HP iPAQ PocketPC running Microsoft Windows Mobile 2003, both run unmodified on the iMate Smartphone II running Windows Mobile Smartphone 2003 (Figures 1d and 4b respectively).

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 prototyped email and mapping as two interactive examples within LaunchTile.

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 formative evaluation which gathered user reactions to each shell design, as well as users' 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 tool glass, a map, or a list of email headers. In AppLens, gestures only affect the 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 although 12 of the participants considered themselves advanced computer users, only 6 regularly used a PDA.

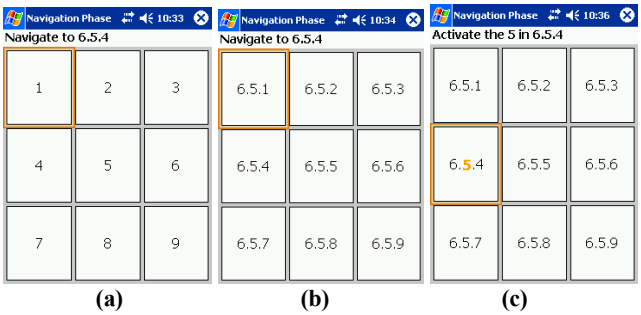

Figure 5. Example gesture study environment states: (a) the top level, (b) the third level after ACTIVATE is performed on 6.5, (c) highlighting within a cell.

MATERIALS: The study was run on an HP iPAQ measuring 4.5x2.8x0.5 inches 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 equal-sized cells, numbered 1-9 like a phone keypad. For the experiment, the hierarchy was limited to 5 levels and *Context* zoom was eliminated so that activating a cell animated the display to the next level. Cell contents 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 (Figure 5b and 5c). 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 test environment, FORWARD and BACKWARD were used for intra-cell selection, allowing participants to cycle a highlight either forward or backward through cell label digits (Figure 5c). 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 required users to perform a gesture when presented with 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 successfully. 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 five nine-point Likert scales: frustrating – satisfying, hard to learn – easy to learn, hard to use – easy to use, slow – fast, and 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 referred to.

The second test phase required participants to perform goal-directed tasks within the information space. These included (**N**)avigation ("Navigate to 6.5.4"), (**A**)ctivation ("Activate the 5 in 6.5.4"), (**NA**)vigation+activation ("Activate the 2 in 4.3.2"), and (**C**)ancellation ("De-activate all digits in 4.3.2") tasks. Participants then recorded their subjective ratings of the interaction experience. Some participants completed the study in as little as 15 minutes, most within 30 minutes, and none required more than 40 minutes.

Results

In the gesture phase of the study, participants correctly performed gestures an average of 87% of the time. By gesture type, 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 the number of times that participants looked at a reference sheet, we assumed that the acts of page-flipping and answer-searching have been reflected in the task time, and thus did not analyze this data further.

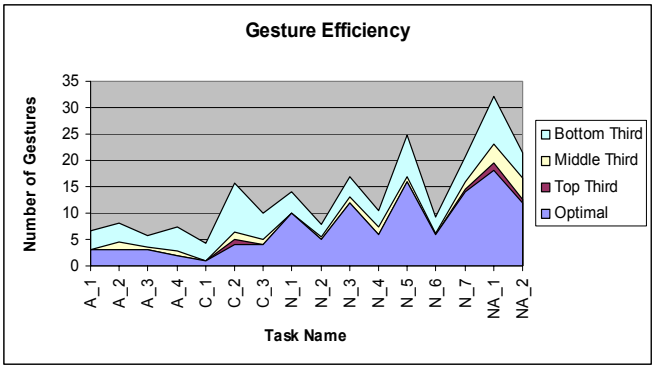

Figure 6. Average number of gestures per task, by performance tier, compared with the optimal.

The second test phase evaluated accuracy and efficiency for goal-directed navigation tasks. The average task success rate was 95%. While both navigation and navigation+activation tasks were performed with 98% accuracy, activation was close behind at 96%. Cancellation 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 Middle third of participants performed comparably to the Top third, but the gaps were wider on the tasks the Top third had trouble with. That is, the tasks for which the Middle third deviated farthest from Optimal, were those for which the Top-tier also deviated. Thus, the most significant difficulties in performing the navigation phase tasks were experienced by only a third of the participants.

The average subjective rating for each of our 5 satisfaction measures, on a scale of 1-9 where 9 was positive, fell within a 1 point span of one another, 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 have an intrinsic spatial mapping. Presumably, this mapping contributes 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 [15] 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 the 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 FORMATIVE STUDY

Due to the early nature of our interfaces, their shared design principles, and our goal of distilling a unified design, we conducted a formative study to understand usability issues for new users of each shell design and to elicit general reactions and comparative preferences.

PARTICIPANTS: We recruited ten participants (8 male, 2 female) from a local private scientific research center. Three participants were in their 20s, five in their 30s, and two were 40 or older. While all participants considered themselves advanced computer users, four used PDAs regularly, and four had never used a PDA.

MEASURES: Participants provided subjective design-specific and comparative reactions to AppLens and LaunchTile through think aloud and verbal 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 application tiles, and answering questions about application content.

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 interaction, 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. We limited each user session to 45 minutes, allotting roughly fifteen minutes to each interface, and the final fifteen for comparative feedback.

Results

Reactions to AppLens were quite consistent across participants. Because only one question in the study focused on a specific interface design feature (*Context* view), we regard commonality in participant responses indicative of the highest-impact interface characteristics. We report here on the strongest trends in opinion. Half the participants commented that they liked the *Notification* view and the ability to access all nine application tiles within both *Notification* and *Context* views. Even though two participants found nothing redeeming about *Context* view, all others found it useful at least some of the time. Seven participants found application navigation easy and enjoyable, but 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 two 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 the gestures for zoom-in vs. zoom-out, difficulty performing the ACTIVATE gesture, difficulty with directional navigation, and frustration due to misrecognition. 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.

Perhaps due to the richness of the LaunchTile environment, reactions were mixed and more complex. Nearly half the participants reacted positively to the aesthetics of LaunchTile, and specifically to Blue. Seven participants appreciated the volume of information available through the two zoom perspectives World and Zone, while six thought that zooming between those perspectives was one of the easiest aspects of the interface. The majority (7) of participants felt comfortable using one hand to perform the tasks, and eight felt they were able to effectively navigate among applications, primarily by tapping via World view.

Surprisingly, a majority (7) of participants had difficulty panning by dragging within LaunchTile, commenting most often that it was unintuitive, but also that it was slow. This subset of participants was slightly skewed toward participants who used the LaunchTile interface first, and thus would not have been biased by experience with the AppLens directional gestures. Six participants struggled with the multi-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. Most of the 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 - half as many as chose AppLens.

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, presumably due to a better balance between the number of applications and the presentation space available for each. 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 stating an a priori preference for one-handed use in all situations. However 2 participants expressed that they would never want to use a PDA with one hand, regardless of the interface design.

DISCUSSION

Two notable themes emerged from the comparative study. The first was the participants' reluctance to use gestures. Results from the first study suggested that directional gestures can be learned quickly, yet for both interfaces, users favored tapping targets over issuing gestures. While this trend may simply be a training issue, it may also be an early warning sign for the adoptability of overlay gesture systems - those that share the same real estate as the interface objects themselves. Second, a large number of users commented on the perceived utility of the simultaneous display of high-value content from multiple applications, bolstering our intuition that flexible notification-based designs may provide an effective balance between functionality and content within the real estate constraints of handheld computing.

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 quality. However, more experience with the two designs might have tipped the scales the other way, as Bederson has pointed out in [4] that even complex interfaces have the potential to be highly satisfying after users have expended the effort to become experts. 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 features. Thus a different participant population may have offered different opinions. While we do not consider LaunchTile in this class of expert interface, clearly 15 minutes is not sufficient for users to become proficient with the variety of interaction techniques supported by the interface.

In [4] 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. 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 nonetheless. 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 designs, and believe notification plays an important role in effective utilization of limited real estate. We have less evidence of the utility of design scalability beyond being an engineering convenience. 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

We appreciate François Guimbretière's early suggestion of considering the ergonomics of human thumbs, and thank Aaron Clamage for his rapid efforts in porting Piccolo.NET to small devices so we could build these prototypes.

REFERENCES

1. Bederson, B. B., Meyer, J. and Good, L. Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. *Proc. UIST* (2000), 171-180.
2. Bederson, B. B. PhotoMesa: A Zoomable Image Browser using Quatnum Treemaps and Bubblemaps. *Proc. UIST*, ACM Press (2001), 71-80.
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. Interfaces for staying in the flow. *Ubiquity 5*, 7 (2004).
5. Bederson, B. B., Grosjean, J. and Meyer, J. Toolkit Design for Interactive Structured Graphics. *IEEE Trans. Soft-Eng. 30*, 8 (2004), 535-546.
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. Hirotaka, N. Reassessing current cell phone designs: using thumb input effectively. *Extended Abstracts CHI*, ACM Press (2003), 938-939.
12. 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.
13. iPod. www.apple.com/ipod/, Apple Computer.
14. Jackito PDA. http://www.jackito-pda.com/.
15. Long, A. C., Landay, J. A., Lawrence, R. A. and Michiels, J. Visual similarity of pen gestures. *Proc. CHI*, ACM Press (2000), 360-367.
16. 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.
17. Perlin, K. Quikwriting: continuous stylus-based text entry. *Proc. UIST*, ACM Press (1998), 215-216.
18. Piccolo.Net. http://www.cs.umd.edu/hcil/piccolo/, Univ. of Maryland Human-Computer Interaction Lab.
19. Potter, R. L., Weldon, L. J. and Shneiderman, B. Improving the accuracy of touch screens: an experimental evaluation of three strategies. *Proc. CHI* (1988), 27-32.
20. 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.
21. Rekimoto, J. Tilting operations for small screen interfaces. *Proc. UIST*, ACM Press (1996), 167-168.
22. Robbins, D. C., Cutrell, E., Sarin, R. and Horvitz, E. Zone Zoom: Map Navigation for Smartphones with Recursive View Segmentation. *Proc. AVI*, ACM Press (2004), 231-234.
23. Robertson, G. G. and Mackinlay, J. D. The Document Lens. *Proc. UIST*, ACM Press (1993), 101-108.
24. Sarkar, M. and Brown, M. H. Graphical Fisheye Views of Graphs. *Proc. CHI*, ACM Press (1992), 83-91.
25. Sazawal, V., Want, R. and Borriello, G. The unigesture approach. *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, Springer-Verlag (2002), 256-270.
26. 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.
27. Spence, R. and Apperley, M. D. An Office Environment for the Professional. *Behaviour & Information Technology 1*, 1 (1982), 43-54.
28. Strachan, S., Murray-Smith, R., Oakley, I. and Angesleva, J. Dynamic Primitives for Gestural Interaction. *Proc. Mobile HCI*, Springer Verlag (2004).
29. Weberg, L., Torbjorn, B. and Hansson, A. W. A piece of butter on the PDA display. *Extended Abstracts CHI* (2001), 435-436.
30. Wigdor, D. and Balakrishnan, R. TiltText: using tilt for text input to mobile phones. *Proc. UIST*, ACM Press (2003), 81-90.
31. Wobbrock, J. O., Forlizzi, J., Hudson, S. E. and Myers, B. A. WebThumb: interaction techniques for small-screen browsers. *Proc. UIST*, ACM Press (2002)
32. Wobbrock, J. O., Myers, B. A. and Kembell, J. A. EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. *Proc. UIST*, ACM Press (2003), 61-70.