The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area

Tovi Grossman, Ravin Balakrishnan
Department of Computer Science
University of Toronto
tovi | ravin @dgp.toronto.edu
www.dgp.toronto.edu

ABSTRACT
We present the bubble cursor – a new target acquisition technique based on area cursors. The bubble cursor improves upon area cursors by dynamically resizing its activation area depending on the proximity of surrounding targets, such that only one target is selectable at any time. We also present two controlled experiments that evaluate bubble cursor performance in 1D and 2D target acquisition tasks, in complex situations with multiple targets of varying layout densities. Results show that the bubble cursor significantly outperforms the point cursor and the object pointing technique [8], and that bubble cursor performance can be accurately modeled and predicted using Fitts' law.

ACM Classification Keywords: H.5.2 [User Interfaces]: Graphical User Interfaces, Theory and methods, Interaction styles.
Keywords: Area cursor, Bubble cursor, target acquisition, Fitts' law.

INTRODUCTION
Pointing to targets is a fundamental task in graphical user interfaces (GUI's). As software gets more complex with an increasing number of selectable user interface elements being crammed into finite sized displays, improvements in pointing performance can have a significant impact on overall software usability. Recognizing this challenge, researchers have proposed several techniques [3-5, 8, 9, 14, 15, 17] that attempt to improve pointing performance by exploiting the fact that virtual pointing can surpass physical pointing by manipulating the control-display parameters.

With few exceptions [8], most of these new techniques have been shown to improve pointing only in situations where targets are fairly sparsely distributed across the display space. When targets are more closely packed together, as is common in many current GUIs, the benefit of these techniques tend to degrade, and can even be detrimental, thus resulting in no advantage in the general case.

CHI 2005, April 2–7, 2005, Portland, Oregon, USA.

In an effort to improve on these previously suggested pointing facilitation techniques we present the bubble cursor, a new technique based upon area cursors [9, 15]. While a standard point cursor has a single point of activation or hotspot, area cursors have larger hotspots defined by the boundary of the cursor (Figure 1a). Problems arise, however, when the area cursor encompasses more than one target, making it difficult to isolate the intended target (Figure 1b). The bubble cursor solves this problem of the area cursor by dynamically updating its size based on the proximity of surrounding targets, such that there is always exactly one target inside the hotspot (Figure 1c,d).

In the following sections, we will review previous efforts at pointing facilitation; discuss the design and implementation of the bubble cursor; evaluate the performance of the bubble cursor in two experiments: first in a simple 1D pointing task and second in a multi-target 2D pointing task with varying target densities; show that the bubble cursor's performance can be modeled accurately by Fitts' law; and conclude by discussing implications for user interface design and future lines of work.

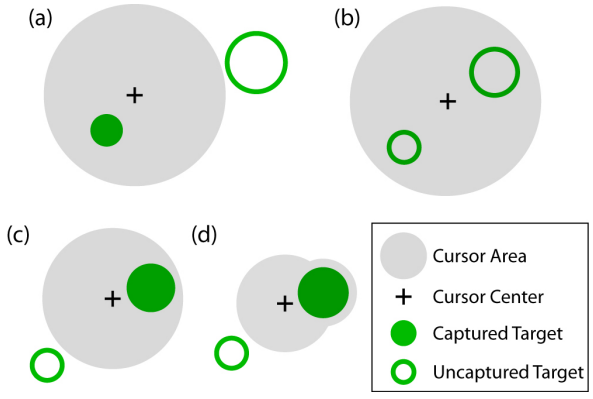

Figure 1. (a) Area cursors ease selection with larger hotspots than point cursors. (b) Isolating the intended target is difficult when the area cursor encompasses multiple possible targets. (c) The bubble cursor solves the problem in (b) by changing its size dynamically such that only the target closest to the cursor centre is selected. (d) The bubble cursor morphs to encompass a target when the basic circular cursor cannot completely do so without intersecting a neighboring target. Note that the same legend is used for Figures 1 to 5.

RELATED WORK

The common approach for studying new selection techniques is to use Fitts' law [6, 11], a highly successful model for predicting movement time in a pointing task. Fitts' Law states that the time (*MT*) to acquire a target with width *W* and distance (or amplitude) *A* from the cursor can be predicted by the equation:

$$MT = a + b \log_2\left(\frac{A}{W} + 1\right)$$

where *a* and *b* are empirically determined constants. The logarithmic term is the index of difficulty (*ID*) of the task. It can be seen from Fitts' Law that if a target's size decreases, or the distance needed to travel to acquire the given target increases, then the time taken to select it increases. Thus target selection can be facilitated by increasing the target width [5, 9, 14, 15, 17], decreasing the amplitude [3, 8], or both [4]. We now review such selection techniques.

Decreasing *A*

An interesting attempt to directly reduce *A* is the drag-and-pop technique developed by Baudisch et al. [3]. In this technique, the system analyzes the directional movements of the cursor and temporarily brings virtual proxies of the potential targets towards the cursor. While this technique was found to be beneficial on a large display for very large *A*, it can be tricky to determine when the user intends to select the remote elements versus items that are in the nearby vicinity. Falsely activated proxies may annoy or even impede the user's selections. As a result, this technique tends to work best in an environment with a relatively sparse layout of targets.

An alternative to bringing the target closer to the cursor is to jump the cursor to the target. Guiard et al. [8] designed a promising interaction technique, called *object pointing*, where the cursor skips across the empty space which can exist between targets, jumping from one selectable target to another. Object pointing was found to be considerably faster than regular pointing in a 1D reciprocal pointing task. However in a 2D environment, it was shown that the degree to which object pointing outperformed regular pointing depended upon the target density. As with the drag-and-pop technique, increased target densities were detrimental to the performance of the object pointing technique.

Increasing W

While the width of the target is typically defined by its size in visual space, the *effective target width* can be defined as the corresponding size of the target in motor space. In standard pointing, the effective target width is typically equal to the width. One of the earliest techniques for increasing the effective width of a target is proposed by Kabbash and Buxton [9] who investigated the use of area cursors. The basic idea is that an area cursor has a larger active region or hotspot for target selection, rather than a single pixel hotspot as in standard cursors. Kabbash and Buxton [9] showed that by setting *W* to be the width of the area cursor, selection of a single pixel target could be accurately modeled using Fitts' law. Thus, very small targets would have a much lower index of difficulty when selected by an area cursor. However, a problem of ambiguity arises when the desktop environment is densely populated with targets, as multiple targets could fall inside the area cursor at one time (Figure 1b).

Worden et al. [15] propose an enhanced area cursor to alleviate this ambiguity, by including a single point hotspot centered within the area cursor, which takes effect when more than one target is within the cursor's bounds. The enhanced area cursor performed identically to regular point cursors when targets were close together, and outperformed point cursors when targets were far apart. This enhanced area cursor is the major inspiration for the bubble cursor which we describe and evaluate in this paper.

In the realm of 3D pointing, Zhai and Buxton [16] implemented a 3D volume cursor and showed that it surpassed a 3D point cursor for target selection. They also showed that semitransparent rendering was a useful aid in discriminating between targets in depth.

Instead of enlarging the cursor's activation zone to increase effective target width, the size of the actual target can be increased. McGuffin and Balakrishnan [14] investigated the potential performance benefits of expanding targets, whose size dynamically increase as the cursors approaches. They found that users were able to take advantage of the larger expanded target width even when expansion occurred after 90% of the distance to the target was traveled. It was also shown that overall performance could be modeled accurately by Fitts' law by setting *W* to the expanded target width. Zhai et al. [17] followed up this study by showing that even when users could not anticipate the expansion, similar performance gains were realized. Cockburn and Firth [5] developed a similar technique based on expanding targets called bubble targets. Instead of increasing the size of the entire target, a bubble would appear around the target as the cursor approached. While expanding targets improve selection times for single isolated targets [14, 17], to date it has not been shown to work well when multiple targets are present in close proximity [13, 14, 17].

Decreasing A and Increasing W

There have been a number of efforts to facilitate pointing by dynamically adjusting the control display (CD) gain. By increasing the gain while approaching a target, and decreasing it while inside a target, the motor space distance and width are decreased and increased respectively. Such adaptive CD gain techniques have been shown to decrease target acquisition times [4, 5, 10, 15]. In a variation called *semantic pointing* Blanch et al. [4] showed that performance could be predicted using Fitts' Law, based on the resulting larger *W* and smaller *A* in motor space. Once again, however, problems arise when multiple targets are present as the intervening targets will slow down the cursor as it travels to its destination target.

BUBBLE CURSOR DESIGN AND IMPLEMENTATION

In designing the bubble cursor, we explicitly sought to address two main shortcomings of area cursors.

First, area cursors in the literature are typically squares. As shown in Figure 2a, situations can occur where a target that is further away from the centre of the cursor is captured preferentially over a closer target, due to the shape of the square. This may make it potentially difficult for a user to plan their movements when heading for a particular target. We solve this problem by making the default shape of our bubble cursor a circle, which ensures that the target closest to the cursor centre is always captured first (Figure 2b)

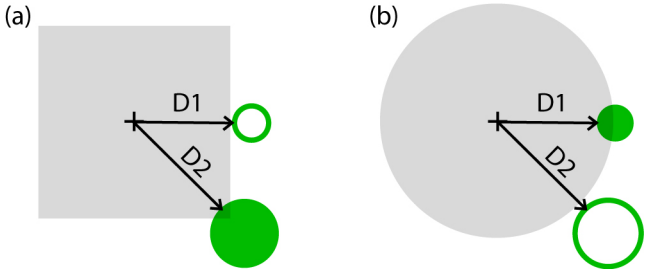

Figure 2. Square vs. circle cursors. D1 < D2. (a) Square shape captures target further away from the cursor centre. (b). Circle shape ensures that the closest target is captured first.

Second, as discussed in the previous section, even with the enhancement suggested by Worden et al. [15], area cursors surpass point cursors only when a single target is inside its hotspot. When the area cursor is too large (Figure 3a), the likelihood of multiple targets being captured by the hotspot increases, and consequently lowers the benefits of an area cursor. In essence, the effective width of the targets regress to their actual values since the area cursor simply behaves as a point cursor in this situation [15]. Conversely, if the size is too small, the full potential of the area cursor will not be realized, as the area cursor may find itself in empty spaces with no targets within its hotspot (Figure 3b). Thus, the overall benefit of an area cursor when used in an interface with multiple targets is highly dependent on its size, and on the layout of the targets. The bubble cursor addresses this problem by dynamically changing its size based on the proximity of the surrounding targets.

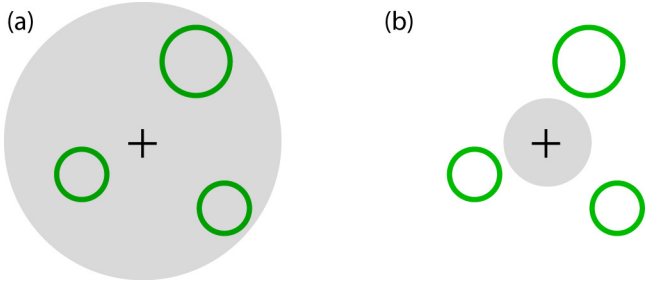

Figure 3. Effect of size on performance of area cursors. (a) A large area cursor tends to encompass multiple targets, thus completely negating its benefits. (b) A small area cursor does not always capture a target, thus reducing its benefits.

By default, the bubble cursor is rendered as a semi transparent circular area cursor. A small crosshair is drawn in the center of the bubble cursor indicating the current location of the standard pointer. A simple algorithm is used to continuously update the radius of the bubble cursor, such that there is always exactly one target within its hotspot. To describe the algorithm in an environment with targets T_1, T_2, ..., T_n we use the following definitions:

Intersecting Distance i ($IntD_i$): The length of the shortest line connecting the center of the bubble cursor and any point on the border of T_i.

Containment Distance i ($ConD_i$): The length of the longest line connecting the center of the bubble cursor and any point on the border of T_i.

A simplified version of the algorithm is as follows:

Set i = index of closest target (T_i) by intersecting distance
Set j = index of second closest target (T_j) by intersecting distance
Set radius of bubble cursor = min($ConD_i$, $IntD_j$)

This algorithm ensures that the bubble cursor will at least intersect the closest target, and possibly completely contain it. Furthermore, it will not intersect the second closest target, and therefore no other target. To prevent the bubble cursor from always touching the second closest target, in practice we slightly modify this algorithm to allow some empty space between the edges of the bubble cursor and the second closest target (Figure 4a). When the bubble only intersects the closest target and does not completely contain it, we morph the cursor by extending a second bubble which quickly expands from the intersection points and envelopes the target (Figure 4b). This acts as a reinforcing visual cue to the user that the target is indeed captured by the cursor.

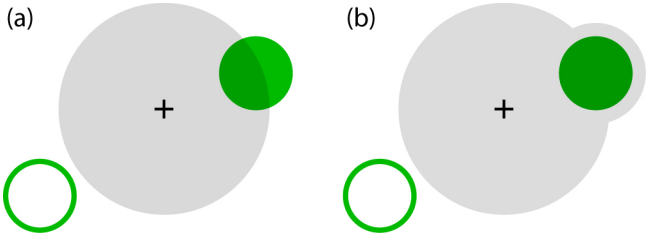

Figure 4. Bubble cursor. (a) Cursor size is dynamically adjusted such that only one target is captured at any time. (b) Cursor is morphed to envelop the target when it is not completely contained by the main bubble.

Effective Width

We now define how the bubble cursor changes the effective width of a target – the size in motor space of a target's activation boundaries. Using the bubble cursor essentially divides up the total space in which all targets reside into regions, such that there is exactly one target inside each region, and that target is the closest target to any point within that region (Figure 5). In mathematical terms, this is referred to as a Voronoi diagram [2]. The activation

boundaries of each target are thus equivalent to its corresponding region in this Voronoi diagram. This in effect increases the target's size in motor space to the maximum possible extent. As can be seen from Figure 5, a target's region need not be rectangular. In this paper, we restrict our experiments to targets with square activation zones, and define the effective width as the square's width. To model selection times in the case where the region is not square, more advanced techniques are required, which we have developed and published elsewhere [7].

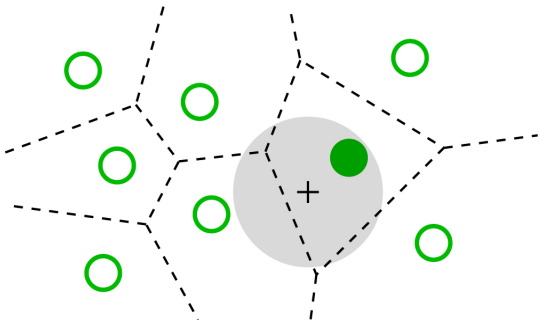

Figure 5. Voronoi diagram defining the regions surrounding each target. The activation boundaries of each target, or *effective width*, are defined by the corresponding region.

EXPERIMENT 1

Given that the bubble cursor enlarges the effective width of every target in a deterministic manner, we should be able to use Fitts' law to model and predict the time it takes to acquire targets using the bubble cursor. From previous work on expanding targets [14, 17], we would expect that Fitts' law would hold in situations like this where the target width dynamically changes during the acquisition process. However, the bubble cursor has a few properties that make it difficult to directly apply the results of the previous work on expanding targets without additional experimental verification. First, a bubble cursor that is constantly changing size may be visually distracting to the user and thus affect the performance in an irregular way that may make it difficult to model with Fitts' law. Second, users may find it difficult to predict how far they need to move the pointer before the bubble cursor captures the desired target, thus affecting the planning and execution of the required motor movements. Thus, it is important to empirically determine if Fitts' law holds for bubble cursors. This is the first goal of Experiment 1.

Even if Fitts' law is shown to model bubble cursor performance accurately, it does not necessarily mean that the bubble cursor provides a significant advantage over point or regular area cursors. By definition, the bubble cursor enlarges the effective width of every target to the maximum possible extent. As such, in the best case scenario, performance should be governed by the effective width rather than the actual width of the target. In other words, selecting a target with actual width *W* and effective width *EW* with a bubble cursor should be equivalent to selecting a target with actual width of *EW* with a regular point cursor. In practice, however, this may not be the case and performance may be somewhere between the limits determined by *W* and *EW*. Thus, the second goal of experiment 1 is to determine if performance is governed by and makes maximal use of the effective width.

To answer these questions in a systematic manner, we begin by studying bubble cursor performance in the simplest possible pointing task: 1D target acquisition.

Apparatus

The experiment was conducted on a 3.2Ghz Pentium4 PC running Windows XP with OpenGL for graphics, and a 20" LCD display at 1600x1200 resolution. Mouse acceleration was set to 0, with a control-display ratio of 1/2. All dimensions are measured in units (1 unit = 0.2 cm).

Participants

Ten volunteers (4 female, 6 male) participated in the experiment. Participants ranged in ages from 18 to 25, were all right-handed, and controlled the input device and consequently the cursor with their right hand.

Procedure

The task was a reciprocal 1D pointing task, which required participants to select two fixed sized targets back and forth in succession. The targets were rendered as solid circles, equidistant from the centre of the display in opposite directions along the horizontal axis. The target to be selected was colored green, and the other grey. Movement of the cursor was restricted along the horizontal axis. When participants correctly selected a target, the targets would swap colors, as an indication that the participant had to now move to and select the other target. Participants had to successfully select the green target before the colors would swap, even if it required multiple clicks. This removes the possibility that participants may try to "race through the experiment by clicking anywhere".

If we only had two targets in the scene, the effective width of each target would be half the size of the display, making selection trivial. To approximate a more realistic target acquisition scenario, distracter targets of varied sizes were pseudo-randomly placed in the scene, in a manner such that the effective width of both goal targets was controlled. Distracters were rendered as grey outlined circles. The bubble cursor was rendered in semitransparent blue, and any goal or distracter targets it covered also appeared blue. In the baseline condition using a point cursor, goal and distracter targets were highlighted blue when the point cursor was over them.

Design

A repeated measures within-participant design was used. The independent variables were cursor type *CT* (Point, Bubble), amplitude *A* (192, 384, 768 units), width *W* (8, 16, 24 units), and effective width *EW* (32, 64, 96 units). A fully crossed design resulted in 54 combinations of *CT*, *A*, *W*, and *EW*.

Each participant performed the experiment in one session lasting approximately one hour. The session was broken up by cursor type, with nine blocks of trials completed for each cursor. In each block participants completed trial sets for each of the 27 combinations of *A, W,* and *EW*, presented in random order. A trial set consisted of 5 clicks (i.e., four reciprocal movements between the two targets).

Before using each cursor, participants were given a single warm-up block to familiarize themselves with the cursor and task. Participants were randomly divided into 2 groups of 5 each, with one group using the point cursor first, and the other group using the bubble cursor first.

Results

Movement Time

Movement time is the main dependent measure, and is defined as the time taken to move to and successfully select the active green target. Repeated measures analysis of variance showed a significant main effect for *CT* ($F_{1,9}$ = 10870, $p < .0001$), *W* ($F_{2,358}$ = 991, $p < .0001$), *EW* ($F_{2,358}$ = 374, $p < .0001$), and *A* ($F_{2,358}$ = 4487, $p < .0001$) on movement time. The overall mean movement times were 1.250 seconds for the point cursor and 0.879 seconds for the bubble cursor. Post hoc multiple means comparison tests showed that for each of the 27 conditions, the bubble cursor was significantly faster, all at the $p < .0001$ level. These results clearly show that the bubble cursor can result in improved performance, even when the effective width is only 33% larger than actual target width (*W*=24, *EW*=32). There were also interaction effects *CT* x *EW* ($F_{5,445}$ = 235, $p < .0001$) and *CT* x *W* ($F_{5,445}$ = 491, $p < .0001$) for movement time. As Figure 6 illustrates, performance of the bubble cursor is dependent on *EW* rather than *W* whereas performance of the point cursor of course depends on *W*.

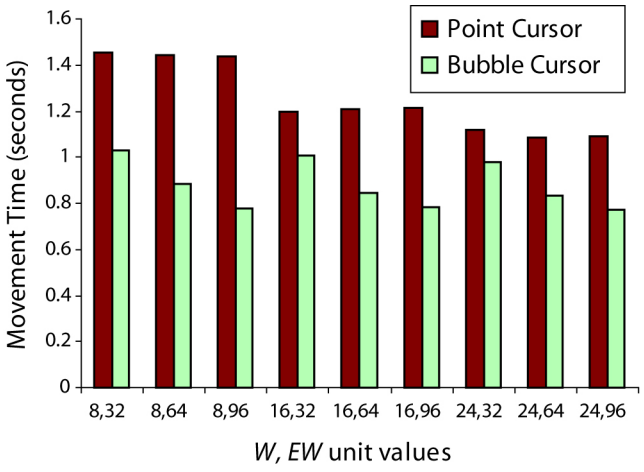

Figure 6. Movement time by *W*, *EW* values for both cursors, averaged over all *A* values.

Figure 7 plots the movement time as a function of the index of difficulty (*ID*). For the point cursor, we define *ID* as $log_2(A/W + 1)$, while for the bubble cursor, $log_2(A/EW + 1)$. Linear regression analysis showed that both conditions fit the Fitts' law equation with r^2 values above 0.95. Moreover, the r^2 value of the combined data is 0.9858. This is an excellent result as it shows that not only can selection using the bubble cursor be accurately modeled using Fitts' Law, but selection is just as fast as if the target had a actual width of *EW* and a point cursor were used. This is similar to the findings in [14, 17] where the time taken to acquire expanding targets could be modeled accurately using the final expansion width, and that movement times could be just as fast as if the target started at the expanded width.

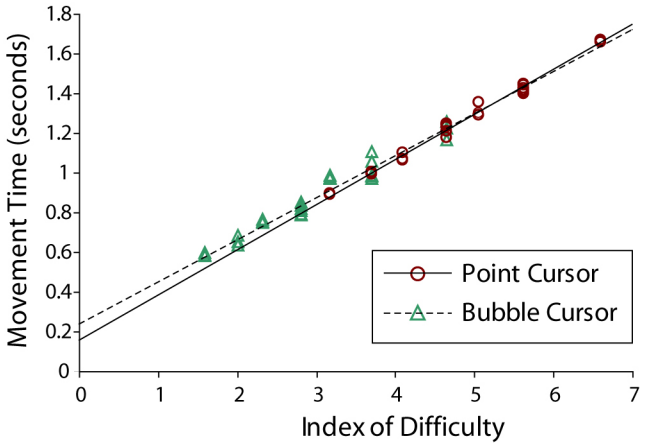

Figure 7. Movement time by *ID* for both cursors

It is interesting to note that even though the regression analysis suggests almost no overhead to using the bubble cursor, there was a significant effect of *W* ($F_{2,178}$ = 16.51, $p < 0.0001$) on movement time, as well as a significant *W* x *EW* ($F_{2,178}$ = 3.76, $p < 0.005$) interaction, in bubble cursor conditions (Figure 8a). Post hoc tests show that increasing *W* does significantly decrease movement time for *EW* = 32 and 64 ($p < 0.001$). However there is no decrease in *MT* for *EW* = 96. A possible explanation for this is that as *EW* approaches *W*, users rely more on the visual target bounds, instead of on the expansion of the bubble.

Another interesting observation is that there was a significant *CT* x block number interaction ($F_{17,153}$ = 6.43, $p < 0.0001$). The different slopes of the curves in Figure 8b illustrates that more learning occurs with the bubble cursor, probably due to users overcoming the slight overheads which were discussed in the previous paragraph.

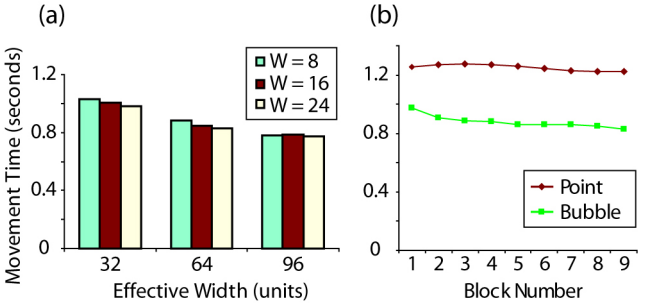

Figure 8. (a) Movement time by *W*, *EW* for the bubble cursor, illustrating differences in effect of *W* depending on *EW*. (b) Movement time by *block number*, illustrating differences in learning effects for the two cursors.

Error Rate

Recall that the experiment design was such that participants had to successfully select each target before proceeding to the next trial, even if it required multiple clicks. As such, all trials are ultimately "successful". However, we classified the cases where targets were not selected on the first click as errors, and analyzed this error data.

There was a main effect for *CT* ($F_{1,9} = 11.68$, $p < .001$), *W* ($F_{2,358} = 3.79$, $p < .05$), *EW* ($F_{2,358} = 19.55$, $p < .0001$), and *A* ($F_{2,358} = 8.84$, $p = .0001$) on error rate. Overall error rates were 3.16% for the point cursor and 2.35% for the bubble cursor – well within the < 4% range one typically finds in standard Fitts' law experiments [11].

EXPERIMENT 2

In experiment 1, we determined that the bubble cursor can indeed reduce target acquisition times, taking maximum advantage of the increased effective target width, and that its performance is accurately modeled using *EW*. The 1D reciprocal pointing task used in that experiment was well suited to answering the initial fundamental questions we asked. However, it may not be representative of the more complex target acquisition tasks one commonly encounters in real 2D GUI interfaces. In this second experiment, we explore the bubble cursor's performance in a more realistic environment with multiple 2D targets in varying layout densities.

By definition, we know that the bubble cursor will have no advantage when acquiring a target which is completely tiled in by its surrounding targets since the effective width of that target would be equivalent to its actual width. In experiment 1, we saw that the bubble cursor could improve selection times even when the effective width was only 33% larger than the actual width. In this experiment, we probe this issue further, and see how much surrounding void space a target would need to take advantage of the bubble cursor. We also look at the impact that intervening distracter targets have on the bubble cursor's performance. In this second experiment, in addition to comparing the bubble cursor to the standard point cursor, we include a third technique – Object Pointing [8] – which as discussed in the related work section is perhaps the most promising existing technique for improved target acquisition. Since the bubble cursor is a direct extension of the static area cursor, there is no reason to expect it to perform *worse* than the area cursor. This was confirmed in pilot studies, and as such we did not include the area cursor in our experimental comparison.

Object Pointing

Object pointing [8] is a pointing technique which essentially ignores void space between targets. When the crosshair cursor leaves a target's "safety zone", which can be larger than its visual boundaries, it analyzes its current direction, velocity, and acceleration. If its velocity and acceleration exceeds a threshold values, it jumps to the boundary of the closest target within an angular slice of its current direction. If no target is found, or its velocity or acceleration is below their respective threshold values, the cursor returns to the boundary of the current target. In our implementation, the safety zone was a 32 unit radius (100 pixels), the velocity threshold was 90 pixels per second, and it searched angular slices of 20, 25, and 30 degrees in sequence. We chose not to use an acceleration threshold as we found it hindered the performance of the cursor when crossing over a large number of intermediate targets. Although it was found in [8] that hiding the crosshair cursor had no effect on movement times, we chose to display the crosshair, as it proved beneficial in our pilot studies.

Apparatus

The apparatus used was the same as in experiment 1.

Participants

Twelve volunteers (3 female, 9 male) participated in this experiment. Participants ranged in age from 18 to 28, were all right-handed, and controlled the input device and consequently the cursor with their right hand.

Procedure

At the start of each trial, a starting solid green goal target was placed in the center of the screen. As in experiment 1, participants had to successfully select the target before continuing to the next goal target. After each successful selection, a new goal target and set of distracter targets were displayed, with the goal target appearing in an unpredictable location. Distracter targets were also positioned so that the effective width of the goal target was controlled. We chose to redraw the entire scene after every successful click, instead of simply jumping the goal target to one of the distracter targets, so that we could carefully control the goal target's effective width and number of intermediate distracter targets along the straight line ideal movement path from the start to goal target. Remaining distracters which were not along the ideal movement path were randomly placed in the scene. Distracters were again rendered as grey outlined circles, and were always the same size as the goal target. The feedback we provided was stronger than in experiment 1, with both goal and distracter targets being filled red when any of the three cursors captured them. The bubble cursor was rendered in a semitransparent grey, which was slightly more visually subtle compared to the blue used in experiment 1, and made it easier to see the red highlighted targets beneath it.

Independent Variables

We define two factors resulting from the environment density which may affect acquisition time in any 2D selection technique (Figure 9). One is the amount of void space immediately surrounding a target (as characterized by *EW* when using the bubble cursor). In this experiment we vary the ratio between the target's effective and actual widths. The other factor is the density of intermediate targets which must be passed over en route to the goal target. This has been shown to affect acquisition times in other selection techniques [8] and could affect the bubble

cursor's performance due to the visual distraction of the bubble cursor growing and shrinking as it passes over these intervening targets. To control the effective width we placed two distracter targets along the direction of movement, one before and one after the goal target, and another two distracter targets perpendicular to the direction of movement, one above and one below the goal target. This created a square activation boundary for the target when using the bubble cursor, simplifying the calculation of *EW*. We controlled the density of the intermediate targets by fixing the number of targets within a 20 degree slice originating at the start target and pointing towards the new goal target positions. With a density value of 0, there were no intermediate targets, except for the one placed before the goal target to control its effective width. With a density of 1, intermediate targets were almost tiled from the start to goal target positions, and then offset in the direction perpendicular to the line of movement by a pseudo-random length, such that each one remained within the 20 degree slice. A density value of 0.5 would produce half as many intermediate targets. The number of remaining distracter targets which were not in the 20 degree slice was controlled to closely match the density of targets within the slice.

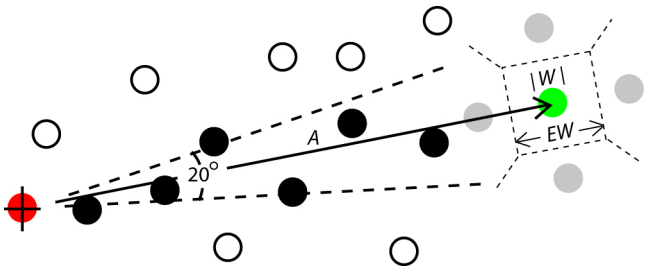

Figure 9. Experimental setup. Red circle is the start target, green circle the goal target. Placement of the grey filled circles is manipulated in the experiment to control the *EW/W* ratio. The black filled circles are placed to control the intermediate target density. The black outline circles are randomly placed distracters. Note that the grey and black colors, and lines, are for illustration only – in the experiment, there were no lines and all the distracter targets were grey outline circles.

Design

A repeated measures within-participant design was used. The independent variables were cursor type *CT* (Point, Bubble, Object), amplitude *A* (256, 512, 768 units), width *W* (8, 16, 32 units), effective width to width ratio *EW/W* (1.33, 2, 3), and distracter density *D* (0, 0.5, 1). A fully crossed design resulted in 243 combinations of *CT*, *A, W,* and *EW/W*. Each participant performed the experiment in one session lasting approximately 90 minutes. The session was broken up by cursor type, with 4 blocks of trials completed for each cursor. In each block participants would complete trial sets for each of the 27 combinations of *W, EW/W*, and *D* presented in random order. A trial set consisted of selecting 9 targets in sequence (not counting the initial click on the start target). The distance between the targets was one of the three *A* conditions, with each of the *A* conditions occurring three times but randomly distributed within each trial set. Before using each cursor, participants were given six warm-up trial sets to familiarize themselves with the cursor and task. Presentation order of the cursors was counterbalanced, resulting in six orderings. Participants were randomly divided into 6 groups of 2, with each group performing one of the six orderings.

Results

Movement Time

Repeated measures analysis of variance showed a significant main effect for *CT* ($F_{2,22} = 7947$, $p < .0001$), *W* ($F_{2,286} = 6659$, $p < .0001$), *EW/W* ($F_{2,286} = 72.92$, $p < .0001$), *A* ($F_{2,286} = 2038$, $p < .0001$) and *D* ($F_{2,286} = 206.3$, $p < .0001$) on movement time. The overall mean movement times were 1.41 seconds for object pointing, 1.08 seconds for the point cursor, and 0.93 seconds for the bubble cursor. The following interaction effects were observed: *CT* x *W*, *CT* x *EW/W*, *CT* x *D*, and *CT* x *A* indicating that the different cursors were affected differently by the manipulated target parameters. Figures 10, 11, and 12 illustrate these effects.

As seen in Figure 10, the bubble cursor is faster than the point cursor, even at the smallest *EW/W* value of 1.33 ($F_{1,11} = 157.6$, $p < .0001$), and is able to take maximal advantage of the increased effective width as the *EW/W* ratio increases. In contrast, it is interesting that object pointing performance degrades as *EW/W* ratio increases ($p < .0001$).

Although the density of distracter targets had a significant main effect on selection times, this effect was remarkably different for the object and bubble cursors as indicated by the significant *CT* x *D* interaction ($F_{8,376} = 203.2$, $p < .0001$). For the bubble cursor, contrary to what may be expected, movement time increased when the density was *decreased* ($p < .0001$) (Figure 11). We believe this is due to the visual distraction of the bubble cursor becoming extremely large when no targets are in its vicinity. This shortcoming could possibly be alleviated by introducing a maximum value for the cursor radius, or altering its visual appearance when it is larger to reduce its intrusiveness. For object pointing, performance was severely degraded as the density of distracter targets increased ($p < .0001$). There are several explanations for the poor showing. First, in [8] the tested target environments contained at most 60 objects. In our experiment, there could be up to 200 targets depending on the distracter target density and target size. Secondly, our experimental design differed from [8] in that we always had a distracter target after the goal target along the expected movement path. This meant overshooting the goal target had a cost associated with it, whereas in [8] the absence of a distracter target after the goal target in some cases effectively made the goal target's width extremely large along one side. It is also possible that users need more time to learn strategies for using the object pointing cursor. Indeed the *block* x *CT* interaction was significant ($F_{11,121} = 5.31$, $p < .0001$), with object pointing showing the most learning (Figure 12).

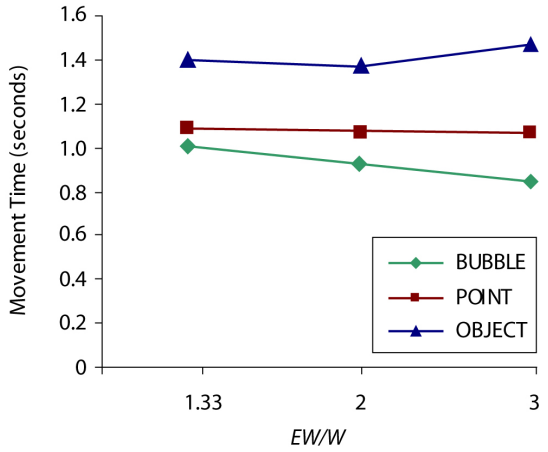

Figure 10. Movement time for each cursor type by *EW/W*

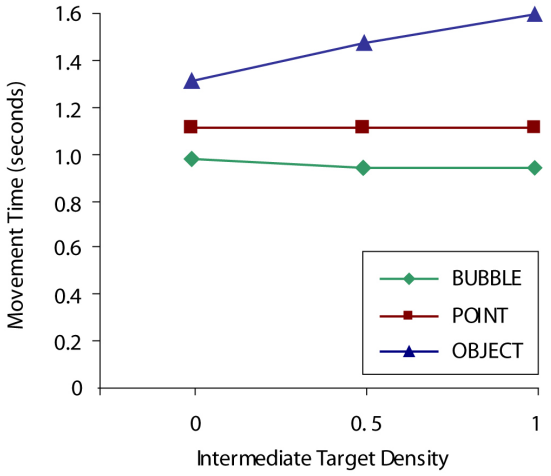

Figure 11. Movement time for each cursor type by distracter target density.

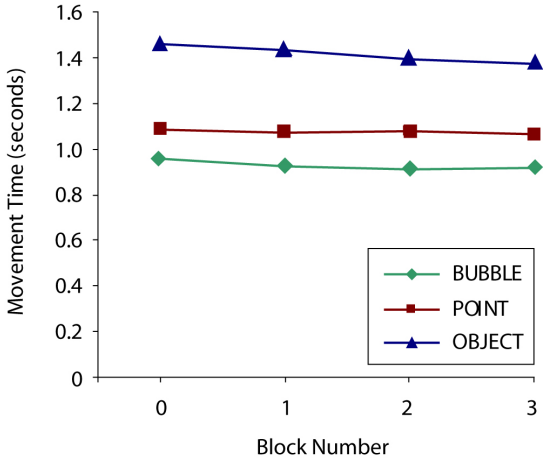

Figure 12. Movement time for each cursor type by block number.

Figure 13 plots the movement time as a function of the index of difficulty (*ID*). As in experiment 1, for the point cursor, we define *ID* as $log_2(A/W + 1)$, while for the bubble cursor, $log_2(A/EW + 1)$. Note that it has been shown that acquisition of square and circular targets can accurately be modeled using their respective widths in the Fitts' Law equation [1, 6, 12]. We omitted the object pointing data from this analysis as the technique cannot be accurately modeled using Fitts' law [8]. Linear regression analysis showed that both conditions fit the Fitts' law equation with r^2 values above 0.96. The combined data r^2 value is 0.966. This confirms that the results seen in the simplified task in experiment 1 continue to hold in the more general 2D target selection task of experiment 2, even with dense target layouts.

Error Rate

We calculated error rate in the same manner as in experiment 1. Error rate was significantly effected by *CT* ($F_{2,22} = 25.64$, $p < .0001$), *W* ($F_{2,286} = 43.85$, $p < .0001$), and *EW/W* ($F_{2,286} = 3.18$, $p = .0416$). Total error rates were 2.98% for the point cursor, 2.31% for the object pointing cursor, and 1.58% for the bubble cursor, again all well within the typical < 4% range seen in target acquisition studies.

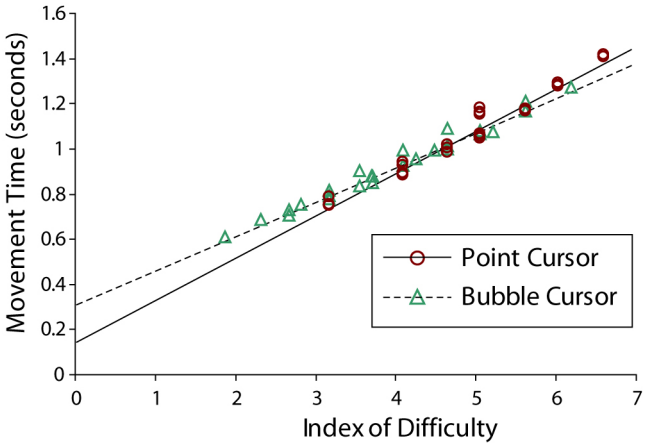

Figure 13. Movement time by *ID*. To calculate the *ID* values, *W* is used for point cursor and *EW* is used for bubble cursor.

DISCUSSION

We have presented the bubble cursor, a new selection technique which enhances area cursors by growing and shrinking to maximize the effective width of targets. In two experiments we have demonstrated that the bubble cursor significantly reduces target acquisition times in both simple and complex multi-target environments.

Unlike many previously described techniques which facilitate selection, the bubble cursor continues to provide advantages in dense target environments. In the second experiment, we varied both the densities of targets immediately surrounding the goal target, and of intermediate targets along the expected movement path. In

all cases we found the bubble cursor to be beneficial, even with relatively little empty space between targets. Because we evaluated the cursor in such varied conditions, and showed that Fitts' law accurately models its performance, we can confidently use the results to predict the performance of the bubble cursor in a general 2D user interface.

While our conditions did vary the essential parameters, we were forced to make minor simplifications in the second experiment. First, each target in the environment was set to the same size, to ease the control of the target density parameters. Second, the distracter targets surrounding the goal target were placed in an organized fashion so that the goal target would have a square activation zone. It may be worthwhile to investigate what, if any, overhead costs would be introduced in conditions which do not make such simplifications.

Our experimental method and analysis warrants some discussion. Firstly, we defined movement time as the time taken for users to make a successful selection, rather than the time until the first click. By forcing subjects to make a successful click to complete a trial, it is in the subject's best interest to perform honestly and minimize errors. This method is also a closer approximation to a realistic GUI pointing task, where users typically do not abort a selection simply because they missed it the first time.

To ensure that this method did not bias our results in any way, we repeated our analysis using the time until the first click, whether it was a successful selection or not. Indeed, the basic patterns, and significant differences which we report, are upheld. For example, in Experiment 1 we report average movement times of 0.879 and 1.250 seconds for the bubble cursor and point cursor respectively. Average times until the first click were 0.866 and 1.232 seconds respectively. Similarly, in Experiment 2, instead of the movement times of 1.08 and 0.93 seconds which we report, times would have been 1.067 and 0.922 seconds.

A related issue was our decision not to use the effective width adjustment for accuracy, as described by MacKenzie [11]. One reason we did not use this adjustment, is so we could analyze movement time data broken up by the specific width conditions in our experiment. Figure 6, for example, would be impossible, or at least misleading, if the widths didn't take on exact values or if it represented data points when a selection was made outside the specified width. Another reason why we chose not to make this adjustment was because the bubble cursor had lower movement times *and* lower error rates, negating the need for any in-depth analysis of the speed-accuracy trade-off.

In both studies we found that not only did the bubble cursor reduce pointing times, but its performance in selecting targets with actual width of W and effective width of EW was essentially equivalent to using a point cursor to select targets with actual width of EW. In other words, targets can remain visually small, but can be selected as though they were as large as can possibly be in motor space. Most importantly, the bubble cursor's performance remains extremely good regardless of the number or density of targets in the environment. In contrast, many of the other selection techniques surveyed in our related work section tend to break down in dense multi-target situations.

Another advantage of the bubble cursor is that it achieves these performance gains without changing the visual characteristics of the underlying interface elements. The size and position of all targets remain unmodified, rather it is the cursor that dynamically changes in size.

We also found the bubble cursor to be superior to object pointing, one of the more promising selection techniques in the literature. However we must be careful before drawing too many conclusions about the relatively poor performance of object pointing in our experiment, as there are a number of parameters which could conceivably be modified to improve this technique. It would be interesting to see if object pointing could benefit from adaptive algorithms which continuously update their own parameters, much like how the bubble cursor radius is dynamically calculated.

It may also be possible to apply the concepts of the bubble cursor to expanding targets. In current implementations, expanding targets have a final predetermined expansion width (such as 200%). This causes the target to overlap or push away neighboring targets within this expansion zone. An alternative strategy would be to expand the target into the region of space as defined by the Voronoi diagram. This would increase its activation zone the same extent which the bubble cursor does.

The positive results from our experiment suggest that the bubble cursor could be a beneficial addition to user interfaces. Because the interface layout is unmodified, individual software programs do not need to be re-written, instead the mouse driver software could be updated to have the bubble cursor available in any program. It would be trivial to incorporate the bubble cursor into user interfaces in which users only need to click on targets, and never in void space. A web browser is a good example of such an interface. Special considerations are required for interfaces in which users can also click on the void space in between targets. In a word processor for example, a user can click anywhere to reposition the cursor location. One solution would be to have some sort of switching mechanism, as suggested in [8] to only use the crosshair positioned in the middle of the bubble. Such a switch could be done in a fluid manner since the crosshair position is already visible to the user. An alternative would be to require users to explicitly select the bubble cursor from a menu or icon. This would be very appropriate for interfaces which already have explicit "selection" modes. In Adobe Illustrator for example, the user must click an arrow icon before selecting drawing objects such as vertices, edges, and control points. This arrow icon could simply be changed to a bubble icon, which would activate the bubble cursor.

FUTURE WORK

Our experiments indicate that the bubble cursor is a very promising selection technique with significant performance advantages over the status-quo. However, there are several directions that can be pursued to extend the current work.

First, while our results could be used to predict the effectiveness of the bubble cursor in any general user interface, it may be interesting to explicitly evaluate it for specific target layouts resembling particular user interfaces.

It may also be interesting to combine some of the prediction algorithms found in [3, 8] into the bubble cursor. Currently, the bubble cursor expands and shrinks in each direction equally, such that it is always a circle. It may be interesting to experiment with some more interesting shapes. One could imagine the cursor expanding in the direction of movement, and shrinking in the direction perpendicular to the movement. Such an elliptical shape could further increase effective target widths by biasing against targets which clearly are not along the direction of movement.

It would also be useful to explore interaction techniques for switching between the discrete selections of the bubble cursor, and the continuous pointing capability of the standard point cursor. As mentioned earlier, the bubble cursor's crosshair would play a critical role in such a switch. It may even be useful to give the crosshair additional functionality when using the bubble cursor, beyond the currently simple function of simply indicating the center of the bubble.

Another interesting extension would be to explore methods for group selections using the bubble cursor. There could be a button or gesture which allows the user to modify the shape or size of the bubble such that it can cover multiple objects for selection when desired.

Lastly, we feel that the bubble cursor could be an effective tool for selection in 3D environments, where the bubble is a spherical volume instead of circular area. Such a cursor would build on the silk cursor of Zhai and Buxton [16], just as the 2D bubble cursor builds on area cursors.

ACKNOWLEDGMENTS

We thank Michael McGuffin for his insights into dynamically sized cursors and directing us to Voronoi diagrams. We also thank John Hancock, members of the DGP Lab, and our experiment participants.

REFERENCES

1. Accot, J. and Zhai, S. (2003). Refining Fitts' law models for bivariate pointing. *ACM CHI Conference on Human Factors in Computing Systems*. p. 193-200.
2. Aurenhammer, F. and Klein, R. (2000). *Voronoi Diagrams. Chapter 5 in*, in *Handbook of computational geometry*, J. Sack and J. Urrutia, Editors. North-Holland: Amsterdam, Netherlands. p. 201-290.
3. Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., and Zierlinger, A. (2003). Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch- and pen-operated systems. *Proceedings of Interact*. p. 57-64.
4. Blanch, R., Guiard, Y., and Beaudouin-Lafon, M. (2004). Semantic pointing: improving target acquisition with control-display ratio adaptation. *ACM CHI Conference on Human Factors in Computing Systems*. p. 519-525.
5. Cockburn, A. and Firth, A. (2003). Improving the acquisition of small targets. *British HCI Conference*. p. 181-196.
6. Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47. p. 381-391.
7. Grossman, T. and Balakrishnan, R. (2005 - in press). A probabilistic approach to modeling 2D pointing. To appear in *ACM Transactions on Computer Human Interaction*.
8. Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. (2004). Object pointing: a complement to bitmap pointing in GUIs. *Graphics Interface*. p. 9-16.
9. Kabbash, P. and Buxton, W. (1995). The "Prince" technique: Fitts' law and selection using area cursors. *ACM CHI Conference on Human Factors in Computing Systems*. p. 273-279.
10. Keyson, D. (1997). Dynamic cursor gain and tactual feedback in the capture of cursor movements. *Ergonomics*, 12. p. 1287-1298.
11. MacKenzie, S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7. p. 91-139.
12. MacKenzie, S. and Buxton, W. (1992). Extending Fitts' law to two-dimensional tasks. *ACM CHI Conference on Human Factors in Computing Systems*. p. 219-226.
13. McGuffin, M. (2002). Fitts' law and expanding targets: an experimental study and applications to user interface design, M.Sc. Thesis, Department of Computer Science, University of Toronto.
14. McGuffin, M. and Balakrishnan, R. (2002). Acquisition of expanding targets. *ACM CHI Conference on Human Factors in Computing Systems*. p. 57-64.
15. Worden, A., Walker, N., Bharat, K., and Hudson, S. (1997). Making computers easier for older adults to use: area cursors and sticky icons. *ACM CHI Conference on Human Factors in Computing Systems*. p. 266-271.
16. Zhai, S., Buxton, W., and Milgram, P. (1994). The "Silk Cursor": Investigating transparency for 3D target acquisition. *ACM CHI Conference on Human Factors in Computing Systems*. p. 459-464.
17. Zhai, S., Conversy, S., Beaudouin-Lafon, M., and Guiard, Y. (2003). Human on-line response to target expansion. *ACM CHI Conference on Human Factors in Computing Systems*. p. 177-184.