ARMouse: using the mouse as a foundation for a universal interaction paradigm for hybrid AR environments on desktops

Ben van Klingeren
ICA-4257383
Utrecht University
Utrecht, Netherlands

January 15, 2021

This work proposes ARMouse with the aim of providing a precise and reliable solution for desk-oriented AR interaction. ARMouse is a novel interaction paradigm that uses the mouse as the main interaction device for both desktop computing and plane based mobile AR 3D object translation, providing a seamless transition between AR and virtual spaces. In a within-subjects study featuring a selection and translation task ARMouse outperformed a touch based method in terms of perceived comfort, translation speed and translation accuracy. Furthermore, ARMouse unexpectedly also elicited higher levels of immersion among participants. This work shows the potential of a mouse based AR interaction paradigm and provides an initial step towards a universal interaction paradigm that supports hybrid AR desktop environments.

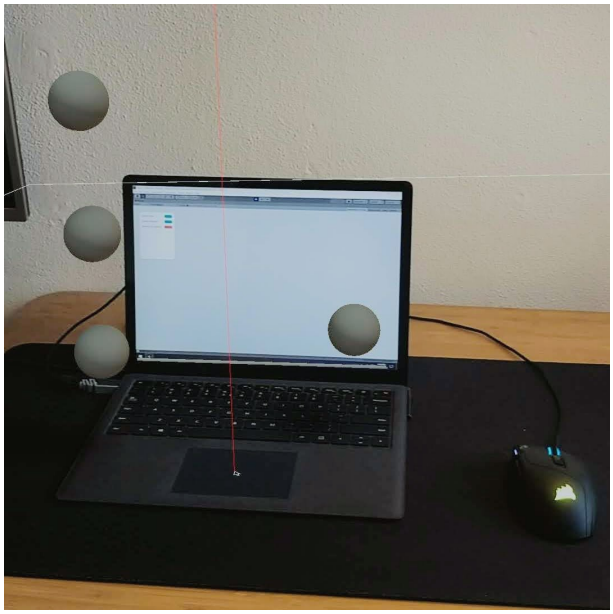

Figure 1. ARMouse setup featuring the cursor, selection line, transition line (the grey line on the desk surface spanning the with of the screen), and spheres to interact with, viewed from the perspective of the smartphone.

Author Keywords

Augmented Reality; Mobile AR; Mouse Interaction; Design; 3D Object Manipulation; Immersion; Desktop Computing

INTRODUCTION

Augmented Reality (AR) allows us to place virtual objects and information into our real surroundings, thus enabling us to utilize the vast space of the virtual world from the comfort of the more familiar real world. However, as more use cases for AR are discovered it becomes apparent that not all interaction should take place in AR. Users may need to be able to utilize both AR and purely virtual tools at the same time [27]. As such, more research is being conducted on systems that feature both an AR environment and a virtual environment: hybrid systems. In the context of this work, hybrid desktop AR refers to a hybrid system that can display content either on the desktop screen or in AR space, with a seamless transition and desktop computing at its core. Finding an interaction method that effectively supports a system essentially featuring two environments is however a non-trivial task. One would either have to switch interaction methods in order to switch environments, thus limiting the seamless transition, or a single interaction method that optimally supports both environments would need to be constructed. Furthermore, current AR interaction methods are unsuited for seated scenarios or prolonged use, making them less optimal for a desktop-oriented system [28]. Therefore, the aim of this work is to propose and evaluate a potential foundation for a hybrid desktop AR universal interaction paradigm. What is understood by a universal interaction paradigm is the goal of keeping the interaction device and paradigm constant, regardless of the device that is used for viewing the AR content. Given this definition, a set of principles is defined that enables the findings of this study to be generalized to other AR devices.

Existing research suggests that in the context defined in this work mouse based interaction offers various advantages (Section 2). Therefore, we propose using the mouse as a basis for this interaction paradigm, naming the first prototype ARMouse, which is implemented using smartphone based mobile AR. To verify the suggested effectiveness of a mouse based AR interaction paradigm the research goal is defined more concretely as: *"Does ARMouse outperform a well-established mobile AR interaction method in a selection and a translation task?"*

RELATED WORK

Recently, more research is conducted on leveraging the increased immersion and space AR offers to improve the understanding of three dimensional data. However, as not all interaction should happen in 3D space, hybrid systems emerge [4]. One such research is DatAR, which aims to support neuroscientists conducting literature research by providing a 3D representation of linked data in AR [22]. Another is an observational study that found "physicists to appreciate a hybrid data exploration setup with an interactive AR extension to improve their understanding of particle collision events" [27].

The mouse is a remarkably effective input device that is still able to outperform various newer inventions. Berard et al compared the mouse to 3D input devices in a 3D placement task and found the mouse to be preferred when high accuracy is required and that its higher accuracy and lower levels of induced stress compensate for the need to break down 3D operations into multiple 2D operations [3]. In *The evaluation of tangible desktop AR UIs*, Dunser et al concluded the mouse to be able to reach a high accuracy faster than the other proposed input devices. However, the authors argued that the 2D manipulation tasks performed during the experiment are similar to regular desktop interaction, therefore explaining the mouse's success [7]. Teather as well as Wang also found the mouse to outperform a 3 degrees of freedom (DOF) tracker in a 3D object manipulation task [23, 26]. Johnsgard compared a VR glove to the mouse with regard to Fitts's Law via a selection task. He found that glove users tended to undershoot and mouse users tended to overshoot. Nonetheless, the mouse proved to be faster and, unlike the glove, adding gain to the mouse did not decrease the performance. Johnsgard argues the results of his findings may be explained by the mouse being operated while resting on a stable surface [11]. Schultheis et al compared a 6+6 DOF two-handed interface to a 6 DOF wand interface and a 2 DOF mouse interface for 3D operations. The authors found the two-handed interface to perform best, especially due to the task requiring many object and viewport manipulations, but argue the mouse to be efficient when fine control or interaction with various other types of content such as menus, dialogs and text inputs is required [21]. In contrast, Masliah et al found users performing a docking task with 6 DOF devices to break down the operation into multiple 3 DOF operations, suggesting 6 DOF operations not to be as intuitive to perform as they seem [14].

The effectiveness of hand gesture recognition is limited by arm fatigue and the preciseness of the human motor system [2, 16, 15], although strides to mitigate this effect are being made [9]. Still, the hand remains a big and awkward pointing device that easily obscures smaller objects [19].

METHODOLOGY

ARMouse design

The goal of ARMouse is to unify the interface between desktop computing, AR and the user. As a result it aims so satisfy a few distinct requirements: fluent and intuitive transitioning between environments to streamline the hybrid nature of the system; optimal interaction in both environments; comfortable interaction during prolonged use.

The mouse is proven to be a good candidate for the role of input device due to its high precision compared to other input devices/methods; its ability to retain comfort during prolonged use and it already being the established interaction device for desktop computing. Therefore, being intuitive for virtually all potential users. Furthermore, the mouse also fits the desk-oriented setup of a professional environment well. Thus, use of the mouse already satisfies a considerable portion of the requirements.

Main concept

The main concept of ARMouse is founded on three principles. The first is using plane based interaction to unify the environments: both the screen and the desk represent planes on which a cursor can travel. This constrains the movement enough to allow for high accuracy and ease-of-use, and limits the available AR space to fit the need of a desk-oriented setup. The planes defined by the screen and the desk are extended and the intersection between these planes defines where the cursor can transition between the purely virtual space (on the computer screen) and AR space (Figure 1). This allows for a fluent and intuitive transition between the spaces: as the cursor crosses the bottom of the screen it transitions to AR space and is placed at the intersection point on the desk. Conversely, as the cursor crosses the intersection line in AR space it transitions to virtual space on screen. This complements the notion of multiple systems and environments forming a coherent whole.

The second principle is that as a zero-dimensional selection tool (the tip of the cursor selecting a 2D object) supports a two-dimensional environment, a one-dimensional selection tool (a selection ray cast from the cursor into 3D space) supports a three-dimensional environment. Therefore, ARMouse features a line emerging upwards from the cursor with which objects can be selected. In raycast based selection this is implicitly realized, but the decoupling of perspectives (as explained in the next paragraph) warrants a more explicit visualization of this principle as potential selection targets are not as easily apparent to the user.

The third principle is leveraging the increased immersion AR offers to allow the cursor to detach from the camera plane (the plane that is always perpendicular to the direction the camera is looking) and be constrained to a horizontal plane in instead (i.e. the desk), increasing the effectiveness of mouse based interaction. Constraining the cursor to a plane in world space decouples the camera's perspective from the cursor's perspective. As seen in Figure 2, this means that an object (assuming it is not enclosed by another object) is always visible to either the cursor or the user. To clarify, in a regular setup the cursor and the camera share the same perspective because the cursor traverses the camera plane, but if the cursor were to traverse a plane in world space its perspective is decoupled from that of the camera. Only a high level of understanding of the three dimensional space as offered by AR may have the potential to enable the use of this particular configuration. This principle is henceforth referred to as the decoupling of perspectives.

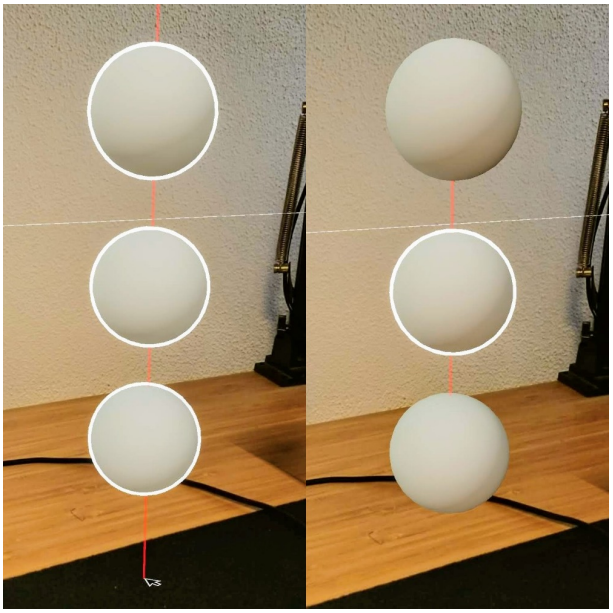

Figure 2. The selection line pointing upwards from the cursor and three spheres in the hover state (left), leading to the secondary selection state during which occluded objects can be more easily selected (right).

Object selection

Object selection is performed with the help of the selection line and the hover feature. The selection line facilitates the selection of objects that are positioned in the space above the desk. The hover feature highlights every object that is intersected by the selection line, providing feedback to the user on which objects are in this position selectable by the cursor. As shown in Figure 2, when multiple objects are intersected by the selection line all will be hovered and the user can move the mouse up and down to switch between the hovered objects and more deliberately select the correct object with a button click. This extra step is here referred to as secondary selection and is similar to the operation found in virtual 3D applications to aid in selection, usually done via a list of objects that appears as the user attempts to select an occluded object (e.g. by holding down the left mouse button). However, due to the decoupling of perspectives possible in AR the user can more directly and intuitively select the correct object when cursor occlusions occur, as these objects are from the user's perspective not occluded.

Object translation

Object translation on the horizontal plane is performed by dragging the selected object with the mouse. Vertical translation is performed by holding the right mouse button and moving the mouse up or down. This allows vertical translation to be of an equal movement scale and precision as horizontal translation. Translating an object horizontally and vertically at the same time is not possible.

Implementation

The prototype for ARMouse is built using the AR Foundation framework [24] within the Unity engine [25]. A video showcasing this prototype can be found at the following url: https://youtu.be/ibtDSi5Pseo. Mouse input is handled by Mirror [18], which sends the raw mouse data and button states over the network to the smartphone. To reduce any noticeable latency, cursor update methods run on a higher frequency than the rest of the code and any visible stuttering is reduced by linearly interpolating the updated cursor position by a constant value. This also masks any lag introduced by a drop in the frames per second. To increase immersion, shadows and materials of virtual objects react on the lighting direction and intensity. This allows the objects to better blend into the real world, and does not negatively affect their visibility, given enough daylight.

3DTouch

To evaluate the effectiveness of ARMouse, it is compared to the most common smartphone AR interaction method that is applicable to the context of this research. For object selection this is obviously touch based raycasting (the user taps the object on the screen, causing a ray to be cast from that position into the 3D space), but for object translation various options exist. The most recent and promising object manipulation methods that fit the requirements of this study's setup are 3DTouch and HOMER-S [17]. HOMER-S is a 6 DOF interaction method that features simultaneous rotation and translation by directly mapping the smartphone's pose onto the object. HOMER-S was however rejected in favor of 3DTouch because 6 DOF operations are not needed and even undesired [14], and 3DTouch is expected to be more accurate overall as the object can only be translated on one axis at a time. Objects are translated with 3DTouch by dragging the finger vertically or horizontally over the smartphone's screen. This movement translates the object along the horizontal x axis and vertical y axis when the smartphone is held more vertically, or along the horizontal x and z axes when the smartphone is held more horizontally (camera pointing downwards). The border between a vertical and horizontal device orientation is denoted by a 45 degree angle. Objects are selected and deselected by tapping them on the screen. For this study two elements of 3DTouch are modified. Firstly, each line representing a translation axis is made transparent when translation along its axis is not possible given the current device orientation. This provides the user with feedback on which axes are active at any given moment. Secondly, gain is added to the translation speed defined by the speed of the finger dragging across the screen. This change allows users have more control over the precision and speed of object translation. With a fast swipe over the screen an object is translated much further than the projected distance of the swipe, allowing for quick but imprecise translation over further distances. By slowly dragging the finger across the screen the object is translated over a shorter distance than the projected swipe distance, allowing for more precise object positioning. By controlling their swipe speed users are with this feature able to position an object more accurately, even over longer distances, without having to lift their finger off the screen.

Hypotheses

H1 Research suggests that the mouse offers a higher precision than various other input methods or devices. Therefore, ARMouse is expected to have a higher accuracy than touch based raycasting in an object selection task.

H2 Research suggests that the mouse offers a higher precision than various other input methods or devices. Therefore, ARMouse is expected to have a higher accuracy than 3DTouch in an object translation task.

H3 Due to the mouse's 2 DOF movement and its expected higher precision resulting in fewer adjustments, ARMouse is expected to position an object in an object translation task in a shorter time than 3DTouch.

H4 Participants using 3DTouch are expected to move the smartphone around more in order to select and translate objects. Also they are expected to keep both arms up at all times during the task. Because mouse users can in contrast remain seated in a more relaxed position ARMouse is expected to be more comfortable during an object translation task.

Experiment Design

This user study has a within-subjects design to maximize the amount of data generated by each participant, minimize noise, and to provide fuel for a potential open discussion. At the beginning of a session the participant fills out a consent form that explains the reasons for the session and informs the participant of the potential risks. After which a training phase starts to allow the participant to get accustomed to both implementations, which concluded once the participant feels comfortable performing all operations offered by the interaction method, never lasting longer than 5 minutes. After the training phase the participant performs a selection task and subsequently a translation task with each interaction method, resulting in a total of four tasks. In turn, each of these tasks consists of several rounds. Participants with an uneven participant number start with ARMouse and those with an even participant number start with (3D)Touch. This results in each participant completing four tasks total. The total duration of a session ranged from 30 minutes to an hour, mostly dependent on the duration of discussions.

To minimize the scope of this research the only implemented object manipulation operation is translation, as it is the most common of the three base operations.

Because this study is conducted during the Covid-19 pandemic the number of participants is limited and standard Covid-19 safety protocols have been followed.

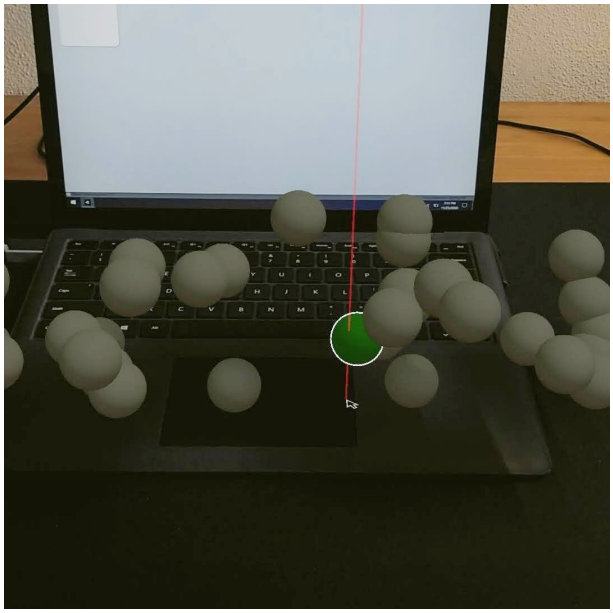

Figure 3. Selection task configuration, hovering over the target sphere.

Participants

Ten participants are selected for this study, two female and eight male, ages 22 to 26. These participants are chosen based on a few parameters: they have to be familiar with mouse interaction as well as touch interaction and they need to be considered average in their experience with 3D virtual environments. In practice, highly experienced participants could not be found and potential participants who did not have any experience with 3D environments were declined. Given these parameters a lower variance between participants is expected.

Setup

The setup of the experiment consists of a desk and chair, laptop (Microsoft Surface Laptop 2, 8th gen i5 and UHD 620), mouse (Corsair Sabre), Android smartphone (Oneplus 5T), and a strong internet connection (15 ms ping, 75 mbps upload-/download), see Figure 1.

Selection task

The selection task (Figure 3) consists of two sets of eleven rounds and a contained area on the desk of roughly 12 by 48 centimeters, positioned at roughly 50 centimeters away from the face of the participant. These values were chosen to limit the reach of participants should they want to move the smartphone up close to the objects and to keep the entire virtual area in view of the smartphone camera when the participant is in a relaxed seated position. This space is subdivided into four boxes of equal size, among which 20 spheres of 3 centimeters in diameter are equally distributed. Within each box the five spheres are in turn randomly distributed. One of the 20 spheres is colored green, designating it as the selection target. Participants are asked to select the target as quickly and as accurately as they could. Once the target object is selected the round is completed and the spheres are redistributed to start the next round. In between sets participants are granted a five second break. A set is made up of eleven rounds, the first of which is not used to collect data but rather to set the initial configuration to be used for the time and distance measurements. This results in 20 samples per variable per participant. Objective data gathered during the task includes:

- Distance: the measured distance in millimeters between the previous selection target and the current selection target.
- Time: the time in seconds measured from the start of the round to the moment the current selection target is selected.
- Speed: the speed in millimeters per second as calculated from the measured distance and time.
- Errors: the number of failed selection attempts during a round (whenever a selection attempt (tap on the screen or left mouse click) did not cause the correct object to be selected).
- Device distance: the distance in centimeters between the smartphone and the selection target at the moment of selection. This variable tells us when the participant is no longer in a relaxed seated position and shows whether the participant had to move closer to the object in order to select it.

After the task is completed using both interaction methods participants are asked to provide two advantages or strengths of the used interaction method for this specific task they noticed while performing the task. Participants could either note these down themselves or communicate these verbally and they would be transcribed afterwards. After this, potential follow-up questions are asked aimed at discovering any reasoning missing from the aforementioned notes, potentially also leading up to a brief open discussion. Any other questions, remarks or discussion items are also noted.

Translation task

The setup of the translation task is similar to that of the selection task. The difference here is that only two spheres are randomly spawned at the start of each round: one white and one green in color, each in a different box. Participants are tasked with positioning the white sphere such that it overlaps with the green sphere, a configuration signified by *z-fighting*, a position at which the two spheres seemingly form one that is half green and half white. To confirm the placement of the white sphere participants are given a button within thumb's reach on the smartphone, which can be tapped quickly and comfortably during one-handed use. Participants are asked to perform this task as quickly and as accurately as they could. Gathered data includes:

- Distance: the distance in millimeters measured from the initial position of the white sphere to the target sphere (distance between the origin of the objects).
- Time: the time in seconds measured from the start of the round to the moment the confirm button is tapped on the smartphone.
- Speed: the speed in millimeters per second as calculated from the measured distance and time.

- Target distance: the distance in millimeters measured from the white sphere to the target sphere (distance between the origin of the objects).
- Device distance: the distance in centimeters between the smartphone and the target sphere at the moment of confirmation.

After participants complete each translation task they answer a NASA Task load Index (NASA-TLX) form [1]. This assessment tool allows for the quantification and comparison of the subjective task load. This tool, alongside participant remarks and the measured device distance, helps in uncovering the differences in influence on the perceived workload between the interaction methods. During this study an unweighted version is used because the relatively short duration of tasks negatively affects the reliability of weights chosen by the participants. When the task is completed using both interaction methods and all forms have been filled in participants are asked the same questions as after the selection task and the same protocol for communicating these is followed. This once again may result in a brief open discussion. At the end of the session participants can provide any final remarks or ask any final questions.

RESULTS

Performance

Automatically collected objective data is analyzed via a comparison of mean values and a two-factor ANOVA with replication (95% confidence interval).

Selection accuracy

The selection accuracy of interaction methods is defined by the number of errors. The use of ARMouse has resulted in fewer selection errors on average than touch based raycasting (30 vs 45), but the difference is not significant ($F_{1,371} = 0.344, p = 0.558$).

Translation accuracy

The translation accuracy of interaction methods is defined by the distance between the selected object and the target object at the moment of confirmation. During the translation task the use of ARMouse accounted for a significantly higher accuracy ($F_{1,371} = 7.543, p = 0.006$). On average a distance of 4 millimeters was measured between the objects for ARMouse, and 6 millimeters for 3DTouch respectively.

Speed

Figure 4 shows the recorded speeds of both interaction methods during both tasks. Touch based raycasting was significantly faster than ARMouse during the selection task ($F_{1,371} = 77.274, p = 0.000$). In contrast, ARMouse was significantly faster than 3DTouch during the translation task ($F_{1,371} = 14.924, p = 0.000$).

Device distance

As shown in Figure 5, ARMouse device distance was higher during both tasks. This difference was significant both during the selection ($F_{1,371} = 98.676, p = 0.000$) and the translation task ($F_{1,371} = 173.430, p = 0.000$).

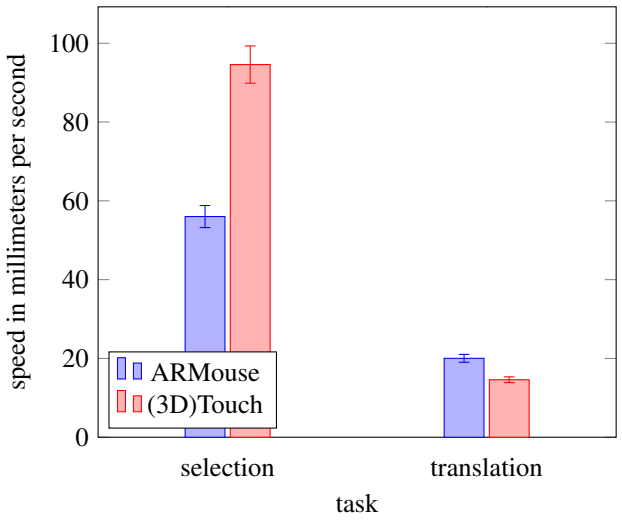

Figure 4. Average selection and translation speeds (higher is better, error bars: 95% confidence interval).

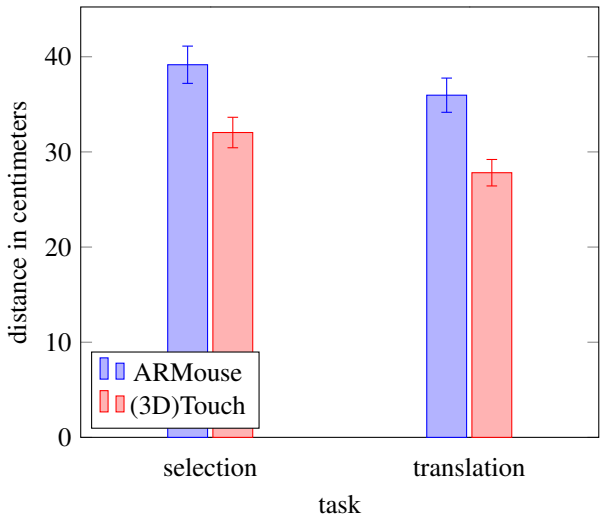

Figure 5. Average distance between the smartphone and the target object at the moment of selection/confirmation (error bars: 95% confidence interval).

Task load

NASA-TLX

Table 1 shows the average NASA-TLX scores. ARMouse scores lower than 3DTouch in every category. By conducting a two-factor ANOVA without replication (95% confidence interval) this difference is concluded to be significant ($F_{1,5} = 17.910, p = 0.008$), meaning that the perceived task load is lower when using ARMouse as compared to when using 3DTouch.

Qualitative analysis

Firstly, participants reported both interaction methods to be intuitive and comfortable to use. Other comments include the mouse to be more comfortable, likely even more noticeable during prolonged use. The mouse felt more precise and participants reported feeling more confident when selecting objects, especially due to the hover and secondary selection

	Mental demand	Physical demand	Temporal demand	Performance	Effort	Frustration
ARMouse	3.4	3.8	4.9	2.1	3.2	1.7
Touch	4	5.8	5.8	4.1	6	5.4

Table 1. Average NASA-TLX scores, on a scale of 1 to 10 (lower is better).

features. Furthermore, participants reported not to notice any input lag when using ARMouse and the transition line (as seen in Figure 1) to be an intuitive mechanic for switching the cursor between environments.Touch selection was reported to be simpler and faster. Regarding object manipulation, ARMouse was perceived to be faster, more precise and more comfortable. Participants tended to overshoot when using 3DTouch, resulting in having to readjust the position of the object multiple times. Participants also reported that the translation task became tedious halfway through the task when using 3DTouch.

Half of the participants reported increased immersion when using ARMouse: participants were more inclined to move the phone around the environment when using touch based interaction in order to keep the 3D illusion alive, explaining that otherwise the AR world felt two dimensional. In contrast, these participants reported this not to be an issue when using ARMouse. Two participants reported quickly developing the habit of moving the cursor and objects without pointing the smartphone at them during the translation task; translating the objects just by feel, allowing themselves to shift their gaze instead on the target position.

DISCUSSION

Results show that no significant difference was found between the selection accuracy of the interaction methods. Therefore, **H1** is rejected. This finding may be explained by touch based raycasting being more intuitive and the steeper learning curve of ARMouse, causing participants to get stuck in the secondary selection step, as a consequence they selected the wrong object and lost momentum, further increasing the observed speed difference during the selection task. It can be argued that making the spheres increasingly smaller could better differentiate the interaction methods' selection accuracy. ARMouse does show a significantly higher translation accuracy during the translation task, meaning hypothesis **H2** is accepted. This result could be explained by the mouse's high resolution and the stable surface on which it rests, as supported by prior work [11, 23]. It can be argued that the high level of proficiency participants already have with the device also contributed. The shortcoming of touch based translation as observed here may also be explained. Participants tended to overshoot their target when small movements were required, possibly as a result of subconsciously applying more force on the screen than necessary, causing their finger to have more inertia than anticipated.

As expected, touch based raycasting, most likely due to its simplicity and ease-of-use, proved significantly faster during the selection task. However, given the requirements of ARMouse the trade-off between selection speed and comfort is a trade-off worthwhile of making. During the translation task ARMouse was found to be significantly faster. Therefore, **H3** is accepted. Likely this is the case because initial positioning over the xz plane is fast and intuitive with ARMouse, after which the participant only has to move up, using the selection line as a guide. This results in fewer operations necessary to complete the task, when compared to 3DTouch. Moreover, the increased precision ARMouse offers prevents readjusting moves.

By collecting data on the distance between the smartphone and the target object, whether it be during selection or translation, the distance is found to be significantly lower during touch based interaction. Furthermore, mouse use elicited significantly lower NASA-TLX scores, especially lower perceived physical demand (Table 1). Adding to this the remarks of five participants stating they effectively performed all tasks with ARMouse without moving around as much as with touch based interaction, hypothesis **H4** is accepted.

An unexpected result was the higher immersion ARMouse elicited, as reported by half of the participants. This is also somewhat supported by the found larger device distance, meaning participants barely moved from their initial position (Figure 5). This opposes the findings of prior work. Georgiou et al compared mouse interaction to a tangible user interface and a gesture recognition system in AR, concluding the mouse to have the lowest immersion [8]. Therefore, it could be possible that mouse based AR interaction is situated somewhere in between hand gesture recognition and touch based interaction on the immersion spectrum. Another explanation, as supported by Choi et al, is that the particular implementation of ARMouse yields a higher level of usability, in turn contributing to the higher level of immersion [5]. Some participants explained the higher immersion to be due to the mouse being a physical object directly controlling a world-space virtual object: the cursor, or the spheres during translation. Perhaps solely the implementation of a world-space cursor instead of a screen-space cursor causes the increased immersion compared to other AR mouse interaction implementations. To summarize, it remains unclear why exactly a considerable number of participants experienced this, but future research may provide an answer. For example, the framework of context immersion in mobile AR may be used [12].

The evaluation of ARMouse shows it being able to elicit a remarkably high level of immersion with a seemingly unintuitive AR interaction device and a simple and relatively inexpensive setup. As Wang et al explains, the novelty factor of using a device like the Microsoft Hololens might introduce bias as participants are easily excited by devices breathing technological advancement [27]. By having participants use devices they were already accustomed to: the smartphone, mouse and laptop, such bias was avoided. Furthermore, this implementation showed no noticeable input lag and participants were able to use it effectively within a short period of time, problems often plaguing AR research prototypes.

Future work

Future research on ARMouse may include adding rotation and scale to the list of transformations and extend the functionality of the current implementation. In addition, given the three principles on which this interaction method is based (section 3), other AR enabled devices can also be supported. Head-Mounted Displays (HMDs), which are considered to be the most comfortable [6] and the best candidate for professional applications [13], can in theory easily be supported without violating any of the defined principles. Future research could explore these options. Note that the three principles allow for other configurations, potentially increasing the cursor's traversable space to more surfaces reachable by the cursor when upholding the first principle. It would be interesting to see whether the principles for mouse based AR interaction as defined here provide any guarantee for the success of different possible setups. This would also require further research.

Future work may also include a comparison between AR-Mouse and virtual mouse interaction to assess whether the theoretical advantage of the decoupling of perspectives proves to be an advantage in practice too. Potential directions may also include multiple users remotely interacting in the same AR space, similar to the system proposed by Hartmann et al [10].

Limitations

Limitations include the number of participants and ratio of male to female participants. While there is no indication of there being a difference between male and female participants, an equal distribution would have been preferred. In addition, device distance is measured at the moment of selection or confirmation. This variable would be more accurately measured if it were sampled more often.

To detect the position of the desk AR Foundation's plane detection algorithm was used. This resulted in occasional tracking issues due to close proximity of the smartphone and the desk. Image tracking was also considered, but AR Foundation's image tracking solution also proved suboptimal. Using a robust image tracking solution as offered by Vuforia [20] would have been preferable.

CONCLUSION

During this research the prototype for ARMouse is proposed: the foundation for a universal interaction paradigm for hybrid AR environments on desktops. ARMouse is found to outperform an established mobile AR interaction method in an object selection and object translation task, given the context of a desk-oriented setup. Furthermore, ARMouse may provide support for other AR enabled devices such as HMDs, providing interesting topics for future research. However, the most unexpected observation among these findings is ARMouse causing an increase in the level of immersion among participants. This observation combined with the decoupling of perspectives to better handle occlusions allows future users to confidently use ARMouse to interact with a hybrid AR desktop environment.

ACKNOWLEDGMENTS

The high level of support and feedback Wolfgang Hürst as supervisor and Lynda Hartman as second examiner provided is greatly appreciated, especially throughout the obstacles and delays encountered during the pandemic. Every participant's willingness to contribute to this study is also greatly appreciated.

REFERENCES

[1] NASA AMES. 2019. (2019). `https://humansystems.arc.nasa.gov/groups/tlx/`

[2] Ferran Argelaguet and Carlos Andujar. 2013. A survey of 3D object selection techniques for virtual environments. *Computers & Graphics* 37, 3 (2013), 121–136.

[3] François Bérard, Jessica Ip, Mitchel Benovoy, Dalia El-Shimy, Jeffrey R Blum, and Jeremy R Cooperstock. 2009. Did "Minority Report" get it wrong? Superiority of the mouse over 3D input devices in a 3D placement task. In *IFIP Conference on Human-Computer Interaction*. Springer, 400–414.

[4] Natalia Bogdan, Tovi Grossman, and George Fitzmaurice. 2014. HybridSpace: Integrating 3D freehand input and stereo viewing into traditional desktop applications. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 51–58.

[5] Hyoenah Choi, Youngwon Ryan Kim, and Gerard J Kim. 2019. Presence, Immersion and Usability of Mobile Augmented Reality. In *International Conference on Human-Computer Interaction*. Springer, 3–15.

[6] Luís Fernando de Souza Cardoso, Flávia Cristina Martins Queiroz Mariano, and Ezequiel Roberto Zorzal. 2020. A survey of industrial augmented reality. *Computers & Industrial Engineering* 139 (2020), 106159.

[7] Andreas Dünser, Julian Looser, Raphaël Grasset, Hartmut Seichter, and Mark Billinghurst. 2010. Evaluation of tangible user interfaces for desktop AR. In *2010 International Symposium on Ubiquitous Virtual Reality*. IEEE, 36–39.

[8] Yiannis Georgiou and Eleni A Kyza. 2017. The development and validation of the ARI questionnaire: An instrument for measuring immersion in location-based augmented reality settings. *International Journal of Human-Computer Studies* 98 (2017), 24–37.

[9] Jeffrey T Hansberger, Chao Peng, Shannon L Mathis, Vaidyanath Areyur Shanthakumar, Sarah C Meacham, Lizhou Cao, and Victoria R Blakely. 2017. Dispelling the gorilla arm syndrome: the viability of prolonged gesture interactions. In *International Conference on Virtual, Augmented and Mixed Reality*. Springer, 505–520.

[10] Björn Hartmann, Meredith Ringel Morris, Hrvoje Benko, and Andrew D Wilson. 2009. Augmenting interactive tables with mice & keyboards. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. 149–152.

[11] Todd Johnsgard. 1994. Fitts' Law with a virtual reality glove and a mouse: Effects of gain. In *Graphics interface*. Canadian Information Processing Society, 8–8.

[12] Mi Jeong Kim. 2013. A framework for context immersion in mobile augmented reality. *Automation in construction* 33 (2013), 79–85.

[13] Kiyoshi Kiyokawa. 2012. Trends and vision of head mounted display in augmented reality. In *2012 International Symposium on Ubiquitous Virtual Reality*. IEEE, 14–17.

[14] Maurice R Masliah and Paul Milgram. 2000. Measuring the allocation of control in a 6 degree-of-freedom docking experiment. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 25–32.

[15] Daniel Mendes, Fabio Marco Caputo, Andrea Giachetti, Alfredo Ferreira, and J Jorge. 2019. A survey on 3D virtual object manipulation: From the desktop to immersive virtual environments. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 21–45.

[16] Alexandre Millette and Michael J McGuffin. 2016. DualCAD: integrating augmented reality with a desktop GUI and smartphone interaction. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*. IEEE, 21–26.

[17] Annette Mossel, Benjamin Venditti, and Hannes Kaufmann. 2013. 3DTouch and HOMER-S: intuitive manipulation techniques for one-handed handheld augmented reality. In *Proceedings of the Virtual Reality International Conference: Laval Virtual*. 1–10.

[18] Mirror Networking. 2020. (2020). `https://mirror-networking.com/`

[19] Jeffrey S Pierce, Andrew S Forsberg, Matthew J Conway, Seung Hong, Robert C Zeleznik, and Mark R Mine. 1997. Image plane interaction techniques in 3D immersive environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*. 39–ff.

[20] PTC. 2020. (2020). `https://developer.vuforia.com/`

[21] Udo Schultheis, Jason Jerald, Fernando Toledo, Arun Yoganandan, and Paul Mlyniec. 2012. Comparison of a two-handed interface to a wand interface and a mouse interface for fundamental 3D tasks. In *2012 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 117–124.

[22] Ghazaleh Tanhaei, Lynda Hardman, and Wolfgang Huerst. 2019. DatAR: Your Brain, Your Data, On Your Desk-A Research Proposal. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. IEEE Computer Society, 138–1385.

[23] Robert J Teather and Wolfgang Stuerzlinger. 2008. Assessing the effects of orientation and device on (constrained) 3D movement techniques. In *2008 IEEE symposium on 3D user interfaces*. IEEE, 43–50.

[24] Unity Technologies. 2020a. (2020). `https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html`

[25] Unity Technologies. 2020b. (2020). `https://unity.com/`

[26] Guangyu Wang, Michael J McGuffin, François Bérard, and Jeremy R Cooperstock. 2011. Pop-up depth views for improving 3D target acquisition. In *Proceedings of Graphics Interface 2011*. Canadian Human-Computer Communications Society, 41–48.

[27] Xiyao Wang, Lonni Besançon, David Rousseau, Mickael Sereno, Mehdi Ammi, and Tobias Isenberg. 2020. Towards an Understanding of Augmented Reality Extensions for Existing 3D Data Analysis Tools. In *ACM Conference on Human Factors in Computing Systems*.

[28] Dennis Wolf, John J Dudley, and Per Ola Kristensson. 2018. Performance envelopes of in-air direct and smartwatch indirect control for head-mounted augmented reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 347–354.

APPENDIX

DATA

NASA Task Load Index

Participants filled out a NASA-TLX [1] form after each translation task. Table 2 shows the results for ARMouse. Table 3 shows the results for 3DTouch. Table 4 shows the two-factor ANOVA without replication results.

Objective data ANOVA results

Tables 5 through 10 show the results for the two-factor ANOVA with replication results. Note that these show the ANOVA results regarding the interaction methods, participants and the interaction between these two groups. In this study only the interaction method ANOVA results were used, but the other results may be useful for future studies.

Participant remarks

The major remarks made by each participant are listed below:

P1 **Selection task:** during ARMouse use I felt more relaxed and my posture was better, also I was able to more confidently select objects due to the hover function. Touch selection was easier and faster, but possibly also more fatiguing as I was moving the phone more. It was also more stressful and I did not feel as confident because I had no feedback.
Translation task: with ARMouse I was able to translate the object more precisely, I also felt more relaxed during the task, both physically and mentally. Using 3DTouch my performance suffered from overshoot and the task was fatiguing. I also think the movement was too constrained and tilting the phone to control the z axis was uncomfortable as I had to lean over, putting strain on my back.

P2 **Selection task:** regarding ARMouse it is pleasant to be able to rest my hand on the desk and selection with it feels more comfortable and less fatiguing. I also think the hover function would be advantageous for contextual data when hovering over objects in e.g. a 3D graph. If the object is out of screen I have to move both the mouse and the smartphone. Touch selection is more intuitive for a selection task and is easier for beginners.
Translation task: ARMouse feels more precise and also faster as I'am not making as many mistakes. There is less overshoot and when it does happen, it is easier to correct. I feel confident when translating objects using ARMouse. I noticed myself moving the object always over the desk first, and then upwards. I also kept the phone relatively still during the task when using ARMouse, which might reduce immersion, but adds comfort. translating objects using 3DTouch may be easier when moving them over longer distances, imprecise translation also felt easier with this interaction method. I was however constantly readjusting.
Other remarks: The transition line feels intuitive, the tasks were fun to do.

P3 **Selection task:** ARMouse feels more precise, especially when objects are close together because I can use the hover function. Due to controlling the cursor in AR I feel more in touch with the AR world; better immersion. Touch is faster and simpler
Translation task: translating objects using ARMouse feels simpler, faster, and more precise. Again, better immersion likely due to controlling an AR object directly with the mouse.
Other remarks: The concept of planes in AR for mouse movement is intuitive. The mouse was just as intuitive in AR as it is in 2D and due to my experience with the mouse, touch was not necessarily more intuitive.

P4 **Selection task:** Due to the selection line the position of the mouse is easily known, I have no trouble selecting the correct sphere without moving the phone, even when occlusions occur. Touch is faster as I so not have to physically move the pointer. Moving the phone itself when using touch does not feel intuitive yet, but might be advantageous during prolonged use. I think touch is better for this task although it may be less accurate.
Translation task: when using ARMouse, the selection line helps with aligning the objects. It also required less physical work as I did not have to move the phone around as much as with touch. Moving the object over a plane feels faster, more accurate and reduces frustration. When using 3DTouch I have to check each axis independently, this might result in a higher accuracy, although sometimes I forgot to check all axes.

P5 **Selection task:** I am already very adept with the mouse, so selection is incredibly fast. The secondary selection function allows for high precision. Touch is very intuitive, it's nice to be able to move around the space with the phone and simply tap it.
Translation task: dragging an object is more intuitive than moving over axes. When using ARMouse, I started dragging objects from off-screen, which felt very natural. It also was not strenuous and felt very fast. ARMouse felt immersive because even when I was not moving the phone, moving the mouse in 3D also moved the object in 3D. I get less disoriented when physical movement is matched to virtual movement (just like motion sickness). When using 3DTouch it was very clear to see where the object is moving towards as it is constrained to one axis. I needed to keep moving a lot to keep the feeling of 3D. I did have a very clear feeling if the overlap was perfect or not.

P6 **Selection task:** I already have much experience with the mouse so movement feels intuitive. Touch feels simpler and faster.
Translation task: ARMouse has a steeper learning curve, but the translation of objects felt more fluent as I did not have to move the phone around. Also movement and translation was more precise. when using 3DTouch I can't develop RSI. When doing this task for a full day I expect ARMouse to be the best ineraction method, especially when using an AR hmd for example.
Other remarks: I feel more connected to the AR world when using ARMouse, it feels like I am interacting with the AR world and not with a screen. I like that I was able to transition between virtual and AR spaces when

Participant number	Mental demand	Physical demand	Temporal demand	Performance	Effort	Frustration
1	2	3	2	2	2	1
2	2	2	4	2	3	2
3	3	5	3	2	3	1
4	2	3	6	1	4	3
5	4	2	8	2	4	2
6	7	7	4	1	2	1
7	3	1	7	8	3	1
8	1	5	3	1	2	2
9	6	6	5	1	4	1
10	4	4	7	1	5	3
Average	3.4	3.8	4.9	2.1	3.2	1.7

Table 2. ARMouse NASA-TLX scores.

Participant number	Mental demand	Physical demand	Temporal demand	Performance	Effort	Frustration
1	6	8	4	7	7	7
2	3	2	6	2	5	5
3	4	5	3	3	4	3
4	3	7	9	1	7	6
5	7	8	8	7	7	8
6	2	3	1	2	4	3
7	2	3	9	10	8	10
8	3	8	7	2	6	5
9	7	7	3	4	7	4
10	3	7	8	3	5	3
Average	4	5.8	5.8	4.1	6	5.4

Table 3. 3DTouch NASA-TLX scores.

using ARMouse, this seems beneficial when for example storing documents in the AR world.

P7 **Selection task:** the mouse is intuitive for manipulating digital spaces and offers more control than a fingertip on a screen. I felt more confident when using the mouse and selecting objects. When using touch I was more inclined to move around, also just tapping the screen is very simple and understandable, also moving the phone closer helped me make less mistakes.
Translation task: 3DTouch feels a bit easier to look around with, but is horrendous when translating objects, being forced to move one axis at a time is disheartening.
Other remarks: ARMouse offered free control of the object and works well as a tool. Furthermore, having multiple buttons is also very useful.

P8 **Selection task:** when using ARMouse the hover function helped in confidently selecting the correct object and moving around the scene is easier when you don't have to move around as much physically. When using touch zooming in on the object helps with confidently selecting it and I felt more inclined to explore. Touch feels more accurate but ARMouse feels faster.
Translation task: ARMouse feels faster and more accurate and I feel more in control when small movements are required. 3DTouch can be used with one hand.

P9 **Selection task:** due to the 2D feel of touch interaction I am less likely to look around and am therefore faster.
Translation task: ARMouse was faster and initial placement was easier. It was also more comfortable and had a higher precision. I also did not need to move the phone around as much to accurately place the object. When using 3DTouch I could more easily gauge the position of an object above the desk with the help of the visible axis lines.
Other remarks: Higher immersion when using ARMouse, because of this I was more inclined to look around and the interaction felt more interesting. I could also easily feel where the mouse was even though I was not looking at it. The transition line was intuitive and added to my immersion due to how it connects the two spaces. ARMouse is cool for now, but perhaps it is hard to imagine use cases today. However, I see potential for the future.

P10 **Selection task:** ARMouse felt intuitive, faster and more precise because mouse movement is unconstrained to movement of the smartphone. Subconsciously I used the secondary selection function already when the target object was heavily occluded. Touch selection requires only one hand and felt more familiar because I was handling a smartphone.
Translation task: when using ARMouse it was fast and intuitive to move the object horizontally first and then vertically. It was also more precise and had no overshoot. The visible axes of 3DTouch aided in accurate placement, but switching between x and z translation was sometimes frustrating.

ADDING OBJECT MANIPULATION FUNCTIONALITY

Rotation and scale transformations would form a likely addition to ARMouse in the future. Therefore, some thought was already given to how these manipulation types could be

Source of Variation	SS	df	MS	F	P-value	F crit
Interaction methods	12	1	12	17.91044776	0.008231882158	6.607890969
Category	7.446666667	5	1.489333333	2.222885572	0.2005985092	5.050329058
Error	3.35	5	0.67			
Total	22.79666667	11				

Table 4. Two-factor ANOVA without replication NASA-TLX results.

Source of Variation	SS	df	MS	F	P-value	F crit
Interaction methods	0.16	1	0.16	0.343697004	0.5580510395	3.866045953
participants	19.26	9	2.14	4.596947428	0.000008868384065	1.904537733
Interaction	11.59	9	1.287777778	2.766283525	0.00380147223	1.904537733
Within	176.9	380	0.4655263158			
Total	207.91	399				

Table 5. Two-factor ANOVA with replication selection accuracy results.

implemented. The (assumed) requirements for DatAR provide the context for these recommendations.

For quick and imprecise rotation a simple script can be used. This translates the mouse's x and y movement to rotation values:

```
transform.Rotate(new Vector3(Input.GetAxis("
    Mouse Y"), Input.GetAxis("Mouse X"), 0) *
     Time.deltaTime * rotationSpeed);
```

Uniform scaling can be added by mapping the mouse's y movement value to a scale factor. Dependent on the context, objects that have been scaled up can be repositioned to sit on top of the desk again to prevent them from clipping through.

Transitioning between manipulation types can for example be realized by using the middle mouse button, either via clicking or scrolling. The left mouse button can then be held down to perform the manipulation, keeping rotation and scale functionality consistent with translation. Feedback to the user on which manipulation type is currently in use can be realized by a brief (or persistent) dialog on the smartphone's display as the user switches manipulation types, or a more immersive type of user feedback would for example be a different cursor or selection line color based on the current manipulation type.

DOCUMENTATION

Note this documentation does not include evaluation and 3DTouch code.

Frameworks and Libraries

- Unity version 2019.4.0f1 LTS
- Packages
  - AR Foundation preview.3 4.0.0
    AR foundation is an abstraction layer on top of ARKit and ARCore built to be able to support both iOS and Android more easily.
  - AR Subsystems preview.3 4.0.0
  - ARCore XR Plugin preview.3 4.0.0
    Android's AR framework.
  - ARKit XR Plugin preview.3 4.0.0
    iOS' AR framework.
  - ARKit Face Tracking preview.3 4.0.0
- Assets
  - Mirror version 16.9.0
    Mirror handles client/server communication.
  - Low-Poly Simple Nature Pack version 1.1
  - LeanTween version 2.5.0
    Library used for linear interpolation of various gameobject properties.

Classes and Functions

Mouse Controller

Description

The mouse controller is where all mouse input is handled; objects are spawned, selected and manipulated; and all communication between the client and the server takes place (after initialization). This script is attached to the mouse prefab itself and becomes active as soon as the client/server connection is established.

Due to the use of the networking library Mirror, all server and client code is contained in this script. This also means that all code is automatically run by both the server and the client. As such, different keywords can be used to differentiate between client and server code:

- [Server]
  The server tag indicates that this function is only executed on the server.
- [Client]
  The client tag indicates that this function is only executed on the server.
- [Command]
  The command tag indicates that this function is executed on the server, but can only be called by the client.
- [ClientRpc]
  the clientRpc tag indicates that this function is executed on the client, but can only be called by the server.

The mouse controller changes operation based on the given context:

- Initialization: no mouse input is handled.
- Interaction: mouse position is updated and mouse clicks are handled.

Source of Variation	SS	df	MS	F	P-value	F crit
interaction methods	0.0003385507765	1	0.0003385507765	7.543321283	0.0063098277	3.866045953
Participants	0.007065865082	9	0.0007850961203	17.4928923	0	1.904537733
Interaction	0.0004137224652	9	0.0000459691628	1.024248615	0.4197678461	1.904537733
Within	0.01705472832	380	0.00004488086399			
Total	0.02487286664	399				

Table 6. Two-factor ANOVA with replication translation accuracy results.

Source of Variation	SS	df	MS	F	P-value	F crit
Interaction methods	0.1487811046	1	0.1487811046	77.2738994	0	3.866045953
Participants	0.04428788886	9	0.004920876539	2.555803842	0.007323952881	1.904537733
Interaction	0.04440951852	9	0.004934390946	2.562822952	0.007167133935	1.904537733
Within	0.7316418632	380	0.001925373324			
Total	0.9691203752	399				

Table 7. Two-factor ANOVA with replication selection speed results.

- Selection: multiple interactable objects can are hit in the hover function, so in this context the user can select the correct object.
- Manipulation: x and y interactable object manipulation.
- ManipulationZ: z interactable object manipulation.

Note that Mirror considers the mouse prefab to be the Player object, meaning that it represents the client and is automatically instantiated once the client/server connection is established. **Functions**

1. **Start**

```
void Start()
```

Description
Instantiates the mouse and its components, removes any gameobject that is tagged with the "serverOnly" tag on the client.

2. **Initialize**

```
public void Initialize(GameObject p,
    Vector3 pos)
```

Description
Because we do not want the cursor to be visible until a valid plane is designated the initialization of the mouse cursor is split up into the Start and Initialization function. Here, the cursor is actually made visible and placed on the plane.
Parameters
GameObject p: the plane gameobject.
Vector3 pos: the position on the plane where we want the cursor to spawn.

3. **SetCursorPos**

```
[Command]
public static extern bool SetCursorPos(
    int X, int Y);
```

Description
External function (Windows only) that sets the system cursor position, this function is called when the cursor transitions from AR space to virtual space.
Parameters
int x: x screen coordinate.
int y: y screen coordinate.
Returns
Whether the function was successful in placing the cursor.

4. **GetCursorPos**

```
[Command]
public static extern bool GetCursorPos(
    out Point pos);
```

Description
External function (Windows only) that gets the position of the system cursor, this function is called when the cursor transitions from virtual space to AR space.
Parameters
out Point pos: point containing the x and y screen coordinate of the mouse cursor.
Returns
Whether the function was successful in getting the cursor position.

5. **CmdDestroyObject**

```
[Command]
public void CmdDestroyObject(string name)
```

Description
Destroys gameobjects on the server and can only be called from the client.
Parameters
string name: the name of the gameobject.

6. **ServerCursorUpdate**

```
[Server]
void ServerCursorUpdate()
```

Description
This function takes the mouse input on the server and sends it to the client. Note that this function runs more frequently than the regular update methods.

7. **RpcSpawnObject**

Source of Variation	SS	df	MS	F	P-value	F crit
Interaction methods	0.002961668136	1	0.002961668136	14.92419065	0.0001315539692	3.866045953
Participants	0.01724576263	9	0.001916195848	9.655934039	0	1.904537733
Interaction	0.00229889295	9	0.00025543255	1.287154367	0.241995003	1.904537733
Within	0.07541004519	380	0.0001984474873			
Total	0.0979163689	399				

Table 8. Two-factor ANOVA with replication translation speed results.

Source of Variation	SS	df	MS	F	P-value	F crit
Interaction methods	0.5075438451	1	0.5075438451	98.65621771	0	3.866045953
Participants	1.308903839	9	0.1454337599	28.26936987	0	1.904537733
Interaction	0.9762518257	9	0.1084724251	21.08483688	0	1.904537733
Within	1.954936705	380	0.005144570275			
Total	4.747636215	399				

Table 9. Two-factor ANOVA with replication selection device distance results.

```
[ClientRpc]
public void RpcSpawnObject(string
    objectName, Vector3 pos, Vector3
    scale)
```

Description
This function can spawn objects by name, to spawn an object its prefab must be located in the resources folder.
Parameters
string objectName: the gameobject name.
Vector3 pos: the position at which the gameobject will be places.
Vector3 scale: the scale of the gameobject.

8. **RpcCursorUpdate**

```
[ClientRpc]
void RpcCursorUpdate(float x, float y)
```

Description
This function receives the mouse input from the server and updates the cursor Note that this function runs more frequently than the regular update methods.
Parameters
float x: the x component of the mouse input.
float y: the y component of the mouse input.
Note that these values are not coordinates, but directional vectors relative to the mouse position in the previous frame.

9. **FixedUpdate**

```
private void FixedUpdate()
```

Description
Updates the rest of the mouse controller at the standard frame rate.

10. **RpcMouseDownLeft**

```
[ClientRpc]
void RpcMouseDownLeft()
```

Description
Handles left mouse button holds. Used for object translation.

11. **RpcMouseUpLeft**

```
[ClientRpc]
void RpcMouseUpLeft()
```

Description
Called when the user has released the left mouse button, used for debouncing and object selection.

12. **RpcMouseDownRight**

```
[ClientRpc]
void RpcMouseDownRight()
```

Description
Handles right mouse button holds. Used for object translation.

13. **RpcMouseUpRight**

```
[ClientRpc]
void RpcMouseUpRight()
```

Description
Called when the user has released the right mouse button.

14. **ServerTransitionLineCollision**

```
[Server]
void ServerTransitionLineCollision()
```

Description
Called when the virtual cursor hits the bottom of the screen, records the x component of the cursor position and calls RpcEnableARMouse to transition the cursor to AR space.

15. **TransitionLineCollision**

```
[Client]
void TransitionLineCollision()
```

Description
Called when the AR cursor crosses the screen line, stores the x component of the cursor position and calls CmdDisableARMouse to transition the cursor to virtual space.

16. **PlaceSelectionLine**

Source of Variation	SS	df	MS	F	P-value	F crit
Interaction methods	0.6638356256	1	0.6638356256	173.430497	0	3.866045953
Participants	1.316254743	9	0.146250527	38.20870799	0	1.904537733
Interaction	0.5857385068	9	0.06508205631	17.00302444	0	1.904537733
Within	1.454516606	380	0.003827675278			
Total	4.020345481	399				

Table 10. Two-factor ANOVA with replication translation device distance results.

```
void PlaceSelectionLine(Vector3 begin,
    Vector3 end)
```

Description
Positions the selection line to span between two points.
Parameters
Vector3 begin: begin point of the selection line
Vector3 end: end point of the selection line

17. **CmdDisableARMouse**

```
[Command]
void CmdDisableARMouse(float mouseX)
```

Description
Server command that enables the virtual cursor and positions it at the designated x position at the bottom of the screen.
Parameters
float mouseX: the normalized x component of the cursor position.

18. **RpcEnableARMouse**

```
[ClientRpc]
void RpcEnableARMouse(float mouseX)
```

Description
Enables the AR cursor and positions it at the screen line according to the normalized x component of the virtual cursor position.
Parameters
float mouseX: the normalized x component of the cursor position.

19. **HoverCheck**

```
[Client]
void HoverCheck(bool click = false)
```

Description
Performs raycasts from the cursor upwards to set hover and selection states of interactable objects, also transitions to the selection context if necessary.
Parameters
bool click: true if the left mouse button is pressed during this frame to set interactable object states from hovered to selected.

20. **RpcDestroyAllObjects**

```
[ClientRpc]
public void RpcDestroyAllObjects()
```

Description
Destroys all gameobjects with the "SpawnedObject tag".

21. **CmdLog**

```
[Command]
public void CmdLog(string s)
```

Description
Debug function allowing you to send strings to the server's command line from the client.

SetupHelper
Description The setup helper is attached to the AR Session Origin and handles the instantiation of the mouse, mouse plane and the screen line.
Functions

1. **Start**

```
void Start()
```

Description
Initialized the setup helper and the necessary managers.

2. **Awake**

```
private void Awake()
```

Description
Initializes other managers.

3. **OnEnable**

```
private void OnEnable()
```

Description
Here we can subscribe to manager's events.

4. **OnDisable**

```
private void OnDisable()
```

Description
Here we can unsubscribe to manager's events.

5. **OnImageChanged**

```
public void OnImageChanged(
    ARTrackedImagesChangedEventArgs args)
```

Description
Called when images tracked by the tracked image manager are updated, added or removed. As such, this is where the screen line is instantiated.
Parameters

ARTrackedImagesChangedEventArgs args: all data concerning the tracked images, provided by the tracked image manager.

6. **TryGetTouchPosition**

```
bool TryGetTouchPosition(out Vector2
    touchPos)
```

Description
Gets the touch position in screen coordinates.
Parameters
out Vector2 touchPos: the touch position in x and y screen coordinates.
Returns
Whether the registration of the touch was successful.

7. **Update**

```
void Update()
```

Description
Primes the mouse controller once the cursor is placed and the screen line is instantiated.

NetworkAutomation
Description

Since the system only has one server and one client the connection process can be automated. With hardware defines there can be distinguished between the client and the server before this relationship is established by Mirror. If the code is run in the Unity editor, on a Windows machine, or on a Mac the server is started. If the code is run on any other device (Android or iOS) the system will search for a server.
Functions

1. **OnValidate**

```
private void OnValidate()
```

Description
Sets the network discovery component when running in the editor.

2. **Start**

```
void Start()
```

Description
Starts the server on Mac and PC.

3. **Awake**

```
void Awake()
```

Description
Starts server discovery on Android and iOS.

4. **OnDiscoveredServer**

```
public void OnDiscoveredServer(
    ServerResponse info)
```

Description
Reports discovered servers. Starts client/server connection with the first server that it finds.
Parameters
ServerResponse info: properties of the newly discovered server.

UIManager
Description

Handles UI events and calls corresponding mouse controller methods.
Functions

1. **ObjectSpawnButton**

```
public void ObjectSpawnButton(GameObject
    obj)
```

Description
Can be hooked up to buttons to spawn objects.
Parameters
Gameobject obj: reference to the gameobject.

2. **SelectionTest**

```
public void SelectionTest(GameObject obj)
```

Description
Spawns a series of interactable spheres to test selection and manipulation.
Parameters
Gameobject obj: reference to the gameobject.

3. **DestroyAllObjects**

```
public void DestroyAllObjects()
```

Description
Destroys all objects with the tag "SpawnedObject" by calling the equivalent method in the mouse controller.

InteractableObject
Description

Any gameobject that is be interactable (can be selected and manipulated) has this script attached to it. Here, selection and hover effects are generated and collision with the ceiling and mouse plane is handled.
Functions

1. **Start**

```
void Start()
```

Description
Initializes colors and reference to the mouse controller.

2. **LateUpdate**

```
void LateUpdate()
```

Description
Animates changes in color based on the state of the gameobject.

3. **CheckPlaneIntersection**

```
public void CheckPlaneIntersection()
```

Description
Handles collisions with the ceiling (50cm above the mouse plane) and the mouse plane. Differentiates between objects with a center and bottom pivot.

4. **SetHoverDone**

```
void SetHoverDone()
```

Description
Sets hover boolean to true when the hover animation completes.

5. **SetHoverNotDone**

```
void SetHoverNotDone()
```

Description
Sets hover boolean to false when the de-hover animation completes.

6. **SetSelectDone**

```
void SetSelectDone()
```

Description
Sets selection boolean to true when the selection animation completes.

7. **SetSelectNotDone**

```
void SetSelectNotDone()
```

Description
Sets selection boolean to true when the deselect animation completes.

8. **SetColor**

```
void SetColor(Color c)
```

Description
Sets the color of the outline shader to the correct value as the animation plays.

Settings
Description

Scriptable object that allows for easy editing of global values and variables such as hover and selection color.

Known Issues

When starting the app the smartphone must be in line with the z axis to set the world orientation.