Lehrstuhl für Medieninformatik
Institut für Information und Medien, Sprache und Kultur
Universität Regensburg

Interaction Techniques and Technologies Assignment 5: Pointing Experiment

Summer Semester 2019

Submission due: Wednesday, 29. May 2019, 23:55

Hand in in groups of max. two. All submitted code and text must contain comments detailing which group members contributed to it.

Goals of this assignment

- get more comfortable with Python, the Jupyter Notebook, numpy and matplotlib
- conduct analysis of your experimental data and document the results
- read up on pointing devices and study design

5.1: Implement a Pointing Experiment

Create a Python script that can be used to conduct a pointing experiment.

In particular, your script should have the following features:

- read test configuration from a .ini file[1] or .json file[2].
- present multiple targets on the screen that the user can click (example: Bubble Cursor (video)[3]. In most cases, colored circles are a good choice - although you may present file icons, words, etc. instead. You may also use a background image as distractor (see e.g., the Shift study). One of the targets should be highlighted in some way to indicate that the user should try to click on this one.
- conduct multiple rounds with varying sizes/distances (colors, … - whatever you like) of targets.
- presents conditions in a counter-balanced order.
- outputs all important information (e.g., start/end position of pointer, errors, task completion time, condition) on `stdout` in CSV format.

Notes:

- Have a look at `fitts_law_experiment.py` in GRIPS for some ideas on how to implement the script. Be aware that this is ugly code for several reasons.
- One important design decision you have to make: where to place the targets and the pointer. The approach used in `fitts_law_experiment.py` has advantages and limitations. Have a look at other pointing experiments (e.g., on YouTube) for inspiration on pointer/target placement.
- Try to use a modular implementation where experiment design (which task should be run next, etc.) is implemented seperately from target display and pointer tracking.

Hand in a file **pointing_experiment.py**

[1] https://docs.python.org/3/library/configparser.html
[2] https://docs.python.org/3/library/json.html
[3] https://www.youtube.com/watch?v=AEnfV4cTrvQ

Points

- **1** The script has been submitted, is not empty, and does not print out error messages and follows PEP8.
- **1** The presentation is beautiful and follows best practices for user interfaces.
- **1** The script reads in the test configuration from a file.
- **2** The script presents targets of which one is highlighted and can be clicked
- **1** The script presents multiple pointing tasks to the user
- **1** The script outputs all necessary information on stdout in CSV format
- **1** Workload distribution among team members is documented in comments

5.2: Implement a Novel Pointing Technique

Conduct a few test runs of your pointing experiment and invent or find an interaction technique that might help the user in selecting objects on a computer screen with a certain input device (e.g., mouse, touch screen). Either implement a technique from the literature (see tips below) or design your own.

Examples:

- the paper "Probabilistic Pointing Target Prediction via Inverse Optimal Control"[4] contains a good overview of general approaches on page 1.
- Bubble Cursor[5]
- switch between high and low CD gain depending on task (e.g., high CD gain if no target is close by, low CD gain if multiple targets are close by)
- magnetic targets: once the cursor is close to a target (and no other potential targets are close), 'pull' the cursor towards the target (An Evaluation of Sticky and Force Enhanced Targets in Multi Target Situations[6])
- extrapolate the pointer trajectory and highlight the target that will be probably selected, allowing the user to click it earlier.

The pointing technique should be implemented as a separate Python class/object that extends your experiment script from assignment *5.1*. This object gets passed a list of available targets (and other necessary information) upon instantiation. Every pointer event (or a set of coordinates) is then passed to a function `filter()` of this object. This function returns the new (optimized) coordinates for the pointer which are then used by . (You are free to implement this any other sensible way, as long as you keep the implementation of the pointing technique clearly separated from the rest of the application.) Include a comment at the top of the file explaining concisely how the pointing technique works. The test script from assignment *5.1* should be extended in a way that allows you to enable/disable use of the pointing technique programmatically (e.g., via the configuration file, via a shortcut, and/or via a command-line parameter) . The CSV output of the test script should also indicate whether the pointing technique was activated.

Hand in a file **pointing_technique.py** (which does not need to be executable on its own).

Points

- **1** The script has been submitted, is not empty, and does not print out error messages and follows PEP8.
- **2** The pointing technique is sensible and described in appropriate detail
- **2** The pointing technique is implemented well (i.e., using it seems to subjectively improve pointing performance)
- **1** Workload distribution among team members is documented in comments

[4] http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/bziebart/publications/pointing-target-prediction.pdf
[5] http://www.dgp.toronto.edu/~tovi/BubbleCursor/
[6] https://www.researchgate.net/profile/Martin_Hitz/publication/221248152_An_evaluation_of_sticky_and_force_enhanced_targets_in_multi_target_situations/links/0912f51091f5f38665000000.pdf

5.3: Evaluate your novel pointing technique

Conduct a small experiment evaluating your pointing technique with at least four participants.

Hand in a Jupyter notebook which contains the following:

- a short documentation of your test design (pointing technique, hypotheses, variables, participants)
- a structured analysis of your test results (i.e., programmatically reading in the CSV files, grouping data by conditions, etc.)
- appropriate visualizations that indicate absolute and relative pointing performance (task completion times, errors) of pointing technique and standard pointer behavior and of individual participants
- summary statistics and visualizations that show whether the initial hypotheses seem to be correct.

Hand in a file **pointing_technique_experiment.ipynb** and one or more CSV files (**data.csv or data1.csv … dataN.csv**) which must be in the same directory as the Jupyter notebook.

Points

- **1** The notebook has been submitted, is not empty, and does not print out error messages and generally follows the Python style guide (PEP 8).
- **2** The experiment is described in sufficient detail and clarity.
- **1** The experiment design is sensible and does not contain errors.
- **2** The notebook correctly reads the data from the file(s) and outputs the required visualizations and statistics
- **2** Visualizations are self-explaining and contain units and axis description.
- **2** The results are discussed in sufficient detail and clarity.

Submission

Submit via GRIPS until the deadline

All files should use UTF-8 encoding and Unix line breaks. Python files should use spaces instead of tabs.

`Have Fun!`