On the Costs of Multiple Trajectory Pointing Methods

Philip Quinn,[1] Andy Cockburn,[1] Kari-Jouko Räihä,[2] Jérôme Delamarche[3]
[1]Dept. of Computer Science, University of Canterbury, New Zealand;
philip.quinn@canterbury.ac.nz, andy@cosc.canterbury.ac.nz
[2]School of Information Sciences, University of Tampere, Finland; kari-jouko.raiha@cs.uta.fi
[3]Polytech' Paris-Sud, Orsay, France; jerome.delamarche@gmail.com

ABSTRACT

Several enhanced pointing techniques aim to reduce the Fitts' law targeting distance by providing multiple target trajectories in the hope that a shorter path is available. However, these techniques introduce a search or decision component to pointing – users must examine the alternatives available and decide upon the trajectory to use. We analyse these difficulties, present a methodology for examining them as well as other behaviour issues, and report empirical results of performance with pointer wrapping and Ninja cursors. Results show that offering multiple trajectories incurs a significant search or decision cost, and that users are therefore poor at capitalising on the theoretical benefits of reduced target distance.

Author Keywords
Pointing, Fitts' law, Ninja cursors, wrapping cursors, multiple trajectories, search/decision.

ACM Classification Keywords
H5.2. Information interfaces and presentation: User Interfaces – Evaluation/methodology.

General Terms
Human Factors, Theory, Verification

INTRODUCTION

Several approaches to improving pointing performance with a mouse focus on reducing the index of difficulty (*ID*) of the pointing task, as described by Fitts' law [8]. That is, by manipulating the representations of the cursor or the target, they reduce the target distance or increase target width.

One approach for reducing target distance is to provide multiple pointing trajectories. Kobayashi and Igarashi describe *Ninja cursors* [7], in which multiple cursors are controlled in unison with a single mouse, allowing a different trajectory for each cursor. Another multiple trajectory approach is to allow the cursor to 'wrap around' the screen edge: for example, exiting the East side enters the West.

However, such approaches introduce a search or decision element to the task of pointing. When there is only one trajectory, the choice of how to move the cursor is obvious; but when multiple alternatives are introduced, users must first select a trajectory and then execute the movement. There is a risk that the search/decision cost could reduce (or eliminate) the benefits of providing these shortcut paths. Figure 1 illustrates efficient (black) and inefficient (red) trajectories for pointer wrapping and Ninja cursors when moving from the right to left targets.

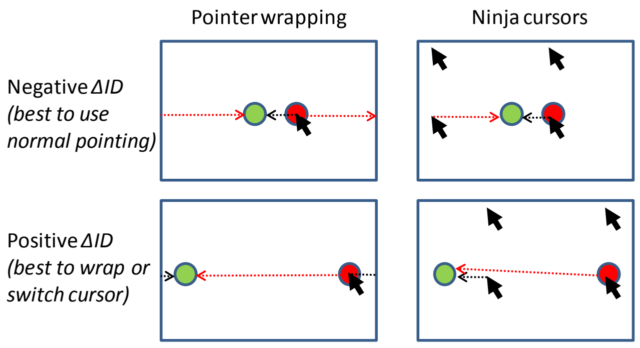

Pointer wrapping
Ninja cursors
Negative ΔID (best to use normal pointing)
Positive ΔID (best to wrap or switch cursor)

Figure 1. Multiple trajectories for pointer wrapping (left) and Ninja cursors (right). Top row shows a cost to using the alternative trajectory; bottom row shows a benefit.

We developed a methodology for examining the decision component of pointing techniques by comparing them to a user's expected direct pointing performance as described by Fitts' law. We were also interested in examining the selected trajectories chosen when presented with these techniques – do users make the right decisions?

The following section briefly reviews research on pointing models and multiple trajectory techniques. We then present a method for examining performance with these techniques. Finally, we report experimental results for *pointer wrapping*, where the cursor can wrap around screen borders, and *Ninja cursors* in configurations of four and nine cursors.

BACKGROUND

Fitts' law [8] is a well-established model of target acquisition, predicting movement time $MT = a + b \times ID$, where $ID = \log_2(D/W + 1)$, and D and W are target distance and width. Since the seminal work of Card *et al.* [2], decision times in interaction have also been examined in HCI, with recent examples including Hinckley *et al.*'s [6] analysis of 'Springboard' mode control mechanisms, and Cockburn and Gutwin's [3] investigation of hierarchical navigation.

Pointing Techniques

Ninja cursors [7] display multiple cursors that are moved in unison with the mouse. While only one cursor can be 'active' at any time, users can use any cursor to acquire a target. To prevent targeting ambiguity a queuing algorithm determines which cursor is 'active'. For each cursor on the screen, an independent *ID* and *MT* can be calculated; with a good distribution of cursors, the minimum *ID* can be significantly less than that of single-cursor pointing.

Kobayashi and Igarashi [4] evaluated Ninja cursors in several cursor configurations and distracter densities, using tasks that intentionally removed the search/decision component of pointing. Ninja cursors significantly outperformed traditional pointing, except in conditions with high target density. Researchers have subsequently used eye gaze to eliminate the inefficiency of the queuing algorithm [1, 9]. Blanch and Ortega [1] separately analysed reaction time and movement time, observing high reaction times for their *Rake Cursor* technique, which they attributed to the need to decide between alternative cursors.

Another technique, *pointer wrapping*, uses a single cursor with the traditional appearance. However, the cursor does not stop when it reaches the edge of the screen; instead, it 'wraps' around to the opposite edge of the display. For example, when the cursor tries to move beyond the bottom of the display, it re-appears at the corresponding horizontal position at the top of the display (and similarly for the left and right edges). Pointer wrapping introduces multiple trajectories through the possible wrapping actions: users can either move directly, or wrap around edges of the screen.

Related techniques eliminate all non-target space: the *Bubble Cursor* [4] enlarges the cursor activation area to always enclose a target, and *Object Pointing* [5] analyses motion kinematics to jump between targets. These techniques may incur search/decision costs, even though they do not create multiple trajectories.

EVALUATING DECISION COSTS

Ninja cursors and pointer wrapping both introduce a search or decision component to pointing interactions by giving users a choice between multiple valid trajectories. Our goals are to examine the size and impact of this component on performance and to examine how well users select paths.

Neither of these techniques alters the representation of the cursor or target; once a trajectory/cursor has been chosen, the task is a direct pointing task and can be modelled by Fitts' law. With this observation, if the Fitts component is known, then whatever additional time remains in the pointing task can be attributed to search/decision factors. Therefore, by calibrating the Fitts *a* and *b* parameters for a user, their expected pointing times can be calculated and compared to actual pointing times with other techniques.

In order to examine trajectory choice, the benefits of each possible trajectory must be controlled. Pointing experiments typically control the *ID* of the target selection; however, this is insufficient when there are multiple possible *ID*s available, as single *ID* does not control the cost or benefit of choosing a different trajectory over the direct pointing path.

When multiple pointing trajectories are present, the user needs to determine: (1) if they need to switch cursors; and (2) which path to switch to. For Ninja cursors, this is a decision between cursors against the one that has their present focus (assumed to be the one used to select the previous target). For pointer wrapping, the decision is between direct pointing or wrapping around the screen (for brevity, we will refer to both of these activities as 'switching').

The switching decision requires a comparison between all possible *ID*s, but ultimately the decision is between the best possible *ID* for not switching and the *ID* for switching: that is, $\Delta ID = ID_{ns} - ID_s$. Controlling ΔID allows isolation of the switching cost/benefit. It also allows examination of how switching behaviour varies with theoretical benefit.

ID_{ns} is the direct pointing *ID*, when not switching cursors or wrapping around the screen. ID_s is the minimum *ID* achievable when switching cursors – the *ID* from the closest cursor (exclusive of the one currently used), or when wrapping around an edge of the screen. A negative ΔID indicates a theoretical cost to switching, while a positive ΔID indicates a benefit, as illustrated in Figure 1.

EXPERIMENT

We conducted an experiment to investigate pointing and decision time behaviour in three interfaces: pointer wrapping, Ninja cursors with four cursors ('Ninja-4'; in a 2×2 arrangement), and Ninja cursors with nine cursors ('Ninja-9'; in a 3×3 arrangement). Additionally, each participant completed a Fitts' law calibration phase.

Our implementation of Ninja cursors matched that described by Kobayashi and Igarashi [7]. Cursors were spaced evenly across the screen, matching the aspect-ratio of the display. As there were no distracter targets, no queuing algorithm was necessary and there were no limitations on which cursor could be used to make selections. Cursors wrapped to the opposing edge when they reached the edge of the display, but targets were controlled so that on-screen cursors were initially closer to the target than wrapped ones.

Apparatus and Participants The experiment ran on an Intel Core 2 Duo machine running Fedora 12 and was presented on a 22″ LCD monitor at a resolution of 1680×1050. Input was collected through a Microsoft Wheel Mouse Optical.

Fifteen volunteers (three female) participated in the experiment. All were post-graduate computer science students and each received a $10 gift certificate.

Task and Stimuli All participants completed four experimental stages: a Fitts calibration (direct pointing) first, and then pointer wrapping, Ninja-4, and Ninja-9 in a counterbalanced order.

Two target acquisition tasks were used: *random* selections and *bi-directional tapping*. In *random* tasks, targets were selected randomly from the collection of possible targets (see below); each target was a red circle, 40 pixels in diameter and was the only element visible on the screen (aside from the cursor(s)). In *bi-directional tapping* tasks, pairs of targets were chosen and presented to participants; the target to select was presented as a solid red circle, while the paired target was shown as a hollow black outline. The order of *random* and *bi-directional tapping* within each stage was counter-balanced between participants.

In *random* stages, participants selected 46 targets (the first target was discarded, as its position relative to the cursor was not controlled). In *bi-directional tapping* stages, fifteen pairs of targets were presented serially, and participants selected each target in the pair four times (the first target of the pair was selected five times, but the initial selection was discarded). *Random* and *bi-directional tapping* tasks were used to provide insights into how users make trajectory decisions in different types of activities. *Random* tasks involve a unique decision for every target, but *bi-directional tapping* involves repeated reciprocal trajectories, so the costs of an initial decision can be dissipated across improved performance in many trials.

In the Fitts calibration stage, participants selected or tapped between targets with *ID*s of {1, 2, 3, 4, 5} bits, within a tolerance of 0.0001 bits. Each *ID* value was used for nine trials. The remaining stages used a similar process, except targets were generated for Δ*ID*s {-0.5, 0, 0.5, 1, 1.5} with wrapping and {-1, 0, 1, 2, 3} for Ninja cursors. The Δ*ID* ranges differ between wrapping and Ninja cursors due to target distance being bounded by cursor separation in Ninja cursors. Consequently, results cannot be directly compared between the wrapping and Ninja cursors conditions.

Additionally, the distance of the target was controlled to ensure that targets at high Δ*ID*s for the Ninja-9 condition were distributed over the full range of possible targets.

Procedure and Design The dependent measure was selection time: from the instant the target was displayed to successful acquisition. It was analyzed for each interface using a 2×5 within-subjects ANOVA for the factors *task* (*random* selection and *bi-directional tapping*) and Δ*ID*.

Each trial continued until successful target selection. After each selection, the cursor was snapped to the centre of the target to ensure an accurate *ID* in the following trial.

In summary, the experiment consisted of:

4 interfaces × 5 *ID*s or Δ*ID*s ×
(3 tap pairs × 8 trials per pair + 9 random targets/trials)
= 660 target selection trials per participant

Prior to each stage, participants were instructed on the interface and completed fifteen sample *random* selection trials. Participants were instructed to perform the tasks as quickly and accurately as possible; they could take breaks between each task, and completed a questionnaire after each stage. The experiment lasted approximately 20 minutes.

RESULTS

For each participant, the Fitts calibration data stage was used to determine their *a* and *b* parameters. These values were used to calculate the participant's expected pointing times with the three multi-trajectory interfaces (wrapping, Ninja-4, Ninja-9), each analysed separately below.

Pointer Wrapping

An analysis of variance revealed significant main effects on the selection time for the factors *task* ($F_{1,14} = 75.9, p < .001$) and Δ*ID* ($F_{4,56} = 18.4, p < .001$).

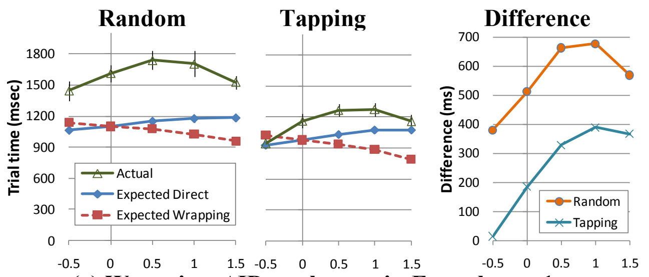
Figure 2(a) shows results for pointer wrapping in the *random* (left) and *bi-directional tapping* tasks (middle). Each plot shows actual mean times as well as Fitts' predictions for direct (ID_{ns}) and wrapping (ID_s). They show that actual performance was substantially slower than Fitts' law predictions for both the optimal trajectory *and* for the suboptimal trajectory (when the user takes the wrong path). The rightmost plot shows the difference between actual and predicted optimal performance. Even with *bi-directional tapping*, where users could recover the initial decision cost through multiple trials, actual performance was substantially worse than the optimum Fitts' predictions. It is interesting that actual performance was best when the theoretical costs of switching were highest (negative Δ*ID*) and worst when Δ*ID* was slightly positive (representing a marginal benefit for switching). This suggests that hard decisions, where the trajectories are evenly balanced, take longer to make (nearly 700 ms, as shown in Figure 2(c)), making performance much worse than if the multiple trajectory option was absent.

Enabling multiple trajectories reduced participants' performance beyond that attainable with normal pointing – the decision costs of between 380 and 677 ms (for *random* targets) overwhelmed any potential benefit in all conditions.

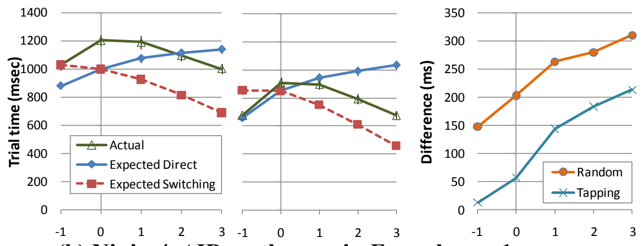
Ninja cursors

Analysis of variance also revealed significant main effects on the selection time for the Ninja-4 interface for the factors *task* ($F_{1,14} = 250.7$, $p < .001$) and Δ*ID* ($F_{4,56} = 39.1$, $p < .001$); and similarly for Ninja-9, *task* ($F_{1,14} = 144.5$, $p < .001$) and Δ*ID* ($F_{4,56} = 28.7$, $p < .001$). Figures 2(b,c) show results for Ninja-4 and Ninja-9 respectively.

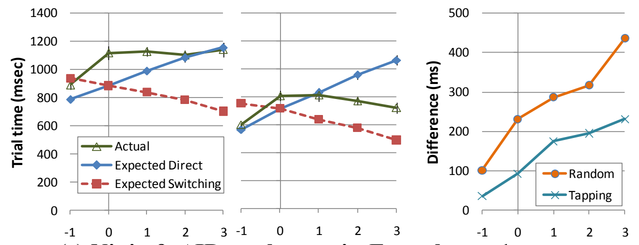
With *random* Ninja-4 targets (Figure 2(b), left), participants' actual performance matched that of normal pointing ('expected direct') at Δ*ID* = 2, and outperformed it at Δ*ID* = 3, showing that for very distant targets Ninja cursors can offer a performance benefit. However, the plot also shows that Ninja cursors' multiple trajectories reduced performance beyond that expected for normal pointing with Δ*ID* in the range –1 to +1. The plots for Ninja-9 (Figure 2(c)) show similar performance trends: for *random* trials it reduced performance beyond normal pointing in all but the most favourable conditions. The search/decision costs for *random* targets ranged from 148 to 310 ms with Ninja-4, and from 102 to 436 ms with Ninja-9. Overall, Ninja cursors reduced performance with *random* targets.



(a) Wrapping. ΔID on the x-axis. Error bars ±1 s.e.m.



(b) Ninja-4. ΔID on the x-axis. Error bars ±1 s.e.m.



(c) Ninja-9. ΔID on the x-axis. Error bars ±1 s.e.m.

Figure 2. Results for the three interface conditions with random and tapping tasks (left and middle columns). Lines show actual and Fitts' predicted times. The right column shows the difference between actual and Fitts' predicted optimal times.

Performance was better in *bidirectional tapping*, with both Ninja-4 and 9 showing performance benefits when $\Delta ID > 1$.

DISCUSSION

In almost all conditions, actual performance was substantially slower than the optimal Fitts' law predictions. Performance with multiple trajectory methods was also worse than normal pointing Fitts' law models (without multiple trajectories) in all but the most favourable conditions.

The search/decision times associated with pointer wrapping overwhelmed any potential advantage of shortened trajectory – even when the theoretical advantages or penalties (allowing it to be easily ignored) of wrapping were large.

Participant comments identified major differences in the nature of the search/decision costs between pointer wrapping and Ninja cursors. With wrapping, participants referred to *'getting confused'*, '*hard to decide', and 'easier to ignore'*, indicating that it demanded a cognitive decision about the trajectory, as supported by the high time difference between theoretical and actual times (nearly 700 ms, Figure 2a, right). With Ninja cursors, however, participants referred to *'seeing the nearest cursor straight after beginning'*, indicating a visual search pop-out effect. The rightmost plots in Figures 2(b,c) also show comparatively small differences between theoretical and actual performance with Ninja cursors (up to 300 ms with Ninja-4 and up to 450 ms with Ninja-9).

Very few experimental conditions showed benefits for multiple trajectory techniques with *random* targets (only Ninja-4 in the $\Delta ID = 3.0$ level). This shows a major challenge for multiple trajectory techniques because the random condition best resembles real world pointing (users seldom repeatedly tap between targets as tested with *bi-directional tapping*). In all other conditions, the decision cost of these interfaces was too high for their theoretical advantages to be realized. It might be tempting to think that users could be given the option to 'turn on' Ninja cursors for distant targets, but these results suggest that each additional decision costs time, and adding further decision points is likely to further harm performance unless targets are very distant.

CONCLUSIONS AND FUTURE WORK

We presented an examination of the search/decision cost in pointing interfaces that provide multiple possible target trajectories. We carried out a controlled experiment where pointing with two such interfaces – pointer wrapping and Ninja cursors – were tested and compared with Fitts' law predictions for optimal trajectories and for traditional pointing with a single cursor. Results showed that actual performance was far from optimal, and that traditional pointing outperforms multiple trajectories in many conditions – the search/decision costs overwhelm the benefits.

Further work includes evaluating multiple trajectories with very large screens and multi-monitor environments, further analysis of the independent effects of target distance and width, and analysis of other enhanced pointing techniques such as Bubble Cursor [4] and Object Pointing [5].

REFERENCES

1. Blanch, R. and Ortega, M. Rake cursor: improving pointing performance with concurrent input channels. in *Proc. CHI'09*, ACM, (2009), 1415-1418.
2. Card, S.K., Moran, T.P. and Newell, A. *The Psychology of HCI*. Lawrence Erlbaum Associates, 1983.
3. Cockburn, A. and Gutwin, C. A predictive model of human performance with scrolling and hierarchical lists. *HCI J. 24*, 3 (2009), 273-314.
4. Grossman, T. and Balakrishnan, R. The bubble cursor. in *Proc. CHI'05*, ACM, (2005), 281-290.
5. Guiard, Y., Blanch, R. and Beaudouin-Lafon, M. Object pointing: a complement to bitmap pointing in GUIs. in *Proc. Graphics Interface'04*, (2004), 9-16.
6. Hinckley, K., Guimbretiere, F., Baudisch, P., Sarin, R., Agrawala, M. and Cutrell, E. The Springboard: multiple modes in one spring-loaded control. in *Proc. CHI'06*, ACM, (2006), 181-190.
7. Kobayashi, M. and Igarashi, T. Ninja cursors: using multiple cursors to assist target acquisition on large screens. in *Proc. CHI'08*, ACM, (2008), 949-958.
8. MacKenzie, I.S. Fitts' law as a research and design tool in human-computer interaction. *HCI J. 7* (1992), 91-139.
9. Räihä, K. and Špakov, O. Disambiguating Ninja cursors with eye gaze. in *Proc. CHI'09*, (2009), 1411-1414.