iControl

Project Report by
Dirk Widmann & Halin Kim

Winter Semester 2019 / 2020

Interaction Engineering

Prof. Michael Kipp

Augsburg University of Applied Sciences

Abstract

For this research study, our aim was to find out how well micro-gestures work in combination with eye tracking when used for aiming at and interacting with virtual objects on a screen and how well the eye tracking input format can emulate the tasks of a mouse when tackling simple interactive tasks. We developed two simple interfaces for each condition with the interactions being to drag, rotate and scale a simple geometry on screen. To find out how well the new gestural interface performs in terms of intuitivity and ease-of-use, a small user test was conducted participants, who were inexperienced with eye-tracking technology. The evaluation of this user test resulted in a clear preference in favor of the traditional mouse interface, while showing potential for the gestural alternative. This was derived from a strong learning factor visible in our data and the statements of our participants, that a more fleshed out technology and prototype would be very appealing to learn and replace a traditional mouse with.

Our Motivation

We can manipulate motion digitally by moving the on-screen pointer and directing task. There are already various kinds of systems in the world, including remote controls, touch-pads on laptops, and mouse. And most manipulators are the form of holding something by hand and moving. We wondered how to manipulate it in a new and more creative way. Ordinary people use a basic PC mouse the most when they use a computer. However, to use the mouse to activate the pointer on the screen, you must look at the monitor, check the desired position, and then move the pointer there by moving the mouse. Also, if you use a mouse for a long time, you will feel wrist tunnel syndrome, and your arms will move more frequently, which will quickly make you feel tired. When using a mouse, it is being done simultaneously to achieve its purpose in two separate ways, looking at the monitor and moving the mouse using the arm. So we tried to separate the two processes. Use the eyes simply to move the pointer on the screen and use the hand to direct the pointer's motion. In other words, move the pointer through eye-gaze tracking to select the object and perform tasks such as drag, rotation, and scaling using simple gestures. This method can reduce the time it takes to move the pointer and is expected to compensate for the disadvantages of using the mouse. Also, unlike other studies that use only one interaction, the synergies that the synthesis of two interaction techniques brings will be positive and attractive. We proceeded with a project to compare whether there was a possibility of replacing the mouse interface with a combination of eye-gaze tracking and gestures. It is expected that the new 'iControl' interface will be more useful than the familiar and comfortable mouse interface.

Related Work

HTTPS://PRECISIONGAZEMOUSE.ORG

This open source project lies as a base for this study to continue upon. The project uses eye- and head tracking to give the mouse cursor a specific location. However it still uses switches on a keyboard or mouse to activate the uses of the mouse buttons. The iControl-Interface gets rid of this physical interaction through gestural- and clutch-input realized through hand tracking, which allows the user to be away from any traditional computer interfaces.

HTTP://WS.IAT.SFU.CA/PAPERS/SINGLEHANDMICROGESTUREELICITATION.PDF

In this research study, the research team created a set of finger-scale micro-gestures, that are user-focused without a specific interface in mind, which allows for a more intuitive experience.
When constructing the iControl-Interface, we made a few tests to incorporate some of the gestures mentioned in this paper, especially because of the found intuitivity the paper has found out for them. Unfortunately, using the Leap Motion Hand Tracking Technology, there were problems with the tracking of two finger tips, that are touching each other. Even though this was supposed to be the fundamental motion for clicking and dragging, we had to go with a different approach using simpler hand positions.

HTTP://WWW.PROAKADEMIA.EU/GFX/BAZA_WIEDZY/428/NR_23_35_42_2.PDF

The Research Group in this paper evaluated different gestures that are shipped with the Leap Motion Hand Tracking System. We intended to implement our own gestures to improve user-experience and intuitivity.
Even though our best-case scenario didn't work out, we ended up on a few hand gestures and positions that translate the wanted effect quite well to the real world, for example rotating the hand to rotate a virtual object or varying the distance to the table to scale the size of the object.

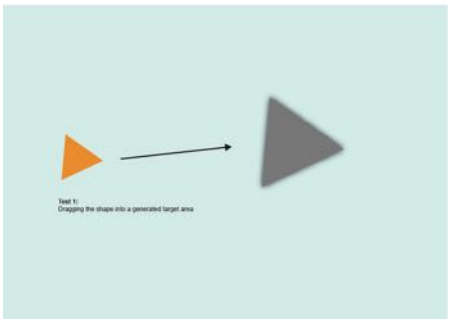
Concept

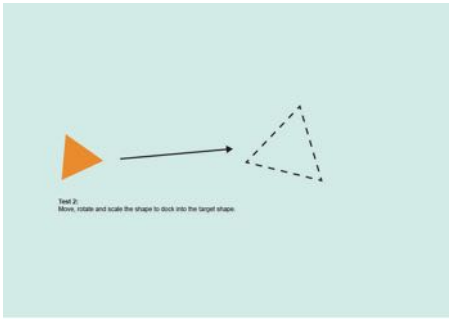
The conceptual baseline for this project derived from an idea for a mini-game, that would use gaze to aim on targets and corresponding hand gestures to fire on different targets. While the idea itself is certainly fun to explore, we noticed that before any practical project with these two input-interfaces are developed, it would be quite interesting to test the interaction layout in its purest form and compare it to the standard used interface in this role - the computer mouse. The experiment is to see how good an interface performs that takes the physical component out of mouse-like interaction. This has the instant danger of losing some of the haptic components, which a mouse provides, for example the little click you feel when clicking or the rotation-ticks you feel when rolling the scrolling-wheel. This has to be compensated through for example visual feedback, that signals to the user when a gesture is completed correctly, or when an object is selected through gaze. Another possible disadvantage could be the physical effort of holding a hand up for the requested gestures instead of resting the hand on the mouse. We think however, that placing the gesture tracker in a similar position as the mouse will enable the user to comfortably rest his arm on the table to perform the different interaction steps.

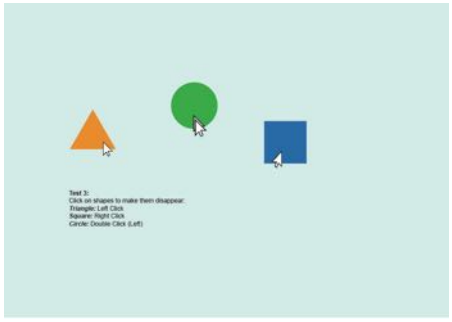
To further increase the comfort for the user when performing these everyday tasks, our first strategy was to use hand-scale micro-gestures to emulate the functions of the mouse. This would have had the advantage of not having to move the hand around in 3D-space above the hand tracker. The micro-gestures we were thinking of were published in the paper that is referenced in the related work section. The special aspect of these gestures are the filigree movements between the single fingers, which are meant to trigger mouse-like functions. For example scaling is translated to the hand by moving the tip of the thumb along the index finger - creating a certain range that can be mapped to digital elements.

Combined with the gaze coordinates acting as a mouse pointer, a (in theory) quick and easy to use interface is created. Unfortunately we weren't able to use the micro-gestures from the paper and had to move to gestures of slightly larger scale to guarantee solid gesture recognition through our gesture tracker, the Leap Motion. These new gestures take away from the thought-after effortless nature, because the user has to move his hand through 3D-space above the Leap Motion. However, the gestures are much simpler to learn and understand and less prone to errors because of their less filigree movement patterns.

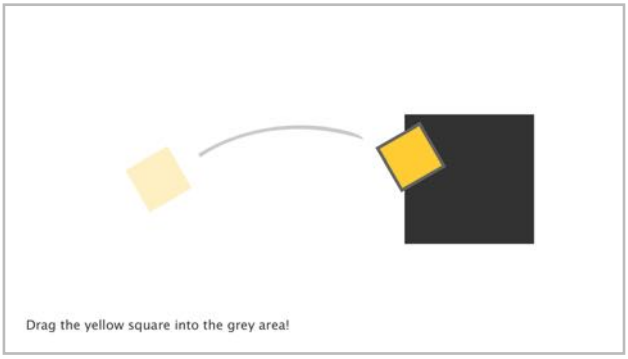
The first sketched interaction idea was a distribution of the mouse function into three tasks the user would have to go through: Movement of an object (translation through dragging), Orientation of an object through rotating and scaling and the selection of different objects through different types of mouse click.

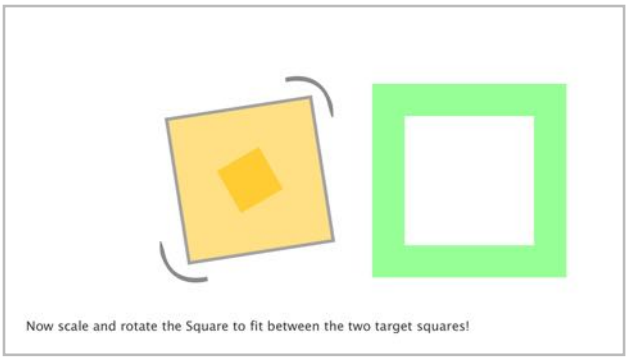






This concept was based on the idea to cover every input type a computer mouse has to offer, to establish the new interface as a whole package to replace the usage of a mouse. The plan was here to test the new interface on its own and analyze the performance of the own prototype through subjective judgment of participants of a user test. We quickly realized that implementing this many functions in one prototype could not only overwhelm the participant, but also give us some difficulties to evaluate our findings accordingly, because of too many too different data sets that would be mostly based on subjective experience without a control condition to compare it to.
These findings led us to two clear goals: less complexity for the user and some kind of data based way of evaluating our user test.
We reduced the number of tasks to just two, both being simple Fitt's law tests. The first task consists of the user dragging an object to a designated area, the second task challenges the user to rotate and scale this object into a given orientation and size.





As is usual in Fitt's law studies, we implemented a function that measures the time it takes the user to complete each task. The timer for the first task starts when the user first hovers over the object and initiates a click and runs until the object is released from the dragging motion and rests in the designated area. The calculation is similar for the second task: The timer starts when triggers the input for rotating or scaling for the first time and stops automatically when the given size and orientation is reached. The automatic generation of this time data gives us a dataset, which gives a good overview over the performance of our interface prototype. We also introduced an identical interface for the mouse as a control-condition. Every participant of our user test was required to do the same tasks ten times on both interfaces to achieve an even set of data. This also

gives the participants themselves the ability to give comparative feedback on the gestural prototype, which would have been more difficult to do just from memory.
To give the user test a little dynamic aspect, we randomized the position of both the interactive object and the target area on each try. This had the effect that the participants couldn't rely on muscle memory while the test went on. However, the position of the target area was always the same relative to the intractable rectangle to keep the Fitt's law study fair and comparable. So in theory it would have been possible to do the same motion from the generation point of the rectangle each time, fortunately this wasn't an issue.

Implementation

For the final implementation of our iControl prototype, we used the Processing framework because of its quick and barrier-free approach to prototyping. Processing itself is an open-source programming environment, which uses a simplified form of Java to create a quick way to sketch generative and function based visualizations on a digital canvas. A big strength is the open-source community around this project, which provides a large amount of libraries that work lots of interactive hardware- and software-interfaces into the environment. These libraries include for example a connection to the Microsoft Kinect or even OpenCV, while native support for mouse- or touch interactions are already given. There is also a web-pathway that is derived from the processing community, called P5.js. This Javascript library allows a similar working environment for quick canvas-based prototyping to create similar tools but embedded in a website, which makes it very powerful for online distribution. Ultimately we chose Processing because of the support for both of our input interfaces and our previous experience with the environment.

The landscape for eye-tracking on the consumer-market is currently led on by the swedish company Tobii. While manufacturing powerful options for enterprise use, there is also a low-budget option, originally intended for use with gaming, called Tobii 4C. It is very lightweight and plugs in over just one USB-Port, which allows for a quick and easy setup, even when on the go. Tobii provides their own software, which is needed to make the tracker work with windows. This also provides some features for everyday use, for example teleporting the mouse cursor to the current gaze position, or selecting a Windows-Tab by looking at it.
The corresponding Processing Library (*http://hci.soc.napier.ac.uk/GazeTrack/*) has the great functionality to give the continuous position of the gaze of the user in 2D Pixel coordinates, which are calibrated to the used monitor. To enable the mapping of your eye-position to the pixel-position on screen, a calibration program has to be executed through the Tobii-Experience Software. With the gaze coordinates, the processing code can freely check, whether the user is looking at an area of interest or a specific object. This is of course combinable with all the other functionalities of Processing, which makes it easy to use gaze as an input, clutch or trigger for all kinds of interactions.

The gaze implementation for our current prototype is held quite simply. Just like a mouse cursor, the eye-focus of the user is used to select or deselect, what object you want to manipulate. So looking at a rectangle has the same use for us as hovering over the rectangle with the mouse. We check this in the code by comparing our streamed X- and Y-position of the gaze to the X- and Y-position. If the gaze position has is near enough to the rectangle by a certain threshold, the rectangle is selected and ready to be interacted with.

```
if (gazeTrack.gazePresent())
{
  latest_gaze_x = gazeTrack.getGazeX();
  latest_gaze_y = gazeTrack.getGazeY();

  stickDistance = dist(latest_gaze_x, latest_gaze_y, rectPosX, rectPosY);
}

if (stickDistance < rectScaleFactor) {
  selected = true;
}
```

While the user is just looking around on screen, the gaze position is not displayed graphically, because it was noticed while testing, that some people would follow the dot their eyes were generating. In use, this means that the user sees, which object is selected through a thick outline, which appears only in that case.
Another implemented gaze-based interaction is the translation of selected objects. While the clutch is activated through hand-tracking and the user is looking at the rectangle, its position snaps to the coordinates of the user's gaze. This leads to a rapid method of translation, because the rectangle is moving to the targeted location as fast as the eyes can move. This made us realize that the rapid, jittery movements, which the eye-tracker detects, can be quite irritating for the user, when he just wants to move his rectangle from point A to point B. Our current workaround is a simple Low-Pass Filter, which offsets the current gaze position with the filtered gaze position of the last frame. This leads to a smoothed movement.

```
if (stick == true)
{
  rectPosX = latest_gaze_x;
  rectPosY = latest_gaze_y;
}

filtered_x = rectPosX * filterfactor + filtered_x * (1 - filterfactor);
filtered_y = rectPosY * filterfactor + filtered_y * (1 - filterfactor);

translate(filtered_x, filtered_y);
```

The danger with this kind of filtering is some delay in the movement if the value from the last frame is weighed too heavily. This is why we settled on a rather low smoothing-factor (0.75 in this function) to keep the object fixated on the user's gaze position.

Tracing the movement of the human body has been tackled by a handful of devices, camera frameworks and even machine learning algorithms. While software options like OpenCV or OpenPose are very practical to detect the body of a user, we were looking to use something that could be used in the same scenario as a computer mouse - ideally between desk and hand. This is why we chose the Leap Motion System to track the hand and fingers of our users. This tracking device has the size, use area and tracking capabilities we were looking for. As with the Eye-Tracker, it was needed to install a SDK-Package to prepare Windows for the functionalities and properties of the Leap Motion sensor. The device itself shipped with some software to get into the topic, for example some visualizations to see the tracked hands, or some small virtual playgrounds to test the tracking with some 3D-objects. The Leap Motion API and the vast support for different Frameworks makes it easy to get to the actual tracking numbers you would need for your interactive application. For the Processing environment we were setting up, we were using an outdated library from the agency "onformative", that gives really basic info about the finger position and some prefabricated gestures like a circling or swipe motion with the index finger. Unfortunately we noticed too late that there was a newer, more comprehensive library available (by Darius Morawiec), which would have also given some information about arm- and hand-position and -orientation. Nevertheless we were able to calculate all our needed data from the position of the fingertips, although it might have taken a little more effort than needed.

The three hand-tracking-features implemented in our prototype are scaling an object, rotating an object in size and initiating a clutch-mechanism to translate an object with gaze-input. All of these functions are realized by analyzing the position of the individual finger tips. The Leap Motion gives out fingertip position relative to the device you place on the table. The axis are defined along the length of the device (X-axis), across the length of the device (Z-axis) and orthogonal of the device (Y-axis). Each fingertip brings a continuous stream of values along every axis to define a position in 3D-space.
These defined positions help us determine what the user is currently doing with his hand. To realize the scaling-function, we calculated the average height (Y-axis) of all the fingers of the hand and compared it to the height of the hand when the gesture started, to guarantee scaling, which is relative to the original scale of the rectangle. Through this, we prevent big unwanted jumps while scaling the object, which could occur when implementation of the tracking is absolute.
The clutch-mechanism for object translation is engaged if the user sets his hand in a vertical position. To check this behavior by code, we compared the X-axis position of each finger to the average X-axis position of all fingers of the hand. The clutch is only activated as long as all finger-X-positions are under a certain distance to the average X-position, which practically means that the fingers have to line up when viewed from above. To rotate the object, the user is supposed to do a rotary motion with his hand around the axis of the arm. The base idea is very similar: We calculate an average position of all tracked fingers and compare it to the position of just one finger, which should ideally rotate around the middle point (=average) of the hand, while performing the gesture. We then calculate a vector between this middle point and the designated finger and measure the angle of this vector to a normalized "up-vector" to calculate the rotation angle of the whole hand. Just like the scaling gesture, this angle is applied to the rectangle in relative fashion to avoid unwanted jumps in rotation.

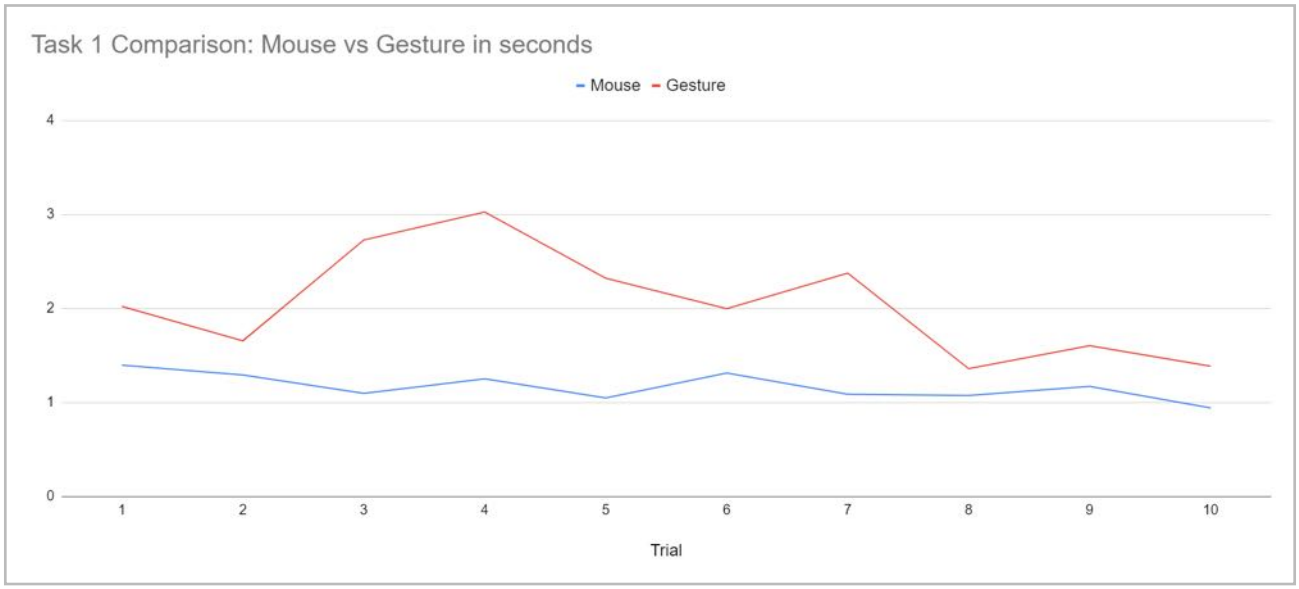
The two interactive input-techniques are both essential for the implemented functions. To realize the functions as planned, the data from both have to work together. A base-

line step was to let the hand-gestures only have effect while the user is looking at the object. This led to many if-clauses before every gestural interaction step, to check if the distance between the gaze coordinates and the rectangle is small enough. Vice versa, the vertical hand position is needed to initiate translation through gaze, which means that similar status checks have to happen on both sides continuously. This is mostly done through booleans, that turn to "true" as long as the wanted status is active. These kind of logic-steps were also important to automatically measure the time it took the participants of our user test to complete the tasks. This was done through object-based proximity checks, as well as controls to see whether the user is still interacting or not. The functions were also slightly adjusted depending on how far the user has come through the tasks. For example when both tasks were completed, any interaction was disabled until the sketch was reset for the next trial.

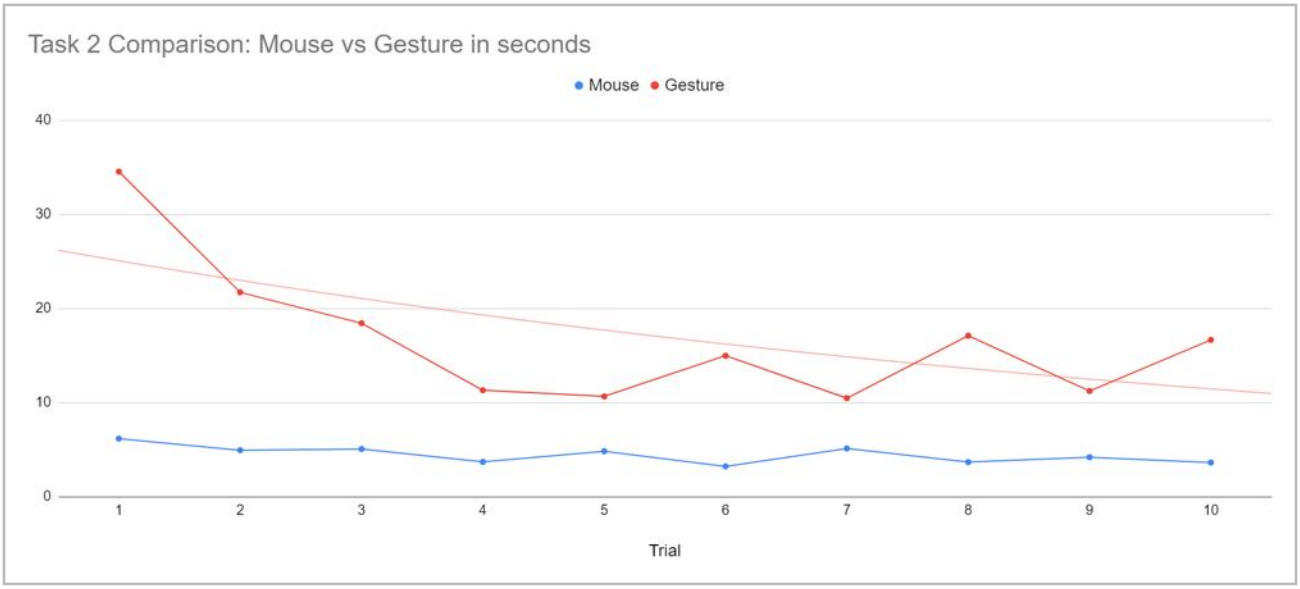
Evaluation

To see how the two interfaces perform against one another, both Processing sketches were coded to generate data output automatically. After the test we were left with two numeric values for each try (each task one value), for ten tries times the two interfaces times seven participants. All of these numbers were summarized in an excel-sheet for evaluation.

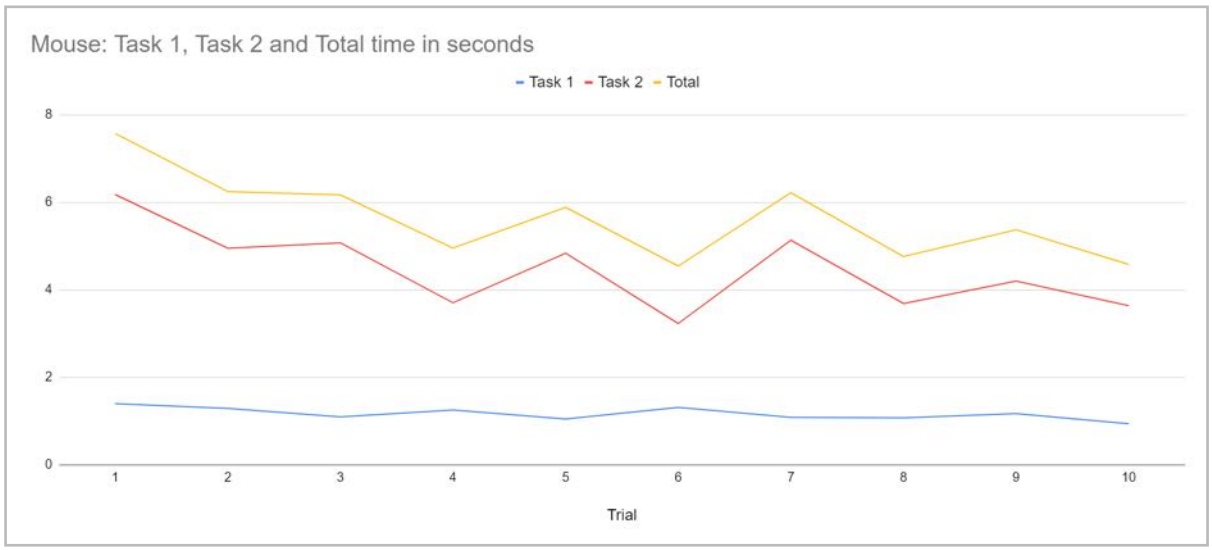
First they were placed into the tables without a clear strategy in mind, however it was advised to calculate the averages of each try and not of each participant, because it wasn't really important to see the average times of each user but to see an average evolution of the Task times over the ten tries. First of all we can safely take away that the mouse interface is significantly more efficient than the gestural interface - the overall average completion time for both tasks on the mouse variant is 5.63s, while the gestural average is 18.78. So let's look where this discrepancy came from. First, let's look at the interfaces in comparison for each task.

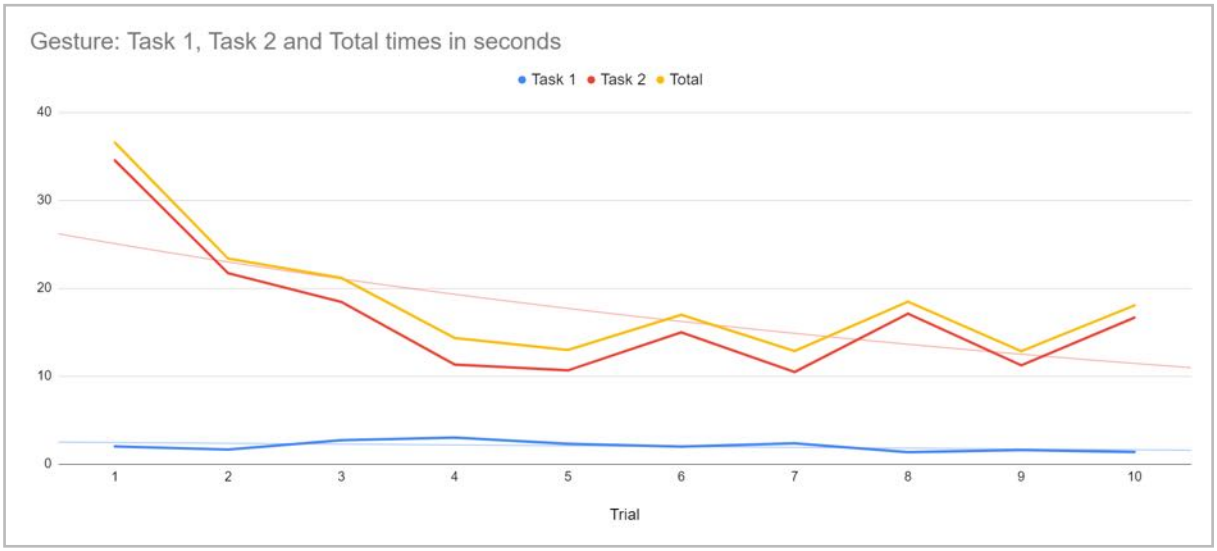


While the graphs for the times for the dragging task show some clear differences in amplitude, the overall distance between the lines fluctuates quite strongly between showing almost the same time like in try 2 or 8 and being separated by a factor of over 2 like in try 3 and 4. The mouse-interface values seem to be very stable and low, as we expected from the control condition, because of the great experience our participants all have with dragging something with the mouse. On the other hand lie the almost arbitrary values of the gestural alternative, which show that the participants were not used to this kind of interaction and had some problems controlling it consistently. Interestingly, the graph shows that when operated properly, the gestural interface almost rivals the mouse interface in time when it comes to dragging an object from point A to point B.



Task 2 required the user to both rotate and scale a virtual rectangle to fit a designated orientation shown on screen. Here the differences are even more significant. Even on the try where the gestural interface performed best and the mouse performed second-to-worst (try 7: 10.49s to 5.14), the gestural completion time doubles the mouse counterpart. These differences are at their most extreme on try 1, where the participants took an average of 34.56 seconds to complete the task. It makes sense for the very first try to be the worst because every participant had to deal with a new interface and its rough edges. A big takeaway from this graphic was for us the quick descent, which the graph for the gestural times shows. We think that this shows how quick our participants were getting used to the new interface and that with further practice it could have some potential for real life uses. Let's take a look at both interfaces for themselves.





It is clearly visible that Task 2 was more challenging for our participants in comparison to the first one. Not only was the second task pretty much two tasks in one, but it confronted the users with two new functions at the same time, which they had to get used to first - even in the mouse variant. Looking at the graphs it could also be said that this prompted a similar learning curve for both interfaces on Task 2. Overall, the participants had most of their problems with this second task of the gestural interface, which is also where the huge difference in overall average completion time comes from. Going into the test, we expected the gestural interface to do worse than its mouse counterpart and the unwanted inputs and filigree movements the interface required were too much to really compete with the mouse. Another anomaly in our data are the wavy patterns over the second half of the tries. Over the course of the test, our users seem to have an alternating harder and easier time back to back to complete Task 2. As this is the case for both the mouse and the gestural prototype, we believe that this is to blame on the low user number of 7 participants and their rather random rates of success and speed while using the interfaces.

While the user test went on, we watched closely on how the users themselves react to their given circumstances. One large factor was definitely the technical affinity of the user to understand the interfaces of the Leap Motion and the Eye-Tracker. Understanding how the hand tracking mechanism works, gave some users the advantage of knowing how to position their hand relative to the tracking device. For other participants it was some luck and some bad luck that led to the large range of completion times.
Since all of our participants were inexperienced with gaze-controlled interaction, this aspect of the test was quite a hindrance for some. Intuitively, they tried to use more natural bodily movements to try and translate the rectangle on screen, like changing the head orientation and position relative to the eye-tracker or moving the hand along the X- and Z-Axis of the Leap Motion to control the object through hand position. We also noticed some signs of exhaustion in our users. This came in different ways:
Some of them remarked, that holding the hand over the table through the duration of the test was exhausting, others said how having to concentrate on looking at the right place for the gaze tracking all the time was quite tiring, too. Another common problem while interacting with the prototype were false positives while rotating the rectangle on screen. Because of the recognition of both the rotary motion and the vertical hand position was calculated based on the average finger position of the hand, there were some

instances where a vertical hand position was detected by the prototype, even though the user was just rotating the object. The then triggered unexpected translation led to frustration for some users, as the rectangle was now at a wrong spot and they had to re-position it again before continuing with the orientation task. Another source of setbacks was the initiation of the vertical hand gesture, because in order to get to this position, you have to rotate your hand into it, which leads to a slight rotation being recorded as rotary gesture to rotate the rectangle. In theory, some of these issues could have been resolved by deselecting the rectangle (by looking away), but this was
- understandably - almost never used.

During the user test, we were able to identify various types of behavior shown to users. Users familiar with the use of the mouse tried to move their hands as if they were moving the mouse while making gestures with their hands. Users also wanted to move objects by moving their heads because they felt awkward at first eye-gaze tracking. However, these observations gradually decreased over the course of 10 tests. This also means that users adapt to eye-gaze tracking and gesture systems.
We could hear the feeling of testing from users through simple questions after the test. These are the questions of the test:

1. How did you feel about the iControl interface?
2. Which do you like better? Why?
3. If you have to do a lot of things with it, which is better?
4. If you have to do an elaborate job with it, which is better?
5. Would you like to use the iControl interface if it could replace the mouse interface?
6. Do you have any idea about this interface?

Most people gave positive answers such as ‚fun and new' to the iControl interface. But there were also many negative opinions, such as 'arm is very tired' and 'eye-gaze tracking is not well recognized' This is because our prototypes do not have armrests, so users have to make gestures in the air while testing. Also, because the angle of the Tobii eye tracking device was fixed during the test, it was not recognized well according to the user's eye level. The iControl interface needs to be able to consider the user's eye height and the installation of an armrest.
Most people chose the existing mouse interface as their preferred interface. Because it is faster and more comfortable for using the more familiar mouse interface. They also said that poor eye-gaze tracking and fatigue in the arms are the reasons. Users typically chose the mouse interface as the system they would use to do more elaborated or huge amounts of work. This is also because the mouse interface can move faster, more comfortably and finer. Using the Tobii eye tracker can result in errors of 1 to 4 cm between where the actual field of gaze is staring and where the machine is being captured. Therefore, it still seems difficult to make delicate and minute movements with eyes. However, for visual tasks such as 3D, some users said it would be easier and more comfortable to manipulate the task with a eye-gaze tracking and gesture interface, while others said using eye-gaze tracking and gesture interface when using a large monitor would be faster and easier to operate. Or there was another opinion that it would be highly utilized in places where hands are dirty or difficult to use, such as factories.

While developing the gestural interface, it was clear that to use the interface perfectly, you have to get to know all the interaction functions and practice them. A more developed prototype that is well calibrated to every user for an audience that knows the

interaction types and has practiced them a bit will most likely have a chance to achieve similar usage times. It is also important to further look for improvements for the gestures themselves, especially while problems with the detection of false positives are still at hand. From a developing standpoint, we see most potential in this kind of interface, when the variants of usable gestures exceed those of a mouse, theoretically making it a more versatile and powerful tool to interact with objects on screen.

Conclusion

Overall, the iControl prototype and the testing process gave us a lot to learn about how new types of interaction are picked up and how important it is to put lots of effort into developing interactions that are easily understandable and easy to use. It also taught us how much of a difference large amounts of experience make when tackling interactive interfaces, especially when it comes to strange input types like gaze-tracking.
We think that it would be best for the project to try out even more gestures for this kind of interaction and to maybe test how different gesture layouts perform compared to each other and compared to a control condition like the mouse. As mentioned before, the gestural aspect has the big strength of not being limited to a certain number of buttons, the motions of a hand are versatile enough to generate a large catalogue of gestures for different types of interaction. Another chance regarding this combination of eye- and gesture-tracking would be to go into the third dimension, since the hand is tracked in 3D-space anyway. This would allow for new interesting combinations to manipulate 3D-objects through hand position and orientation.

According to the user test, the prototype of iControl interface was generally less attractive to users than to mouse systems in practical terms. Because of fatigue, speed, efficiency and technical problems. However, we found the benefits of being creative, interesting and easy if users get used to the iControl interface. When asked if they were willing to use advanced controls, all users answered yes. From this point of view, without technical and armrest problems, iControl's interaction skills which are merged with eye-gaze tracking and gesture interface have considerable potential for digital manipulation.