OhSnap: Helping Users Align Digital Objects on Touch Interfaces

Jennifer Fernquist, Garth Shoemaker, Kellogg S. Booth
Department of Computer Science
University of British Columbia
201-2366 Main Mall, Vancouver, BC Canada V6T 1Z4
{adara, garths, ksbooth}@cs.ubc.ca

ABSTRACT
In this paper we describe...

Author Keywords
Alignment, snapping, multi-touch, touch input

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Graphical user interfaces*

INTRODUCTION

RELATED WORK
Baudisch [2] Lécuyer [4] Bier [3] Ahlström [1] Mandryk [5]

OHSNAP TECHNIQUE
OhSnap is a snapping technique designed specifically for touch interfaces that does not require being toggled on or off and maintains its relative position underneath a user's finger. Objects can be position both at and near lines with OhSnap attraction. In addition, when objects that are snapped to an OhSnap line become unsnapped, they are imperceptibly returned to their original position underneath the user's finger.

When dragging a digital object to a snap line, users experience the object moving normally towards the line and then being stopped once reaching the line. With traditional snapping, users would experience the object being pulled suddenly away from their finger to the line once the object position was within some threshold distance from the line. With OhSnap, if a user snaps an object and then drags their finger beyond the line (intending the snapped object to follow), the object remains snapped for a time and then "catches up" to their finger.

When a user touches a digital object, the position of their finger relative to the object is recorded. If the position of the object becomes the same as a line with OhSnap attraction, that object is flagged as being snapped. As long as the position of the user's finger (relative to its original position in the object) is within a predefined *snap width*, the object will remain snapped to the line. If the user's finger position goes beyond the snap width, in either direction, then the object's position is determined by a linear interpolation function. This technique is visualized in Figure 1.

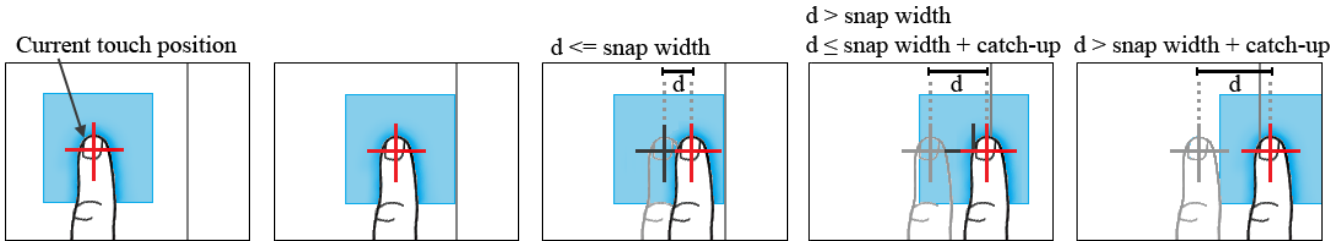
The linear interpolation function calculates the object's position proportional to: the snap line position, the distance between the original and current finger positions, the snap width, and the *catch-up width*. Pseudo code for the algorithm is presented in Listing 2.

Snap width and catch-up width
When the object position is computed via the linear interpolation function, the object appears to move faster than one pixel at a time so that it can catch up with the user's finger. In fact it moves (catch-up width + snap width) / (catch-up width) pixels at a time. This ratio denotes a *super pixel* so that a snapped object moves one super pixel at a time. The object is flagged as snapped as long as the user's finger position is less than or equal to (catch-up width) + (snap width) pixels distant from its original position. Once the finger position is beyond that, then the object is flagged as unsnapped and it has returned back in its original position relative to the user's finger.

The (catch-up width + snap width) / (catch-up width) ratio and, thus, the size of the super pixels, must be carefully chosen. As the ratio approaches 1, the catching up of objects to fingers is slower and less perceptible. When the ratio is equal to 1, the objects never catch up. As the ratio increases, the objects catch up faster. When the ratio approaches infinity (i.e. the catch-up width is 0), then the object will jump to its original position underneath the user's finger when unsnapped, emulating traditional snapping.

The (snap width) + (catch-up width) size should ideally be less than the width of a typical finger (about (ref)). The snap width should be large enough to handle users overshooting a target. Plus, a balance must be struck with the ratio so that the catching-up is imperceptible but occurs quick enough. One issue if the ratio is too small is that certain positions near the snap line are unreachable when the object is snapped. For example, if the ratio (and super pixel size) is 2, then the position 3 pixels away from the snap line is unreachable.

Submitted for review to CHI 2011.



(a) Object is moved to snap line. (b) Object snapped to line. (c) Object snapped, finger moves. (d) Object catching up to finger. (e) Object has caught up.

Figure 1: Example of the OhSnap behaviour. A user moves an object towards a snap line (a) and it becomes snapped when its right edge touches the line (b). As the user continues to drag their finger to the right, the object remains snapped to the line while their finger keeps moving (c). Once their finger has travelled farther than the snap width from the starting point when the object was first snapped, the object becomes unsnapped from the line and it starts to catch up to their finger (d). Once the distance between the current finger position and the original snapped finger position is greater than (snap width + catch-up width), the object returns to its original position underneath the user's finger (e).

Benefits of the OhSnap technique

One benefit of the OhSnap technique over techniques such as Snap-and-Go is that it retains the position of the user's touch point on a digital object relative to that object. This feature is especially useful when users drag objects across lines that have OhSnap attraction when they have no intention of aligning the object to those lines. Although the object would be temporarily snapped to those lines as it got dragged across them, due to the catch-up functionality of OhSnap, the object would catch-up with the user's finger and return to its original relative position underneath it. This feature is especially important for touch tables or other direct-input interfaces.

Another benefit of the OhSnap technique is that it allows users to place digital objects near snap lines as well as aligning them with snap lines without having to toggle the snap capabilities off.

With the OhSnap technique, users of the multi-touch tabletop do not have to toggle the snap capabilities on and off in order to place digital objects close to lines rather than aligned with lines. This is especially important in collaborative environments where toolbars that may hold the snap toggle function may be inaccessible to some users. In addition, snap toggling is a global action that would affect all users and its use may cause interference or disruptions.

IMPLEMENTATION

We implemented OhSnap in C# for all movement types: 1D translation, 2D translation, and rotation. For the translation movements, the position of each edge of an object is checked against the position of all environment lines it is parallel to. Edges in the 2D environment can be snapped independently. For rotation, the snap width is measured by angle rather than pixel.

If a user snaps an object to a line and then lifts their finger from the screen, the object is no longer flagged as being snapped, whether it is aligned with the line or within the (snap width) + (catch up width) region. This is primarily useful if a user wishes to place an object near a snap line but has snapped the object and is having difficulty reaching their destination due to the object moving along super pixels.

```
OhSnap(x, snapLine_x, snapWidth, catchUp)
{
    if (OhSnap_active) // OhSnap
    {
        if (x == snapLine_x)
            isSnapped = true;
        if (isSnapped)
        {
            if (x <= snapLine_x + snapWidth)
                return snapLine_x;
            else if (x > snapLine_x + snapWidth &&
                    x <= snapLine_x + snapWidth + catchUp)
                return linearInterpolation(
                    x, snapLine_x, snapWidth, catchUp);
            else
                isSnapped = false;
        }
        return x;
    }
    else  // Traditional snapping
    {
        if (x >= snapLine_x − snapWidth ||
            x <= snapLine_x + snapWidth)
            return snapLine_x;
        else
            return x;
    }
}
```

Listing 1: Pseudo code fragment for the 1D OhSnap function. In this code x is (current finger position - original finger position + snap line position) and snapLine_x is the position of the snap line.

```
linearInterpolation(x, snapLine_x, snapWidth, catchUp)
{
    return ((x − snapLine_x − snapWidth) / catchUp) ∗
        (snapWidth + catchUp);
}
```

Listing 2: Pseudo code fragment of the linear interpolation function that returns the position of the object if the object is snapped and the finger position is in the 'catch-up' area.

EXPERIMENTS

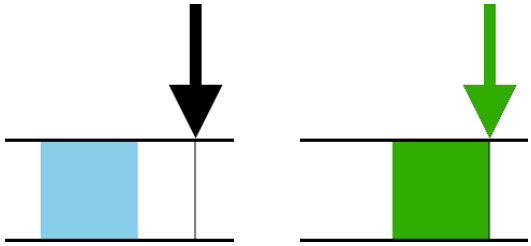
In order to objectively evaluate the performance of the OhSnap technique, we performed two experiments. In the first experiment, the participants carried out three alignment tasks: 1D translation, 2D translation, and rotation. Participants were asked to drag and align a digital rectangle with environment lines. This experiment focused on evaluating the effectiveness of OhSnap versus no snapping and traditional snapping. The second experiment investigated participants' ability to drag a digital block close to a line with OhSnap attraction on.

EXPERIMENT 1: COMPARISON OF SNAPPING TECHNIQUES

The purpose of this experiment was to evaluate the OhSnap technique and compare its performance to traditional snapping as well as no snapping. We investigated performance with three movement types: 1D translation, 2D translation, and rotation.

Task

The participants' task was to move a digital blue square, as fast as possible, so that it was adjacent to one or more target lines. They were instructed to move the square with a single finger on their right hand and align specific edges of the square so that they were adjacent to one or more target lines in the environment. The target lines were indicated with a black arrow. When the square was adjacent to a target line, the respective arrow turned green to indicate the square was aligned, as in Figure 2. When the square was adjacent to all target lines, the square turned green, informing a participant that the square is successfully aligned.



(a) Square not aligned with target line.

(b) Square aligned with target line.

Figure 2: If, during a task, the square edge was not aligned with its target line, the square remained blue and the arrow remained black. When the square edge was aligned with its target line, the square and arrow turned green.

At the start of a trial, a button labelled 'GO' at the bottom of the screen was inactive for one second. When touched, the button disappeared, the square became active (indicated by changing its colour from grey to blue), and the participant could then begin the trial. The timer began when the user first touched the square and it stopped when the appropriate edges of the square were aligned to the target lines and the participant lifted their finger from the screen. Each trial required successful alignment, so that if a participant lifted their finger when the square was not fully aligned, they would have to touch it and move it again to complete the trial.

Participants were asked to conduct three different movement types: 1D translation, 2D translation, and rotation. For the 1D translation task, participants moved the square so that its right edge was aligned with a single target line. The right edge of the square began at a distance of 305 pixels to the left of the target line.

In the 2D translation task, the square was to be aligned so that its top edge and left edge were both aligned to horizontal and vertical target lines, respectively. At the start of a trial, the top left corner of the square was 490 pixels from the corner intersection of the target lines.

In the rotation task, participants rotated a rectangle, anchored at one end, so that the center line protruding from it was parallel to the environment line. The target line was horizontal and the anchored rectangle began vertically at the start of a trial, 90 degrees from the target line.

Each of the tasks were designed so that their components occupied the lower portion of the screen so that participants could sit at the table and comfortably reach the digital objects. A screenshot of the start of each of the tasks (after the 'GO' button has been pressed) is shown in Figure 3.

Interfaces

There were three snapping interface conditions: *no snapping*; *OhSnap*; and *traditional snapping*. The trials appeared identical across conditions, but the attraction type of the target lines differed. In the no snapping condition, the target line had no attraction and the edge of the square had to be placed within +/- 2 pixels of the appropriate target line. Pilot testing revealed that the no snapping tasks were nearly impossible to complete without a small tolerance due to the touch sensing limitations of the SMART Table.

In the OhSnap condition, the target lines had the OhSnap attraction type with a snap width set to 10 pixels and the catch-up width set to 10 pixels. In the traditional snapping condition, the target lines had the traditional attraction type with a snapping threshold of 10 pixels so that if the appropriate square edge was within that threshold of the target line, it would automatically be translated to the target line position.

Experimental Design

The study design was fully counter-balanced, within subjects 3 x 3 (Snapping Technique x Movement Task) with 20 trials for each treatment. For each trial, we recorded the task completion time and the number of times participants lifted their finger from the table.

Participants were given training and the opportunity to practice each movement task (with no snapping) at the start of their session. For the purposes of data analysis, we discarded the first 5 trials of each treatment of 20 trials to reduce learning effects.

Participants

Eighteen volunteers (2 female) between the age of 21 and 40 (μ = 26.7) were recruited from our institution. Six participants had previously used a tabletop display, but only briefly during demos or to play games. Each participant received $10 for their time.

Apparatus

The experiment was conducted on a SMART Table from SMART Technologies. The table has a screen with a size of 57.2cm x 42.9cm, resolution of 1024 x 768 pixels, and 70Hz refresh rate. The application used for this experiment was written in C# with the SMART SDK.

Hypotheses

We had two hypotheses: (1) Participants would perform faster with OhSnap than with no snapping. (2) Participants would be slightly slower with OhSnap than with traditional snapping, but not significantly so.

Results

To compensate for learning effects and human response time, we discarded the first 5 trials per treatment set of 20 and average the trial time over the remaining 15 trials.

Snapping vs. no snapping

We performed a 3 x 3 (Snapping Technique x Movement Task) within-subjects repeated measures analysis of variance. There was a significant main effect of both snapping technique ($F(2,16)=16.078$, $p< .0005$, $\eta^2=.668$) and movement task ($F(2,16)=31.287$, $p< .0005$, $\eta^2=.796$). Pairwise comparisons of snapping techniques revealed additional significant effects between OhSnap and no snapping ($p< .0005$) and traditional snapping and no snapping ($p< .0005$). Thus, hypothesis 1 was supported (Figure 4). There were no significant interaction effects.

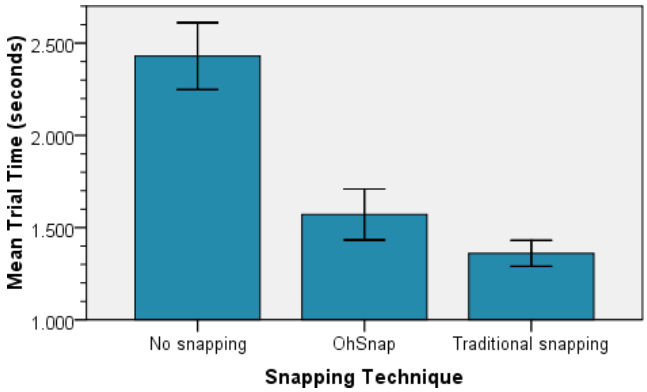


Figure 4: Mean trial time by snapping technique across all movement types.

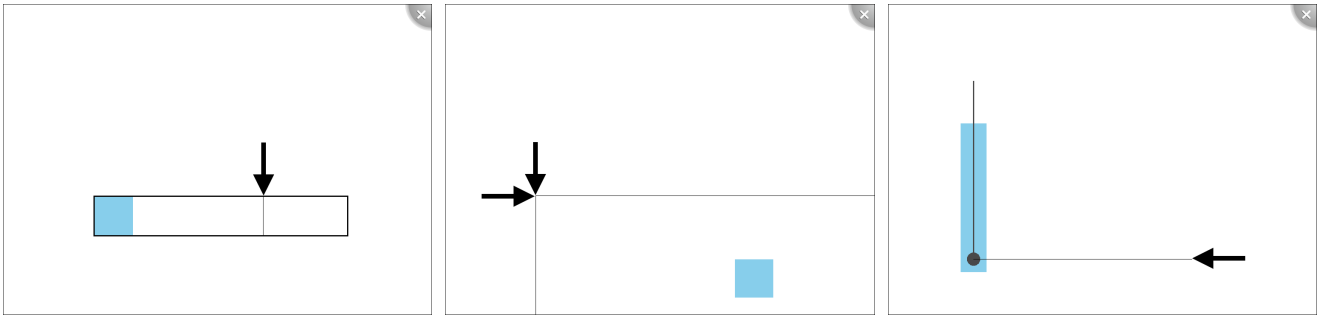
Pairwise comparisons of movement types revealed significant differences between all pairs: 1D and 2D ($p<.005$), 1D and rotation ($p< .0005$), and 2D and rotation ($p< .0005$) (Figure 5).

Multiple repeated-measures analysis of variance of snapping techniques with individual movement types (3 x 1) revealed a significant main effect for rotation ($F(2,16)=17.634$, $p<.0005$, $\eta^2=.688$), 1D translation ($F(2,16)=14.411$, $p<.0005$, $\eta^2=.643$) and 2D translation ($F(2,16)=10.128$, $p<.001$, $\eta^2=.559$). Pairwise comparisons showed that there were significant differences between no snapping and OhSnap, no snapping and traditional snapping for all movement types with $p<.001$. Mean trial times for all snapping techniques across all and individual movement types is shown in Table 1.

OhSnap performed 167% faster than no snapping overall, 145% faster for 1D, 211% faster for 2D, and 132% faster for rotation.

OhSnap vs. traditional snapping

We performed a 2 x 3 (Snapping Technique x Movement Task) within-subjects repeated-measures analysis of variance. We did not find a significant difference between OhSnap and traditional snapping ($F(1,17)=2.522$, $p< .13$), thus hypothesis 2 was supported.



(a) 1D translation task (b) 2D translation task (c) Rotation task

Figure 3: Screenshots of each of the three tasks at the start of a trial in Experiment 1.

Mean Trial Time (seconds)				
Snapping Technique	**Movement Type**			
	All	**1D**	**2D**	**Rot.**
No snapping	2.507	2.301	3.517	1.703
OhSnap	1.513	1.582	1.667	1.287
Traditional snapping	1.387	1.362	1.651	1.147

Table 1: Mean trial time for each of the snapping techniques across all movement types.

Pairwise comparisons of repeated-measures analysis of snapping techniques with specific movement types (3 x 1) revealed no significant time differences between OhSnap and traditional snapping for any of 1D translation, 2D translation, or rotation.

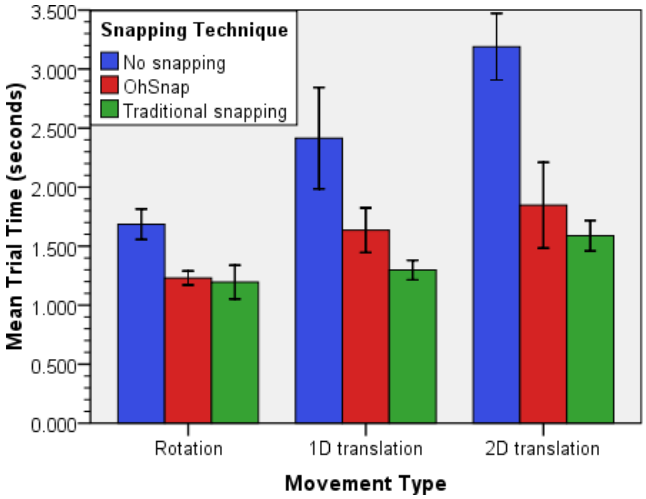


Figure 5: Mean trial time by movement type grouped by snapping technique.

Number of touch-ups

We also conducted 3 x 3 (Snapping Technique x Movement Task) within-subjects repeated-measures analysis of variance for the average number of touch ups in a set of trials. There was a significant main effect of both snapping technique ($F(2,16)=14.838$, $p<.0005$, $\eta^2=.65$) and movement task ($F(2,16)=26.076$, $p<.0005$, $\eta^2=.765$). Pairwise comparisons showed that participants lifted their finger up significantly more in no snapping ($\mu=1.438$) compared to both OhSnap ($\mu=1.19$) and traditional snapping ($\mu=1.185$) with $p<.0005$. There was no significant difference in the number of touch ups between OhSnap and traditional snapping. There were no significant interaction effects.

Questionnaire data

(TODO)

EXPERIMENT 2: VARIATION OF OHSNAP PARAMETERS FOR SNAP LINE PROXIMITY

This experiment was designed to investigate the variation of OhSnap parameters and their effect on user performance for aligning digital objects in close proximity to a line with OhSnap attraction. We also sought to compare the OhSnap attraction variations with no snapping for the proximity tasks.

The apparatus and the participants were the same as in Experiment 1.

Task

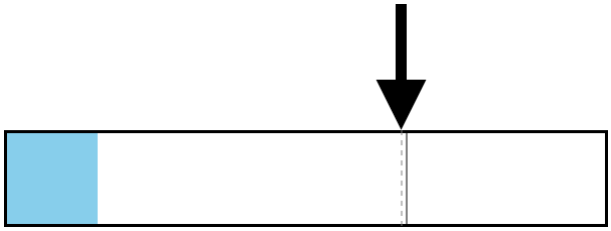
The participants' task was to translate a square so that it was close to a line with OhSnap attraction. The visuals for this task were nearly identical to the 1D task in Experiment 1. A dotted target line, with an arrow pointing down at it, was placed just before or just after the OhSnap line. Participants were asked to drag the square so that its right edge was aligned with the dotted line. As in Experiment 1, once the square was adjacent to the target line both the arrow and the square turned green to indicate a successful alignment.

A screenshot of the start of each of the tasks (after the 'GO' button has been pressed) is shown in Figure 6.

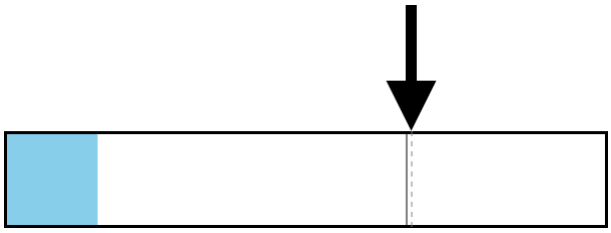
Interfaces

There were four sets of snapping parameters: *no snapping*, *1:3 ratio with 5 pixel snap width*, *1:2 ratio with 10 pixel snap width*, and *1:3 ratio with 10 pixel snap width*.

Experimental Design



(a) Target line before the snapping line.



(b) Target line after the snapping line.

Figure 6: Screenshots of each of the two tasks at the start of a trial in Experiment 2.

The study design was partially counter-balanced, within subjects 4 x 2 (Snapping Parameter Set x Proximity Task) with 20 trials for each treatment. One movement asked participants to align right edge of the square to the dashed target line 5 pixels before the line with OhSnap attraction, and the other task asked participants to align the right square edge to the dashed line 5 pixels after the attraction line. For each trial, we recorded the task completion time and the number of times participants lifted their finger from the table.

As in Experiment 1, participants were given training and the opportunity to practice each movement task (with no snapping) at the start of their session. For the purposes of data analysis, we discarded the first 5 trials of each treatment of 20 trials to reduce learning effects.

Hypotheses

We had one hypothesis: (1) Participants would perform fastest with the largest catch-up size to snap width ratio and the largest snap width. When the ratio is large (i.e. the catch-up size is 3 times larger than the snap width) the size of the super pixels is decreased thus it is most likely easier to perform fine-grained movement of a digital object that is snapped.

Results

We conducted a 4 x 2 (OhSnap Parameter Set x Proximity Task) repeated measures analysis of variance. There was a significant main effect of both OhSnap parameter set and proximity task ($F(3,15)=22.797$, $p<.0005$, $\eta^2=.82$). Pairwise comparisons revealed that the no snapping condition was significantly faster than all parameter sets ($p<.002$), but participants did not complete trials significantly faster with any OhSnap parameter set compared to the others (Figure 8. There were significant interaction effects ($F(3,15)=3.884$, $p<.031$, $\eta^2=.437$) though the effect size was small. Mean trial times for all OhSnap parameter sets and proximity types are presented in Table 2. Mean trial times for all parameter sets are presented in Figure 7.

Surprisingly, the proximity task that asked participants to position the digital square just before the snap line took significantly longer than the task positioning the square just after the line. We anticipated that since the objects could avoid being snapped when placing them before the snap line, this proximity task would be less difficult and faster as a result. Squares positioned after the snap line must be snapped.

It is perhaps the case that participants knew the snapping would occur in the 'after' task and learned how to adjust their movements to accommodate it. Conversely, in the 'before' task, participants may have worked slower so that they did not snap to the line. In addition, participants may have occasionally overshot the target position, though not always, resulting in inconsistent behaviour preventing them from mastering the task.

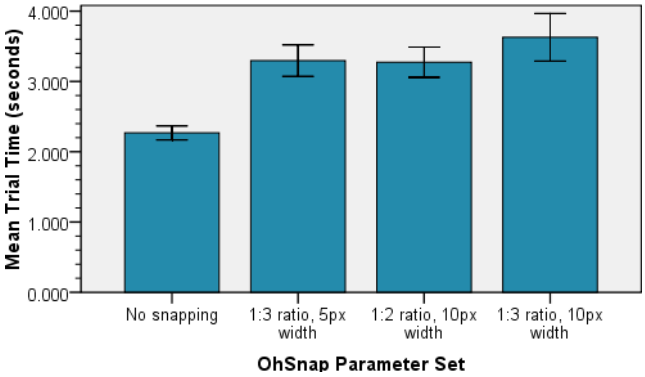


Figure 7: Mean trial time for all OhSnap parameter sets over both proximity types.

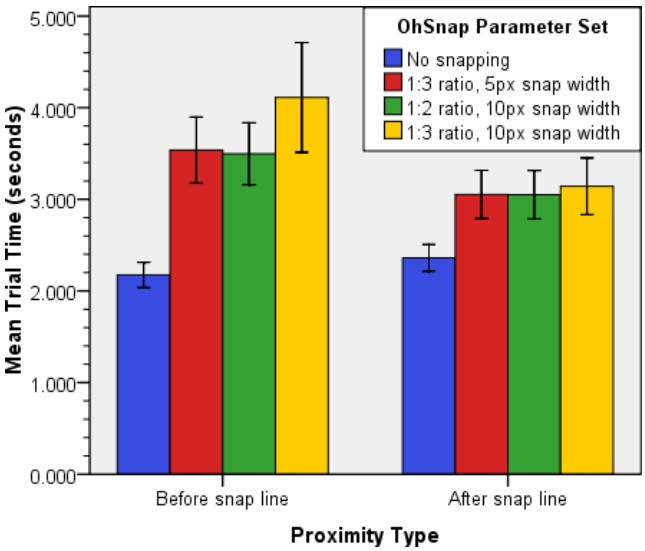


Figure 8: Mean trial time by proximity type grouped by OhSnap parameter set.

Number of touch ups

We conducted a 4 x 2 (OhSnap Parameter Set x Proximity Task) repeated measures analysis of variance for the average

Mean Trial Time (seconds)			
OhSnap Parameter Set	**Proximity Type**		
	Both	**Before**	**After**
No snapping	2.309	2.236	2.382
1:3 ratio, 5px snap width	3.350	3.689	3.012
1:2 ratio, 10px snap width	3.559	3.919	3.198
1:3 ratio, 10px snap width	3.662	4.128	3.196

Table 2: Mean trial time for each of the OhSnap parameter sets across both proximity types (just before and just after the snap line).

number of touch ups in each treatment. There was a significant main effect of both OhSnap parameter set ($F(3,15) = 8.473$, $p<.002$, $\eta^2=.629$) and proximity task ($F(3,15)=5.236$, $p<.035$, $\eta^2=.235$). Pairwise comparisons revealed that the no snapping condition had significantly fewer touch ups than the 2nd parameter set ($p<.002$) and 4th parameter set ($p<.008$), but not the 3rd.

Questionnaire data
(possibly TODO, check if there's something interesting first)

DISCUSSION

REFERENCES

1. D. Ahlström, M. Hitz, and G. Leitner. An evaluation of sticky and force enhanced targets in multi target situations. In *NordiCHI '06: Proceedings of the 4th Nordic conference on Human-computer interaction*, pages 58–67, 2006.

2. P. Baudisch, E. Cutrell, K. Hinckley, and A. Eversole. Snap-and-go: helping users align objects without the modality of traditional snapping. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 301–310, 2005.

3. E. A. Bier and M. C. Stone. Snap-dragging. *SIGGRAPH Comput. Graph.*, 20(4):233–240, 1986.

4. A. Lécuyer. Simulating haptic feedback using vision: A survey of research and applications of pseudo-haptic feedback. *Presence: Teleoper. Virtual Environ.*, 18(1):39–53, 2009.

5. R. L. Mandryk and C. Gutwin. Perceptibility and utility of sticky targets. In *GI '08: Proceedings of graphics interface 2008*, pages 65–72, 2008.