The Springboard: Multiple Modes in One Spring-loaded Control

Ken Hinckley[1], **Francois Guimbretiere**[2], **Patrick Baudisch**[1],
Raman Sarin[1], **Maneesh Agrawala**[1], **Edward Cutrell**[1]
[1]Microsoft Research, One Microsoft Way, Redmond, WA 98052
[2]University of Maryland, A.V. Williams Building, College Park, MD 20742.
{kenh, baudisch, ramans, cutrell}@microsoft.com; francois@cs.umd.edu; maneesh@cs.berkeley.edu

ABSTRACT
Modes allow a few inputs to invoke many operations, yet if a user misclassifies or forgets the state of a system, modes can result in errors. Spring-loaded modes (*quasimodes*) maintain a mode while the user holds a control such as a button or key. The *Springboard* is an interaction technique for tablet computers that extends quasimodes to encompass multiple tool modes in a single spring-loaded control. The Springboard allows the user to continue holding down a nonpreferred-hand command button after selecting a tool from a menu as a way to repeatedly apply the same tool. We find the Springboard improves performance for both a local marking menu and for a non-local marking menu ("lagoon") at the lower left corner of the screen. Despite the round-trip costs incurred to move the pen to a tool lagoon, a keystroke-level analysis of the true cost of each technique reveals the local marking menu is not significantly faster.

Author Keywords
Modes, tablet, pen, marking menus, keystroke-level model

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Input

INTRODUCTION
Modes are all about using one input device to do multiple things. Most of the relevant literature on modes dates from the era of keyboard-only interfaces and text editors that were infamous for cryptic key combinations, such as *vi* and *emacs* [18], that could trap unsuspecting users in seemingly inescapable modes. The essence of the problem was that all system functions were invoked using the same set of keys that served primarily for text entry. The situation led Tesler, in a classic 1981 *Byte* article, to lament *Don't mode me in!* to describe his feeling of entrapment by modes [22]. Such problems seem antiquated, yet we now face pen-operated devices that lack a keyboard. All functions are invoked using the same pen that serves primarily for inking on the screen. The resulting modal traps seem all too familiar.

For tablet computers, common modes include pen/inking mode, gesture mode, selection mode, eraser mode, highlighter mode, panning and zooming modes, and object creation modes (e.g. drag out a rectangle or ellipse). Such modes plague many note-taking and drawing applications, including *Windows Journal*, *OneNote*, *Alias Sketchbook*, and even the classic *MacPaint* interface. These programs all exhibit a strong default mode (inking or drawing) where users are expected to spend most of their time, but users also need frequent but temporary access to tool modes. These applications have an *iCi task structure*, where *i* is the default inking mode and *C* is the temporary command mode. *Quasimodes*, also known as *spring-loaded modes*, are well suited to such *iCi* tasks [20,21]. But quasimodes only provide a spring-loaded control for one mode. It is not practical to have a button for all of the modes listed above; even if it were, hitting the wrong button would undermine the benefits of providing a quasimode for each tool [10]. As a result, designers must use quasimodes sparingly.

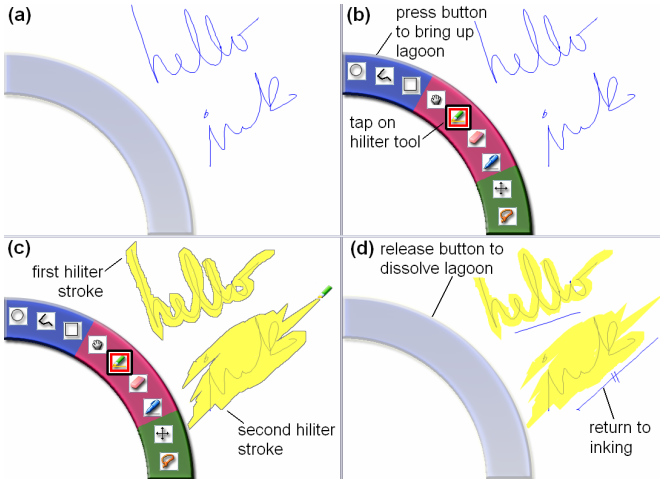

Fig. 1. Illustration of the Springboard using a tool lagoon with icons. *(a)* Inking is the default. *(b)* The user holds down a nonpreferred-hand button to call up the lagoon, and taps the hiliter tool. *(c)* Pen strokes apply the hiliter as long as the user holds the button. *(d)* Releasing the button returns to inking.

To address this, we propose the *Springboard* as a way to extend a quasimode associated with a single spring-loaded control to multiple modes. For clear illustration, *Fig. 1* shows a version of the Springboard design with icons, but the implementation we study in detail uses marking menus. To use the Springboard, the user presses and holds a nonpreferred-hand button (COMMAND) located on the screen bezel. The user then makes a menu selection to choose the desired tool mode. The key difference compared to quasimodes is that as long as the user continues to hold COMMAND, he can apply the *selected* tool mode by making one or more pen strokes. Releasing COMMAND turns off the tool mode, and returns the application to its default mode.

CHI 2006, April 22–27, 2006, Montréal, Québec, Canada.

The Springboard can be applied to a "local" menu that the user activates from the current pen position, or to "remote" menus that lie at the edges of the screen. For the local design, we implemented a marking menu that was triggered by stroking while pressing a nonpreferred-hand button [14]. For the remote design, we implemented a marking menu that was triggered from a "tool lagoon" [6] in the lower left corner. In this case, the location of the pen stroke already signals that the mark is a command, so the Springboard uses the COMMAND button press to keep the selected tool mode active. This button press is an extra cost, but it might save time if it can eliminate round trips.

We contribute the Springboard as a design option for mode switching. We also contribute an experimental paradigm for rigorous quantitative analysis of mode and command selections. Our paradigm contrasts different interaction patterns to operationalize in-situ uses of a technique, and uses a subtraction methodology [5,14] to isolate physical articulation costs via a keystroke-level model. This paradigm ensures that we can identify any extra costs over the full life-cycle of a mode switch. Our analysis reveals that on a tablet-sized device, the local marking menu was not significantly faster than a marking menu on the tool lagoon. Interface designers can use our models and experimental approach to extend these results to variations on the specific interface designs that we tested, or to reason about design tradeoffs in other similar inking/command tasks that naturally arise in pen gesture interfaces.

RELATED WORK

A user commits a *mode error* when he or she fails to comprehend the current state of the system and thus performs an action which is incorrect given the true state of the system [17]. Quasimodes may reduce the potential for mode errors by helping users maintain awareness of the system's state. Sellen et al. [21] show that holding a foot pedal reduces errors in a text editor, but a latching foot pedal that holds its state for the user does not. This suggests the user's active maintenance of muscle tension is the crucial quality of a quasimode. For example, holding SHIFT to temporarily capitalize letters is a quasimode, but CAPS LOCK is not because tapping the key persists the mode. Another good example of a quasimode is the panning tool in *Photoshop*. The user presses and holds the spacebar, and can then click and drag multiple times to pan an image; releasing the spacebar returns to the prior mode. Note that the spacebar accesses one and only one temporary function.

Tracking menus are menus that follow the pen [6]. *Alias Sketchbook 2.0* activates a pan/zoom tracking menu when the user holds the spacebar. The user may pan by quickly "pawing" [6] at the screen (just as in *Photoshop*), or may instead target the center of the tracking menu to zoom. Releasing the spacebar dismisses the tracking menu and returns to drawing mode. Thus the tracking menu offers a quasimode with one other rapid-access function.

Marking menus use the direction of a pen stroke to select commands [13]. Several systems allow extension of the stroke to transition from command selection to a dragging mode specific to the selected command [8,9,19]. In all of these systems, the pen must stay in continuous contact with the screen: the dragging phase ends as soon as the user lifts the pen. The Springboard enables the user to draw multiple strokes in the tool mode by continuing to hold the button, rather than forcing a single stroke syntax [1,24].

The Springboard shares some properties of the hotbox in *Alias Maya* [12]. The user activates a hotbox by pressing and holding the spacebar; the hotbox disappears when the user releases the spacebar. The hotbox offers multiple marking menus that the user can select from; the user can also issue multiple commands in a single posting. The Springboard extends the hotbox in one critical dimension: when the user marks to select a command, the user *keeps holding* the button to apply the resulting tool mode as many times as desired by stroking the pen.

There are several strategies to improve command efficiency. Some applications persist a mode if the user presses SHIFT while clicking on an icon. The Xerox Star had an AGAIN key, and some modern keyboards have a REDO key [16]. These keys repeat the last command. Pressing CTRL while clicking on a color chip in *Microsoft Paint* lets users reapply that color whenever they hold CTRL; releasing CTRL returns to the prior color. Unlike *Paint*, the Springboard emphasizes and deemphasizes the visual representation of the interface with the COMMAND button press, and generalizes the approach to tool modes. Also, our experiment is the first to study this design option and demonstrate that it offers significant advantages.

Dillon et al. [5] assess the true cost of command selection. Their study compares a *baseline task* without commands to a *compound task* with command selections; subtracting the two yields the true cost. We extend this *subtraction methodology* by contributing a new methodology that leverages keystroke-level analyses to carefully tabulate all costs associated with mode selection. This approach allows us to show that locally-activated marking menus do not benefit tool mode switching despite the common belief that round trips to the edge of a tablet's screen are slow [6,9].

Li et al. [14] show that pressing a button with the nonpreferred hand is an effective method to switch between ink and gesture modes. We evaluate techniques for switching between ink and several other command modes. The efficiency of command selection depends on the surrounding operations [2,15]. Some designs optimize *alternation* between commands, while others optimize *repetition* of a command. For example, ToolGlass [3] has a tool palette that users position with the nonpreferred hand, but users must position and click through the ToolGlass every time they apply a tool, making repetitive tool use inefficient. Our experiment controls for this by including both *alternation* and *repetition* task patterns.

INTERACTION TECHNIQUES FOR MANAGING MODES

Current applications use several techniques for managing modes. The ***Persists*** technique keeps the selected mode active until the user chooses a new tool. For example, in

Windows Journal, clicking on the lasso icon turns on selection mode, which stays on until the user taps the pen icon. This technique amortizes the cost of the command selection across several operations, but returning to pen mode requires an extra step to reselect the pen tool.

The ***Once*** technique turns on the selected mode for one use only, and then automatically reverts to the prior mode. For example, when a user selects the *Insert Space* command in *Windows Journal*, dragging the pen inserts white space. When the user lifts the pen, the interface reverts to the default inking mode. This technique works well for tools that the user tends to employ one time, but is tedious if the user applies the same tool multiple times in a row.

Quasimodes such as the spacebar for panning in *Photoshop* offer another approach. Quasimodes are only suitable for temporary modes, as users cannot hold a key indefinitely, and even on a keyboard, only a few keys for quasimodes are available, so designers must use them sparingly.

The Springboard

The *Springboard* is a technique to get more mileage out of quasimodes. Instead of mapping one control to one mode, the *Springboard* allows users to pass through two sub-modes. Pressing COMMAND starts a command selection sub-mode (by presenting commands representing various tools in a menu). After the user selects a tool, the *Springboard* transitions to a command performance sub-mode where the user can apply the selected tool multiple times. Like a traditional quasimode, releasing COMMAND always returns to the application's default mode.

The *Springboard* thus brings the benefits of a quasimode to multiple tool modes, because it encompasses all modes in a pop-up menu within a single spring-loaded control. Like the *Persists* technique, *Springboard* amortizes a command selection across several operations. But unlike *Persists*, it removes most of the extra cost required to turn off the mode. This should make *Springboard* more efficient than *Persists* in both one-time and multiple-use scenarios.

SpringOnce

During pilot testing of the *Springboard*, test users liked holding the button as a way to apply a tool multiple times. But when the user's intention was to apply a tool one time only, some users would release the COMMAND button immediately after picking the tool from the menu but before applying it. The system would return to inking, thus causing users to mistakenly ink rather than applying the tool. We found this error hard to avoid even after lots of practice. The problem seemed to be particularly vexing for the local marking menu version of the *Springboard*, but did occasionally arise with the *Springboard Lagoon* as well.

We realized that a hybrid of *Springboard* and the *Once* technique might help to resolve this issue. If a user selects a tool without applying it and then lets go of the button, it is unlikely that he immediately wanted to transition back to inking. So our *SpringOnce* design keeps the tool active for one use in this case. If the user instead continues holding COMMAND until he starts applying the tool, *SpringOnce* then allows the user to keep holding COMMAND to apply the tool multiple times, just like the *Springboard* design.

This approach is more forgiving for users who tend to release the button too early when applying a tool one time. If the user actually did select a command by mistake, and his intention was to immediately return to inking, the user can press COMMAND again to cancel.

EXPERIMENT

We conducted a formal experiment to evaluate the effectiveness of techniques for transitioning between multiple modes on tablet computers. We evaluate the traditional *Persists* and *Once* techniques as well as the *Springboard* and *SpringOnce* designs. We decided to include both of these designs because it was not clear which one would actually perform the best, and because the *Springboard* builds on previous work that suggests quasimodes can facilitate mode switches [20,21].

Experimental Factors

Several factors may influence the efficiency of tool modes:

Behavior is the user interface's mode switching policy, i.e. the status-quo *Once* and *Persists* behaviors, plus our proposed designs, *Springboard* and *SpringOnce*.

TaskType. Since *Once* and *Persists* support different task patterns, it is important to test tasks with *alternation* between ink and command modes, as well as tasks that require *repetition* of the same command. MacKay [15] notes that when users are performing mechanical copying and modification tasks, they tend to batch commands together. But during problem solving, it is difficult to anticipate the sequence of commands that will be needed, so users tend to interleave different operations. Hence our experiment includes examples of each class of task.

MenuType. We wanted to evaluate the effectiveness of the Springboard designs both for a local marking menu activated close to the user's current screen location, as well as for more traditional interface widgets that are located near the edges of the screen. We chose the tool lagoon design as it is now an established approach (used by *Sketchbook* and *ArtRage*) that seems well suited to the pen.

Fig. 2. The Lagoon's active and inactive states as used in the experiment. *Far right*: Using the Local Marking menu to pick the lasso tool in close proximity to a dot stimulus.

To make our *Local Marking* menu and our *Lagoon* menu designs comparable, unlike the design pictured in Fig. 1, our experiment places all tool modes in a single marking menu on the lagoon (*Fig. 2*, *Fig. 6*). Since our conditions require users to draw identical marks to select modes, any observed effects will not be due to a difference in how the menus work. *Sketchbook* also uses marking menus on its

lagoon [6], so this is an established approach. Compared to our lagoon with icons (*Fig. 1*), marking from the lagoon of *Fig. 2* has the virtue that the user never has to decide which icon to aim for, which may speed performance.

For the Springboard designs only, our *Lagoon* condition highlights the arc when the user presses the COMMAND button. The lagoon transitions to a semi-transparent state when the button is released (*Fig. 2*). The COMMAND button was not used at all for the lagoon's *Once* and *Persists* techniques, as we wanted our implementation to be as close as possible to status quo mode techniques.

Experimental Task

Inking is the "normal" mode that we expect users to spend the most time using in applications that leverage the unique capabilities of a pen-based computer. Thus, in order to test the full life-cycle of a mode switch, we needed a compound *inking/command selection* experimental task that required the user to start with inking, apply one or more tool modes, and then return to inking. This task pattern ensured that our experimental task could capture *all* of the costs that might accrue due to (1) switching from inking to a tool mode, (2) applying a tool mode, and (3) reverting back to the normal inking mode when finished with a tool. Our experiment also included all-ink baseline tasks (*Fig. 3*) to provide a reference condition without any mode switches.

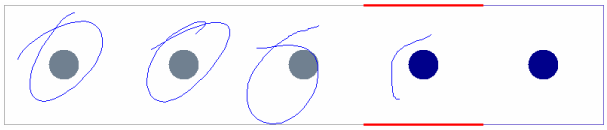

Fig. 3. Baseline (inking-only) task in progress.

For the inking portion of the task, we decided to have users draw series of circles, rather than handwriting or scribbling. Circling captures the essence of handwriting (which fundamentally consists of oscillatory motions [23]) while providing a well-defined task that is amenable to quantitative study. A series of dot stimuli, which the user always had to work through from left to right, provided a reference for where the circle had to be drawn.

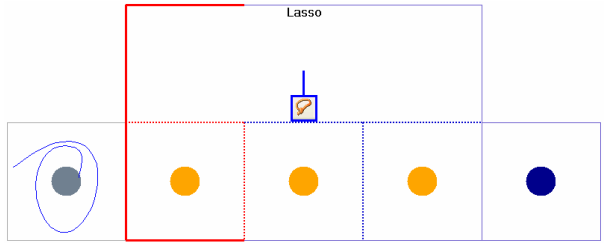

Fig. 4. Repetition task stimuli.

We wanted the task to be ecologically valid, so we used commands that mimic the *Windows Journal* Tablet PC application. The pen left a thin blue ink trail. Applying the hiliter tool created a thick yellow trail, and the lasso tool made a dotted red line. The eraser tool left a gray swath, as we wanted users to be able to see where they were applying each tool. All tools were always applied by circling the current dot stimulus. We told users to think of the eraser as "erasing what's inside the circle" rather than as a tool they had to scrub back and forth.

Each task consisted of a set of 5 subtasks, which we call *segments*. The first and last segment are always inking. For *repetition* tasks (*Fig. 4*), segments 2, 3, and 4 all require the user to apply the same command, and thus the entire task has an *iCCCi* pattern, where *i* is inking, and *C* is a command. The user's task was to circle each dot using the correct mode. To prevent any errors from cascading, users could not proceed until they successfully completed the current segment. If the user made an error a short 'oops' sound played. Users were instructed that their circles could be drawn casually, but should be larger than the dot and smaller than the box. Drawing the circles was thus a quick and fluid inking movement, much like jotting down a quick note, rather than a visually guided steering task or precise pointing task. If the circle did not contain the dot, or if it started outside the box, it resulted in an error.

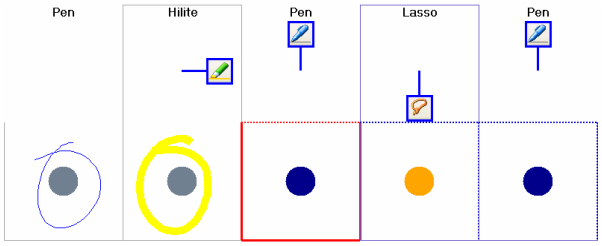

Fig. 5. Alternation task in progress (with Pen mode marking reminders for *Persists*, which requires pen reselection).

For *alternation* tasks (*Fig. 5*), the task pattern is *iCiCi*, where segment 2 is a command, segment 3 is ink, and segment 4 is a different command. During pilot testing, users would sometimes skip the ink (segment 3) if the two *C* subtasks were the same due to a tendency to chunk two identical commands together. Thus, we used different commands for the *C* segments of the *iCiCi* pattern.

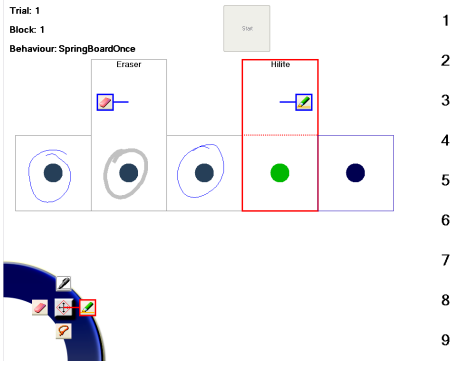

Fig. 6. Screen layout for experiment, with a user marking from the lagoon (lower left). The digits at right were not used.

The stimulus for the trial was not revealed until the user tapped on a Start button (*Fig. 6, top center*). Each stimulus shows the user where the command is in the marking menu. These cues appear above the dots so that the user's hand does not occlude them during the task. These are just static cues, not active icons (we trained users not to tap on them). For the dots that only required inking, we omitted the border around the top half of the box to emphasize that the task segment was different and required returning to the "normal" inking state. However, for the *Persists* condition only, we did provide reminders of how to reselect the pen (*Fig. 5*). The dots were color-coded using the dominant

color from the icon for its mode. These cues are intended to help users anticipate the sequence of actions, instead of reacting to the stimuli one box at a time.

We also provided users with clear visual feedback of the currently selected mode. When users completed a marking selection, the icon for the marking command remained on the screen as continuous feedback of the selected tool. Each mode also provided a custom tracking cursor.

The center of the lagoon icon was 125 pixels from the left and 125 pixels from the bottom (*Fig. 6*). The lagoon mark had to start within a 50x50 pixel square around this point. The first stimulus dot was 104 pixels from the left and 363 pixels from the top. Each dot was separated by 159 pixels.

Experimental Design

Independent variables were *MenuType* (Local Marking, Lagoon), *Behavior* (Once, Persists, Springboard, SpringOnce), and *TaskType* (Alternation, Repetition, or Baseline). Because fully crossing all of these factors results in too many conditions for a single subject to complete in a one hour session, we ran *Behavior* and *TaskType* as within-subjects factors and *MenuType* as a between-subjects factor.

We employed a 4x4 Latin square across the *Behavior* conditions to minimize order effects, with 4 participants each in 4 different orders (2 participants in each *Order* used the *Lagoon*, and 2 used the *Local Marking* menu). Thus in total, our experiment includes:

16 Participants x
  4 Behaviors x
    6 blocks (2 practice, 4 experimental) consisting of
      1 baseline trial +
      6 command trials (with 3 Alternation tasks + 3 Repetition tasks mixed in random order);
    = 42 trials (28 experimental: 4 baselines + 12 Alternations + 12 Repetitions) per Behavior.
  = 168 Trials per subject (112 experimental)
= 2688 total trials (1792 experimental).

These trials include a total of 5*2688 = 13,440 dot-circling subtask segments (8960 experimental). Half of the participants used *Lagoon* and half used *Local Marking* menus, for 2688/2 = 1344 trials (896 experimental) in each.

Dependent measures were completion time and error rate. We recorded the total trial time, which is the total time to complete all 5 task segments. To facilitate our keystroke level analyses, we also recorded separate time intervals for each of the 5 segments and for each elemental subtask contributing to the final (error-free) circling of the dot stimulus, i.e. the times to draw the mark, lift the pen, circle the dot stimulus, and move the pen between segments.

Participants & Training

16 right-handed users participated in the study. Most participants had used a tablet or a handheld, and all had used laptops or other mobile devices. All participants self-reported normal color vision, and we ensured that all participants positioned the tablet so that they could clearly distinguish the different colors of the task stimuli.

To familiarize subjects with our experimental stimuli and task, subjects started with a formal practice session consisting of 4 baseline practice trials of 5 segments each + 4 marking menu practice trials of 5 segments each). For the Local Marking conditions, we trained users to invoke the menu on or near the dot stimulus. They continued to use the Local Marking menus this way throughout the experiment.

Apparatus

Each participant used a Toshiba Portege TabletPC, running Windows XP SP2 Tablet Edition, with a 24.5 x 18.5 cm (1024 x 768 pixel) display. The tablet was used in slate mode and angled by about 10 degrees using a short monitor stand to help users achieve a desirable viewing angle. Users were not allowed to hold the tablet in their lap. We used the Enter key of a Targus USB Numeric Keypad as the COMMAND button. We were not able to use the built-in tablet bezel buttons as they are on the right side and running these Toshibas in the upside-down screen orientation significantly degrades interactive performance.

RESULTS

Participants took approximately 1 hour to complete the experiment including all practice, experimental trials, and qualitative comments on each technique.

Qualitative Results

Participants ranked the four Behaviors according to their preference. Analysis of the rank data revealed a significant overall effect for *Behavior* (Friedman's $\chi^2_{(3,N=16)} = 8.1$, $p<.05$). Seven of the 8 *Lagoon* users chose *Springboard* or *SpringOnce* as their favorite technique and one chose *Once*. For the 8 *Local Marking* menu users, 5 chose *Springboard* or *SpringOnce*, 2 chose *Persists*, and one chose *Once*. Note that participants could not contrast *Local Marking* with the *Lagoon* since *MenuType* was a between-subjects factor.

Total Trial Completion Time

This includes all time after tapping Start (*Fig. 6*) until circling the 5th dot stimulus in ink. We conducted a 4x3x4 repeated measures ANOVA on 4 *Behaviors* (*Once, Persists, Springboard, SpringOnce*) by 3 *TaskTypes* (*Baseline, Alternation, Repetition*) by 4 *Blocks*. The *MenuType* (*Local Marking, Lagoon*) and *Order* (4 presentation orders) were between-subjects factors.

We used the median completion time to correct for typical skewing of reaction time data and to remove any outliers. The median was calculated for each cell of the design at the *Behavior x TaskType x Block* level for each subject. At this level the design has three observations per subject, so we discarded the best and the worst observation, leaving the subject's median trial completion time for data analysis.

Note that some of these median trials do include errors. We left these trials in this analysis, because the costs resulting from decision making, hesitations, and errors are part of what we are trying to measure to accurately characterize the true cost of command selection [5] for each technique. Because the user had to finish all task segments correctly to complete a trial, making errors never allowed users to complete the task more quickly. We also recorded the times

for the error-free portion of each segment and used those times to determine the elemental building blocks of each technique in our keystroke-level analyses (discussed later).

Our analysis revealed a significant effect for *Behavior*, $F_{(3,24)}=6.17$, $p<.005$. Planned comparisons revealed that overall *Springboard*, *SpringOnce*, and *Once* were all significantly faster than the *Persists* condition ($p < .05$).

TaskType was highly significant, $F_{(1,8)}=145.5$, $p<.001$. *Repetition* tasks were faster than *Alternation* tasks ($p<.001$). This is a robust effect because all levels of *Behavior* (except the *Once* technique) required fewer tool mode selections for *Repetitions*. The *Baseline* was fastest ($p<.001$).

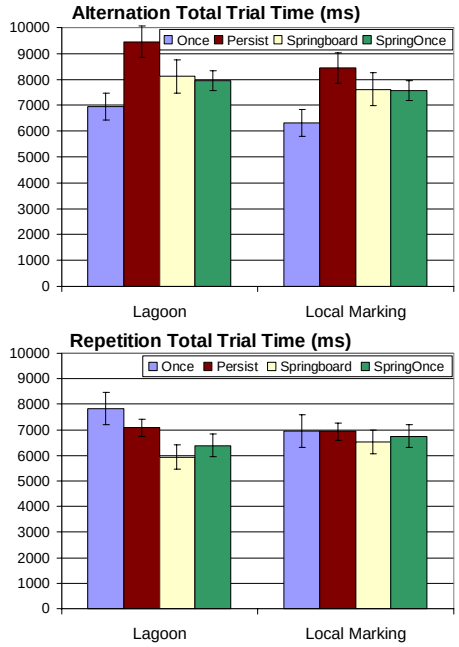

Fig. 7. Total task times from the experiment.

Behavior x TaskType was also significant, $F_{(1,8)}=27.78$, $p<.001$. This means the best mode Behavior depended on the *TaskType*, which underscores the importance of controlling for different task types as advocated by Mackay [15] and Appert et al. [2]. *Once* was the fastest behavior for *Alternation* tasks ($p<.001$). *SpringOnce* was significantly faster than both *Once* and *Persists* for *Repetitions* ($p<.05$) and faster than *Persists* for *Alternations* ($p<.05$). Although *Springboard* had the fastest absolute mean time for *Repetition* tasks, it had a higher standard deviation than *SpringOnce*. So *Springboard* was significantly faster than *Once* ($p<.001$) but not significantly faster than *Persists*.

Block was also significant, $F_{(3,24)}=15.59$, $p<.001$, indicating that participants became more proficient with increasing experience. The *Block x TaskType* interaction was also significant, $F_{(6,48)}=2.75$, $p<.05$. This suggests the *Repetition* and *Alternation* tasks exhibited a stronger learning effect across blocks than the *Baseline* task.

The *Order* between-subjects factor had no main effect, but *Behavior x Order* $F_{(9,24)}=1.93$, $p<.05$ was significant. Investigation of the means revealed the *SpringOnce-first Order* contained the slowest participant in the experiment. Our impression was that this user preferred to be neater that most participants and therefore was more deliberate when circling the dots. Repeating the analysis without this user's data showed the same significant results. If the *SpringOnce-first Order* resulted in a negative training effect, we would have expected the *Springboard-first Order* to also exhibit this effect. Since it did not, this interaction probably just reflects a participant effect.

Results for Error Analysis

A trial was an error trial if the user exhibited any mistake or sub-optimal behavior, such as unnecessarily reselecting the current mode, for any of the 5 segments. We computed the error rate for each *Behavior* and *TaskType* across all 4 *Blocks*. Since errors were not possible in the *Baseline* task, we only have 2 levels for *TaskType* in this analysis. Thus we conducted a 4 (*Behavior*) x 2 (*TaskType: Alternation* or *Repetition*) repeated measures ANOVA on error rate with *MenuType* and *Order* as between-subject factors.

There were no main effects on error rate for any within or between-subjects factors. The error rates for each *Behavior* were 9.6% for *Once*, 10.2% for *Persists*, 12.2% for *Springboard*, and 11.2% for *SpringOnce*. None of these differences were significant. However, *Behavior x TaskType* ($F_{(3,24)}=3.89$, $p<.05$) was significant. This interaction indicates that the *SpringOnce* design actually did succeed in reducing errors versus the *Springboard* for the *Alternation* task (17.7% for the *Springboard* versus 11.5% for *SpringOnce*). However, the pure *Springboard* worked well for *Repetition* (6.8% errors).

Failure to Detect any Significant Effect of *MenuType*

Perhaps the most surprising result from the experiment was that the between-subjects factor *MenuType* was not significant ($F_{(1,8)}=0.04$, $p=.85$). Overall, the *MenuTypes* only differed by 330ms (excluding baseline trials, 7137 ms for *Local Marking* vs. 7467 ms for *Lagoon*.). Prior to the experiment we estimated that a round trip to the lagoon should take about 1.5 seconds, yet experts can invoke a marking menu command in less than 500ms [11,13], which should give the *Local Marking* menu an advantage.

An interaction between independent variables potentially could negate any main effect for an experimental factor, but the only significant interaction with *MenuType* was a *Behavior x TaskType x MenuType* three-way interaction, ($F_{(6,48)}=2.53$, $p<.05$). Investigation of the means suggested that this interaction occurred because the Lagoon version of *Springboard* and *SpringOnce* tended to be faster for *Repetition* than the *Local Marking* versions of the techniques. In fact, the lowest absolute mean completion time for any technique in the entire experiment was the *Springboard Lagoon* for *Repetition* tasks, at 5938 ms. But even for the traditional *Once* and *Persists* mode techniques, which demand increased round trips to the lagoon, the advantage for *Local Marking* menus was limited.

Another possibility is that some uncontrolled effect could have obscured any effect of *MenuType*, or that our experiment lacked sufficient statistical sensitivity on *MenuType* since it was a between-subjects factor. But because we collected detailed timing information for all

elemental actions contributing to the task times, we can construct a keystroke-level model of what users had to do to complete the task and compare that to what we actually measured in the experiment. This model lets us account for all of the time that users spent completing the tasks.

KEYSTROKE-LEVEL ANALYSIS

Our keystroke-level analysis required three steps. First, we defined all of the unit actions required to perform our tasks, and translated these into keystroke-level models (KLM's) that compute the total task time in terms of these unit tasks. These models omit mental pause operators [4], because we will estimate any mental pause times from empirical data.

Second, we used our experimental observations from the *Baseline* tasks (*Fig. 3*) and the *Once* condition (which doubled as a control condition) to estimate values for these unit costs. These values can be plugged into our keystroke-level models to compute an *expected physical articulation time* based on users' performance in the control conditions.

Third, we compared the KLM models to the observed times for the *Behaviors* in the *Lagoon* and *Local Marking* menus. This shows where the techniques deviate from the model, indicating the presence of hidden costs. These costs might include increased reaction time resulting from planning what to do next, mental pauses, or delays while the user attends to visual feedback after performing an action. Our methodology cannot attribute these costs to a specific cause. It just lets us deduce that a hidden cost must exist in a specific portion of the task. This is sufficient to generate many insights as to where the bottlenecks to performance lie and what parts of a technique might be improved.

Step 1: Define KLM and Compute Unit Task Costs

A trial consisted of five subtasks (segments), each with a dot stimulus that the user circled with the correct tool. For some segments, the user already has the correct tool. The time A_P to Apply the tool after pointing to the segment is:

$\boldsymbol{A_P} = (\boldsymbol{P_S} + \boldsymbol{D_C})$, where *(Eq.1)*

$\boldsymbol{P_S}$ = Point the pen at the segment

$\boldsymbol{D_C}$ = Draw a circle around the dot

If the user needs to change the tool, the time M_C to use the Menu to pick a new command is:

$\boldsymbol{M_C} = (\boldsymbol{P_C} + \boldsymbol{D_M})$, where *(Eq.2)*

$\boldsymbol{P_C}$ = Point where the command will be activated

$\boldsymbol{D_M}$ = Draw the mark

Then, it takes time A_M to Apply the tool after marking:

$\boldsymbol{A_M} = (\boldsymbol{P_M} + \boldsymbol{D_C})$, where *(Eq.3)*

$\boldsymbol{P_M}$ = Point from end of the mark to the dot

(D_C is again the time to circle the dot)

We can now populate tables to summarize the overall costs for each technique by following several rules: (1) a segment that does not require a mode switch just takes time A_P; (2) any task segment that requires a tool mode switch requires time $M_C + A_M$ to complete; and (3) if the user may need to press COMMAND, this is noted by tallying each mode-in cost m_{in} and mode-out cost m_{out}. These always come in pairs ($m_{in} + m_{out}$) since the user eventually lets go.

The tables below, prominently labeled **R** for the Repetition task models (*Fig. 8*) and **A** for the Alternation task models (*Fig. 9*), summarize the resulting KLM models.

R	**Repetition Task**					***R* KLM**'s Computed total time
	1	2	3	4	5	
	i	**C**	**C**	**C**	**i**	
SpringBoard, SpringOnce	A_P	m_{in} M_C A_M	A_P	A_P	m_{out} A_P	1 (m_{in}+m_{out}) 1 M_C 4 A_P 1 A_M
Persists	A_P	m_{in}+m_{out} M_C A_M	A_P	A_P	m_{in}+m_{out} M_C A_M	2 (m_{in}+m_{out}) 2 M_C 3 A_P 2 A_M
Once	A_P	m_{in}+m_{out} M_C A_M	m_{in}+m_{out} M_C A_M	m_{in}+m_{out} M_C A_M	A_P	3 (m_{in}+m_{out}) 3 M_C 2 A_P 3 A_M

Fig. 8. Keystroke-Level Models for the *Repetition* task.

A	**Alternation Task**					***A* KLM**'s Computed total time
	1	2	3	4	5	
	i	**C_1**	**i**	**C_2**	**i**	
SpringBoard, SpringOnce	A_P	m_{in} M_C A_M	m_{out} A_P	m_{in} M_C A_M	m_{out} A_P	2 (m_{in}+m_{out}) 2 M_C 3 A_P 2 A_M
Persists	A_P	m_{in}+m_{out} M_C A_M	m_{in}+m_{out} M_C A_M	m_{in}+m_{out} M_C A_M	m_{in}+m_{out} M_C A_M	4 (m_{in}+m_{out}) 4 M_C 1 A_P 4 A_M
Once	A_P	m_{in}+m_{out} M_C A_M	A_P	m_{in}+m_{out} M_C A_M	A_P	2 (m_{in}+m_{out}) 2 M_C 3 A_P 2 A_M

Fig. 9. Keystroke-Level Models for the *Alternation* task.

Step 2: Estimate Values for Unit Task Costs

Because the *Once* condition was so repetitive, it required very little decision making while performing the task. Thus the component times for *Once* tended to be faster than the other conditions. This fast pace tended to continue through the baseline trials that users encountered during *Once*. For our purposes, we want the KLM models to estimate the pure physical articulation costs of each technique. Thus in our experiment the *Once* condition, as well as the baselines performed during the *Once* condition, yielded the best estimates for these pure physical articulation costs. For the times related to drawing the mark and returning to the dot after marking, for each *MenuType* we took the average time for the task segments in *Once* that contained tool selections.

Note that in our experimental data collection, we recorded each of the 5 individual time intervals defined above ($\boldsymbol{P_S}$, $\boldsymbol{D_C}$, $\boldsymbol{P_C}$, $\boldsymbol{D_M}$, and $\boldsymbol{P_M}$) for each of the 5 segments of every trial. Thus our full data set includes these 25 additional dependent measures, which allows us to calculate the *expected physical articulation times* (without decision or mental preparation costs) for each of the unit actions:

For the Local Marking Menu (denoted with subscript *m*):

$\boldsymbol{P_{Sm}}$ = 347 ms (point to the segment)

$\boldsymbol{D_{Cm}}$ = 433 ms (draw a circle around dot)

$\boldsymbol{P_{Cm}} = \boldsymbol{P_{Sm}}$ (mark in-place near dot)

$\boldsymbol{D_{Mm}}$ = 494 ms (draw the mark)

$\boldsymbol{P_{Mm}}$ = 168 ms (move to dot after marking)

$\boldsymbol{A_{Pm}}=(P_{Sm} + D_{Cm})$ **= 780 ms**

$\boldsymbol{M_{Cm}}=(P_{Cm} + D_{Mm})$ **= 841 ms**

$\boldsymbol{A_{Mm}}=(P_{Mm} + D_{Cm})$ **= 601 ms**

For the Lagoon Menu (denoted with subscript *l*):

$\boldsymbol{P_{Sl}}$ = 307 ms (point to the segment)
$\boldsymbol{D_{Cl}}$ = 440 ms (draw a circle around dot)
$\boldsymbol{P_{Cl}}$ = 558 ms (point to the lagoon)
$\boldsymbol{D_{Ml}}$ = 667 ms (draw the mark on lagoon)
$\boldsymbol{P_{Ml}}$ = 283 ms (time to return from the lagoon)
$\boldsymbol{A_{Pl}} = (P_{Sl} + D_{Cl})$ **= 747 ms**
$\boldsymbol{M_{Cl}} = (P_{Cl} + D_{Ml})$ **= 1225 ms**
$\boldsymbol{A_{Ml}} = (P_{Ml} + D_{C})$ **= 723 ms**

Thus, we can see immediately from these estimates that the time penalty for making a round trip to the lagoon was not that large. The largest advantage for *Local Marking* comes from the time to pick a command from the menu $\boldsymbol{M_C}$, with (1225 - 841) = 384ms being saved versus *Lagoon*. It only took users 558ms to move the pen to the lagoon, and the time to draw the mark on the lagoon also took (667 – 494) = 173ms longer. This may have been because users were moving the pen quickly and were still coming to a stop as they started drawing the mark on the lagoon.

The small value for $\boldsymbol{P_{Mm}}$ (time to move from the end of the mark to the dot=168ms) confirms that users did mark close to the dot. However, the segments containing each stimulus were fairly large targets, so the time $\boldsymbol{P_{Ml}}$ to return to the vicinity of the stimulus from the lagoon was only 283ms, just 122ms more than $\boldsymbol{P_{Mm}}$ for the *Local Marking* menu.

Step 3: Compare Computed Costs to Observed Costs

Reaction Time

We estimated the time required to react to the stimulus for each *MenuType*. The time to move from the Start button to begin circling the dot in the first segment includes the time to physically move the pointing device plus the unknown reaction time $\boldsymbol{R_t}$. We know the time to point from the Start button to the first dot must be a minimum of $\boldsymbol{P_{Sl}}$ = 307 ms (the average time in the *Lagoon* conditions to point to the next dot when no tool selection was required). If we attribute all of the remaining time to reaction time, this gives us a conservative estimate, yielding average reaction times of $\boldsymbol{R_{tl}}$ = 434ms for the *Lagoon* and $\boldsymbol{R_{tm}}$ = 643ms for *Local Marking*, favoring *Lagoon* by 209ms.

Mode Switching Time from Pressing the COMMAND Button

For the mode switching time (m_{in} + m_{out}), [14] reports 139ms for their nonpreferred hand button technique. From our data, the KLM for the *Repetition* task predicts the *Once* technique with the *Local Marking* menu takes:

$$= 3\boldsymbol{M_{Cm}} + 2\boldsymbol{A_{Pm}} + \boldsymbol{A_{Mm}} = 5886 + 3(m_{in}+m_{out}) \quad \textbf{(Eq.4)}$$

The actual value we observed in our experiment for this condition was 6946 ms, which includes $\boldsymbol{R_{tm}}$ = 643 ms. Assuming that all of the unaccounted time is due to pressing the button, we solve:

$$5886 + 3(m_{in} + m_{out}) = 6946 - 643, \text{ yielding} \quad \textbf{(Eq.5)}$$
$$\boldsymbol{(m_{in} + m_{out}) = 177ms}$$

This agrees fairly closely with Li et al. [14] and our task was more complex so it makes sense that mode switching time could have increased slightly. Thus we will use (m_{in}+m_{out}) = 177ms for both *Local Marking* and *Lagoon*, and we will assume $m_{in} \approx m_{out}$ = (177 / 2) = 88.5 ms.

Computed Costs: Average Total Trial Times

We can now calculate the average total trial times we would expect based on the unit task times. To do this, we compute the sum of the KLM models for all 4 *Behaviors* and the 2 tasks (*Repetition* and *Alternation*):

Local Marking expected average trial time $\boldsymbol{MT}$: **(Eq.6)**

$$\boldsymbol{MT} = [(17\boldsymbol{M_{Cm}} + 23\boldsymbol{A_{Pm}} + 17\boldsymbol{A_{Mm}}) + 17(m_{in}+m_{out})] / (4*2)$$
$$= 5758\text{ms total (5306ms excluding } m_{in}+m_{out})$$

The *Lagoon* conditions require fewer (m_{in}+m_{out}) actions because they do not use the button for *Once* or *Persists*. Furthermore, we observed that for *Springboard* and *SpringOnce*, users tended to press the button while moving the pen to the lagoon. Thus, it seems likely that some or all of the m_{in} mode-in cost for the Lagoon occurs in parallel with the $\boldsymbol{P_{Cl}}$ unit task, so we only count m_{out} time for the *Lagoon* conditions where the button was needed:

Lagoon expected average trial time $\boldsymbol{LT}$: **(Eq.7)**

$$\boldsymbol{LT} = [(17\boldsymbol{M_{Cl}} + 23\boldsymbol{A_{Pl}} + 17\boldsymbol{A_{Ml}}) + 6(m_{out})] / (4 * 2)$$
$$= 6354\text{ms total (6287ms excluding } m_{out})$$

Erosion of the Time-Motion Benefits for Local Marking

Without the button presses, the models above predict the advantage for *Local Marking* should be (6287-5307)= 980ms. This is the theoretical time-motion efficiency advantage that *Local Marking* menus should exhibit due to elimination of round trips, as reflected in the smaller values for its $\boldsymbol{M_{Cm}}$ and $\boldsymbol{A_{Mm}}$ unit task costs compared to $\boldsymbol{M_{Cl}}$ and $\boldsymbol{A_{Ml}}$ for the *Lagoon*. But once we consider the (m_{in}+m_{out}) button presses, this advantage is reduced 310ms due to the increased button press costs for *Local Marking*. Adding in the 209ms increased reaction time further undermines the advantage for *Local Marking*, reducing the theoretical 980ms advantage by a total of 519ms to yield a net predicted advantage of only **461ms** compared to *Lagoon*.

This agrees quite closely with the average *Local Marking* vs. *Lagoon* total trial time difference of 330ms that we observed in our experiment, leaving only 131ms unaccounted for. This small remaining difference may result from limitations of our models, the occasional time that participants lost due to errors, or other random between-subject variations. Thus, altogether these analyses confirm that it was unlikely we could have observed a significant advantage for *Local Marking* menus for our experimental task.

Mental Preparation Costs in the Mode Behaviors

We can identify hidden costs in the *Once*, *Persists*, *Springboard*, and *SpringOnce* techniques by comparing the observed times for each segment to the values predicted by our models. For this analysis, for each task segment, we computed the sum of all the times for command selection (if any) and circling of the dot stimulus in the correct mode.

Fig. 10 shows the results. Due to random deviation from the average unit task times, users sometimes completed subtasks faster during *Once* than our models predicted. For *Alternation* tasks, the *Springboard* and *SpringOnce* designs both appear to require extra decision making time during the two *C* command subtasks, but despite this were still

faster overall than *Persists*. This suggests that *Persists* (and *Once*) are mostly constrained by physical articulation time, whereas the bottleneck for *Springboard* and *SpringOnce* is decision time. For *Repetition* tasks, *Persists*, *Springboard*, and *SpringOnce* all appear to have a mental set-up cost in segment 2 (the first command *C*) as users decide whether or not to use the tool multiple times in a row.

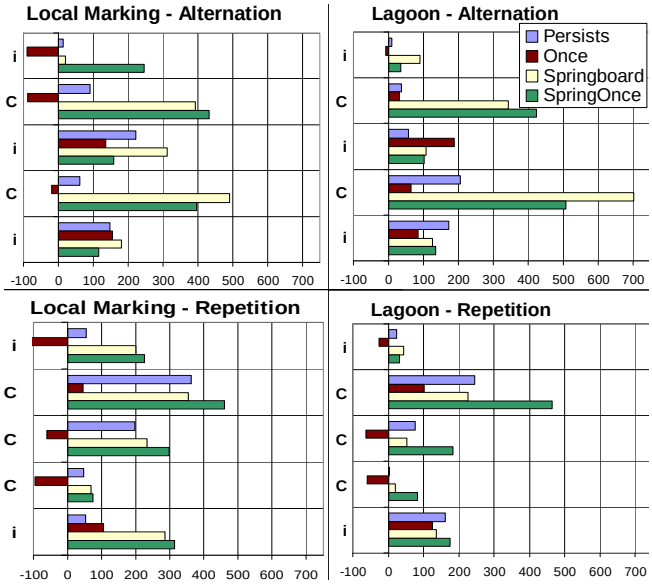

Fig. 10. Difference in observed times vs. computed times for each *Behavior* and *MenuType*. The vertical axis shows subtask segments 1 (top) to 5 (bottom). Time (ms) is horizontal axis.

DISCUSSION

Can Local Marking Perform Better?

Our models suggest that *Local Marking* would have fared better if no movement had been required to proceed to the next task segment. The *Local Marking* menu defined the time to move to the location where the mark is drawn as $\boldsymbol{P_{Cm}} = \boldsymbol{P_{Sm}}$ =347ms, which is just saying that it is the time to point to the next segment. If the user can keep working in the same spot, $\boldsymbol{P_{Cm}}$ would drop to some minimum time needed to lift the pen between commands, say 100ms. In this scenario our models predict that the local marking menus would gain another 1200ms performance advantage over the *Lagoon*, because the round trip time is still required to pick a command with the lagoon and cannot be eliminated. Thus, in a sense the time to move between the experimental stimuli was an added cost for *Local Marking*, but not for the *Lagoon*, because the round trip time absorbs the time needed to move between stimuli. It is not clear how often real-world tasks would have sufficiently high locality of reference for the *Local Marking* menu to avoid this 'extra cost' and gain the calculated benefit of 1200ms.

We can also consider *N* repetitions of a command rather than just 3 repetitions. Our observed time for the *Local Marking* menu's *Persists* condition was actually slightly slower than *Once* even for *Repetition* tasks (*Fig. 7*). By generalizing the model for the *Persists* repetition task in terms of *N*, the number of commands performed in a row, we can solve for *N* and determine that the break even point for *Persists* is *N*=4.8, so at least 5 repetitions would be necessary for the *Persists* technique to be worthwhile with the *Local Marking* menu. Similar calculations show that our *Springboard* designs had paid for the extra set-up time to invoke them by the second command invocation on the *Lagoon* and by the third command for *Local Marking*.

We can also consider use of round-trip designs such as the lagoon on larger displays or for tasks that require greater precision of pointing. For example, if we double the round trip times $\boldsymbol{P_{Cl}}$ and $\boldsymbol{P_{Ml}}$, this would increase the *Lagoon's* average task times for our experimental tasks by approximately 2 seconds overall, which again would provide a scenario where *Local Marking* offers a benefit.

We anticipate that *Local Marking* may have increased benefit if the starting location of a mark chooses the object to act upon, because it integrates selection of an object with command selection [3,9]. Like right-click, this limits the commands offered in a menu to those valid for the selected object. For these reasons, our current results should not be extrapolated to commands that require a prior selection.

Finally, localized user interfaces (whether marking menus or something else) may offer benefits to users that were not captured by our experimental task. For example, localized interfaces may reduce the physical effort needed to move around the screen. Recent results suggest they may also prevent visual diversion of attention from one's work [7].

Design Issues for the *Springboard*

There are several design questions for the *Springboard*:

Does a tool stay selected across invocations? Three users commented that the *Springboard Lagoon* should reactivate the prior tool if they hit COMMAND and stroke the pen without going to the lagoon. We find this feature works well (see video). It allows users to quickly interleave inking with another mode, thus facilitating tasks such as panning while annotating a document, or going through a document with a highlighter while jotting down notes. It may not be possible to support this desired feature for *Local Marking*.

How to provide a spring-loaded control for the nonpreferred hand on a tablet? Our users found the Enter key acceptable for use on a supporting surface, but felt that COMMAND should be part of the tablet. We are currently exploring placement of a button on the Tablet PC bezel, as well as other spring-loaded control designs. A control should be accessible for both left and right handed use from any of the 4 screen orientations, and it should not cause inadvertent activations. Also, some new Tablet PC designs make the keyboard more accessible, and some Wacom desktop tablets offer side buttons suitable for COMMAND.

Are other design hybrids with the Springboard possible? Our KLM models show that *Springboard* and *SpringOnce* can theoretically go as fast as the *Once* technique for *Alternation* tasks (*Fig. 9*), yet in our experiment users were not able to realize this level of performance. It is possible that experts could use the techniques more efficiently, or further design hybrids may be possible. A lagoon that always stays active could support both the *Once* behavior and the *SpringOnce* behavior. If the user employs the

lagoon without the COMMAND button, the lagoon could default to the *Once* behavior, which is easy for novices to learn, and is efficient in many situations. This is also useful if a device lacks COMMAND, or if the user eschews COMMAND due to fatigue or laziness. But if the user instead presses and holds COMMAND prior to completing the first application of the tool mode, this could trigger *SpringOnce*.

Does the lagoon stay visible when inactive? We tried a fully disappearing lagoon, but it seems important to have a visual target in order to make a ballistic motion from the work area to the lagoon. Without a target to aim for, users must move to the general area, and then make a corrective movement when the lagoon appears, which might reduce performance. Future studies should investigate this further.

CONCLUSION AND FUTURE WORK

Our experimental results showed that users preferred *Springboard* and *SpringOnce* to the status-quo *Once* and *Persists* techniques. One user felt strongly that using the button was a pain, but most users liked having a choice that let them control how long a mode stayed active. Although *Springboard* and *SpringOnce* performed similarly in the experiments, the *SpringOnce* design did tend to result in a lower error rate than the *Springboard*, particularly for *Alternation* tasks. This suggests that *SpringOnce* can provide an effective mechanism for users to manage multiple tool modes from a single spring-loaded control.

The new designs were faster than *Persists* overall and were faster than *Once* for *Repetition* tasks. However, *Once* remains the technique of choice for alternation; in this case it would be hard to beat *Once*, unless the user keeps interleaving the same command and uses the *Springboard* design option suggested above where the tool remains selected across invocations of the lagoon. Thus, exploration of design hybrids with *Once* offers a promising avenue for mode behaviors that can support a variety of task contexts.

Our keystroke-level analyses showed that the *Local Marking* menu has a time-motion efficiency advantage in theory, but in practice this advantage is smaller than we anticipated and is eroded by several ancillary costs, such as mode switching and reaction time, that appear to favor the *Lagoon* menu design. Failing to find a statistically significant difference cannot prove that there is no difference between the *Lagoon* and the *Local Marking* menu. But the failure to detect this effect combined with the results of our keystroke level analysis, which suggest why *Local Marking* failed to significantly outperform the *Lagoon*, together make a convincing case that it is difficult for the *Local Marking* menu to provide a substantial benefit in the task context that our experiment studied.

Thus, elimination of round trips does not necessarily improve the speed of tool switching on tablet computers. Our hope is that our results and methodology can offer better insights into factors influencing performance for pen and gesture interfaces. We must continue to explore alternative task contexts, novel localized interface designs, or other metrics of user performance where pen and gesture interfaces can offer significant performance advantages.

ACKNOWLEGEMENT

This work has been supported in part by the Microsoft Center for Interaction Design and Visualization at the University of Maryland and NSF under grant IIS-0414699.

REFERENCES

1. Apitz, G., Guimbretiere, F. *CrossY: A crossing based drawing application*. UIST 2004, 3-12.
2. Appert, C., Beaudouin-Lafon, M., Mackay, W. *Context matters: Evaluating interaction techniques with the CIS model*. Proc. of HCI 2004, Springer Verlag, 279-295.
3. Bier, E., Stone, M., Pier, K., Buxton, W., DeRose, T. *Toolglass and Magic Lenses: The See-Through Interface*. Proceedings of SIGGRAPH 93, 73-80.
4. Card, S., Moran, T., Newell, A., *The Keystroke-Level Model for User Performance Time with Interactive Systems.* Communications of the ACM, 1980. **23**(7): p. 396-410.
5. Dillon, R., Eday, J., Tombaugh, J., *Measuring the True Cost of Command Selection: Techniques and Results.*CHI'90, 19-25.
6. Fitzmaurice, G., Khan, A., Pieke, R., Buxton, B., Kurtenbach, G. *Tracking Menus*. UIST 2003, 71-79.
7. Grossman, T., Hinckley, K., Baudisch, P., Agrawala, M., Balakrishnan, R. *Hover Widgets: Using the Tracking State to Extend the Capabilities of Pen-Operated Devices*. CHI 2006.
8. Guimbretiere, F., Winograd, T. *FlowMenu: Combining Command, Text, and Data Entry*. UIST 2000, 213-216.
9. Hinckley, K., Baudisch, P., Ramos, G., Guimbretiere, F. *Design and Analysis of Delimiters for Selection-Action Pen Gesture Phrases in Scriboli*. CHI 2005, 451-460.
10. Johnson, J., *Modes in non-computer devices.* Int. J. Man-Mach. Studies, 1990. **32**(4): 423-438.
11. Kurtenbach, G., Buxton, W. *The Limits of Expert Performance Using Hierarchic Marking Menus*. INTERCHI'93, 482-487.
12. Kurtenbach, G., Fitzmaurice, G., Owen, R., Baudel, T. *The Hotbox: Efficient Access to a Large Number of Menu-items*. CHI'99, 231-237.
13. Kurtenbach, G., Sellen, A., Buxton, W., *An emprical evaluation of some articulatory and cognitive aspects of 'marking menus'.* J. Human Computer Interaction, 1993. **8**(1).
14. Li, Y., Hinckley, K., Guan, Z., Landay, J. A. *Experimental Analysis of Mode Switching Techniques in Pen-based User Interfaces*. CHI 2005, 461-470.
15. Mackay, W. E. *Which Interaction Technique Works When? Floating Palettes, Marking Menus and Toolglasses Support Different Task Strategies*. ACM AVI 2002, 203-208.
16. McLoone, H., Hinckley, K., Cutrell, E. *Ergonomic Principles Applied to the Design of the Microsoft Office Computer Keyboard*. IEA 2003 International Ergonomics Association.
17. Norman, D. A., *Categorization of Action Slips.* Psychology Review, 1981. **88**(1): p. 1-15.
18. Poller, M., Garter, S., *The Effect of Modes on Text Editing by Experienced Editor Users.* Human Factors, **26**(4): 449-462.
19. Pook, S., Lecolinet, E., Vaysseix, G., Barillot, E. *Control Menus: Execution and Control in a Single Interactor*. CHI 2000 Extended Abstracts, 263-264.
20. Raskin, J., *The Humane Interface: New Directions for Designing Interactive Systems*. 2000: ACM Press.
21. Sellen, A., Kurtenbach, G., Buxton, W., *The Prevention of Mode Errors through Sensory Feedback.* J. Human Computer Interaction, 1992. **7**(2): p. 141-164.
22. Tesler, L., *The smalltalk environment.* Byte **6**(8): p. 90-147.
23. Wilson, F. R., *The Hand: How its use shapes the brain, language, and human culture*. 1998, New York: Pantheon.
24. Zhao, S., Balakrishnan, R. *Simple vs. Compound Mark Hierarchical Marking Menus*. UIST 2004, 33-42.