RAPPORT DE RECHERCHE

L R I

VECTOR POINTING : OBJECT VS. PIXEL SELECTION IN GRAPHICAL USER INTERFACES

GUIARD Y / BLANCH R / BEAUDOUIN-LAFON M

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud-LRI

12/2003

Rapport de Recherche N° 1379

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 650
91405 ORSAY Cedex (France)

Vector Pointing: Object vs. Pixel Selection in Graphical User Interfaces

Yves Guiard [a] **Renaud Blanch** [b] **Michel Beaudouin-Lafon** [b]

[a] Laboratoire Mouvement et Perception
CNRS & U. de la Méditerranée, Marseille, France
guiard@laps.univ-mrs.fr

[b] LRI & INRIA Futurs
Université Paris-Sud, Orsay, France
{blanch, mbl}@lri.fr

ABSTRACT
Pointing, a cornerstone of our graphical user interfaces (GUIs), has been conceptualized and implemented so far as the act of selecting pixels in bitmap displays. This bitmap pointing (BMP) technique is generally sub-optimal because it requires the production of continuous information with the mouse whereas the system just needs the discrete specification of objects. We introduce vector pointing (VP), a novel interaction technique based on discrete object selection with a special screen cursor that skips empty spaces, thus eliminating this waste of input information. We report data from 1D and 2D Fitts' law experiments showing not only that VP outperforms BMP, but also that the performance facilitation increases with task difficulty. We discuss the implementation of VP in current GUIs.

Keywords
Input and Interaction Technologies. Analysis Methods. Empirical Methods. Quantitative.

INTRODUCTION
Since the advent, about two decades ago, of graphical user interfaces (GUIs), which led to the so-called WIMP paradigm (Windows, Icons, Menus, and Pointers) for human-computer interaction (HCI), sustained efforts have been made to optimize pointing, the key elemental act that permits selection among graphical objects such as icons, buttons, menu items, or hypertext links.

Supra-Normal Facilitation of Pointing in HCI
Interestingly, it is only recently that HCI researchers realized that interface designers can do better than just making pointing to a graphical object as easy as pointing to a real-life object—one should try to make pointing in a GUI definitely *easier* than normal [6,10,16,19].

It has been known, from the beginning of the WIMP era [7], that in GUIs target acquisition time (or movement time, *MT*) is almost entirely determined by the ratio of target distance *D* and target width *W*, as stated by Fitts' law [8,17]:

$$MT = a + b \log_2(D/W + 1), \quad (1)$$

with *a* and *b* standing for empirically adjustable coefficients ($b>0$) and the expression $\log_2(D/W + 1)$ defining what Fitts termed the index of difficulty (*ID*).

Equ. 1 suggests two non-exclusive ways of facilitating target acquisition in a GUI, both of which have been recently investigated in HCI research. One is to reduce *D*. If, as soon as the system detects cursor motion, it helps the user by shifting the target object—or, more realistically, the set of objects that are likely to include the target object—toward the approaching cursor, the numerator of Equ. 1 will be reduced. This solution, investigated in [6], suffers the potential drawback that cursor motion perturbs the layout of objects. The other trick is to have the system expand whatever object is being approached by the cursor, thus increasing *W* and hence reducing the *ID*. This solution, which has been shown to facilitate performance even for a surprisingly late expansion [16,19], has been implemented in the Mac OS X Dock.[1].

The above solutions do have a potential for facilitating pointing, but they take it for granted that a cursor is a tool for pixel selection. Below we question this basic assumption.

Input Information Waste in Current GUIs
The present study, as well as a parallel study reported in a companion paper [1], was triggered by the observation that in current GUIs the amount of information received by the system is generally far less than that emitted by the mouse. Suppose a graphical desktop displays 40 icons, each 20x30 pixel (px) large, on a 1600x1200px screen. Selecting one of these icons means selecting 600px within a set of 1,920,000px, and this amounts to sending $\log_2(1{,}920{,}000/600)=11.6$ bits to the system. The system, however, just needs the specification of one icon among the 40 icons, and this discrete event will only

[1] Incidentally, the expansion of the target icon is just visual, with no change in pointing tolerance[19].

convey $\log_2 40=5.3$ bits. So in this example, representative of many real HCI situations, the system will receive hardly half of the information produced by the pointing act: a substantial proportion of the input information is wasted.

In fact, what is ignored by the system is unnecessary information. Consider the case in which the graphical desktop is entirely tiled with objects, leaving no empty spaces. In that special case, we have $\log_2(S_s/S_o)=\log_2 N$, with S_s denoting the surface area of the screen and S_o that of the objects (all the same size), and with N denoting the number of displayed objects. If, however, there are empty spaces, as is the case in many real cases, then the information emitted by the user will necessarily exceed that received by the system, and the larger the proportion of empty space in the display, the more bits wasted. So it is the information delivered by cursor motion through the voids of our graphical displays that the system ignores.

Put differently, the continuous input from the mouse contains a proportion of gratuitous information. As the cursor crosses an empty portion of the display, the system keeps on updating the cursor position to reflect mouse motion. However, this is information just for the user—the system just waiting for some discrete selection. In light of Fitts' law, which quantifies the difficulty of aimed movement along the dimension of information [8,17], one is led to the concern that there is something gratuitous to the difficulty of target acquisition tasks in our current GUIs, which in this sense seem to be sub-optimal.

This gratuitous difficulty component, we believe, reflects the fact that current GUIs require users to phrase their selection in the continuous language of pixels, whereas the system speaks the more parsimonious language of discrete objects.

Detailed analysis in 1D space

Let us call the current mode of cursor pointing *bitmap pointing* (BMP), our GUIs being based on bitmap graphics. Figure 1 shows what happens in current GUIs when the cursor moves to some object to select it. Time elapses along the figure's horizontal axis. For the sake of simplicity we assume pointing to take place on a 1D desktop, and so space can be represented on the vertical axis. Let us suppose that the desktop displays two targets, which occupy intervals [AB] and [CD], and the cursor's target is object [CD], starting from point O.

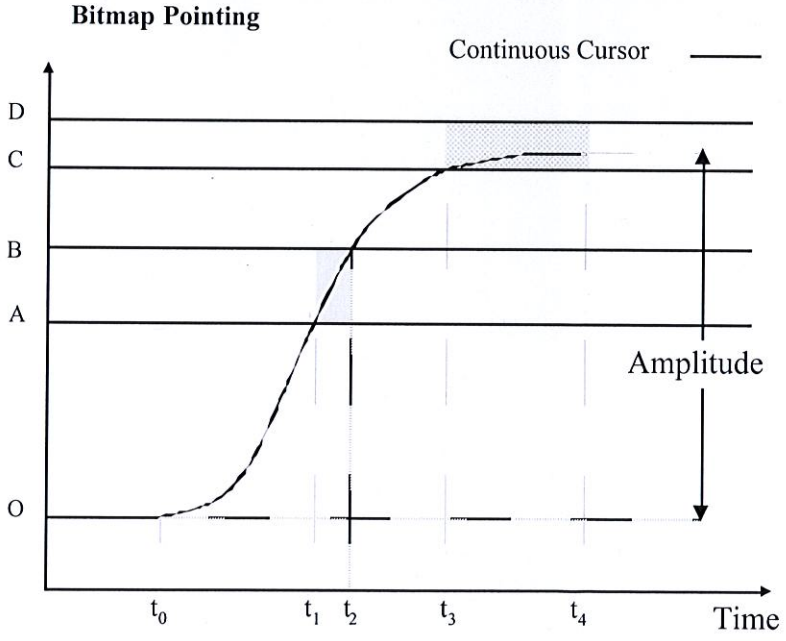

***Figure 1.* Cursor motion in an elemental 1D pointing movement, using the current BMP technology**

At first, suppose the cursor is stationary. At time t_0, when the cursor starts to move upwards, anyone, and hence the software that controls onscreen cursor motion, could easily infer that the user is starting a pointing movement, which must end up in either object [AB] or object [CD]: the only uncertainty left at t_0 is target identity (1 bit).

The system gains no information as the cursor crosses the empty space interval [OA]. At t_1, when the cursor reaches the proximal boundary of the first object, this object gets highlighted (gray filling in the figure), signaling that if a click or an ENTER key press were to happen now, the system would activate that particular object. However, this is information *from* the system—not to the system. From the viewpoint of information input, what is occurring at t_1 was trivially predictable from t_0 because [AB] had to be reached anyway, regardless of whether or not it was the movement's target. The message emitted by the user over the t_0-t_1 interval is essentially redundant.

In fact, the single bit of information the system has been waiting for since t_0 is delivered as a whole at time t_2, when the cursor crosses the upper boundary of target [AB]: at that moment, the probability that the pointing movement aims at target [CD] is about 1. In the chosen example, the cursor is moving rather fast (i.e., the slope of the trajectory is steep) at t_2, so it can be inferred with little risk that the cursor is definitely leaving the object—had the cursor velocity been low, and its acceleration strongly negative (braking) at t_2, obviously one would have been able to guess that the target was interval [AB] and that an overshoot has occurred. At any rate, it is a fact that the rest of cursor motion from t_2 to t_4 will deliver no information whatsoever. Yet the standard technology demands that users carefully finish up their movement. This potentially raises a concern, since the final deceleration of a targeted movement is known to be the part that generally lasts most and involves the greatest cost, in terms of control effort [e.g.,18].

Vector pointing in 1D space

We now introduce what we call *vector pointing* (VP), a new pointing mode designed with reference to the so-called vector graphics mode, which codes graphical displays in terms of discrete objects rather than pixels.

The VP mode involves an extra cursor, which we call the 'timorous' cursor (Tim) because it behaves on the screen as if it were void phobic. Tim's original feature is that it never visits empty regions of graphical space, thanks to an algorithm that uninterruptedly analyzes its kinematics. As soon as the algorithm detects that Tim has left an object, it identifies the current direction of Tim's motion and makes it jump to the first object located in that direction. However, as long as Tim is within an object, it moves normally in parallel to the continuous cursor. Note that in VP mode, the usual system cursor is ineffective for object selection: only Tim's visit to objects can provoke their highlight.

Figure 2 illustrates VP in the case of the simplest 1D pointing task of Figure 1. We suppose the user is initially in BMP mode. At time t_0, with the standard, continuous cursor still at rest, the user switches to the VP mode. Tim would have normally popped up at the same location as the standard cursor, but because it is not within an object, it immediately jumps to the nearest edge of the nearest object, causing this object to highlight. At t_1, the mouse starts to move, causing the two cursors, now in different locations, to move upward in parallel, with Tim 'safely' within the object [AB]. At t_2, Tim reaches point B, the upper boundary of the object, and hence jumps instantaneously to point C, the proximal edge of object [CD], in which it can resume its continuous course parallel to the other cursor. Note that when Tim reaches point D, the mouse is still pushing it upward (as revealed by the trajectory of the other cursor), but there is nothing beyond object [CD] and so Tim stays at D.

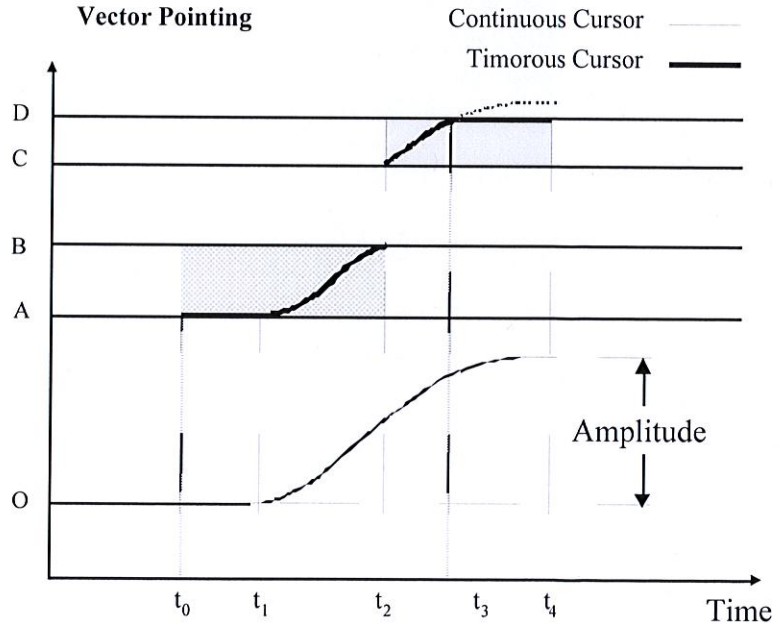

Figure 2. **Tim's jumping course in the VP mode, along with the continuous trajectory followed by the standard cursor**

This rather simple example in 1D space captures the main properties and potentialities of VP, which still hold in 2D or 3D space. One noteworthy property is that neither of the two screen cursors need to be shown to the user. Displaying the standard cursor in VP mode is superfluous because this cursor is ineffective. As for Tim, it is always in some object and that particular object is highlighted, so at every single instant the user receives the appropriate visual feedback, in the form of a highlight that jumps from object to object as the mouse is moved.

The main advantage of the VP mode is that it makes pointing work as though all the objects of the display had been packed together as a compact keyboard. Therefore D, measured in pixel space, is no longer relevant to estimate the ID. The only distance Tim actually needs to cover to reach its target is the sum of the widths of the obstacles it has to cross. Obviously this distance depends on object density in the display, but it can only be shorter than D. Comparing Figures 1 and 2 with special attention to the amplitude covered by the continuous cursor, which in either mode faithfully reflects mouse movements, shows this amplitude saving.

Second, when the target happens to be the last object in the direction of the motion, as is the case for object [CD] in Figure 2, VP makes pointing tolerance virtually infinite in this direction. As illustrated in Figure 2, which shows the user slightly overshooting object [CD], this error has no impact on target acquisition—after t_3, Tim stays at the boundary of the object, which therefore remains highlighted.

Kinematic conditions for the jump

The best algorithm we have found for the control of Tim's jumps in VP mode considers not only Tim's current position, but also its instantaneous velocity and acceleration at the time it is found to leave an object. As soon as Tim is found to occupy a pixel that does not belong to some object, the algorithm evaluates whether two kinematic conditions are met. First, a certain velocity minimum must have been reached: if not, Tim returns to the object's boundary it has just left. Second, Tim's current velocity must not be dropping too sharply: if Tim is decelerating while leaving an object and the absolute value of this negative acceleration exceeds a certain threshold, it returns to the edge of the previous object—so no jump should take place in the case of a mere overshoot during object landing. If the above two kinematic conditions are met, then the algorithm starts to evaluate if a jump is in order.

Vector Pointing in 2D Space

As noted above, the basic features of VP are essentially the same in 1D, 2D and 3D spaces. Below, we focus on the 2D case, keeping in mind the importance of 2D displays in current GUIs.

Identifying the target object in 2D space

In 2D space the direction of Tim's motion must be analyzed in angular terms. Provided the position, velocity and acceleration requirements are met, our algorithm performs a simple linear regression analysis using a small sample of xy coordinates from the recent history of Tim's positions. Once the instantaneous direction of Tim's motion has been identified, the algorithm determines an angular sector centered around Tim's current direction to search for a new object. If the search sector catches at least one object, Tim jumps to the proximal boundary of the nearest object. If the search is unsuccessful, the angular sector is incremented and the search starts again, up to a certain maximum angular sector. If the search fails to find any object in the largest angular sector, then Tim returns to its departure point.

Eye movements in Pointing

One potent argument for the workability of VP is the well-known behavioral fact that in pointing tasks the gaze precedes the hand. Typically, the target begins to be fixated at about the time the cursor starts to move [2,14]. This means that in VP mode, there can be no difficulty for the user to follow the jumping motion of the highlight across the layout of objects (regardless of whether or not Tim is visible). In a normal sequence of events, the highlight will move from the periphery of the visual field to the current point of fixation.

EXPERIMENT 1: RECIPROCAL POINTING IN 1D SPACE

Here we aimed to quantitatively assess VP vs. BMP performance in a highly simplified laboratory task so designed as to allow rigorous experimental control over a minimal number of relevant variables. We investigated three pointing conditions: the standard BMP mode (baseline) and two variants of the VP technique, one with Tim permanently visible (VP_c) and the other with Tim never visible.

Dependent Variables

Movement time (*MT*), defined as the time elapsed between two correctly located clicks, was the main dependent variable. The other dependent variables were the amplitude of on-screen cursor motion, the amplitude of the puck movement on the tablet, and the subjective rating of the difficulty of pointing in the various conditions.

Independent Variables

As explained in Table 1, Exp. 1 was run in two parts. Exp. 1A served to manipulate *Task difficulty* by just varying W at a constant level of movement amplitude, with D set to a constant 800px, to avoid any experimental confound between the effects of difficulty and scale [11]. We used three levels of ID: 3.0, 5.5, and 8.1 bits (i.e., W=114, 18, and 3px, respectively).

Exp. 1B served to manipulate *Task scale* by varying D and W proportionally, so that the ID was a constant 3.46 bits (see Table 1). We used three levels of scale. For the lowest, D and W were set to 10 and 1px, respectively, to create a miniaturized task—for the participant, the problematic feature was obviously W=1px, corresponding to 0.3mm on the tablet. For the next scale level, D and W were set to a comfortable 250 and 25px, corresponding on the tablet to 75 and 7.5mm, respectively. In the third condition, D and W were set to 1230 and 123px to force the puck to cover a rather large 367mm

amplitude on the tablet. This extended scale range was expected to give rise, in the standard BMP condition, to an optimum, V-shaped, relationship between *MT* and scale. Below, we use *D* to identify task scale, it being understood that in Exp. 1B *W* always equaled *D*/10.

Table 1. Task parameters for Exp. 1. Underlined are the values that were expected to be problematic for participants.

	Screen Space (pixels)		Tablet Space (mm)		Task Difficulty	
	D	*W*	*D*	*W*	Ratio *D* /*W*	*ID* (bit)
Exp. 1A	800	114	238.7	34.0	7	3.00
	800	18	238.7	5.4	44	5.51
	800	3	238.7	0.9	267	8.06

	Screen Space (pixels)		Tablet Space (mm)		Task Difficulty	
	D	*W*	*D*	*W*	Ratio *D* /*W*	*ID* (bit)
Exp. 1B	10	1	3.0	0.3	10	3.46
	250	25	74.6	7.5	10	3.46
	1230	123	367.0	36.7	10	3.46

Note that since display-control gain was a constant 3.35px/mm in Exp. 1, the scale manipulation of Exp. 1B simultaneously affected display size on the screen and movement amplitude on the tablet (at least for the standard BMP condition). We wished to evaluate VP in situations that capture the properties of real-world interfaces such as PDAs (or wall interfaces), which miniaturize (magnify) both the display and the movement.

Hypotheses

Three working hypotheses were derived for Exp. 1 from our theoretical analysis of VP.

- Â Overall, pointing performance should be faster in VP mode than in standard BMP mode, owing to the reduction of movement amplitude and—both targets of the reciprocal pointing paradigm being peripheral —the increase of target tolerance.
- Â In VP mode, the slope of Fitts' law should be zero, because the variations of *D* and *W* are handled with immaterial time costs by the system. Since, in contrast, *MT* is known to follow Fitts' law in standard BMP mode, one predicts that the higher the *ID*, the larger the superiority of VP over BMP (Exp. 1A).
- Â In VP mode, *MT* should become scale-insensitive, essentially for the same reason as in the preceding paragraph. Since both interface miniaturization and magnification are known to impair BMP performance [4,9], the larger the deviation from optimal size in either direction, the larger the expected superiority of VP over BMP (Exp. 1B).

Methods

Equipment

The experiment was run on a PC running UNIX with a screen set to a 1600x1200 pixel resolution. The input device was a puck to be moved on an A3 Intuos 12x18' Wacom tablet programmed in the relative mode.

Display and Task

We used Fitts' (1954) classic 1D reciprocal-pointing paradigm [8]. The participants had to click back and forth on two dark-blue-colored vertical strips that extended from the top to the bottom of the screen, appearing on a black background. The pointing movement had to be controlled exclusively along the left-right dimension. The target turned light-red when it was reached by the cursor (the tip of a white arrow, not necessarily displayed), as if highlighted—this feature was necessary for the VP mode to work in the absence of any visible cursor.

Design and Procedure

Exp. 1A involved three pointing modes (BMP, VP with Tim visible, and VP with Tim invisible) and three *ID*s, yielding a nine-cell within-participant experimental design. The same design was recycled in Exp. 1B, with scale simply replacing the *ID* (see Table 1).

Twelve unpaid volunteers participated in the experiment. However, due to errors in handling log files, we were left with a sample of 9 participants (Exp. 1A) and 8 participants (Exp. 1B) to assess the performances. Each participant ran two sets of 27 blocks of trials, the first set for Exp. 1A and the second for Exp. 1B. Each block comprised 10 movements, and so Exp. 1 totaled 540 target acquisitions per participant. Within each set of 27 blocks, order effects were counterbalanced over conditions with Latin squares. The two sets, separated by a comfortable rest, made up a session that lasted approximately one and a half hour.

Results and Discussion

Performance speed

Figure 3 shows mean *MT* as a function of task difficulty for each pointing condition. Note that the independent variable on the *x* axis is the *effective ID*, $ID_e=\log_2(A_e/W_e+1)$, where A_e denotes effective amplitude (the average amplitude of cursor motion) and W_e denotes effective target width (the value of *W* which, given the observed dispersion of clicks, would have yielded 4% errors). Recourse to the ID_e makes it possible to assess Fitts' law satisfactorily even if the correlation between movement-endpoints dispersion and prescribed *W* is less than perfect [17].[2] Note, however, that Fitts' law then becomes a raster plot linking two stochastic variables, which precludes recourse to the analysis of variance.

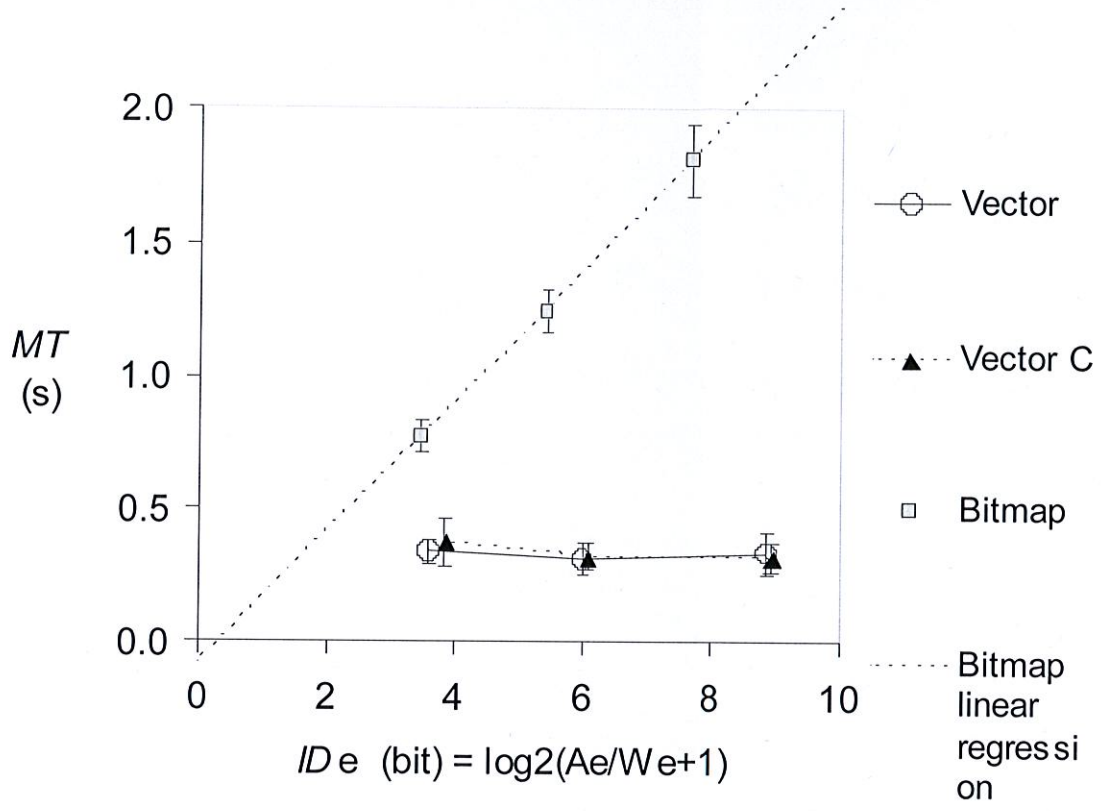

Figure 3. Effect of effective task difficulty on movement time, for each pointing technique (Exp. 1A). Here and in other figures, error bars represent a=.05 confidence limits based on between-participant standard deviation (*SD*)

Figure 3 shows that performance was always much faster for VP (mean *MT*=0.331s) than standard BMP (mean *MT*=1.274s), a 74% time-saving effect which the confidence limits show to be highly reliable. Second, whereas *MT* faithfully obeyed Fitts' law in the BMP condition (for the best-fitting equation, $MT=0.246\ ID_e - 0.081$), it no longer did in the VP condition, where the mean slopes were 0 and -0.01s/bit. Finally, Tim's visibility had no incidence on VP performance.

Figure 4 shows mean *MT* as a function of task scale (i.e., *D*) for each pointing condition, as measured in Exp. 1B. Again, VP yielded considerably faster performance than BMP, the mean time saving amounting to 76% of *MT*.

---

[2] Technically, ID_e was computed on a participant-by-participant and block-by-block basis, before averaging.

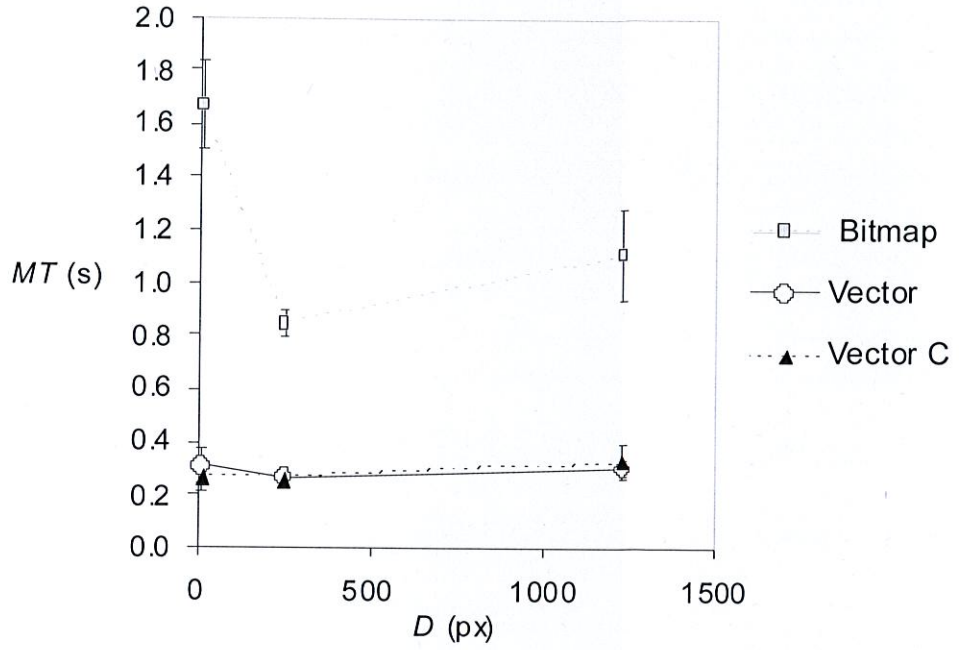

Figure 4. Effect of task scale on movement time, for each pointing technique technique (Exp. 1B)

Most importantly, the predicted interaction between scale and pointing mode was obtained ($F(52.62)=$, $p<.0001$). Whereas performance speed in the BMP mode was strongly dependent on task scale, with the expected V-shaped relationship, in the VP mode it was completely insensitive to the variations of scale. Newman-Keuls pair-wise comparisons revealed that for the BMP condition both task miniaturization and task magnification significantly impaired performance ($p<.05$). In contrast, neither scale pair-wise difference approached significance for any of the two VP variants.

Tablet Amplitudes vs. Screen distances

The data of Exp. 1B, which contrasted very small and very large task scales on the screen, are well suited to illustrate the way in which the participants actually managed in tablet space to cover screen-space distances. Figure 5 shows, for the three pointing technique, how the amplitude actually covered by the puck on the tablet varied with the distance separating the two targets on the screen. Although a slight scale effect was still observed in the VP mode, the amplitude of the puck movement required in VP mode to handle the three levels of D remained very short, in comparison with the BMP mode, revealing a drastic input-device footprint reduction.

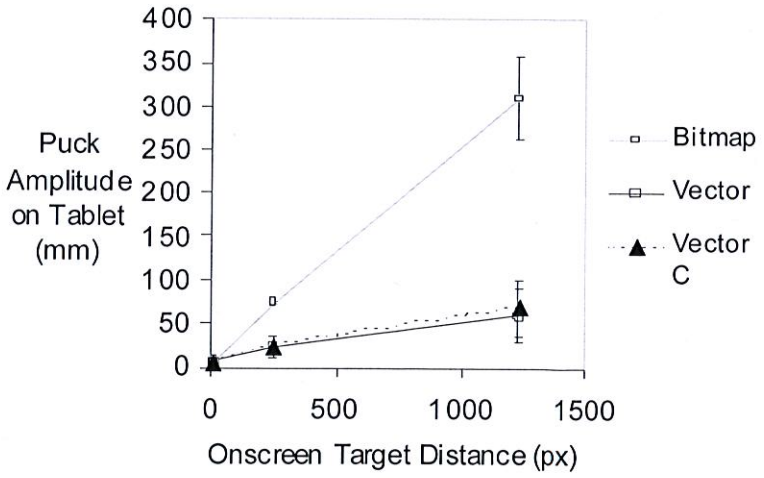

Figure 5. Puck amplitude on the tablet as a function of onscreen target distance, for each pointing technique.

Subjective difficulty scores

The rating, on a 5-point scale, of the difficulty experienced at pointing in Exp. 1 was about the same for the two variants of VP (1.42 and 1.47). But pointing was judged far less difficult with the VP technique (mean±*SD*=1.44±0.74) than the usual BMP technique (3.55±0.44). All but one participant (who equally scored VP and BMP), assigned a reduced score of difficulty to VP (sign test $p<.001$, one-tailed).

EXPERIMENT 2: SERIAL POINTING IN 2D SPACE

This experiment aimed at evaluating VP in 2D space, using a more realistic simulation of a GUI.

Independent variables

Because the results of Exp. 1 had confirmed that the visibility of Tim was superfluous, in Exp. 2 the pointing-mode variable became binary, contrasting BMP vs. the variant of VP that never displays Tim. Pointing mode was crossed with another factor, onscreen target density, with the display exhibiting 6, 12, or 60 objects.

Methods

Equipment and Display

The input device consisted of the same A3-format tablet as in Exp. 1, but the puck was replaced by a standard mouse on which a stylus had been fixed. The mouse served to control onscreen cursor motion in relative mode—with not only position, but also direction being thus defined in the 'egocentric' frame of reference of the hand. The attached stylus allowed the tablet to track the hand's absolute position so as to allow the measurement of footprint.

The screen resolution was set to 1024x768 pixels. The layout of the 6, 12, or 60 objects, all identical (32x32 pixel icons), was randomly drawn by the program on a black background. By default, the displayed objects were dark blue, but the object that was the current pointing target was shown in a more conspicuous light-green color. Highlighting of the currently visited object, whether light-green or dark-blue, was obtained by surrounding this object with a white outline that enlarged the object to 52x52 pixels.

Task

The pointing paradigm was serial, but no longer reciprocal in the sense of a back and forth, and hence stationary, movement. Rather, to track the green target, which every successful click made to jump to another unpredictably determined object, the participant had to produce a series of clicks to follow the path dictated by the program across graphical objects. This task was designed to mimic a rather common situation in GUIs (e.g., reaching and clicking the FILE menu, then the OPEN item, then browsing a tree, etc.). Within each display, 11 successive clicks had to be made, yielding blocks of 10 measurements of *MT*. If a target object was missed by the click, it remained the target, waiting for a correct click, and so a 0% error rate was imposed to participants.

Participants and procedure

Twelve unpaid volunteers participated. Each of the 6 cells of the design (2 modes x 3 object densities) was explored 6 times for each participant, yielding a set of 36 blocks each including 10 movements overall, with possible order effects being counterbalanced with Latin squares. The experiment was completed in a single session that lasted about 40 minutes.

Results and Discussion

Performance speed

Ignoring the *ID* factor, there was no significant main effect of pointing mode on performance (on average, *MT*=1.090s for VP vs. 1.123s for BMP, $F(1,11)<1$). However, as visible in Figure 6, there was a cross-over interaction between mode and object density ($F(2,22)=108.85$, $p<0.0001$), with VP surpassing BMP for displays that included fewer than about 20-30 objects, but doing less well than BMP above that critical density.

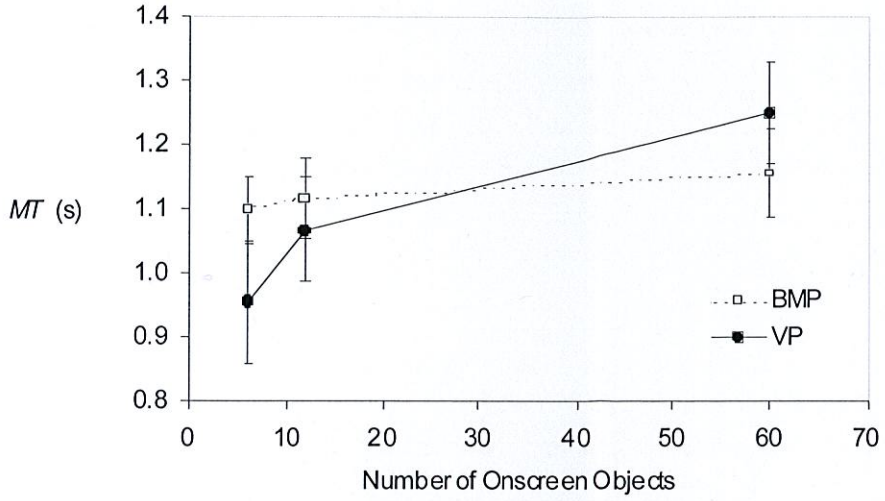

Figure 6. Movement time in Exp. 2 as a function of onscreen object density for the VP and BMP modes

Figure 7 shows the effect of movement difficulty on *MT* (computed from the coefficients of Fitts' law that were estimated in each participant), all object densities collapsed. For levels of ID_e above a modest 3.6 bits (the abscissa of the intersection between the curves), the more difficult the movement, the better VP relative to BMP.

However, the curve intersection that marked the critical level above which VP began to surpass BMP was dependent on object density: its abscissa was 2.7, 3.5, and 7.0 bits for the 6, 12, and 60 object condition, respectively. In sum, in keeping with our expectations, the less dense the display and the more difficult the pointing task, the greater the benefit brought about by the current version of the VP algorithm.

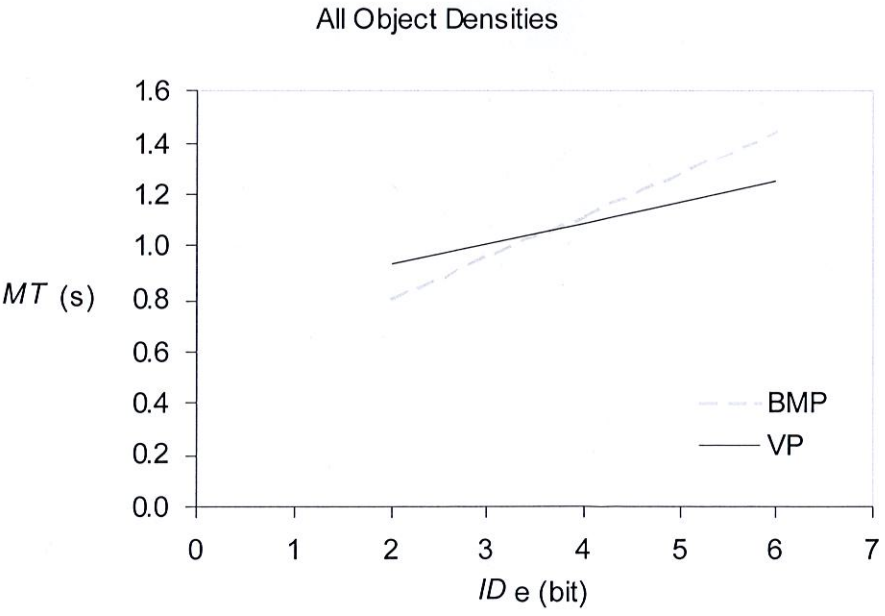

Figure 7. Movement time in Exp. 2 as a function of effective movement difficulty for the VP and BMP modes

Mouse Footprint

We obtained an estimate of the mouse footprint for the BMP and the VP mode by computing on the x and y axis the *SD* of the absolute mouse coordinates delivered by the tablet. Since, with a normal spread of hand positions in both dimensions, a rectangle with its vertical and horizontal sides equal to 2 *SD* should include about 96% x 96% = 92% of all positions,

mouse footprint can be estimated as the length of the rectangle's diagonal, as SQRT(2 SD_x_+ 2 SD_y_), or as the rectangle's surface area, as 2 SD_x * 2 SD_y. We reasoned that even though this method can occasionally lead to overestimations of the foot print in case of a gross repositioning of the mouse, it ought not bias the VP vs. BMP footprint comparison. Indeed, it turned out that using the rectangle's diagonal, mouse footprint was only 23.8% in VP relative to BMP (mean±*SD*=37±0.6mm vs. 155±23mm). In terms of surface area, footprint in VP amounted to a minute 6.2% of BMP (695±234mm_ vs. 11,192±3,191mm_).

Satisfaction scores

Overall, all three object densities mixed, VP and BMP scored about the same in terms of satisfaction concerning ease of use (12.8 vs. 12.3, respectively, $t(11)=0.53$, $p>.1$), but the scores depended on the density, in keeping with the performance speed data. In the low-density condition VP scored 15.8 and BMP 10.8 ($t(11)=4.46$, $p<.001$, one tailed), with all individual judgments, save a tie, favoring VP over BMP. No significant differences were found for the intermediate and higher density conditions.

OVERVIEW

To sum up, input flows in current GUIs suffer a serious waste of information due to the fact that the bitmap language spoken by the mouse does not match the discrete-object language spoken by the system. We have introduced VP, a new mode that dispenses the user of producing redundant cursor moves across the voids of graphical displays. With formal experiments in 1D and 2D space, we have shown that VP indeed facilitates target acquisition, both objectively and subjectively. The data corroborate our hypotheses that the benefit of VP increases as pointing becomes more difficult in either of two senses—a higher *ID* or a more marked departure, in either direction, from optimal interface size—and as object density in graphical space decreases.

IMPLICATIONS FOR HCI DESIGN

The main characteristic of the VP mode is that it considerably reduces the difficulty of target acquisition without constraining the distribution of objects in graphical space (see also [1]). Regardless of the way in which users have spatially arranged their graphical objects, VP is as easy as though all objects were tightly packed together, thus forming a soft keyboard. Since pointing is a basic building block of all our GUIs, we believe the implementation scope of the VP mode is very broad.

That the advantage of the new technique correlates positively with pointing difficulty speaks for VP— obviously, the more difficult a task, the more important it is to carefully optimize the method of carrying it out. The current HCI technology raises two challenges of special relevance to VP. One is the fact that, with more pixels being made available to them for visualization [20], users are likely to face higher levels of *ID* in the future; the other challenge is the current broadening of the palette of interface scales, with efforts toward both miniaturization and magnification of interaction [13]: such trends increase the potential utility of methods like VP.

We have seen that VP leads to a considerable reduction of input device footprint. This feature may be quite useful in practice, if only because users often work on cluttered desks.

The VP technique should not be thought to be a mere mouse counterpart of the classic keyboard direction keys. Two decisive differences can be mentioned. One is that the definition of the jumping direction is far finer with the VP technique than up, down, left, and right, and the other that it does not require a time consuming switch from one input device to another.

In practice, we believe VP should not be thought of as an alternative to BMP, but rather as an *optional mode* made available to the user beside the usual BMP mode. Ideally, users should be allowed to switch opportunistically from BMP to VP, via some simple mouse command, so as to remain able to occasionally cope with pixels (e.g., in a drawing task). So the interface should provide the user with some means to switch back and forth between VP and BMP.

FUTURE WORK

Exp 2 used a newborn version of the VP algorithm for 2D space, which in particular involved no velocity and acceleration criteria to determine the jump. Moreover, little time had been spent optimizing the algorithm's parameters. So, although VP already worked generally better than BMP, certainly there is space to improve its efficiency.

One practical problem that remains to be addressed is the design of an appropriate command to switch back and forth between VP and BMP. Among the possibilities, we may think of a combination of mouse-button presses or a brief oscillation of the cursor.

Among the developments of the VP principle may be contemplated, one is to amend the technique so as to facilitate navigation through pull-down hierarchical menus. In its present design, VP does not yield any benefit in hierarchical menu navigation, because these menus display packed items with no voids. Yet, crossing a horizontal path takes time, a time actually proportional to tunnel length [3]. Thus, horizontal cursor motion within a given item has a cost, but it is entirely

redundant from the system's viewpoint—the probability that the cursor aims at the next level of the hierarchy when it is moving to the right is close to 1. One possible adaptation in the spirit of the VP mode would be to allow the system to make the cursor jump rightward directly to the next level of the hierarchy as soon as a threshold of velocity is reached, a trick reminiscent of marking menus [15].

Another possible development would be to allow, in VP mode, the object highlight to switch not only from object to object within the currently active window, but also from objects in this window to those hosted by surrounding windows. This, for example, could be done by setting different velocity thresholds for jumping within and between the graphical windows.

REFERENCES

1. A. Anonymized (2003). *Semantic pointing: Improving target acquisition with control-display gain adaptation.* Submitted to CHI'04.
2. Abrams, R.A., Meyer, D.A., & Kornblum, S. (1989). Speed and accuracy of saccadic eye movements: Characteristics of impulse variability in the oculomotor system. *Journal of Experimental Psychology : Human Perception and Performance, 15*, 529-543.
3. Accot, J. and Zhai, S. (1997). Beyond Fitts' law: Models for trajectory-based HCI tasks. *Proc. of CHI'97* (pp. 295-302).
4. Accot, J. Zhai, S. (2001). Scale effects in steering law tasks. *Proc. of CHI'01* (pp. 1-8).
5. Albinsson, P.A. & Zhai, S. (2003). High precision touch screen interaction. *Proc. of CHI'03* (pp. 105-112).
6. Baudisch, P. (2002). *Drag and Pop*. http://www.darmstadt.gmd.de/~baudisch/projects/dragandpop/index.html.
7. Card, S.K., English, W.K., and Burr, B.J. (1978). Evaluation of mouse, rate-controlled isometric joystick, step-keys, and text keys for text selection on a CRT. *Ergonomics, 21*, 301-613.
8. Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, 47*, 381-391.
9. Gan, K. C., & Hoffmann, E. R. (1988). Geometrical conditions for ballistic and visually controlled movements. *Ergonomics, 31*, 829-839.
10. Gutwin, C. Improving focus targeting in interactive fisheye views. *Proc. of CHI'02* (pp. 267-274).
11. Guiard (2001). Disentangling relative from absolute movement amplitude in Fitts' law experiments. *Proc. of CHI'01* (pp. 315-316).
12. Guiard, Y. & Ferrand, T. (1998). Effets de gamme et optimum de difficulté spatiale dans une tâche de pointage de Fitts. *Science et Motricité, 34*, 19-25.
13. Hascoët, M. (2003). Throwing models for large displays. *Proc. of HCI'2003, Designing for Society, Vol. 2* (pp. 73-77), British HCI Group.
14. Jacob, R.J.K. (1991). The use of eye movements in human-computer interaction techniques: What you look at is what you get. *ACM Transactions on Information Systems 9(3),* 152-169.
15. Kurtenback, G & Buxton, B. (1994). User learning and performance with marking menus. *Proc. of CHI'94* (pp. 258-264).
16. McGuffin, M. and R. Balakrishnan, R. (2002). Acquisition of Expanding Targets. *Proc. of CHI'02* (pp. 57-64).
17. MacKenzie, I.S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction, 7*, 91-139.
18. Welford, A. T. (1968). *Fundamentals of skills*. London: Methven.
19. Zhai, S., Conversy, S., Beaudouin-Lafon, M., & Guiard Y. (2003). Human on-line response to target expansion. *Proc. of CHI'2003* (pp. 177-184).
20. http://www.research.ibm.com/resources/news/20001110_display.shtml

RAPPORTS INTERNES AU LRI - ANNEE 2003

N°	Nom	Titre	Nbre de pages	Date parution
1364	GOURAUD S D	GENERATION DE TESTS A L'AIDE D'OUTILS COMBINATOIRES : PREMIERS RESULTATS EXPERIMENTAUX	24 PAGES	07/2003
1365	BADIS H AL AGHA K	DISTRIBUTED ALGORITHMS FOR SINGLE AND MULTIPLE-METRIC LINK STATE QoS ROUTING	22 PAGES	07/2003
1366	FILLIATRE J C	WHY : A MULTI-LANGUAGE MULTI-PROVER VERIFICATION TOOL	20 PAGES	09/2003
1367	FILLIATRE J C	A THEORY OF MONADS PARAMETERIZED BY EFFECTS	18 PAGES	09/2003
1368	FILLIATRE J C	HASH CONSING IN AN ML FRAMEWORK	14 PAGES	09/2003
1369	FILLIATRE J C	DESIGN OF A PROOF ASSISTANT : COQ VERSION 7	16 PAGES	09/2003
1370	HERMAN T TIXEUIL S	A DISTRIBUTED TDMA SLOT ASSIGNMENT ALGORITHM FOR WIRELESS SENSOR NETWORKS	32 PAGES	09/2003
1371	RIGAUX P SPYRATOS N	GENERATION AND SYNDICATION OF LEARNING OBJECT METADATA	32 PAGES	10/2003
1372	APPERT C BEAUDOUIN-LAFON M MACKAY W E	CONTEXT MATTERS : EVALUATING INTERACTION TECHNIQUES WITH THE CIS MODEL	14 PAGES	10/2003
1373	BLANCH R GUIARD Y BEAUDOUIN-LAFON M	SEMANTIC POINTING : IMPROVING TARGET ACQUISITION WITH CONTROL-DISPLAY RATIO ADAPTATION	12 PAGES	10/2003
1374	FORGE D KOUIDER M	COVERING OF THE VERTICES OF A GRAPH BY SMALL CYCLES	16 PAGES	10/2003
1375	BIDOIT N CERRITO S THION V	UN PREMIER PAS VERS LA MODELISATION DES DONNEES SEMI-STRUCTUREES PAR LA LOGIQUE MULTI-MODALE HYBRIDE	62 PAGES	10/2003
1376	FRAIGNIAUD P GAVOILLE C PAUL C	ECLECTICISM SHRINKS THE WORLD	20 PAGES	11/2003
1377	KOUIDER M VESTERGAARD P D	GENERALIZED CONNECTED DOMINATION IN GRAPHS	12 PAGES	11/2003
1378	DENISE A GAUDEL M C GOURAUD S D	A GENERIC TOOL FOR STATISTICAL TESTING	22 PAGES	11/2003