SCIENCES SORBONNE UNIVERSITÉ

Sorbonne Université
Master ANDROIDE

Interaction Gestuelle

UE de projet M1

Equipe:
Qingyuan Yao
Alan Adamiak
David Leconte

Encadrants:
Gilles Bailly

2022

Contents

Chapter 1

Introduction

This is a HCI (Human-Computer Interaction) project of 3 students in the first year of our master degree at Sorbonne Université under the guidance of Mr. Gilles Bailly.

The purpose of this project is to compare gestural interaction with keyboard input. We compare the two primarily on execution speed and learning speed, based on an existing paper, we extend the research by adding some new independent variables and hope to obtain new perspectives. In order to do that, an interactive experiment has been developed and sent out to participants to record and gather input data in order to perform analysis.

We would like to start by specifying on the types of shortcuts we have studied:

- **Keyboard shortcuts** : The keyboard shortcuts we tested are executions that come from a **physical keyboard**, with at least 26 Latin letters and modifier keys such as shift, control (command on Mac), alt, etc.
- **Gestural shortcuts** : The gestural shortcuts consists actions with only one pointer and it should come from a mouse or a touch-pad. Touch screen input is not considered in this project.

The git repository which contain the written documents, the code, and tests results is accessible through this link.

Chapter 2

State of The Art

Although much research has been done on both keyboard and gestural shortcuts, the comparison between the two hasn't been on researchers mind, as they exist side by side without interfering each other.

However, we were given a relevant paper that does the comparison on performance speed and learning speed [1]. The researchers designed a set of keyboard and gestural shortcuts to execute various commands, participants were then told to execute the commands with both types of shortcuts with the help of drops down menus as reminders. Each participant had to complete 12 blocks of 60 trials organized into two sessions on two consecutive days.

The paper concluded that: *"Stroke shortcuts can be as effective as keyboard shortcuts in eventual performance, but have cognitive advantages in learning and recall."*

There are several factors in this experimental protocol that we found problematic: First of all, there was almost no semantic relations between the required commands and the designed shortcuts (for example, for the word "karate", the hotkey was Shift+W and the gesture, an U upside down). Secondly, the number shortcuts was overwhelmingly big, it was almost impossible to reach the level of expert in two sessions. Furthermore, there was no consecutive executions mentioned in the experiment, which could differ from the results of one single executions. Another factor that could have been considered is the size of the target, given that many shortcuts are executed on a target, in the experiment, the target is the whole screen.

The main goal of the paper was not to prove the superiority of gestural shortcuts over keyboard shortcuts but to prove the usability and potential of gestural shortcuts. And since the publication of the paper in 2009, the popularity gestural interaction has risen as the usage of smart devices with touchscreen increased exponentially. However, we would like to return to the original experiment and create our own experiment in hope of obtaining new insight on the two types of shortcuts by adding new dimensions.

Chapter 3

Contributions

3.1 Protocols

As this is a HCI project, we've followed the protocol of the quantitative method of the empirical evaluation:

1. Hypotheses
2. Experimental design
3. Data collection
4. Analysis

For the experimental design and tests generation, we have decided to use a modern tool called TouchStone (see 3.3.1), which generates unbiased tests from the given independent variables for every participant.

Different from the testing platform of the original experiment, we have decided to implement our experiment in the web environment (see 3.4). Despite the lack of ideal controlled laboratory conditions, this in-the-field approach gives us the possibility to send the tests to people of all socio-professional backgrounds and countries, which had proven to be very practical under the on-going sanitary conditions.

For analysis, we used Python and some libraries for statistics and graphs such as pandas, seaborn and pingouin, etc. (see 3.7)

3.2 Hypotheses and Independent Variables

Based on the experiments of Zhai and Appert[1], we have decided to define 4 main independent variables which have not been tested before:

- Size of the target
- Number of repeat
- Number of modifiers for keyboard shortcuts
- Number of angles for gestural shortcuts

And for each of the variable, we formulated the hypotheses shown below:

Hypothesis I: Gestural shortcuts will take more time to execute than keyboard shortcuts as the size of the target decreases

Hypothesis II: Gestural shortcuts will take more time to execute than keyboard shortcuts as the number of repeat increases

Hypothesis III: Keyboard shortcuts will take more time to execute as the number of modifiers increases

Hypothesis IV: Gestural shortcuts will take more time execute 2 angles separately than consecutively.

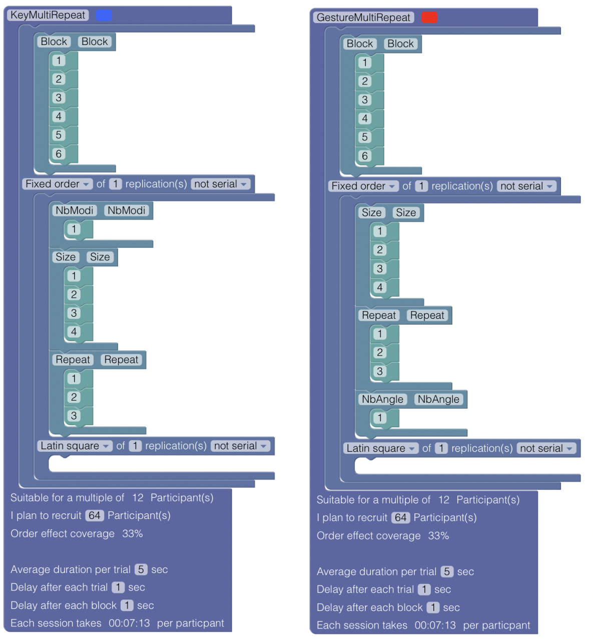
3.3 Experimental Design

3.3.1 Generation of Tests

TouchStone[2] is a web application that generates csv files that contains the totality of trials for each participant based on the given independent variables. It is also possible to randomize the sequence by classical randomization or latin square. Each test contains 6 blocks, each block is the totality of trials that contain all the possible combinations among all the variables (size of the target, number of repeat, etc.). By setting multiple blocks, we are able to see the performance gain as the participant gets familiar with the shortcuts.

As for the modifiers and the letters to use, or the directions of gestures to draw, we choose to generate them in the code (see 3.4).

After several changes, we've decided to make 4 type of experiments to be carried out:



KeyMultiRepeat
Block Block
1
2
3
4
5
6
Fixed order of 1 replication(s) not serial
NbModi NbModi
1
Size Size
1
2
3
4
Repeat Repeat
1
2
3
Latin square of 1 replication(s) not serial
Suitable for a multiple of 12 Participant(s)
I plan to recruit 64 Participant(s)
Order effect coverage 33%
Average duration per trial 5 sec
Delay after each trial 1 sec
Delay after each block 1 sec
Each session takes 00:07:13 per particpant

GestureMultiRepeat
Block Block
1
2
3
4
5
6
Fixed order of 1 replication(s) not serial
Size Size
1
2
3
4
Repeat Repeat
1
2
3
NbAngle NbAngle
1
Latin square of 1 replication(s) not serial
Suitable for a multiple of 12 Participant(s)
I plan to recruit 64 Participant(s)
Order effect coverage 33%
Average duration per trial 5 sec
Delay after each trial 1 sec
Delay after each block 1 sec
Each session takes 00:07:13 per particpant

Figure 3.1: First two TouchStone models

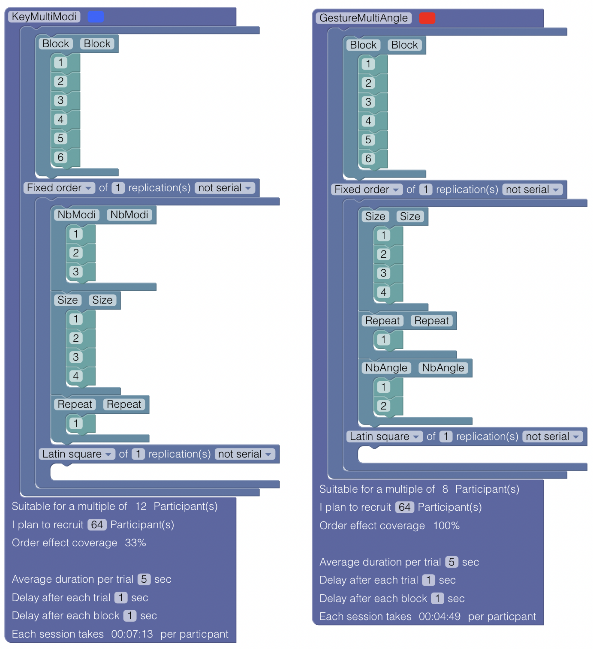
KeyMultiRepeat

In this experiment, for each trial we ask the participant to execute 1-3 keyboard shortcuts while fixing the number of modifier at 1. In the code, for each participant, we randomly draw 6 letters, separate them into 3 groups, and pair each two with one of the three modifiers (CMD/Ctrl, Alt/Option, Shift). Which makes 6 fixed shortcuts different for each participant.

In the current stage, we've even simplified the test by only using only CMD/Ctrl as the modifier, which means we have all 6 letters assigned to CMD/Ctrl.

GestureMultiRepeat

In this experiment, for each trial we ask the participant to execute 1-3 gestural shortcuts while fixing the number of angle at 1. In the code, for each participant, we randomly draw 6 out 8 directions (N, S, E, W, NE, NW, SE, SW). Which forms 6 shortcuts different for each participant.


KeyMultiModi
Block Block 1 2 3 4 5 6
Fixed order of 1 replication(s) not serial
NbModi NbModi 1 2 3
Size Size 1 2 3 4
Repeat Repeat 1
Latin square of 1 replication(s) not serial
Suitable for a multiple of 12 Participant(s)
I plan to recruit 64 Participant(s)
Order effect coverage 33%
Average duration per trial 5 sec
Delay after each trial 1 sec
Delay after each block 1 sec
Each session takes 00:07:13 per particpant

GestureMultiAngle
Block Block 1 2 3 4 5 6
Fixed order of 1 replication(s) not serial
Size Size 1 2 3 4
Repeat Repeat 1
NbAngle NbAngle 1 2
Latin square of 1 replication(s) not serial
Suitable for a multiple of 8 Participant(s)
I plan to recruit 64 Participant(s)
Order effect coverage 100%
Average duration per trial 5 sec
Delay after each trial 1 sec
Delay after each block 1 sec
Each session takes 00:04:49 per particpant

Figure 3.2: Second two TouchStone models

KeyMultiModi

In this experiment, for each trial we ask the participant to execute a single keyboard shortcut with 1-3 modifiers. In the code, for the simplicity of the test, there will only be 3 combinations: CMD/Ctrl, CMD/Ctrl + Shift, CMD/Ctrl + Alt/Option + Shift. For each participant, we randomly draw 6 letters, separate them into 3 groups, and pair each with a combination, forming 6 fixed shortcuts different for each participant.

GestureMultiAngle

In this experiment, for each trial we ask the participant to execute a single gestural shortcuts with 1-2 angles (the recognizer we built isn't able to recognize 3 angles yet).

For each participant, we randomly draw 3 directions and 3 pairs of directions, making it 6 shortcuts in total and the combination is different for each participant.

Due to the delay of the development, despite all the implementation finished, until the deadline of this report, we were only able to carried out the first two experiments. Thus, we're only able to do analysis for the first two hypotheses.

3.3.2 Apparatus

In order to test the speed of both types of input, the devices on which the test is to be done are required to have both input methods available. Which means all tests were done in the browser (ideally Firefox) on a computer with a keyboard and a mouse (or a touch-pad). due to the lack of controlled laboratory environment, the variations expected are:

- Customized hotkeys
- Incompatible browser
- Different modifier layout on different system (Windows with ctrl and Mac with cmd)
- Different letter layout in different language (QWERTY, AZERTY, etc.)
- Screen size

3.4 Website Development

We have developed the website using a vanilla HTML-CSS-JS front-end and node.js back-end. The website is hosted on a server to allow users to participate in the experiments without having to download anything, but also to centralize the results obtained.

3.4.1 UI-UX Design

Landing Page

When the user arrives on the website, they are greeted with a short thank you message and some general instructions about the experience. They are also asked what type of keyboard they are using and whether they are using a mouse or a trackpad. These are the only information we need to ask the user about their setup, as we can get the rest, such as the OS and browser, automatically with JavaScript. They can then proceed to the tutorial by clicking on a button.

Tutorial

When arriving on the page, the user is assigned a random identifier, which is used to have anonymous data, and a random type of experiment between gesture interaction and keyboard shortcut. Depending on the experiment, he receives a written explanation of what he has to do, and a video showing some trials. Once ready, the user can access the experiment by clicking on the "continue" button.

Experiment Page

We decided to make a very basic interface, showing only what is required for the experiments on the screen.

A target which consists of a simple circle that "sinks" (with the help of a shadow and light game) when the user clicks on it. When the user performs a shortcut successfully, a progress circle fills the target according to the number of shortcuts to be done on the trial, and when the shortcut was not correct, the circle flashes red to indicate failure;

A global progress indicator that shows the user how far along they are, and how many in total;

A list of shortcuts that the user will have to perform in the right order, and that turns light grey when they are done;

A button that is only clickable when the step is complete, it also indicates the progress to the user by starting grey, then turning pale green during the step and finally green when complete.

About You Page

At this stage, we ask the user for some personal information, age, gender, time spent on a computer per day and also the possibility to leave a comment to inform us about possible bugs, or just a feedback about their experiment. We will use this data to classify the results according to broad categories to see possible differences emerging.

Thank You Note

The experience finally ends on this last page, with a small message of thanks. We also give the user his identifier and a contact email address if he wishes to have feedback.

3.5 Experiments Interactivity

3.5.1 Adaptation of Tests

In order to let the participants take trials intended for them. We need the program to load the csv files generated by TouchStone (see 3.3.1) in memory. At the start of each trial, the program shows the corresponding instructions, draws the target in the right size, adjusts the recognizer to admit the right combination of keys or gestures, and when the shortcut is correctly executed, the program passes to the next trial.

3.5.2 Input Recognition

One of the difficulties of programming the test would be the implementation of the recognizers for keyboard and gestural input. The recognizers would take in the input and return true or false after being compared with the required command.

Key Recognizer

The flexibility of the keyboard shortcut recognition algorithm was at the heart of its development, so that a single function can handle a shortcut with a single modifier, as well as three, but also several in a row. Once the target is clicked, the keyboard shortcut recognition function is called each time a key is pressed. This function first registers which modifiers and which letter are currently pressed. The first time a key of the current shortcut is pressed, the time since the target was clicked is logged. If the shortcut is valid and there are still others on this trial of the experiment, the next one is selected for the new comparison, otherwise the next trial is loaded.

```
Input: e: the list of currently pressed keys
Input: currentExperiment: a list of shortcuts to do
i ← 0;
if OS is MacOS then
    mainModKey ← e.cmdKey
else
    mainModKey ← e.ctrlKey
end
shiftKey ← e.shiftKey;
altKey ← e.altKey;
key ← e.keyCode;
if mainModKey = true and currentExperiment.mainModKey[i] = true then
    Log elapsed time since target clicked;
end
if shiftKey = true and currentExperiment.shiftKey[i] = true then
    Log elapsed time since target clicked;
end
if altKey = true and currentExperiment.altKey[i] = true then
    Log elapsed time since target clicked;
end
if key = true and currentExperiment.key[i] = true then
    Log elapsed time since target clicked;
end
shortcutSuccess ← mainModKey = currentExperiment.mainModKey[i] and shiftKey =
  currentExperiment.shiftKey[i] and altKey = currentExperiment.altKey[i] and key =
  currentExperiment.key[i];
if shortcutSuccess = true then
    if i < currentExperiment.nbShortcuts then
        i ← i + 1;
    else
        Load next experiment;
    end
end
```

Algorithm 1: Key shortcut recognizer

In addition to the recognition of shortcuts, the function also manages the animations of the target according to the results obtained.

Gesture Recognizer

Two implementations of the gesture recognizer were kept for the experiment : one recognizer for single direction gestural input (up, down, left or right), and another for bi-directional gestural input.

For the one direction inputs, it was simply about getting the first and the last point of the input drawn on the canvas by the participant and deducing a direction from those points. The direction is computed in degrees from the coordinates of the two points through this formula :

$$x, x' \in \mathbb{R}^2$$
$$\theta = \frac{180}{\pi} arctan(\frac{x_2 - x'_2}{x_1 - x'_1})$$
$$angle = -\theta + 180$$

For the two directional inputs, we had to come up with a way to recognize where to split the input in two different segments. We got our inspiration from Kurtenbach and al. [3].

We first ensure that there is a point far enough from the starting point A so there can be a sufficiently wide angle, we call that point C. We then find between A and C the middle point B that maximizes the angle between the three (the maximum angle is considered to be the one purposely drawn by the user). After that, we only need to treat the two separate directions made respectively of the segments $[AB]$ and $[BC]$ the same way we treated the direction of a one direction input.

```
$E \leftarrow$ length of input/2;
$W \leftarrow E * 0.3$;
$Sensitivity \leftarrow 0.75$;
$M \leftarrow$ max breadth of menu;
$Threshold \leftarrow 360/M/2/$Sensitivity;
$A \leftarrow$ stroke starting point;
$C \leftarrow NULL$;

while $C$ is $NULL$ do
    $C \leftarrow$ first point a least distance W from A;
    if $C$ is $NULL$ then
        break
    end
    $L_AB \leftarrow$ first point at least distance W / 8 from A;
    $L_BC \leftarrow$ first point at least distance W / 8 from C;
    find point B between $L_AB$ and $L_BC$ so it makes the maximum $\widehat{ABC}$ angle;
    if $max\ \widehat{ABC} > Threshold$ then
        return $B$;
    else
        $A \leftarrow$ next point after A;
    end
end
```

Algorithm 2: Multi angle recognizer

Multiple Repeat Recognizer

The two main types of experiments (keyboard shortcuts and gestural interaction) are also replicated so that the user has to repeat that type of action more than once, but with different instructions.

Repeated experiment steps only consist of single modifier inputs for keyboard shortcuts and of single direction inputs for gestures (Ctrl + A *then* Shift + B, or draw up *then* draw left).

The recognizer doesn't allow the tester to pass each step if the user hasn't succeeded at each one of the instructions he was given in the order they were given to him.

3.5.3 Data Collection

In order to log the performance of each user, we store the totality of the recorded data in a csv file, where each line represent a trial. The header, which contains the name of each column of data is represented as below.

ExperimentType_ParticipantID.csv

```
header = [
    //Config CSV Data
    "DesignName", "ParticipantID", "TrialID", "Block1", "Block2", "Size",
       "Dir", "NbModi", "NbAngle", "Repeat",
    //For Keyboard
    "letter1", "modifiers1",
    "execTimeCMD1", "execTimeAlt1", "execTimeShift1", "execTimeKey1",

    "letter2", "modifiers2",
    "execTimeCMD2", "execTimeAlt2", "execTimeShift2", "execTimeKey2",

    "letter3", "modifiers3",
    "execTimeCMD3", "execTimeAlt3", "execTimeShift3", "execTimeKey3",

    //For Gestures
    "angle1", "userAngle1", "drawDist1",
    "angle2", "userAngle2", "drawDist2",

    //For Both
    "mouseClick1", "totalExecTime1", "nbOfAttempts1",
    "mouseClick2", "totalExecTime2", "nbOfAttempts2",
    "mouseClick3", "totalExecTime3", "nbOfAttempts3",
    "targetDist", "keyboardLayout", "mouseType"
]
```

In order to have comparison between all the tests, we have decided to have the same structure of header for all the experiment types where the non-relevant column would simply be NaN.

To analyse all the data, a bash script has been used to concatenate all the csv log files into a main one.

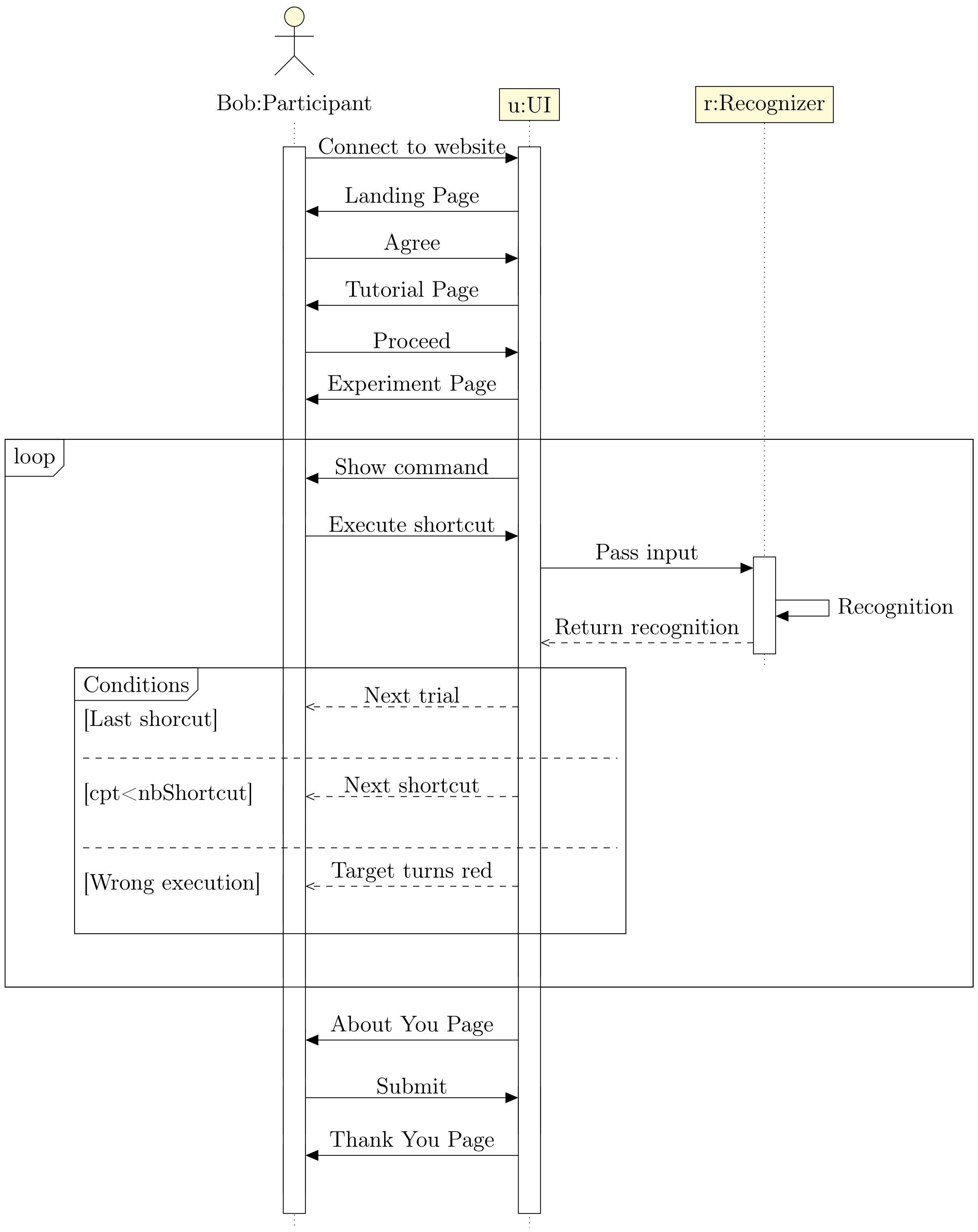
userdata.csv

Apart from user performances, we also want to collect other data from the user. At the start of experiment, we ask the keyboard layout and the mouse type from the user, the type of the web browser is also stored. At the end of the experiment, we ask some more personal, but anonymous and optional data from the user such as their age, gender and frequency of computer usage. We also leave a text area for the user to send us a message, whether it's bug report, description or observation of the experiment, or critiques about what we can do to improve the experience. The data is store in another csv files structured as below:

```
endform = ["DesignName", "ParticipantID", "keyboardLayout", "mouseType",
   "browser", "user_age", "user_gender", "frequency", "comment"]
```

3.5.4 Sequence Diagram

The sequence diagram down below illustrate how a participant (Bob) would interact with the program and how the program would respond.



3.6 Results

Due to the delayed development schedule, we have only collected enough data for the first two experiments, therefore, we will only discuss on the first two hypotheses.

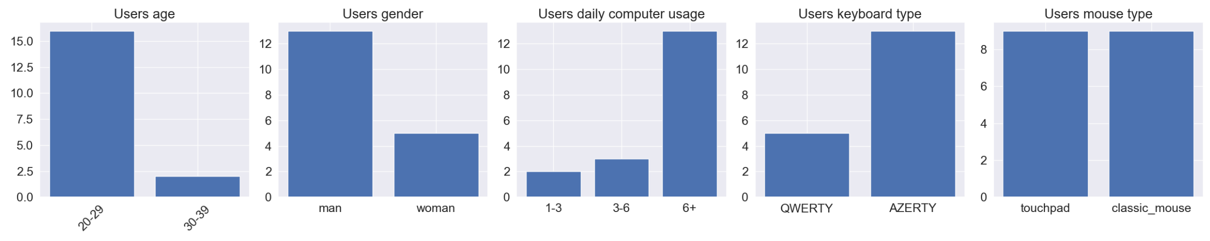


Figure 3.3: Distribution of user data

3.6.1 Participants

Up until the start of the analysis phase, we have collected data from 22 participants, with the following demographics:

3.6.2 Participant Feedback

As mentioned before (see 3.5.3, participants were allowed to leave a comment. Thanks to these comments, we have managed to notice and fix some inconveniences of the website. And in some cases, the comments have also helped us comprehend better the data.

One of the participant stated that *"I became less and less concentrated and my fingers are tired.the direction of left down or up right is a little difficult."*, which explains the drop of performance for the gestural shortcuts, and gives us an insight on the difference of energy consumption and comfort between keyboard and gesture shortcuts.

3.7 Analysis

3.7.1 Filtering

Outliers

After obtaining the raw data, we have found that the performance of some participants are not persistent, there are many trials where the execution time differ significantly from the average performance (figure 3.4). While seaborn already remove outliers with the confidence of 95% when plotting, it does not differentiate performances under different independent variables (for example, the trials with Size=Tiny and Repeat=3 could be mixed with trials where Size=Large and Repeat=1)

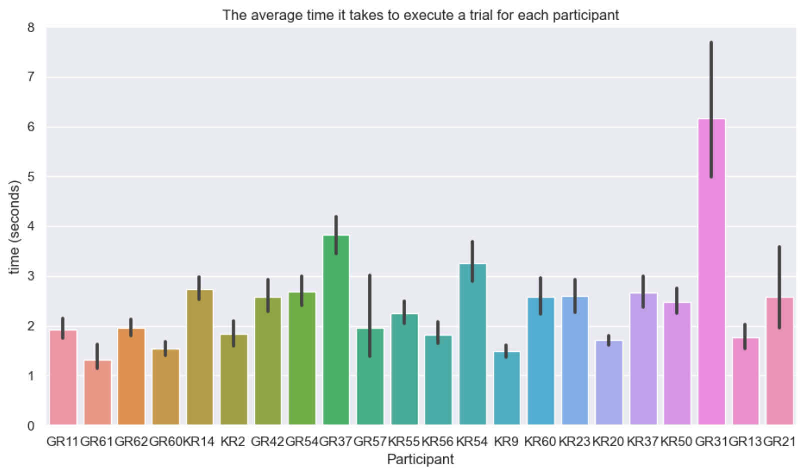


Figure 3.4: Average performance and error space with outliers

To prevent false analysis, we have decided to remove some outliers beforehand. The process is quite simple, for every combination of independent variables, we remove the trials where the difference between execution time and average is greater than the standard deviation.

After removing outliers, we can see the results are much more consistent (figure 3.5). In fact, some participants have been removed completely as their data proves to be more distracting than useful.

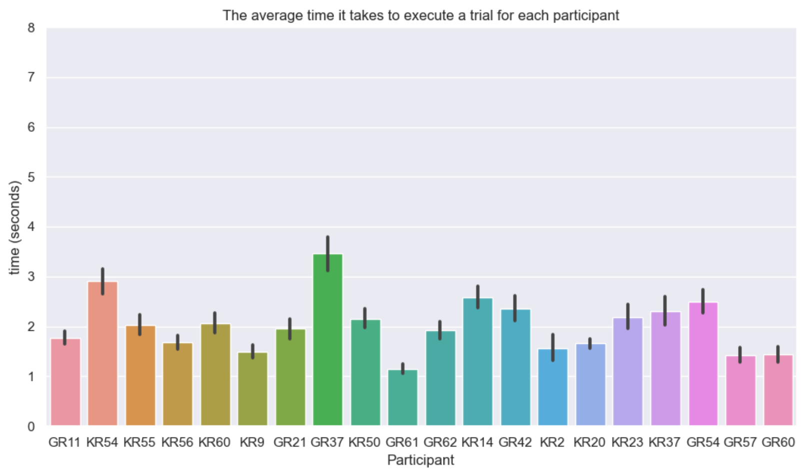


Figure 3.5: Average performance and error space after removing outliers

An interesting remark to make is that, the outliers exist much more often in gestural shortcuts than keyboard shortcuts. Which we suppose comes from the inferior familiarity of gestures than keyboard.

Blocks

From the figure 3.6 representing the average performance by block, it is observed that in the first block, execution time is longer than other blocks. Which is normal considering the need for participants to familiarise with the shortcuts. But for the sake of accuracy, we have decide to exclude the first block out of the analysis.

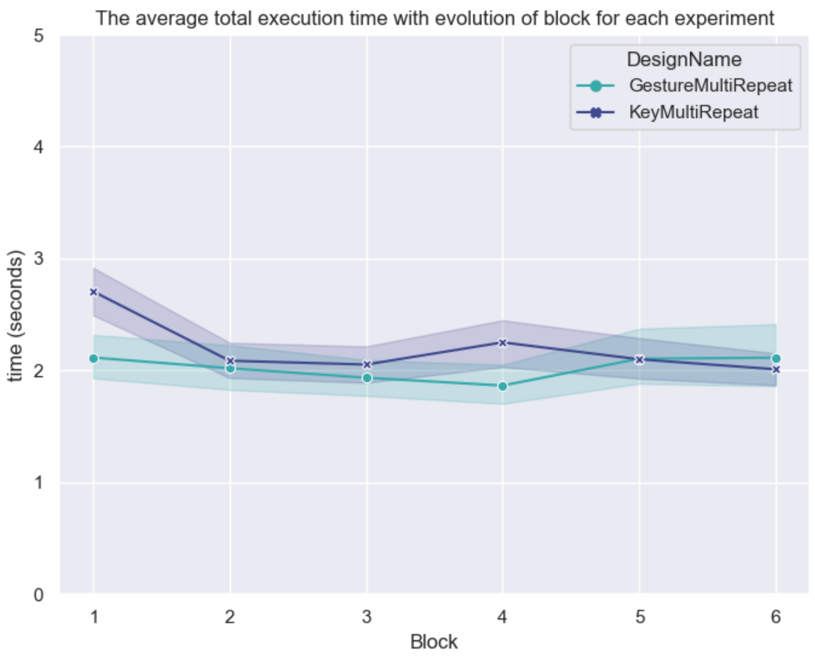


Figure 3.6: Average performance by block

An interesting remark to make is that, in the first block, the execution time for gestures shortcuts is notably faster than keyboard shortcuts. Which could be explained by the simplicity of the designed gestures and our intuitive tutorials. However, as participants do trials for later blocks, the execution time of keyboard shortcuts has exceeded gestural shortcuts. Which we think can be explained by the comments of a participant we mentioned previously (see 3.6.2).

3.7.2 Validations of Hypotheses

Hypothesis I

According to the ANOVA test, the p-values of size for the two experiments are:
KeyMultiRepeat : 2.086864e-03
GestureMultiRepeat : 0.000008
Which proves the relevance for both experiments.

From figure 3.7, we can see the gestural shortcuts are performed slightly faster than keyboard shortcuts, but the slope of gestural shortcuts is steeper that of keyboard shortcuts, especially when passing from Tiny to Small.

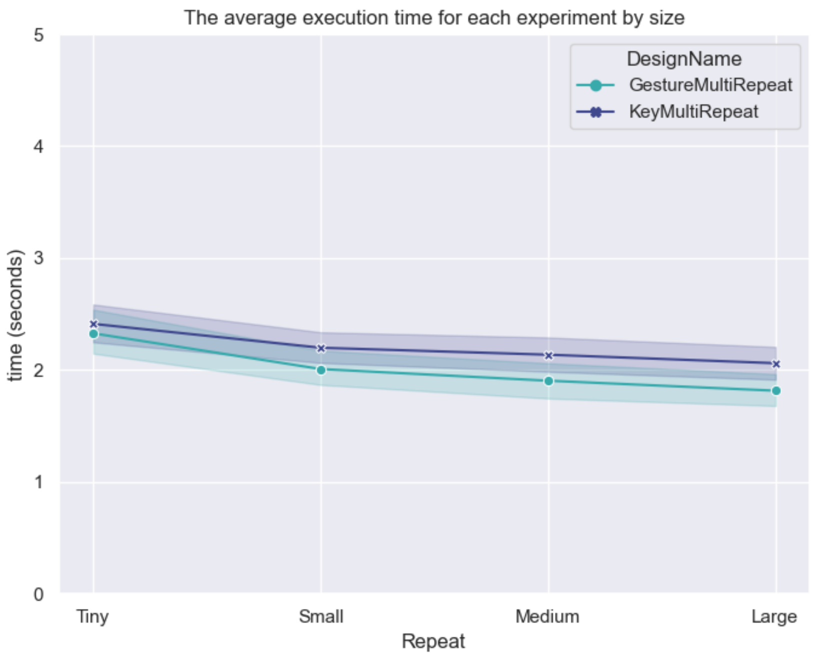


Figure 3.7: Performance by size

In order to further explain this, we have also studied the average time it takes to click on the the target (figure 3.8. Surprisingly, the time for keyboard shortcuts is slightly faster than gestures. Which lead us to deduce that the average time to execute keyboard shortcuts after click on the target is longer than that of the gestural shortcuts.

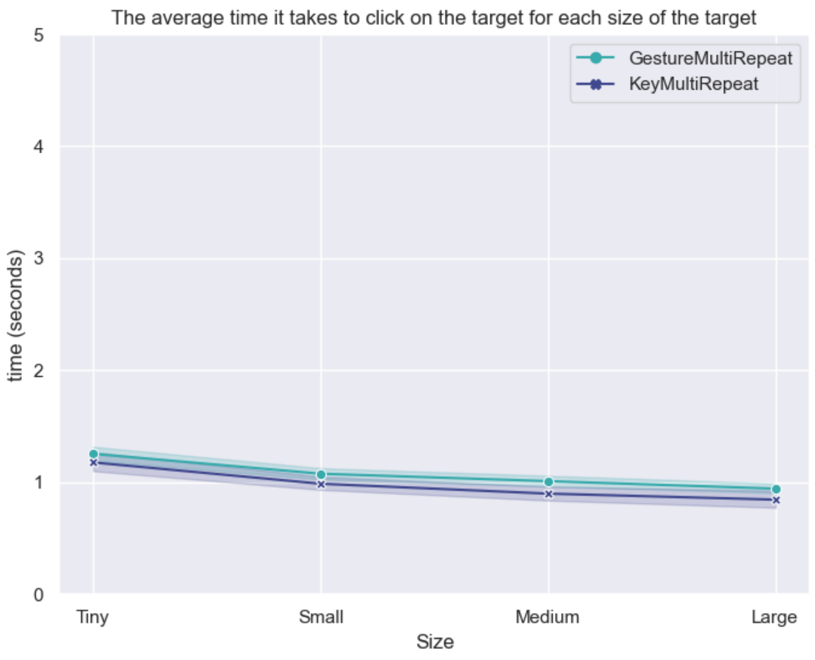


Figure 3.8: Performance

In conclusion, while there is a relevance between performance and size of the target for both shortcuts, we cannot prove the size of the target has interaction effect on keyboard and gestural shortcuts.

Hypothesis II

According to the ANOVA test, the p-values of the number of repeat for the two experiments are:
KeyMultiRepeat : 1.067038e-09
GestureMultiRepeat : 0.000005

Which proves a even bigger relevance than size

In figure 3.9, we noticed an interaction effect in all sizes. At single execution, the gestural shortcuts keep the superiority over keyboard shortcuts as we have observed in the last hypothesis. However, at two executions, the performances of the keyboard shortcuts have gotten closer to gestural shortcuts, even surpassing the latter in some instances. At three executions, it is clear that the keyboard shortcuts have evident dominance over the gestural ones (especially for Tiny targets, but it is still not enough to prove Hypothesis I). Which validate our second hypothesis.

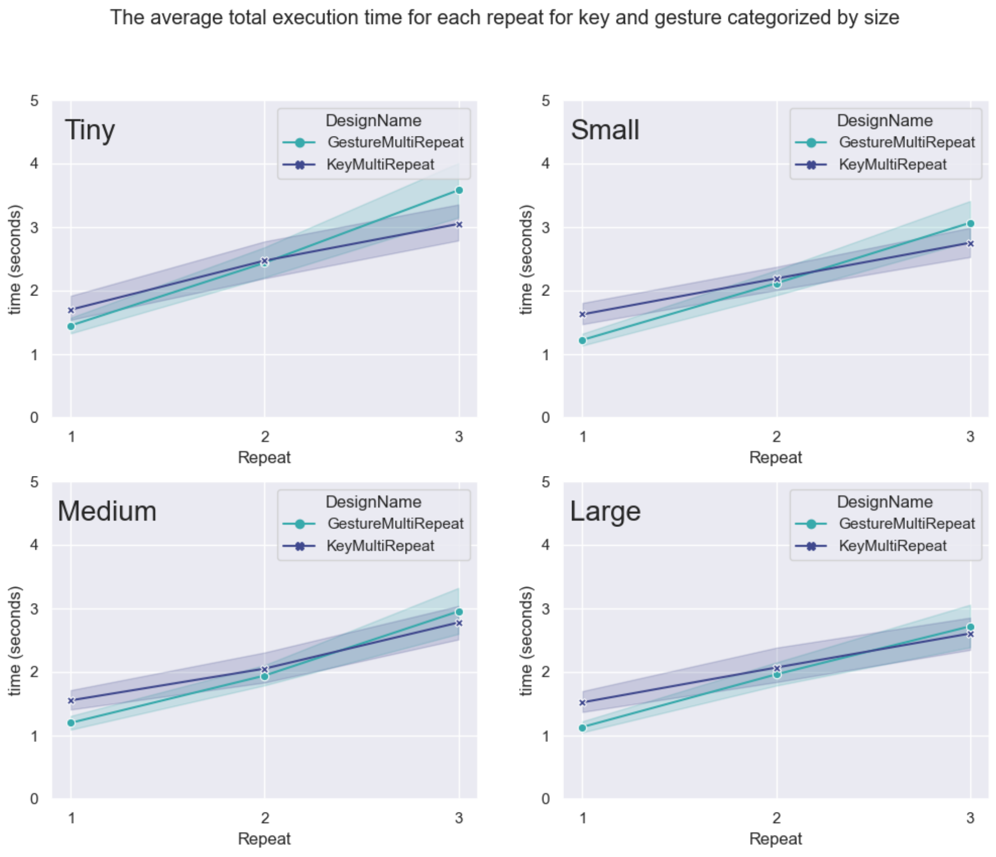


Figure 3.9: Performance sorted by repeat and size for each experiment type

We speculate that this is due to the fact that, in order to execute the second/third shortcut, participants need to move the cursor back to the target in order to draw towards the required direction, while for keyboard shortcuts, participant only need to press on the next key.

Other Analysis

We have made more graphs in order to see other correlations, considering the limited pages, we have decided to leave them on our git repository[4].

3.7.3 Final Words

While it is observed that the average execution time in general for gestural shortcuts is faster than keyboard shortcuts. We don't believe that this proves the faster speed of gestural shortcuts. Instead we think the design of shortcuts affects significantly the execution time, and compared to the design in the experiments of Zhai and Appert, we did lower the difficulty of gestural shortcuts while maintaining the same difficulty for keyboard shortcuts.

However, as observed in our graphs, gestural shortcuts have proven to be more affected by the size of the target decreases and the number of repeat, by having steeper slopes than keyboard shortcuts.

For further analysis, we would like to see how the increase of difficulty (multi-angled gestures, multi-modifiers) would affect both shortcuts.

Chapter 4

Conclusion

In this project, we dived into the domain of Human-Computer Interactions and focused on interactions that seems basic, but plays a crucial part in our modern digital lives. We have based our research on the results of those before us, in order to look further for the correlations. While not all of our analyses have been studied or proved, we've still gained insights on more dimensions than before.

Furthermore, by following the professional protocols, we have learned the scientific way of studying human behavior. And by honing our skills in HTML-CSS-JavaScript, we have come closer to mastering web development.

4.1 On the Agenda

Due to underestimating the develop time of the website, splitting too much the tasks between three people, and not making the code flexible against changes of experimental designs. We have not fully reached the envisioned goal of our project. Here are the things we would like to continue to finish and add if we had time:

- Tests and analysis for the last two experiments
- Add semantically linked commands with shortcuts
- Gamification of the experiment
- Further interaction with participants: interviews on daily usage of shortcuts
- Add left-handed/right-handed in the questionnaire
- Do all experiments in an well-controlled laboratory environment
- More representative and intuitive graphs

Bibliography

[1] Caroline Appert and Shumin Zhai. "Using strokes as command shortcuts: cognitive benefits and toolkit support". en. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Boston MA USA: ACM, Apr. 2009, pp. 2289–2298. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1519052. URL: https://dl.acm.org/doi/10.1145/1518701.1519052 (pages 2, 3).

[2] Alexander Eiselmayer, Chat Wacharamanotham, Michel Beaudouin-Lafon, and Wendy E. Mackay. "Touchstone2: An Interactive Environment for Exploring Trade-offs in HCI Experiment Design". en. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Glasgow Scotland Uk: ACM, May 2019, pp. 1–11. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300447. URL: https://dl.acm.org/doi/10.1145/3290605.3300447 (page 4).

[3] Gordon Kurtenbach, George Fitzmaurice, Azam Khan, and Don Almeida. "Scale Independence in Marking Menus". en. In: (), p. 8 (page 9).

[4] *GitHub repository of the project*. URL: https://github.com/MRVNY/ProjectGI (pages 18, 27).

Appendix A

Cahier des charges

A.1 Introduction

This is a project of 3 students in the first year of our master degree at Sorbonne Université under the guidance or Mr. Bailly.

The purpose of this project is to compare gestural interaction with keyboard input. We will compare the two primarily on execution speed and learning speed. In order to do that, we will make an interactive survey to record and gather input data from users and perform analysis based of that data.

We hope this research project would prove the superiority of keyboard input and serves as an counterargument for papers that overestimate gestural interaction.

Down below, you will find the explanations on each stage of our project and how we plan to tackle them.

A.2 Definitions

Firstly, we would like to specify on the terms we use and comparisons we'll make, as there exists multiple types of gestural input.

A.2.1 Keyboard Input

The keyboard input should come from a physical keyboard, with at least 26 Latin letters and modifier keys such as shift, control (command on Mac), alt, etc.

A.2.2 Gestural Input

The gestural input consists actions with only one pointer and it should come from a mouse or a touch-pad. Touch screen input is not considered in this project.

A.2.3 Platform

In order to test the speed of both types of input, the devices on which the test is to be done are required to have both input methods available. Ideally, all tests should be done on a computer with a keyboard and a mouse (or a touch-pad).

A.3 Stages

The project will be carried out in 5 stages, some stages can be done in parallel with others, and the previous 3 stages would be repeated at least twice in order to optimise collected data.

A.3.1 Modeling (Input Design and Data Structure)

A set of actions, ranging from copy-pasting to spatial movement would be defined as commands that will be activated with a gestural interaction as well as a keyboard input.

The commands and inputs should be related and intuitive enough for everyday use as such is what we intend to simulate.

It's best to also define the data structure. The data to collect would be combination of input time, the evolution of input speed, precision, parameters on the users, etc.

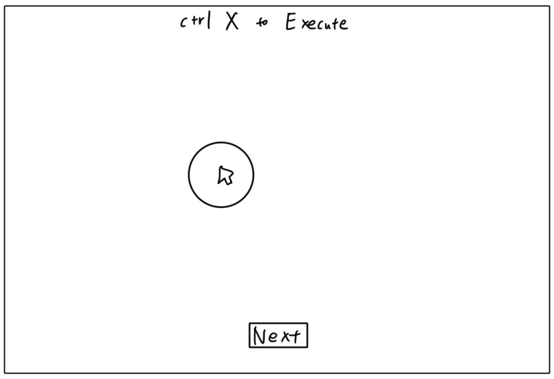
Plan of delivery: By the end of this stage, all the gestures and keystrokes to use would be decided as well as the data structure.

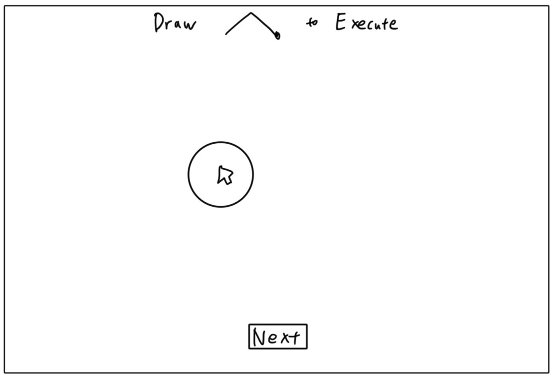
A.3.2 Interactive Test

The program will be a web application coded in JavaScript, it would be hosted on a lip6 server for more people to access and take the test. The gamification of the survey is an option to consider if time allows

Direct Commands

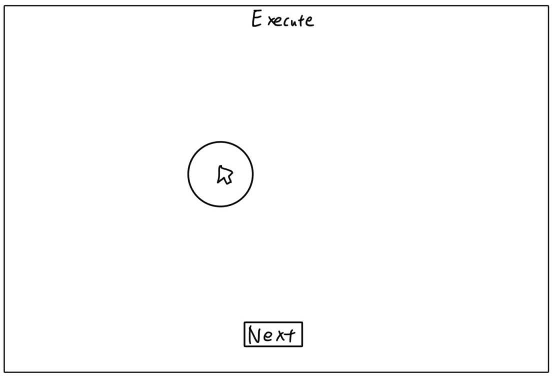
The first part of the test would be command with specific actions such as "Click Control X to execute" or "Draw X to execute"





Natural Commands

The second part is set out to test the learning speed for both types of input, so instead of showing "Click Control X to execute", the command would simply be "Execute"

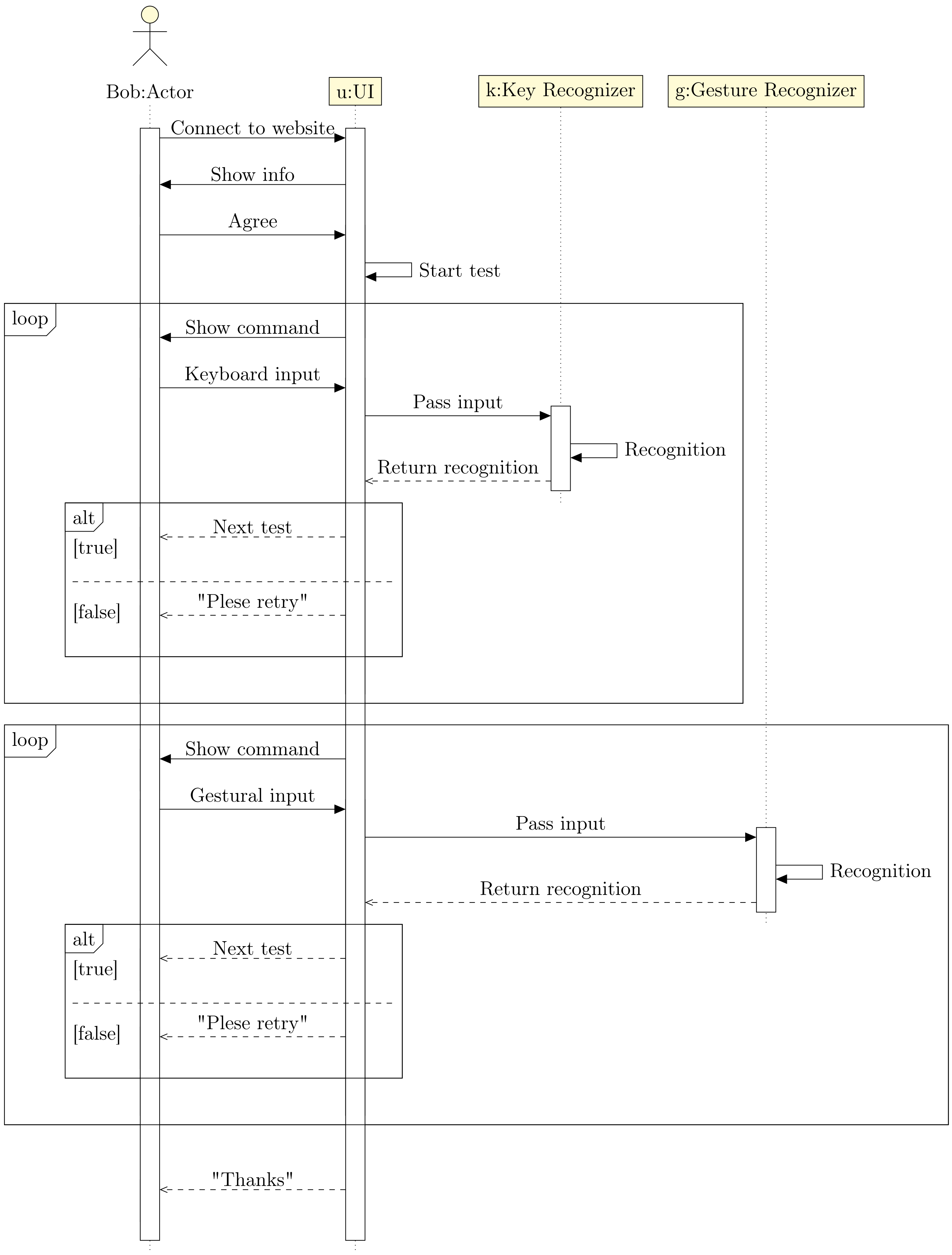


Input Recognition

One of the difficulties of programming the test would be the implementation of the recognizers for keyboard and gestural input. The recognizers would take in the input and return true or false after being compared with the required command.

Sequence Diagram

The sequence diagram down below illustrate how a user (Bob) would interact with the program and how the program would respond.


Bob:Actor
u:UI
k:Key Recognizer
g:Gesture Recognizer
Connect to website
Show info
Agree
Start test
loop
Show command
Keyboard input
Pass input
Recognition
Return recognition
alt
[true]
Next test
[false]
"Plese retry"
loop
Show command
Gestural input
Pass input
Recognition
Return recognition
alt
[true]
Next test
[false]
"Plese retry"
"Thanks"

Plan of delivery: By the end of this stage, the program which realizes the functionalities and steps of sequence diagram would be finished. The gamification is a surplus.

A.3.3 Tests

The test will be carried out online, and ideally, people of all socio-professional background would participate.

Plan of delivery: By the end of this stage, the estimated number of participants should have finished taking the test.

A.3.4 Analysis

After gathering enough data from the test. We will analyse the data in order to compare the execution speed and learning speed for both inputs based the data harvested during the experiment.

The data would be manipulated in Python.

Realization of the learning curve from the data would be a plus for the analysis.

Plan of delivery: By the end of this stage, conclusions would be drawn in order to write the report.

A.3.5 Report

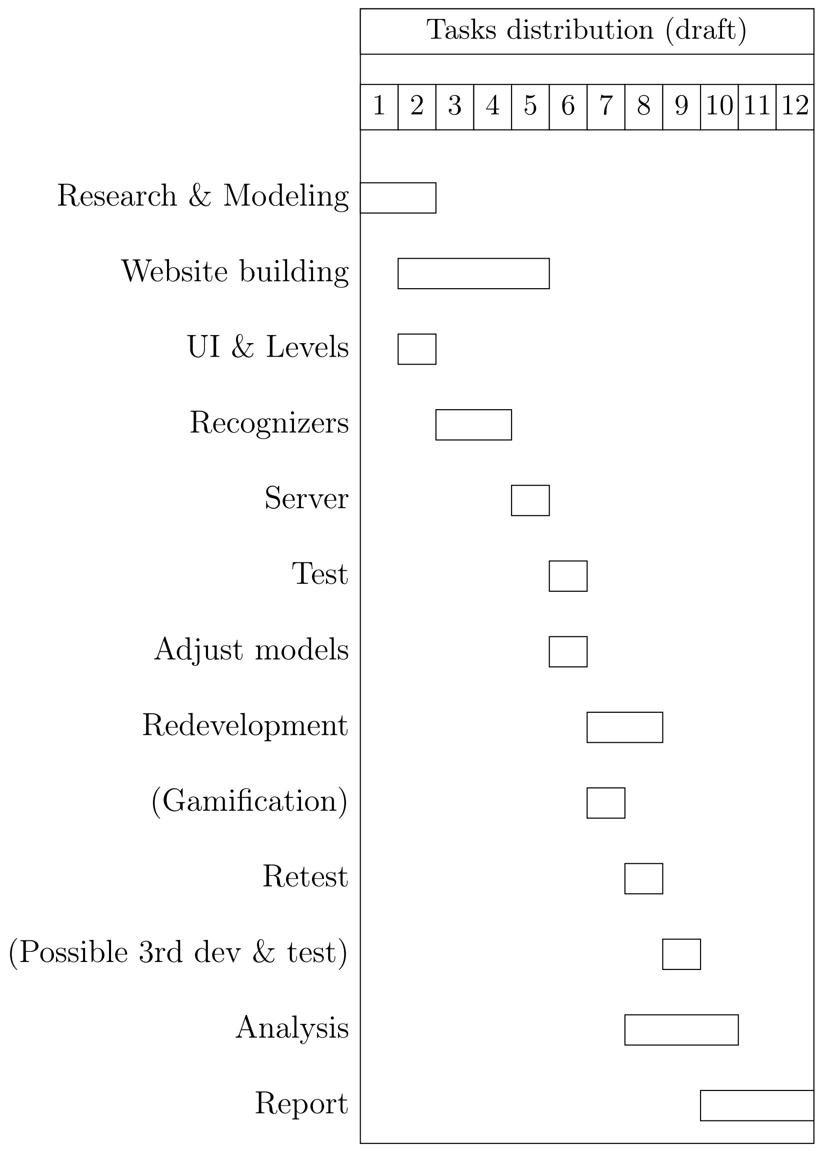
After the analysis, a report will be redacted to explain in details the 5 previous stages, including the references we've used, the decisions we've made, the implementation we've done and the results we've obtained

Plan of delivery: By the end of this stage, the report would be finished.

A.4 Progress Estimation and Tasks Distribution

We estimate the project to be finished before the end of May 2022, which gives us approximately 4 months. A detailed estimation and task distribution is specified down below:

1. **Research & Modeling:** 2+1 weeks, 1-2 persons
2. **Building the test website:** 1 month + 2 weeks, 2-3 persons
    - UI, 1 week
    - Recognizers, 2 weeks Server 1 week
    - Readjustment and optimisation 1-2 weeks
    - (Gamification) 1 week
3. **Public takes the test:** 1+1 weeks
4. **Analysis of the results:** 2-3 weeks, 3 persons
5. **Redaction of the report:** 2-3 weeks, 3 persons


Tasks distribution (draft)
1
2
3
4
5
6
7
8
9
10
11
12
Research & Modeling
Website building
UI & Levels
Recognizers
Server
Test
Adjust models
Redevelopment
(Gamification)
Retest
(Possible 3rd dev & test)
Analysis
Report

Appendix B

User Manual

B.1 Running code locally with Node.js

You will need to install **Node.js**, which is a pack of tools allowing you to write server-side code using the JavaScript language.

Then, to execute the code, you will also need clone the project repository from GitHub[4]. Go to the project folder through your terminal. Once you are there, type and enter both of these commands to install the needed Node.js packages and launch the server on port 4000 (make sure this port is not used by another process):

```
$ npm install
$ node server/index.js
```

The server is now online at address **http://localhost:4000**.

B.2 Taking tests online

Once you get to the main page, you can already choose your material and proceed to see what experience you have been attributed, as well as a tutorial showing you what the experience is.

At the end of the experiment, you are reminded your participant ID. You can find the information about your trials in the **/server/logs** folder, the file depending on what experiment you were part of, and your credentials such as your age and gender are stored in the **/server/logs/userdata.csv** file.

B.3 Logs and Analysis

First you have to concatenate all the csv files by executing in the terminal:

```
$ cd server/logs/
$ bash go.sh
```

Then simply run analyse.py to generate new graphs and see the anova tests.