Pointing at 3D Wiggle Displays

Michael Ortega, Wolfgang Stuerzlinger

Pointing at Wiggle 3D Displays

Michaël Ortega*
University Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France

Wolfgang Stuerzlinger**
School of Interactive Arts + Technology, Simon Fraser University Vancouver, Canada

ABSTRACT

This paper presents two new pointing techniques for wiggle 3D displays, which present the 2D projection of 3D content with automatic (rotatory) motion parallax. Standard pointing at targets in wiggle 3D displays is challenging as the content is constantly in motion. The two pointing techniques presented here take advantage of the cursor's current position or the user's gaze direction for collocating the wiggle rotation center and potential targets. We evaluate the performance of the pointing techniques with a novel methodology that integrates 3D distractors into the ISO-9241-9 standard task. The experimental results indicate that the new techniques are significantly more efficient than standard pointing techniques in wiggle 3D displays. Given that we observed no performance variation for different targets, our new techniques seem to negate any interaction performance penalties of wiggle 3D displays.

Keywords: 3D Interaction Technique, Pointing, Eye Tracking.

Index Terms: H.5.2. Information Interfaces and Presentation: User Interfaces - Interaction styles, Input devices and strategies.

1 INTRODUCTION

Today, virtual 3D environments are used in many industrial and educational domains. For example, in medicine, and more specifically in anatomy learning, the students spend a large amount of time in exploring 3D models of bones, muscles and tendons. In this context, the depth perception provided by perspective projection is a critical component for constructing a correct cognitive model of these anatomical structures.

Monoscopic perspective projection already provides important depth cues from occlusion, relative size, and texture distortion. Other cues, such as binocularity and motion parallax further improve depth perception [2][5]. While most forms of binocular display requires specific devices, such as Head Mounted Displays [10], the need for an external device makes it less accessible to a general audience (students). Motion parallax, and more specifically automatic motion parallax, also called wiggle stereoscopy or short *Wiggle 3D display*, as illustrated in Figure 1, is a better candidate to convey strong depth cues. Automatic motion parallax can easily be used on every person's laptop, desktop or tablet computer.

A basic task in interactive 3D environments is to click on a 3D point to designate/select it for further operations. Yet, one of the biggest drawbacks of wiggle 3D display is that most potential targets points are continually in motion, which substantially decreases potential pointing performance, and could even make each selection stressful and tiresome. In this paper, we present two new techniques that stabilize potential targets by taking advantage of the cursor's current position or the user's gaze direction. To our knowledge, such selection has never been evaluated in wiggle 3D display systems.

Other related work has investigated gaze-based pointing techniques in virtual environments, both with 3D environments [12] or on 2D displays [13]. The most recent work uses HMDs with integrated eye-tracking capabilities [10].

Our contributions are:

- Two new pointing techniques for wiggle 3D displays,
- An evaluation of the techniques in a novel experimental design that integrates distractors into the ISO 9241-9 task [11].

2 WIGGLE 3D DISPLAY

Classic perspective projection uses the pinhole camera principle: a fixed viewpoint where light from the scene passes through onto the projection plane. It provides many important depth cues such as occlusion or relative size. Yet, it is a projection of a 3D environment onto a 2D image and thus information about the third dimension is at least partially lost.

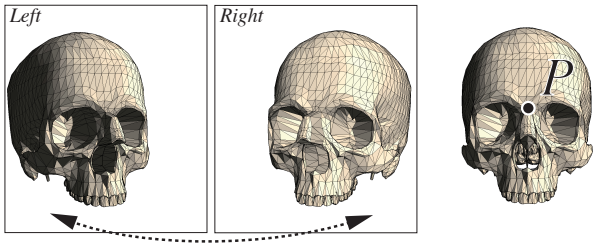

Figure 1: In Wiggle 3D Displays, the viewpoint permanently rotates back-and-forth around a fixed 3D point (*P*). In this figure, the left and center skulls present projections from the extremities of the wiggle 3D rotation. The continuous back-and-forth motion between these two extreme viewpoints enables the user to see the depth of each part of the skull or the protuberance up to the nose. This information is invisible in the frontal projection on the right.

Wiggle 3D display is a perspective projection **where the viewpoint permanently rotates back-and-forth around a fixed 3D point** (*P*). Wiggle 3D display provides a dynamic monoscopic perspective image, yielding changes in parallax and object occlusion, thereby providing more depth cues than the classic static projection [2].

P is a point in the 3D scene, around which the viewpoint rotates back-and-forth. *P is* usually situated in the centre of the 3D content, and the viewpoint rotation is typically horizontal. As a rotation pivot, *P* stays stable on the screen, as are all points situated on a 3D vertical line passing through *P*. The range of movement of any given point of the object (*po*) then increases

* michael.ortega@univ-grenoble-alpes.fr
** w.s@sfu.ca

Author Version of paper published at IEEE VR 2018.

proportionally to the horizontal distance between *po* and *P* (see Figure 2). However, the difficulty of selecting any such point *po* then also increases with the horizontal distance to *P*: points that move faster (further from *P*) are more difficult to select [7][8][9]. In the dynamic environment investigated by Ortega *et al*. [7], targets have unpredictable movements, which leaves the user only with the strategy of following the target until (s)he reaches it, which might never happen, when the target is too fast. With more predictable target movements [8][9], which is similar to the back-and-forth movements in wiggle 3D displays, an efficient pointing strategy might also be to follow the target movement. Yet, when the target moves too fast, another strategy is to position the cursor on a specific point of the target path, and then wait until the target is under the cursor. With both predictable and unpredictable target movements, the user has to change their pointing strategy, which affects pointing performance negatively.

In contrast to previous work that changes the shape of the cursor [3], our new techniques do not change the cursor, but change the display of the scene to make the point that the cursor is over better visible to the user.

For wiggle 3D to be useful for general applications, it is crucial to create pointing techniques that can match the pointing performance with the level of a static image, while maintaining the benefits of wiggle 3D display in terms of enhanced depth perception.

3 Techniques

3.1 Design Rationale

We propose two techniques for pointing in Wiggle 3D displays, which maintain the depth cues provided by this kind of display. The techniques aim to stabilize a target point during the continuous back-and-forth rotation of the viewpoint. In this context, we investigate the co-location of the 3D rotation pivot (*P*) with the target. As mentioned above, *P* and all the points situated on the vertical line passing through *P* stay stable on the screen. A target that coincides with *P* (see Figure 2*)* is then stable on the screen plane. Based on this insight, we propose two methods for computing *P*.

3.2 Gaze Pointing *(GPT)* and Mouse-based Pointing (*MPT)*

Both new techniques compute *P* in real time, based on the well-known ray-casting principle: a point on the screen (*SP*) and the viewpoint position (the center of projection or origin of the view for the 3D scene projection) creates a ray whose intersection with the scene is a 3D point: *P*. Changing *SP*, along the 2D screen, causes *P* to travel in 3D across the (visible parts of the) 3D scene surface(s). In the proposed techniques, if the ray-casting does not intersect the scene, *P* stays at its last computed 3D position.

As one of our motivations is commonly available, low-cost hardware, such as laptops, desktops or tablets, we propose two techniques for determining *SP* with no or low-cost, and then compute *P*.

In the first technique (called *MPT: Mouse Position Technique*) *SP* is simply determined by the mouse cursor position. This technique can then be used on classical computers, with no additional material, and with the mouse as an efficient pointing device.

In the second technique (called *GPT: Gaze Position Technique*), *SP* is the screen location where the user is gazing at. The technique requires an eye-tracker device, nowadays a low cost and widespread device, for knowing the gaze direction. Such an approach could be used on a computer, but also on a tablet. For tablets, hovering is typically not supported, which means that the finger cannot act as a mouse. Thus, *MPT* cannot be used and *GPT* is the only technique that could stabilize a target on a tablet.
As both techniques only change the visual feedback of the scene by modifying the rotation pivot of the view, the sequence of actions in a pointing task is identical to everyday pointing: move the mouse/the finger onto the target and click/touch it.

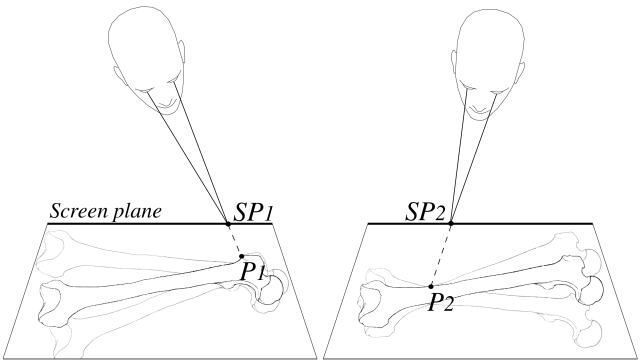

Figure 2: A wiggle rotation is made around a 3D point*: P.* The latter is computed from a screen position *(SP).* Here, *SP1* is the screen location where the user is gazing at, and *P1* is on the bone surface. The whole scene does small back-and-forth rotations around *P1*, which is behind the stable *SP1*. If the user is looking at a different screen position *SP2*, then the display changes to stabilize the target *P2* in screen space.

4 User Evaluation

Here we describe the general procedure and apparatus used in the experiment as well as the analysis of the results. In this study, we evaluate the performance of *GPT* and *MPT* against the status quo for wiggle 3D displays, which uses a fixed rotation pivot at the centre of the screen. We call this condition fixed pivot, *FPT*.

4.1 Hypotheses

From previous work we know that selecting a moving target is more difficult than selecting a static one [7][8][9]. We thus first hypothesize that performance will be better for both *MPT* and *GPT* than for *FPT*. More specifically, *MPT* and *GPT* will show on average faster pointing and fewer errors (*H1*).

However, with *FPT* targets close to the vertical midline of the screen could match with *P* and thus be static on the screen. *H1* should thus be correct "in general", but not for all the targets. We examine this in more detail in the results below.

Moreover, as a target acquisition task typically requires the user to first look at the target before actuating the cursor button [14], we assume that the target should be stabilized earlier with *GPT* than with *MPT*. We thus hypothesize that *GPT* will provide better performance than *MPT* (*H2*).

4.2 Distractors

GPT and *MPT* compute *P* in real time, for any new position of *SP*, determined respectively by the gaze or the cursor position. Then, during a pointing task, while *SP* travels on the screen, on a flat line in between the 2D projection of the starting point and the 2D projection of the target, *P* travels onto the visible faces of the 3D scene, which could be a 3D path with many depth changes. The oscillation amplitude of a final target is thus potentially continuously changing during a pointing movement. This makes our new techniques performance highly dependent on the 3D

scene and thus on the presence of distractors, i.e., potential targets along the pointing path [1].

In order to accurately measure the performance of *GPT* and *MPT*, our software apparatus thus has to simulate such an environment. In other words, only displaying the starting point and the final target does not yield an ecological environment for the task we are interested in. For our experiment, we thus decided to create a new solution for integrating distractors into a pointing task onto the (visible) surface of a virtual environment, and we decided to combine targets in a ISO 9241-9 circle (see Figure 3) with a distractor scene, which is then displayed in a wiggle 3D display system.

In the ISO 9241-9 task, all targets are arranged on a circle (Figure 3), and each circular configuration correspond to an index of difficulty (Fitts' law *ID*). However, in a wiggle 3D display the pointing difficulty depends on the pointing amplitude, the target's width, and also the target's oscillatory movements (speed and amplitude). As the oscillatory movements could change with the target's depth, we decided to put all the targets of an ISO circle at the same depth. We then consider that each ISO 9241-9 circle has a unique ID, even if the wiggle rotation implies different target movements according to their distance to *P*.

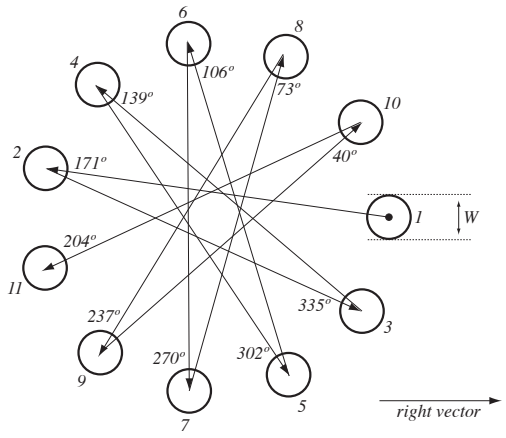

Figure 3: ISO 9241-9 task [11]. Eleven targets are placed in a circular configuration, and the pointing sequence is fixed. In the study, the *angle* variable (in degrees) is the angle between the vector made by two consecutive targets and the right vector (horizontal, from left to right).

Adding distractors means creating a 3D environment that places potential targets along the pointing path and challenges the user during the pointing action as could happen in a real application setting. In our context, such a 3D environment should respect three main constraints:

1. potential targets (*ptargets)* should have different depths, to simulate a (potentially) arbitrary 3D scene.
2. inspired from [1], *ptargets* should respect Fitts' law along the pointing path. Each *ptarget* should have an *ID* "equal" to the final target's one. However, due to their depth differences, their movement amplitudes are different. Thus, a distractor's real *ID* then "slightly" changes from the target's one. For distractors, we consider this deviation to be not critical, as the average *ID* is still correct.
3. both the distractors density (called rho [1]) and their depth amplitude are limited, as they cannot cause the target to be occluded.

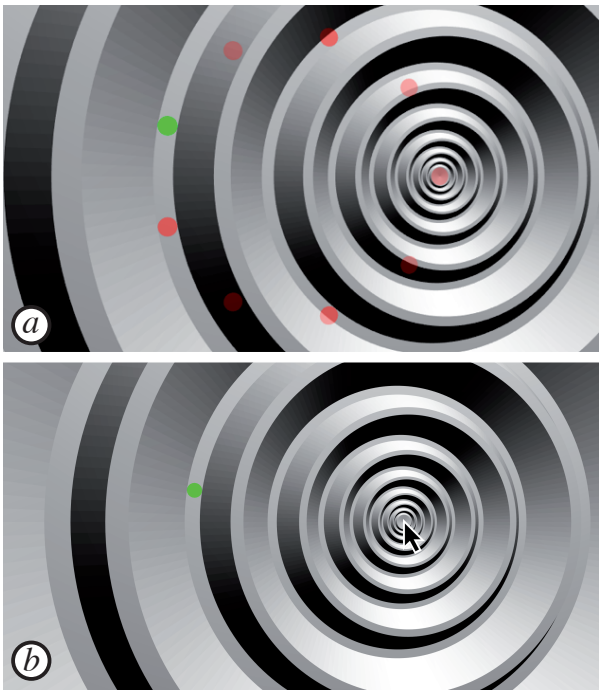

Figure 4: **a:** Illustration of ISO 9241-9 circle (targets are the transparent red circles) displayed on the distractor shape. The participants never saw this image. The current target is the green circle. Potential targets/distractors are the concentric grey circles (centered around the previous target position), whose width varies to have the same *ID* as the target. One in two potential targets/distractors has the target's current depth, while every other is closer to the user, following a linear function: depth = distance_to_start_point/5. **b:** Distractors shape as the user see it. The cursor is on the previous target.

From these constraints, we developed a distractor 3D shape (illustrated in Figure 4) with the following properties:

1. *ptargets* are circles around the starting point, with specific line widths.
2. The width of *ptargets* are the same as the circle's line width. This width increases according to both the distance to the starting point and Fitts' law. We compute the width with the formula from previous work that scaled distractors accordingly [1].
3. *ptargets* alternate between closer or farther from the viewpoint. Visual constraints require that farther *ptargets* be at the final target's depth (otherwise they may be invisible). Other *ptargets* are closer to the user and their visual depth is chosen to follow an empirical linear function: depth = distance_to_start_point/5 (Figure 5). Moreover, the density of the *ptargets*, has to be carefully chosen to avoid that *ptargets* occlude the final target.
4. in order to be more ecological (simulate the viewing of real objects) the gaps in between the potential targets are filled with cones, which makes the depth change smoothly from one *ptarget* to another along the path.

To keep visual consistency, all the targets of an ISO 9241-9 circle cannot be displayed at the same time. Indeed, some of them will be hidden by the distractors shape. Thus, we display only the next target. This is different from the ISO 9241-9 task, but we

consider that this does not influence the results, as after a training step we can assume that every user knows the pointing sequence and does not hesitate about where the next target is.

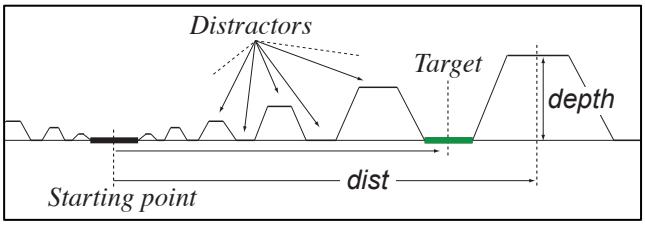

Figure 5: Profile of distractor shape. One in two potential targets/distractors has the target's current depth, while every other is closer to the user. Their "height" follows the linear function: *depth = dist / 5*, where *dist* is the distractor's distance form the starting point.

4.3 Apparatus and Software

The study was conducted on an 15" Apple Macbook Pro Retina, with 2.6GHz Intel Core i7 and an NVIDIA GeForce GT 650M 1024M. We used a 22" monitor, at 1600x1024@60Hz (Apple Cinema Display). Eye tracking was done with a Tobii 4C. The monitor was positioned vertically, with the eye-tracker mounted at the bottom of the screen. We used a traditional mouse as a 2D pointing device. Mouse acceleration was disabled, and the gain was set to 3.54: 1cm movement with the mouse made the cursor move about 3.54 cm on the screen.

The software was written in Python, with PyOpenGL, using shaders in GLSL 3.30. The wiggle 3D display was created by rotating the viewpoint according to the current rotation pivot (*P*) (computed differently for each pointing technique). The viewpoint made back-and-forth rotations with a range of 18 degrees. This is equivalent to 20 cm of lateral head movements at 70 cm from the screen. We used a sinusoidal function for smoothing the changes of direction, and the viewpoint made complete back-and-forth loops in 0.8 *secs*. These values were established through observation of many wiggle 3D displays that one can find on the internet as well as pilot experiment.

The cursor icon was unchanged. We kept the traditional arrow cursor that is typical for desktop operating systems.

4.4 Participants and Procedure

We recruited 12 participants (mean age 28 years, SD 3.4) from the local university. Participants first started with the eye-tracking calibration step. Next, the experimental context was introduced by presenting the wiggle 3D display principle.

To compensate for the lack of familiarity with the different conditions, participants were asked to perform 4 practice trial "circles" with each condition. This was enough to enable participants to feel comfortable with our simple task. Participants were instructed to select the green highlighted target as quickly and accurately as possible, and to only rest after finishing a complete ISO circle. The distractors shape was not displayed during the resting period, making this period easy to identify.

4.5 Design

The study used a 3x4 within-subject design. The first independent variable is the technique: **3** techniques *GPT*, *MPT* and *FPT*. The next variable is the combination of ISO circle and distractor shape, from Fitts' law *ID*s and *rho* (distractor density). The radius of the ISO circles was fixed, for a pointing amplitude of 766 *pixels*. Only the target width changed. In this study, **4** combinations of [*ID*, *rho*] were used: [3, 0.6], [4, 0.3], [5, 0.2], and [6, 0.1], which were created with target widths of 110, 51, 25 and 12 *pixels*, respectively.

Techniques were counter-balanced between participants while the [*ID*, *rho*] combinations were randomly presented. The first dependent variable was the **movement time**. It is the time between the previous click and the click on the current target. The second variable is the **error rate**, which expresses the number of missed target. The last variable is the **throughput**, in bits per seconds, computed following the procedure outlined by previous work [6].

3x4=12 different conditions were presented to each participant, with **10** trials per condition. Each ISO circle had **11** targets, which corresponds to 10 pointing tasks (the time to click on the first target cannot be controlled). Each participant completed a total of 3x4x10x10 = 1200 pointing tasks, for a total of 12x1200 = 14400 recorded trials overall.

After the experiment, which lasted about 30 minutes, participants were asked to rank the techniques in terms of preference, and to record their feelings in terms of performance, fatigue and usability.

4.6 Results

We analyzed the results with repeated measures ANOVA and report the outcomes for alpha = 0.05. As with many pointing experiments, the data was not normally distributed. We first filtered outliers beyond three standard deviations from the mean, and then used a log transform before performing the ANOVA. The data then also passed Mauchly's sphericity test.

4.6.1 Quantitative

Technique had a significant effect on movement time, $F(2,22) = 19.14$, $p < 0.0001$, error rate $F(2,22) = 26.35$, $p < 0.0001$, and throughput $F(2,22) = 21.27$, $p < 0.0001$.

As identified by a post-hoc test, for all significant effects the *FPT* condition is significantly slower/has more error/less throughput than the group formed by both the *MPT* and *GPT* conditions. For example, *MPT* and *GPT* throughputs are 4.92 respectively 4.88 bps, which the *FPT* condition has only 4.19 bps (Figure 6).

Movement angle had a significant effect on movement time, $F(9,99) = 5.91$, $p < 0.0001$, error rate $F(9,99) = 8.81$, $p < 0.0001$, and throughput $F(9,99) = 19.23$, $p < 0.0001$.

There is a significant interaction between technique with angle in terms of movement time $F(18,198) = 3.69$, $p < 0.0001$, error rate $F(18,198) = 6.59$, $p < 0.0001$, and throughput $F(18,198) = 4.99$, $p < 0.0001$.

In terms of throughput a Bonferroni Multiple Comparison Test identifies the following grouping of angles: (171°, 204°) with the lowest throughput, followed by (40°, 335°) and (139°, 237°, 302°), and (73°, 106°, 270°) with the highest throughput.

Considering only the angles at 73°, 106°, and 270°, i.e., movements that are predominantly vertical, there is no significant difference between the *FPT, MPT* and *GPT* conditions (throughputs between 4.55 and 5.28), $F(2,22) = 2.654$, $p > 0.05$. Figure 7 illustrates the differences between the techniques in terms of throughput for each angle.

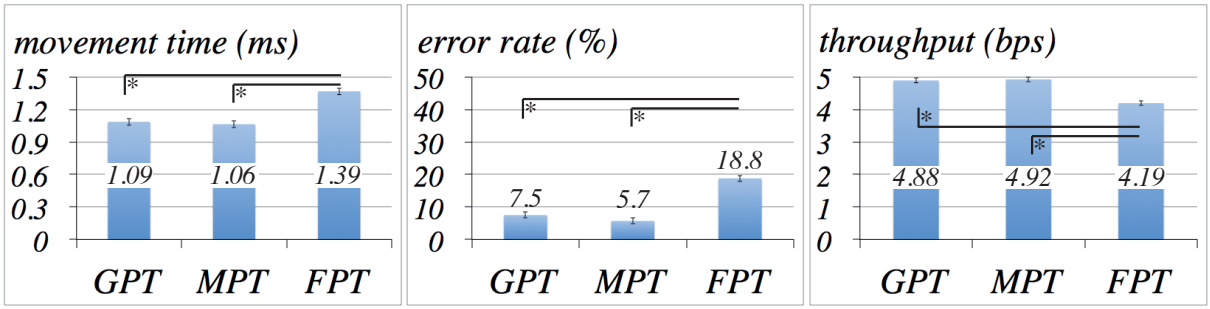

Figure 6: Mean and standard error of movement time, error rate and throughput for each technique. *FPT* is significantly slower, more error prone, and has lower throughput than *GPT* and *MPT*, which are statistically not different.

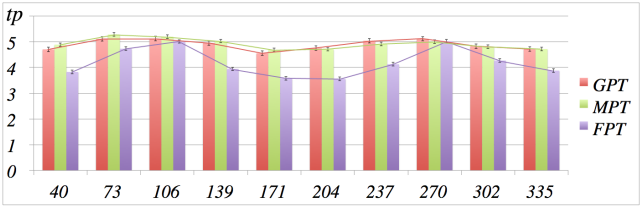

Figure 7: Mean throughput per angle for each technique. *FPT* is highly dependent on the angle, while *GPT* and *MPT* do not vary significantly. *FPT* is equivalent to other techniques when the pointing angle is vertical, i.e up/down to the central pivot, but lower when pointing is horizontal, i.e distant and on both sides of the central pivot.

4.6.2 Qualitative

Taking the average of technique rankings, participants prefer *GPT (8/12)*, followed by *MPT (4/12)*, and finally *FPT* which was always ranked last. Eight participants felt fatigue with *FPT*, while two participants expressed that they experienced slight fatigue with *GPT*: they had the feeling they had to be more concentrated than with other techniques. All the participants felt fatigue about the experiment itself during the last condition: "the distractors shape is too geometric, and the rotational movements creates a visual fatigue effect" (P6).

Two participants thought that they were more efficient with *GPT* because they had the feeling that "the gaze arrives sooner on the target than the mouse cursor" (P5, P6).

4.6.3 Discussion

Our results highlight the benefits and potential of our two new pointing techniques in terms of time, errors, and throughput. Overall, the new techniques, *GPT* and *MPT*, which use the gaze respective mouse cursor, provide both similar results, which validates *H1*, as these techniques significantly outperform the status quo (*FPT*). As expected, this performance benefit does not exist for targets along the vertical midline of the screen, simply because in *FPT* these targets are stable by definition.

Still, even if targets are stable earlier with the gaze pointing technique *GPT* [14], results do not show a significant difference between *GPT* and *MPT*, which means that we have to reject *H2*.

Finally, *GPT* and *MPT* are clearly preferred, and all participants identified benefits in terms of comfort and performance. *GPT* was preferred by eight users, but *MPT* also receive very good reviews, while FPT clearly was the worst condition.

5 *MPT* OR *GPT*?

We point out that our throughput results match the results for 2D pointing [6]. Moreover, as our new pointing techniques stabilize the targets, they enable the user to perform precise manipulations on a 3D object, such as translating a vertex, or rotating/extruding/drawing a specific face. Yet, the wiggle 3D display still provides more depth cues than a static projection.

GPT and *MPT* are statistically not significantly different, and *GPT* is preferred by most of the users. This validates that *GPT* is a good candidate for both tablet and laptop/desktop computers. On tablet computers, *GPT* is the only candidate. Indeed, *MPT* cannot be used on such screens, as there is no cursor on tablet systems (the finger hovering the screen is not detected). Moreover, on such computers, eye-tracking without additional devices is becoming available [4].

On laptop/desktop computers, a few participants complained about the use of eye tracking and the gaze point. Thus, we are not certain of *GPT* is a solution suited for long term use. Still, as the two techniques have the same core function: $f(SP) = P$, it could be interesting to provide both techniques to the user and to offer the user the possibility to switch between them. A longitudinal experiment could reveal which technique is better for long term use.

We could have investigated other techniques that freeze the viewpoint rotation. Some of the alternatives that seem feasible are freezing the rotation as soon as the mouse button is depressed, as soon as an additional button is depressed, or as soon as the mouse moves. Relative to the status quo, any of them might also improve pointing performance, as long as the image is frozen, to the level of a static image. However, freezing limits the number of targets that are accessible during a pointing movement. Compared to a static viewpoint, the back-and-forth rotation of the viewpoint enables the system to display more of object's surface area, such as the sides of the skull in Figure 1. These areas are potential targets, and any freezing-based pointing technique would force the user to wait for the target to become visible to start a pointing movement. This could have a significant impact on pointing performance, because the user now needs to anticipate the viewpoint rotation and to precisely know when to freeze the image so that the target is visible.

Moreover, on laptop/desktop systems the mouse is frequently also used for hovering without pointing. This is particularly useful for illustrating some aspect of the scene to another user. Automatic freezing during mouse hovering decreases the additional depth cues due to motion parallax during this specific task.

In 3D user interfaces, inspecting a 3D object/scene is a very common task that is associated with pointing. Inspecting involves mainly travelling the gaze over a 3D shape for learning its shape or for looking at something on the surface. For this task, *GPT* and *MPT* do not act the same. With *MPT*, the gaze does not modify the wiggle rotation. The scene movements are predictable and there is no "Midas effect". However, when the 3D point that the user looks at is far from the cursor, wiggle movements could be very large, making accurate inspection potentially more difficult. The simple solution is to move the cursor. With *GPT*, the gaze modifies the wiggle rotation. There is a "Midas effect", but it is positive: the scene is stable at the gaze position, and all the rotation movements are thus in the non-central (i.e., peripheral) area of the human visual field of view, still providing depth cues. From this, *GPT* seems a better technique in general.

6 Conclusion And Future Work

This paper presents two new pointing techniques for wiggle 3D displays that outperform the status quo in terms of performance and user preferences. This makes wiggle 3D display a good candidate for displaying 3D content on low-cost display devices, and potentially even for 3D design software.

In order to evaluate the effects of these new pointing techniques in longer term usage, we plan to implement them in a 3D design software package and then to perform a longitudinal evaluation with two categories of users: students in anatomy and 3D designers, and on two categories of computers: desktop and tablets.

Acknowledgments

We thank Özge Nilay and Junwei Sun for their assistance with a pilot study for this work. We thank Simon Chambonnet for its implementation of the first prototype and the first pilot study with the distractor shape. We acknowledge funding by NSERC for this research. We also gratefully acknowledge the support of ANR (French National Research Agency) (ANATOMY2020 ANR-16-CE38-0011, ISAR ANR-14-CE24-0013 and PERSYVAL-lab ANR-11-LABX-0025-01).

References

[1] Renaud Blanch and Michael Ortega. 2011. Benchmarking pointing techniques with distractors: adding a density factor to Fitts' pointing paradigm. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 1629-1638. DOI: https://doi.org/10.1145/1978942.1979180

[2] Eleanor J. Gibson, James J. Gibson, Olin W. Smith, and Howard R. Flock. 1959. Motion parallax as a determinant of perceived depth. In *Journal of Experimental Psychology*, Issue 58, 40-51.

[3] Tovi Grossman and Ravin Balakrishnan. 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 281-290. DOI: http://dx.doi.org/10.1145/1054972.1055012

[4] Corey Holland and Oleg Komogortsev. 2012. Eye tracking on unmodified common tablets: challenges and solutions. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (ETRA '12), Stephen N. Spencer (Ed.). ACM, New York, NY, USA, 277-280. DOI: https://doi.org/10.1145/2168556.2168615

[5] Frank L. Kooi, and Alexander Toet. 2004. Visual comfort of binocular and 3d displays. In *Displays 25*, 23 (2004), 99 – 108. DOI: https://doi.org/10.1016/j.displa.2004.07.004

[6] Scott MacKenzie and Poika Isokoski. 2008. Fitts' throughput and the speed-accuracy tradeoff. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, USA, 1633-1636. DOI: https://doi.org/10.1145/1357054.1357308

[7] Michael Ortega. 2013. Hook: Heuristics for selecting 3D moving objects in dense target environments. In *IEEE Symposium on 3D User Interfaces (3DUI'13),* Orlando, FL, 119-122. DOI: 10.1109/3DUI.2013.6550208

[8] Andriy Pavlovych and Wolfgang Stuerzlinger. 2011. Target following performance in the presence of latency, jitter, and signal dropouts. In *Proceedings of Graphics Interface 2011* (GI '11). Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 33-40

[9] Andriy Pavlovych, Carl Gutwin. 2012. Assessing Target Acquisition and Tracking Performance for Moving Targets in the Presence of Latency and Jitter, In *Graphics Interface 2012*, 109-116.

[10] Ken Pfeuffer, Benedikt Mayer, Diako Mardanbegi, and Hans Gellersen. 2017. Gaze + pinch interaction in virtual reality. In *Proceedings of the 5th Symposium on Spatial User Interaction* (SUI '17). ACM, New York, NY, USA, 99-108. DOI: https://doi.org/10.1145/3131277.3132180

[11] William Soukoreff and Scott MacKenzie. 2004. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. *Int. J. Hum.-Comput. Stud.* 61, 6 (December 2004), 751-789. DOI: http://dx.doi.org/10.1016/j.ijhcs.2004.09.001

[12] Vildan Tanriverdi and Robert J. K. Jacob. 2000. Interacting with eye movements in virtual environments. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (CHI '00). ACM, New York, NY, USA, 265-272. DOI: http://dx.doi.org/10.1145/332040.332443

[13] Eduardo Velloso, Jayson Turner, Jason Alexander, Andreas Bulling, and Hans Gellersen. An Empirical Investigation of Gaze Selection in Mid-Air Gestural 3D Manipulation. 15th Human-Computer Interaction (INTERACT), Sep 2015, Bamberg, Germany. Lecture Notes in Computer Science, LNCS-9297 (Part II), pp.315-330, 2015, Human-Computer Interaction – INTERACT 2015. DOI: 10.1007/978-3-319-22668-2_25

[14] Shumin Zhai, Carlos Morimoto, and Steven Ihde. 1999. Manual and gaze input cascaded (MAGIC) pointing. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (CHI '99). ACM, New York, NY, USA, 246-253. DOI: http://dx.doi.org/10.1145/302979.303053