PointAssist: Helping Four Year Olds Point with Ease

Juan Pablo Hourcade, Keith B. Perry, Aditya Sharma
Department of Computer Science
The University of Iowa
14 MacLean Hall, Iowa City, IA 52242 USA
hourcade@cs.uiowa.edu

ABSTRACT
Children's difficulty in point-and-click tasks using indirect pointing devices such as the mouse has been documented in several studies. This difficulty is manifested in a lack of control near the target, which often results in children clicking inaccurately. This paper presents and evaluates *PointAssist*, a tool that helps children in pointing tasks by detecting the type of motion that occurs when children have difficulty pointing at a target, and triggering a precision mode that slows the speed of the mouse cursor in those cases. We conducted a study with 30 four year old participants who completed point-and-click tasks with and without *PointAssist*. *PointAssist* provided participants with significant advantages in terms of click accuracy, enabling them to be as accurate as 18 to 22 year olds in a previous study with a very similar setup.

Keywords
Human-computer interaction, children, mouse, pointing tasks, sub-movements, algorithm.

ACM Classification Keywords
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Input devices and strategies; Graphical User Interfaces (GUI)*.

INTRODUCTION
Several studies have provided evidence of children's difficulty in pointing tasks when using computers [3, 6, 7, 8, 9, 10, 11]. Studies that have closely looked at children's pointing show that the culprit is a lack of control near the target [6, 7, 8]. Four and five year olds tend to go back and forth over the target, often missing it when they click. In this paper, we present and evaluate *PointAssist*, a tool that helps children in pointing tasks by detecting the type of motion that occurs when children have difficulty pointing at a target, and triggering a precision mode that slows the speed of the mouse cursor in those cases. *PointAssist* thus works as a scaffold [21], helping children complete pointing tasks when they have difficulty, and going away once children have mastered pointing tasks. In this study, conducted with 30 four year old children, *PointAssist* provided participants with significant advantages in terms of click accuracy, enabling them to be as accurate as 18 to 22 year olds in a previous study with a very similar setup.

IDC '08, June 11-13, 2008 Chicago, IL, USA

RELATED WORK

Strategies to Ease Pointing
A number of strategies have been suggested for helping users point when using indirect pointing devices such as the mouse or the trackball. The simplest strategies involve making targets larger or adjusting control-display ratios to slow down the speed of the cursor [18]. Many researchers have worked on more elaborate solutions, none of them designed with children in mind. Worden et al. [22] slowed the speed of the mouse cursor when users moved slowly over icons. They also studied making the active area of the cursor larger. The bubble cursor expands on this by finding the closest target to the cursor within a predetermined range [5]. Others have proposed making the active area of targets larger than their visual area [2]. There has also been research on using force fields to help users point at some targets and stay away from others [1, 20]. Two different groups have researched predicting which target users want to point at and expanding it based on the direction of motion [14, 23]. All of these solutions have the limitation that the algorithms they use need to know where the targets are, meaning that developers need to program applications in a different way, and existing software would not benefit or would have to be rewritten. In addition, expanding the active area of cursors or targets does not provide advantages when targets are clustered as they often are in toolbars. Force fields have the additional problem of slowing down users as they move from one part of the screen to another if there are targets along the way. Predictions of interest on targets used for target expansion may not work well for children due to children's inaccuracy in the direction of their sub-movements as discussed in [6].

In contrast to the currently available approaches, *PointAssist* has the advantage of being able to run in the background independently from the software children are using, sparing developers the need to modify the way they write software and working automatically with all currently available software and anything written in the future. *PointAssist* is also immune to clusters of targets, targets being in the way, and actually takes advantage of children's

inaccurate sub-movements near the target by using that information to detect pointing difficulty.

Fitts' Law

Most studies similar to this one analyze movement time using Fitts' law, a model that predicts pointing movement time based on target size and distance [12, 13, 19]. The equation that defines Fitts' law is the following:

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

where MT is movement time, A is target amplitude (distance from the starting location to the center of the target), W is the width of the target, and a and b are empirically determined constants. In addition, researchers often make use of two other equations derived from Fitts' law:

$$ID = \log_2 (A/W + 1) \quad (2)$$

$$IP = ID / MT \quad (3)$$

where ID is the index of difficulty, and IP is the index of performance. The index of difficulty expresses the difficulty of the pointing task. The index of performance expresses the quality of the performance of participants pointing under the experimental conditions. In addition, when conducting a Fitts' law analysis through linear regression the fit to the model can be assessed by the value of the correlation coefficient R^2 (the closer it is to 1.0, the better the fit).

IMPLEMENTING *PointAssist*

As people complete pointing tasks, their motion towards a target is composed of a series of sub-movements that continue until the target is acquired [4]. See Figure 1 for an example of a pointing path and its sub-movements. *PointAssist* uses an analysis of these sub-movements based on the data on sub-movements from [6] that identifies when children have difficulty pointing. When difficulty is detected, *PointAssist* triggers a precision mode that slows the speed of the mouse cursor by half. *PointAssist* exits the precision mode when difficulty ceases to be detected.

Figure 1. Example of the path traced by a participant when completing a pointing task in this study. The black dots show the boundaries of sub-movements.

Parsing Sub-Movements

We parsed sub-movements in the same way it was done in [6], which adapted a widely used algorithm [9] (see [6] for a full discussion). We identified the potential beginnings of sub-movements by looking for one or more of the following: a change in direction, a change in acceleration from negative to positive, or a relative minimum in absolute acceleration values while the acceleration is negative. We defined direction by classifying the motion between two mouse events as falling into one of four quadrants[1]: *right* (315° to 45°), *left* (135° to 225°), *up* (45° to 135°) or *down* (225° to 315°). A change of direction was triggered when two consecutive mouse motions fell into different quadrants. To be considered a sub-movement each potential sub-movement had to cover a minimum of 4 pixels, last at least 50 milliseconds, and achieve a speed of at least 0.02 pixels per millisecond. These requirements had to be met before the beginning of another potential sub-movement was identified. To reduce noise that we noticed in the raw data from the mouse, we processed mouse events every 45 milliseconds which eliminated noisy speed and acceleration numbers similar to those reported in [16].

Triggering Precision Mode

We identified difficulty pointing following the heuristics presented in [6], triggering precision mode if we identify consecutive slow, short sub-movements with a top speed below 0.08 pixels per millisecond, and a length below 24 pixels. We implemented precision mode by reducing the mouse speed set by the operating system in half. For this study, it reduced the speed from eight to four (eight corresponds to the fifth tick from the left in Windows XP). We exited precision mode if we detected a sub-movement that was more than 24 pixels long.

RESEARCH GOAL

The goal of this study was to find out if *PointAssist* provides any advantages to four year old children conducting point-and-click tasks with a mouse in terms of accuracy or efficiency and if there are any costs associated with these advantages.

STUDY SETUP

Participants

The participants were 30 four year old children (mean 4.6), fifteen boys and fifteen girls, 27 preferred to use their right hand and three their left hand. On average, they used computers for one hour a week and on average were two years and nine months when they first used a computer. We recruited the children from local child care centers.

Materials

We used a Dell Latitude E1505 laptop (Intel 1.83Ghz Core 2 Duo, 2 GB RAM at 987Mhz) running Windows XP Media Center Edition (SP2). We used a Dell optical mouse (6.6cm wide, 12.1cm long, 3.6cm high) as an input device and a 17" (standard not widescreen) Samsung SyncMaster 740n monitor using a 1024 x 768 pixel resolution. The mouse moved the screen cursor approximately 11.9 pixels for every millimeter of physical mouse motion. The monitor width was 337mm, yielding a control display ratio

[1] 0° pointed straight to the right with angles increasing in value counter-clockwise.

of 0.26 (e.g. for every 0.26 inches the mouse moved, the cursor moved 1 inch). Acceleration was turned off.

We wrote study software that very closely mirrors the game-like software used by Hourcade et al. in [7] and [8]. We did not use the same software because it requires Windows 98 and we did not have computers available with that operating system. The only differences in the user interface of the software have to do with the visual feedback on elapsed time and the colors used. While the software in [7] presented cumulative time, the software used in this study shows the number of tasks completed so far. The study software presents children with pointing tasks that involve moving the mouse cursor from the middle of a green square towards a light blue target circle, and clicking on the light blue target circle. The light blue target circles always appear to the right of the yellow square. We presented targets in only one direction because we were not interested in researching the effects of angles on pointing tasks and because previous research has found this effect to be small [12]. Tasks ended as soon as participants clicked, regardless of whether the click was inside or outside the target. More details on how tasks are presented in the study software can be found in [7].

Procedure

We conducted the study in quiet areas at the participants' child care centers in December of 2007. All the rooms had age appropriate furniture. Before starting to use the study software, a researcher made sure children were comfortable using the mouse. Participants then completed a practice sub-block consisting of six tasks with *PointAssist* turned off. These practice tasks were meant for children to become familiar with the tasks, ask questions, and learn how to interpret the feedback from the study software. A researcher instructed children to click on targets accurately and to do it as quickly as possible.

The study software presented target circles of three different diameters: 16, 32 and 64 pixels. See Table 1 for equivalent sizes in millimeters for size on the screen and mouse displacement. The smallest target size (16 pixels) was the size of buttons in scrollbars in Windows XP. The mid-size targets of 32 pixels were the size of the buttons to close applications in Windows XP. The center of the target circles appeared at one of two distances from the starting position: 128 and 512 pixels. The combination of three target sizes and two distances yielded a total of six distinct tasks.

Participants completed a total of three blocks of tasks after practicing, with each block consisting of two sub-blocks, one completed with *PointAssist* turned on and the other with *PointAssist* turned off. Participants were randomly assigned to starting each block with *PointAssist* turned on or off (half started with *PointAssist* on and half with *PointAssist* off). The study software switched *PointAssist* on and off automatically and the children were not told that they would be testing two modes of pointing. Each sub-block in the study consisted of the six distinct tasks for a total of 36 tasks in the study. The study software presented the six tasks in random order.

Screen (pixels)	Screen (mm)	Mouse Displacement (mm)
16	5.3	1.34
32	10.5	2.69
64	21.1	5.38

Table 1. Size of targets on the screen in pixels and millimeters, and equivalent mouse displacement also in millimeters.

Design

The independent variables were: target size, distance to target, block number, and *PointAssist* status (all within-subjects). The dependent variables were: accuracy, movement time and target reentry. Accuracy for each task was 100 if the participant pressed and released the mouse button while over the target, 0 otherwise. Movement time was measured starting when participants moved the mouse cursor from the middle of the green square and ending when they released the mouse button as part of a click. Target reentry refers to the number of times participants entered the target (not counting the first time).

RESULTS

The study software logged every mouse event into a Microsoft Access database. For the statistical analyses, we exported data to SPSS 15.0.

Path Plots

Before conducting statistical analyses, we examined plots of the paths taken by participants to complete tasks to see if we noticed any patterns or differences between tasks completed with and without *PointAssist*. For tasks completed with *PointAssist*, we changed the path color from black to red when precision mode was turned on and switched it back to black when it was turned off. Figure 2 shows a close-up view of the paths taken by all participants to click on 16 pixel targets at a distance of 512 pixels. Note that the paths were rendered with an alpha value of 0.125 (i.e. it takes eight paths over the same pixel to get a solid color with no transparency). The plots of the paths show more activity around the target when *PointAssist* was turned off. This can be seen in the area immediately outside the target. This suggests that *PointAssist* helped reduce back and forth motion near the target. It is also clear from the paths that precision mode was turned on in the right location, with a high concentration of activity near the center of the target, with a majority of black paths (precision mode off) outside the target.

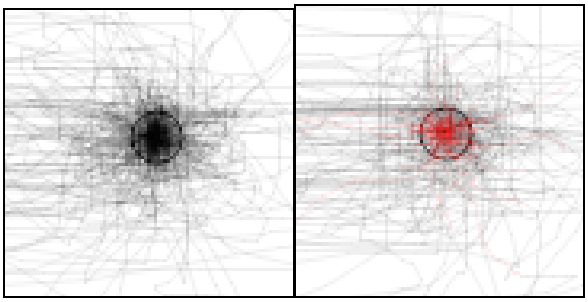

Figure 2. Close-up view of paths taken by all participants to click on 16 pixel targets at a distance of 512 pixels. On the left, tasks completed without *PointAssist*, and on the right tasks completed with *PointAssist* with path colors in red when precision mode was turned on. Paths were rendered with an alpha value of 0.125 (it takes eight paths over the same pixel to get a solid color).

Accuracy

Through repeated measures ANOVAs we found that target size and *PointAssist* had a significant effect on click accuracy while distance to target and block number did not (see details in Table 2).

Independent Variable	Click accuracy F statistic	p value
Target Size	F(2, 28) = 12.215	p < 0.001
Distance to Target	F(1, 29) = 0.738	p = 0.397
Block Number	F(2, 28) = 0.059	p = 0.943
PointAssist	F(1, 29) = 25.495	p < 0.001

Table 2. Result of repeated measures ANOVA looking at significant effects of target size, distance to target, block number and *PointAssist* on click accuracy.

Given these results, we took a closer look at click accuracy rates by target size and *PointAssist* status. Table 3 shows the results, which show that *PointAssist* provided statistically significant advantages for both 16 and 32 pixel targets and almost did so for 64 pixel targets. The smaller the target, the more *PointAssist* helped. Something else to note is that the standard error was smaller for tasks conducted with *PointAssist* turned on, which means that *PointAssist* reduced the variability in performance of the participants.

We also looked more closely at why the click accuracy rates, which involve accurate pressing and releasing of the mouse button, were higher with *PointAssist*. Table 4 shows that while there were differences in press accuracy due to the use of *PointAssist* (and these were statistically significant, $p < 0.001$ for 16 pixel and $p < 0.05$ for 32 pixel targets), *PointAssist* seemed to help even more in improving accuracy when releasing the mouse button. This suggests that slowing down the mouse cursor as participants completed a pointing task helped them keep the cursor inside the target as they clicked.

Click Accuracy by Target Size

Target Size	*PointAssist*	Mean	Std. Error
16	On	91.7	2.6
p < 0.001	Off	79.4	3.9
32	On	95.0	1.8
p < 0.01	Off	87.8	2.6
64	On	98.9	0.8
p = 0.057	Off	96.1	1.3

Table 3. Click accuracy rates by target size and *PointAssist*. The p values under the target sizes were obtained through pairwise comparisons using Bonferroni's correction of the accuracy rates by *PointAssist* for each target size.

Press and Release Accuracy by Target Size

Target Size	*PointAssist*	Press	Release
16	On	95.6	92.2
	Off	86.1	79.4
32	On	97.8	95.0
	Off	93.3	87.8
64	On	98.9	98.9
	Off	98.9	96.1

Table 4. Press and release accuracy rates by target size and *PointAssist*.

We were also interested in learning about how the participants were individually affected by *PointAssist*. Figure 3 shows the accuracy rates for all participants over all target sizes with and without *PointAssist*. The chart shows that more than half of the participants were more accurate with *PointAssist*, and only two were slightly less accurate. Figure 4 shows the same but for 16 pixel targets only. It illustrates the greater difficulty some four year olds have with these small targets, the greater variability in accuracy with this target size, as well as more dramatic gains made by the children when *PointAssist* was turned on.

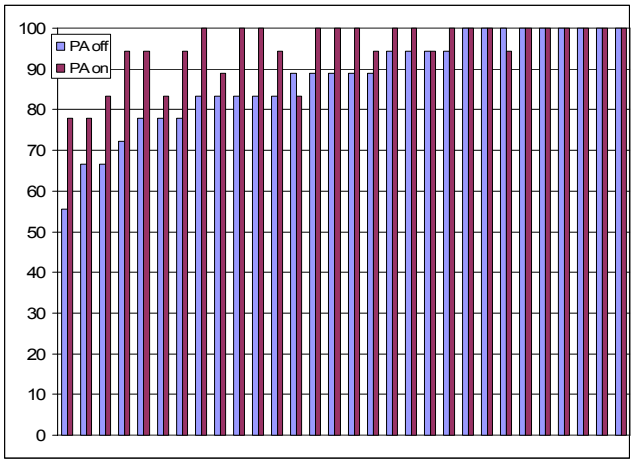

Figure 3. Click accuracy rates for each participant over all target sizes with *PointAssist* turned on and off.

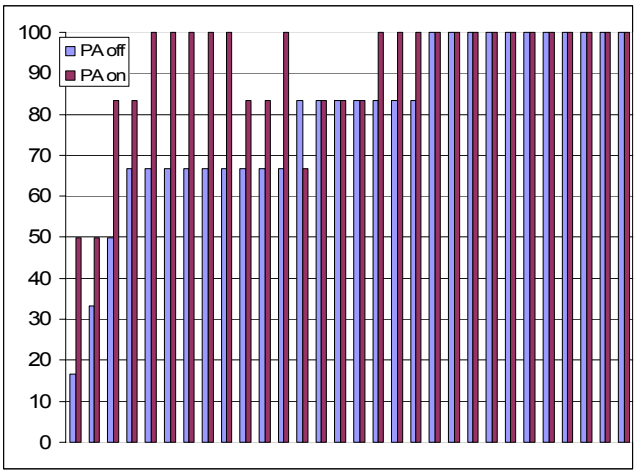

Figure 4. Click accuracy rates for each participant for 16 pixel targets with *PointAssist* turned on and off.

Target Reentry

Through repeated measures ANOVAs we found that target size and *PointAssist* had a statistically significant effect on target reentry while distance to target and block number did not (see details in Table 5). Table 6 takes a closer look at the numbers, showing that the differences were due to higher target reentry rates without *PointAssist* for tasks involving 16 pixel targets.

Independent Variable	Target Reentry F statistic	p value
Target Size	F(2, 28) = 41.746	p < 0.001
Distance to Target	F(1, 29) = 3.031	p = 0.092
Block Number	F(2, 28) = 0.037	p = 0.964
PointAssist	F(1, 29) = 13.835	p < 0.01

Table 5. Result of repeated measures ANOVA looking at significant effects of target size, distance to target, block number and *PointAssist* on target reentry.

Target Reentry by Target Size

Target Size	*PointAssist*	Mean	Std. Error
16	On	1.0	0.09
p < 0.01	Off	1.8	0.26
32	On	0.6	0.09
p = 0.603	Off	0.7	0.13
64	On	0.2	0.05
p = 0.922	Off	0.3	0.05

Table 6. Target reentry by target size and *PointAssist*. The p values under the target sizes were obtained through pairwise comparisons using Bonferroni's correction of the target reentry by *PointAssist* status for each target size.

Movement Time

Through repeated measures ANOVAs we found that target size and distance to target had a significant effect on movement time while block number and *PointAssist* did not (see details in Table 7).

Independent Variable	Movement Time F statistic	p value
Target Size	F(2, 28) = 49.701	p < 0.001
Distance to Target	F(1, 29) = 84.827	p < 0.001
Block Number	F(2, 28) = 0.877	p = 0.422
PointAssist	F(1, 29) = 2.069	p = 0.161

Table 7. Result of repeated measures ANOVA looking at significant effects of target size, distance to target, block number and *PointAssist* on movement time.

In taking a closer look at the data, we noticed that *PointAssist* did seem to have an effect with 16 pixel targets. A statistical analysis confirmed this, with *PointAssist* enabling participants to complete tasks quicker when pointing at 16 pixel targets ($F(1, 29) = 7.46$, $p < 0.05$).

In our Fitts' law analysis of the movement time data we decided against using Mackenzie's effective width and effective index of difficulty because they actually reduced the R^2 values in similar studies with young children [7, 8]. Table 8 shows regression data. The first noticeable difference is that the correlation coefficient (R^2) was much higher when *PointAssist* was on meaning that Fitts' law better modeled the tasks when *PointAssist* was turned on than when it was off. The 0.96 value for R^2 is similar to those reported in studies with adults (e.g. [7, 12]). On the other hand, the R^2 value when *PointAssist* was off, 0.84, is similar to those obtained in previous studies conducted with four year olds (it was 0.86 in [8] and 0.79 in [7]). Hourcade et al. in [7] discuss the reasons for the low correlation coefficients focusing on the difficulty four year olds have in completing tasks once they get close to targets, particularly small ones, moving back and forth without being able to point accurately. This analysis was expanded in [6], which illustrated how four year olds make many inaccurate sub-movements when they get close to the target. Introducing *PointAssist* seems to reduce this problem, helping children when they start moving back and forth over a target by reducing the speed of the mouse cursor.

Regression Data by *PointAssist* status

PointAssist	R^2	a	b
On	0.96	1553	1015
Off	0.84	452.9	1436

Table 8. Fitts' law correlation coefficient and constants *a* and *b* by *PointAssist* status using movement time (in milliseconds) and Fitts' index of difficulty in the linear regression.

In terms of the values of *a* and *b* found in Table 8, it is difficult to reach conclusions due to the relatively low values of R^2 when *PointAssist* was turned off. A statistical analysis showed no statistically significant differences between tasks completed with and without *PointAssist* in Fitts' index of performance.

Gender Differences

We did not set out to investigate gender differences, but since our participants ended up being evenly distributed between boys and girls, we looked for gender differences. We found no statistically significant differences in terms of click accuracy or movement time, although girls were a bit more accurate than boys and boys were a bit quicker than girls. *PointAssist* seemed to benefit both genders equally.

DISCUSSION

Advantages of *PointAssist*

PointAssist provided clear advantages in terms of click accuracy. When compared to tasks conducted without *PointAssist*, it provided higher accuracy rates for pressing and releasing the mouse button inside the target accompanied by lower target reentry rates. It provided these advantages without any apparent drawbacks. As a matter of fact, it even helped participants complete tasks quicker for the most difficult tasks with 16 pixel targets.

In addition, *PointAssist* has the additional advantage of running in the background independent from applications, with no need for developers to write software in a special way in order to use it. By writing the study software and running the study we have also shown that it is possible to implement a precision mode and that the analysis of mouse input data does not slow down system performance (both were cited as concerns in [6]).

Given that *PointAssist* triggers its precision mode when it detects difficulty pointing, users who are more accurate at pointing should trigger precision mode less often and vice-versa. This may not always be true because poor point-and-click accuracy can also be due to users clicking as soon as they get near the target without making an effort to be accurate. In spite of this caveat, we found a statistically significant (Spearman's $\rho = -0.447$, significant at the 0.05 level) participant-level correlation of click accuracy without *PointAssist* and the number of tasks in which precision mode was triggered when *PointAssist* was on. This means that the participants that were more accurate without *PointAssist* triggered *PointAssist's* precision mode less often when *PointAssist* was turned on. This provides evidence of how *PointAssist* can work as a scaffold, and how its precision mode should be triggered less often as children improve their ability to point-and-click.

True Positives

True positives are cases in which precision mode was triggered when participants had difficulty pointing. It is difficult to define true positives in a fair manner because we do not know which targets the participants would have missed had precision mode not been triggered. That said, out of 540 total *PointAssist* tasks, we identified 221 where participants reentered the target. Out of these 221 tasks, 149 (67 percent) triggered precision mode within 64 pixels of the center of the target. Participants missed the target in only three of the remaining 72 tasks, which suggests that *PointAssist* caught most of the cases where participants reentered the target and required assistance. Figure 5 shows examples of true positives. In total, precision mode was triggered within 64 pixels of the center of the target for 297 tasks, or 55 percent of all tasks for which *PointAssist* was turned on. Out of these 297 tasks, 43 percent of them involved 16 pixel targets, 36 percent involved 32 pixel targets and 21 percent involved 64 pixel targets.

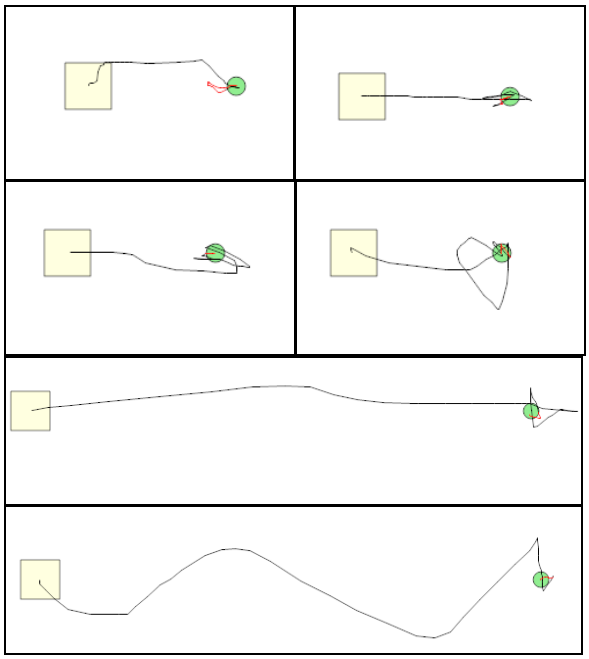

Figure 5. Examples of true positives. Paths are black when precision mode is off and red when it is on.

False Positives

False positives are a concern with any software that tries to interpret what users are doing. In this case, we define false positives as cases where precision mode was triggered away from the target (more than 64 pixels away from its center). We found that false positives occurred in 8.3 percent of tasks. Figure 6 shows that the concentration of paths with precision mode triggered was near the target, with no clear signs of false positives, pointing at their rare occurrence and the ease of pulling out of precision mode when necessary. A closer examination of false positive cases suggests that some cases were related to children moving the mouse back and forth over a small area of the screen to ensure they had control over it, others were related to children readjusting their mouse on their way to the target, and others were due to children beginning to move back and forth near the target, but at a distance greater than 64 pixels (see Figure 7 for examples).

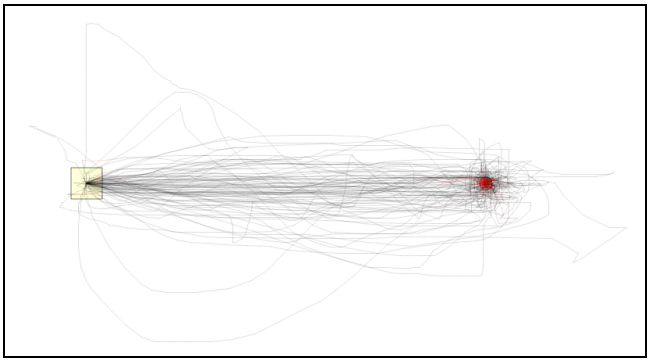

Figure 6. Paths taken by all participants to click on a 16 pixel target a distance of 512 pixels with *PointAssist* turned on. Paths are black when precision mode is off and red when it is on. Paths were rendered with an alpha value of 0.25 (i.e. it takes four paths over the same pixel to eliminate transparency).

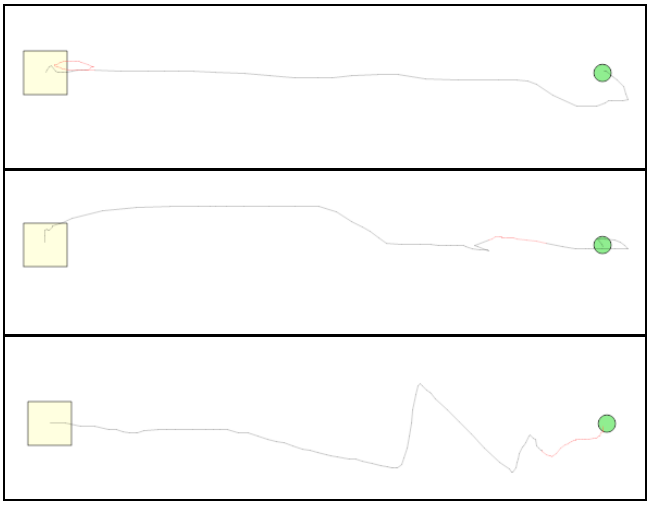

Figure 7. Examples of false positives. Paths are black when precision mode is off and red when it is on.

False Negatives

The encouraging results for *PointAssist* are in great part due to the low number of false negatives. To identify false negatives we looked for cases where there was target reentry (denoting that participants had difficulty pointing), participants were not able to successfully click on the target and precision mode was not turned on within 64 pixels of the target. We found only three such tasks. If we compare this to the total number of tasks for which there was target reentry, it yields a false negative rate of 1.4 percent.

Limitations of the Study

We cannot say that the children in the study are representative of a majority of four year olds in the world given their prior experience with computers. In addition, the study was conducted with particular settings in terms of screen resolution and mouse speed. Research still needs to be conducted on how to adjust *PointAssist* to different settings. We also turned acceleration settings off because we did not want to bring in a confounding variable. In future research we plan to compare *PointAssist* to the use of acceleration settings and to see how *PointAssist* can work with acceleration turned on.

Comparison with Previous Results

Figure 8 compares the click accuracy of participants in this study with that of participants in two previous studies that used the study software that was closely emulated in this study. The accuracy of participants in this study when *PointAssist* was turned off was close to that of five year olds in [7] and four and five year olds in [8] and much higher than that of four year olds in [7]. This confirms the trend seen in [8] of four year olds coming into studies with more experience using input devices, consistent with the increase in the use of computers in preschools in the United States [17]. With *PointAssist* turned on, the accuracy of children in this study was higher than that of children in the previous two studies and similar to that of 18 to 22 year olds in [7].

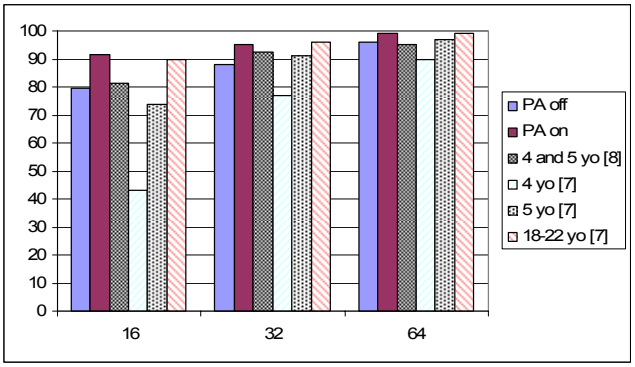

Figure 8. Comparison of click accuracy rates for participants from this study and two previous studies by target size.

FUTURE WORK

As mentioned earlier, we plan to make *PointAssist* work for a variety of screen resolutions and mouse speeds in order to be able to deploy *PointAssist* for real-world use. Once this is completed, we plan to run a longitudinal study by installing *PointAssist* in children's home computers to see how its use in a natural environment changes over time.

In addition, we plan to study the application of *PointAssist* to other populations, such as older adults, who have impaired motor abilities.

CONCLUSION

We have presented empirical evidence that *PointAssist* enables four year old children to significantly increase their accuracy in point-and-click tasks with the mouse reaching accuracy levels similar to those of 18 to 22 year olds. *PointAssist* does so without any apparent drawbacks and many additional advantages: it works as a scaffold, it runs independently of the software with which children interact, it does not require software to be rewritten or written in a specific way, it works both with isolated and clustered targets, it is not affected by targets in the way of a pointing

task, and takes advantage of children's inaccurate mouse movements to detect pointing difficulty.

ACKNOWLEDGEMENTS

We would like to thank the participants and their parents as well as the local child care centers that gave us permission to recruit participants and conduct studies in their facilities: Alice's Rainbow, La Petite Academy and Handicare. *PointAssist* uses the UserActivityHook library by George Mamaladze.

REFERENCES

1. Ahlström, D., Hitz, M. & Leitner, G. (2006). An evaluation of sticky and force enhanced targets in multi target situations. *Proceedings of NordiCHI 2006*. ACM Press: pp. 58-67.
2. Blanch, R., Guiard, Y. & Beaudouin-Lafon, M. (2004). Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation. *CHI Letters, 6*(1), 519-526.
3. Crook, C. (1992). Young children's skill in using a mouse to control a graphical computer interface. *Computers and Education, 19*(3), 199 - 207.
4. Gallahue, D.L. (1989). *Understanding Motor Development: Infants, Children, Adolescents* (2nd ed.). Indianapolis, Indiana: Benchmark Press.
5. Grossman, T. & Balakrishnan, R. (2005). The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. *Proceedings of Human Factors in Computing Systems (CHI 2005)*. ACM Press: 281-290.
6. Hourcade, J.P. (2006). Learning from Preschool Children's Pointing Sub-Movements. *Proceedings of Interaction Design and Children* (IDC 2006). ACM Press: pp. 65-72.
7. Hourcade, J.P., Bederson, B.B., Druin, A. & Guimbretiere, F. (2004). Differences in Pointing Task Performance Between Preschool Children and Adults Using Mice. *ACM Transactions on Computer-Human Interaction* (TOCHI), 11(4), 357-386.
8. Hourcade, J.P., Crowther, M. & Hunt, L. (2007). Does Mouse Size Affect Study and Evaluation Results? A Study Comparing Preschool Children's Performance with Small and Regular-Sized Mice. *Proceedings of Interaction Design and Children 2007*. ACM Press: pp. 109-116.
9. Joiner, R., Messer, D., Light, P. & Littleton, K. (1998). It Is Best to Point for Young Children: A Comparison of Children's Pointing and Dragging. *Computers in Human Behavior, 14*(3), 513-529.
10. Jones, T. (1991). An empirical study of children's use of computer pointing devices. *Journal of Educational Computing Research, 7*(1), 61-76.
11. King, J. & Alloway, N. (1993). Young children's use of microcomputer input devices. *Computers in the Schools, 9*, 39-53.
12. MacKenzie, I. S. (1992). Fitts' Law as a Research and Design Tool in Human-Computer Interaction. Human-Computer Interaction, 7, pp. 91-139.
13. MacKenzie, I.S., Kauppinen, T. & Silfverberg (2001). Accuracy measures for evaluating computer pointing devices. *Proceedings of CHI 2001*. ACM Press: pp. 9-16.
14. McGuffin, M. & Balakrishnan, R. (2002). Acquisition of Expanding Targets. *CHI Letters, 4*(1), 57-64.
15. Meyer, D.E., Abrams, R.A., Kornblum, S., Wright, C.E. & Smith, J.E.K. (1988). Optimality in Human Motor Performance: Ideal Control of Rapid Aimed Movements. *Psychological Review, 95*(3), 340-370.
16. Mithal, A.K. & Douglas, S.A. (1996). Differences in Movement Microstructure of the Mouse and Finger-Controlled Isometric Joystick. *Proceedings of Human Factors in Computing Systems (CHI 96)*. ACM Press: 300-307.
17. National Center for Education Statistics (2005). *Digest of Education Statistics, 2004*. U.S. Department of Education, Institute of Education Sciences, NCES 2006-005.
18. Sanders, M.S. & McCormick, E.J. (1993). *Human Factors in Engineering and Design. Seventh Edition*. New York: McGraw-Hill.
19. Soukoreff, R.W. & MacKenzie, I.S. (2004). Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. *International Journal of Human-Computer Studies, 61,* 751-789.
20. van Mensvoort, K. (2008). *PowerCursor*. Available at powercursor.com.
21. Vygotsky, L.S. (1978). *Mind in Society: The Development of Higher Psychological Processes*. (M. Cole, V. John-Steiner, S. Scribner, E. Souberman, Eds.). Harvard University Press, Cambridge, Massachusetts.
22. Worden, A., Walker, N., Bharat, K. & Hudson, S. (1997). Making Computers Easier for Older Adults to Use: Area Cursors and Sticky Icons. *Proceedings of Human Factors in Computing Systems (CHI 97)*. ACM Press, 266-271.
23. Zhai, S., Conversy, S., Beaudouin-Lafon, M. & Guiard, Y. (2003). Human On-line Response to Target Expansion. *CHI Letters, 5*(1), 177-184.