Experiment name

Manual + documentation

Contents

Manual

Overview

This is a manual describing a python implementation of two measurements of Unilateral spatial neglect (hereafter: Neglect). The two experiments measure a horizontal free viewing bias and horizontal pupil response bias as described by the following publications:

Ohmatsu, S., Takamura, Y., Fujii, S., Tanaka, K., Morioka, S., & Kawashima, N. (2019). Visual search pattern during free viewing of horizontally flipped images in patients with unilateral spatial neglect. Cortex, 113, 83-95.

Ten Brink, A. F., Van Heijst, M., Portengen, B. L., Naber, M., & Strauch, C. (2023). Uncovering the (un) attended: Pupil light responses index persistent biases of spatial attention in neglect. *Cortex*, 167, 101-114.

Starting the application can be done by running the .exe file from a stimulus pc. The pc does not have to be connected to a functioning eye tracker and can be run in dummy mode (using the mouse cursor as simulated gaze position). Additionally, information regarding participants, setups, instructions, and sessions can be viewed and edited without needing an eye-tracker connection.

When the App is started the main window is shown giving an overview of all functionalities is given. The functionalities are described hereunder:

Start Exp: Starts the experiment. By running an experiment, a Session object is created for which a Participant and a Setup object is required. These can be selected when previously defined or created on the fly.

View Sessions: An overview of all previously recorded sessions. Sessions are saved as .ses objects using Python's Pickling method. Session information includes which participant was

tested at which setup, and the results of that experiment. Additionally, a .csv file containing all relevant samples is saved in the same folder. See the section *Post-measurement* for an explanation of how the data is saved.

Before the experiment is started, a setup and a participant should be defined already. When starting an experiment, you are prompted to select each of these objects from the list of existing objects, or you can define a new object on the fly.

View Setups: An overview of all defined setups. These are saved as .set objects using Python's Pickling method. Setup information includes eye-tracker and stimulus screen information.

View Participants: An overview of all defined participants. These are saved as .part objects using Python's Pickling method. Participant information includes demographic and clinical information.

View Instructions: An overview of all defined experiment instructions. These are Dutch and English by default. You can define experiment instructions yourself when required. (see: **Defining a language of instruction**)

Pre-measurement

Before measuring pupil and gaze position bias from any participant, certain information is required to run the experiment. This section describes all the steps required to provide the necessary information.

Overview

To start the experiment, participant and setup information is required. You may fill the information in beforehand by following the steps in the sections **Defining a participant/Defining a setup**, or you may give the information on the go by clicking '*Start Exp'* from the main window. You will first be prompted to select a participant and a setup. Here you can select the previously defined participant and setup information, or you can select 'new'. When you select the 'new' option, you will be prompted all required information. See the

sections **Defining a participant/Defining a setup** for a detailed explanation of all data fields. After participant and setup have been selected, you will be prompted all other information required to start the session. They are described hereunder.

1. Experiment order.
   Which of the possible experiments are performed, and their order. Experiments can be (de)selected using the checkbox under their name and reordered using the arrow buttons under their name.
2. Gaze region scale.
   The experiments use gaze-contingent conditionals e.g. the participant needs to maintain fixation on an ellipse (visual degrees: 1.74° wide, 3.48° high) around the fixation cross for the duration of a trial. Since maintaining fixation is hard for some participants, the software allows this region to be scaled by any factor.
3. Testing Date.
   The date on which the experiment is performed. Here, the current date is automatically filled in.
4. Upload data consent.
   Whether the participant agrees to the uploading of their data to a possible online data repository. This includes all data collected about the participant (demographics, clinical information, and experiment results) as well as the setup information.
5. Dummy mode overwrite.
   Sometimes you might want to run the experiment without connecting an eye-tracker. When a specific eye-tracker has been selected in the setup information, this can be reverted to dummy mode (supplementing gaze position with mouse-cursor position).

Eye-tracker setup

The eye-tracker that is used to collect data should be able to report gaze position and pupil size at a sampling frequency of at least 60 Hz. One tracked eye is sufficient. The experiment relies on

PyGaze, which is able to communicate with eye-tracking devices of Tobii, SR research (EyeLink), and SensoMotoric Instruments.

More information about the eye tracker?

Defining a setup

All data fields relating to a setup are given hereunder, followed with a small explanation.

1. Reference name.

   The name referring to the setup. This could be a reference to a laboratory or office. You can enter any name here (but try to abstain from using characters reserved by windows OS: <, >, :, ", /, \, |, ?, and *).

2. Screen information

   All information relating to the screen on the stimulus pc that is used to present the experiment.

   Screen information consists of multiple different data fields:

   a. Screen used: When multiple screens are used, they are referred to using a number, starting at 0. If you know the number that refers to the screen you want to use you can fill this in manually. If not, we implemented a method that shows you which screen has which number, and prompts you to select any of these screens
   b. Screen resolution: The width and height of the used screen in pixels. If you know these numbers you can fill them in manually. The resolution will be obtained automatically if you selected to automatically obtain Screen Used.
   c. Distance between participant and screen (centimetres): The physical distance between the participant and the stimulus screen.
   d. Screen refresh rate (Hz): How many frames per second are presented on the stimulus screen.
   e. Screen physical size (centimetres): The width and height of the used screen in centimetres. If you know these numbers you can fill them in manually. The size can also be obtained automatically using credit card

calibration. In credit card calibration, the user is prompted to scale a rectangle until their credit card covers it completely on the stimulus screen.

3. Tracker type

   The brand name of the tracker that is used. We have tested the application on EyeLink only, so the other options (smi/tobii) might not work completely yet.

4. Pupil size conversion

   EyeLink eye-trackers report pupil size in arbitrary units of diameter or area. To convert these values to millimetres diameter, a scaling factor can be obtained using a fake eye with a known diameter. If you have this scaling factor it can be filled in here. If you do not have the scaling factor, you can leave the factor at 0. Note that not providing a scaling factor will mean that the pupil and gaze biases will be reported in arbitrary units, and thus not be comparable between different setups.

5. Used screen width

   The pupil bias experiment presents the white and black hemispheres at a fixed width of 24.9 visual degrees each. To ensure generalisability screen size will not be fully utilised when the total width is higher than needed. In the opposite case of the screen not being wide enough to present the slides at the correct width, you will be prompted to move the user closer to the screen. Note that this value is not prompted but calculated using the values provided in screen information.

   The current version of the program cuts off images if the screen is too wide. This means that grey bars will appear on the sides of the screen. We currently do not know whether this impacts the data quality.

Defining a participant

All data fields relating to a participant are given hereunder, followed with a small explanation.

1. Alias

An alias that refers to the participant. This could be the acronym of their first and last name.

2. Age

   The age of the participant at the time of testing. This age will be rounded to the nearest 5 years to decrease identifiability.

3. Sex

   The Sex of the participant at the time of testing. Sex can be male, female, or other. When 'other' is selected, any different description can be filled in.

4. Language of instruction

   The language in which the participant is instructed using on-screen prompts during the experiment. To define a set of instructions in a new language, see the chapter '**Defining a language of instruction'**.

5. Handedness

   The handedness of the participant at the time of testing. Can be right, left, or other. When 'other' is selected, any different description can be filled in.

6. Time post injury

   The time in months since the damage from which the Neglect from which the participant suffers was suffered (relative to the testing date)

7. Type of damage

   The type of damage from which the Neglect from which the participant suffers is presumed to originate.

8. Location of damage

   The brain regions implemented in the damage suffered by the participant.

9. Damaged vessels/arteries

   The vessels and arteries implemented in the damage suffered by the participant.

10. Comments on damage

    Any additional comments regarding the damage suffered by the participant.

11. Previously taken tests

    The number of previously taken diagnostic tests performed on the participant. For each test the following information is prompted:

    a. Test name: Name that refers to the test.

b. Test Score: Description of the score the participant achieved.
c. Months post-injury: How many months post-injury the test was performed.
d. Additional observations: Any additional comments regarding the previous test.

12. Known hemianopia

    Whether the participant has been diagnosed with hemianopia, separately from the neglect diagnosis being tested by this application.

13. Previous brain damage

    Whether the participant has suffered other incidences of brain injury.

14. Comments

    Any comments regarding the participant.

Defining a language of instruction

During the experiment, multiple prompts can be shown to the participant. The default language of instruction is English, and Dutch is provided with the program. To add your own language, press '*New'* from the '*View Instructions*' Screen. You will first be prompted to provide the name of the language. After this, you can view English references to all instruction prompts. When you click a reference, you can see the English text and you can fill in the translation of that text in the text field. Try to keep the instructions as minimalistic as possible.

Once you have defined your own language, press 'Save' to actually save the language. From now on, the new language will appear as option when selecting language of presentation for a participant.

Measurement

After you have pressed 'start Exp' and filled in the participant, setup, and session information the selected experiments will launch. When an eye-tracker is correctly connected, calibration specific to the eye tracker will begin now (this is implemented in PyGaze, see **Calibration**). After calibration each selected experiment will begin in the order set previously, see **Experiments**.

Calibration

Calibrate the eye tracker to the participant. One tracked eye should be enough. Try to use as many calibration points as possible. Patients suffering from neglect might not instantly spot calibration points presented in their neglected hemifield.

EyeLink Calibration

You can use the PyGaze menu to calibrate an EyeLink tracker using the PyGaze calibration tool. This is done in four steps:

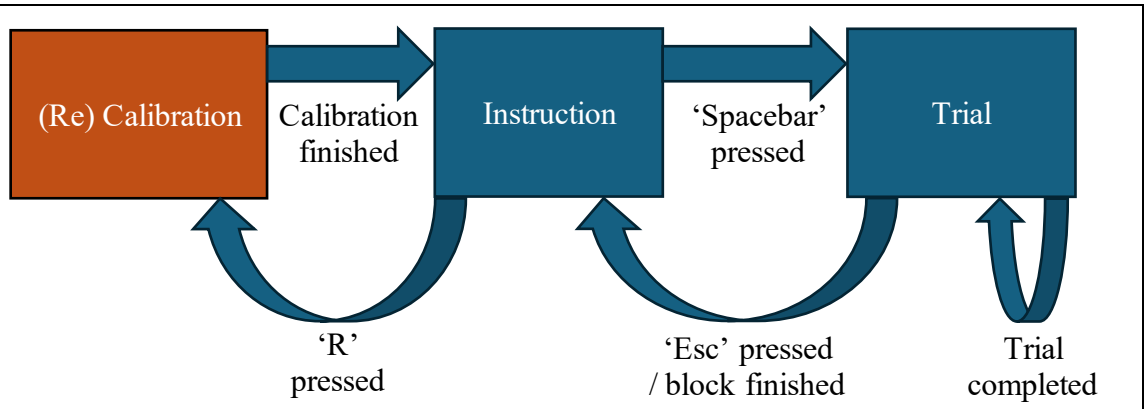
1. Calibrate the threshold values for pupil and corneal reflection (shortcuts: 'Up'/'Down' and '+'/'-' keys respectively). You can look at the camera image from the EyeLink host pc, or press 'Enter' to display the camera image on the stimulus screen and 'Left'/'Right' arrows to switch between different eyes.
2. Calibrate (press 'C', then press 'Spacebar' when the participant is looking the presented point for each point) and Validate (press 'V', then press 'Spacebar' when the participant is looking the presented point for each point)
3. Close the calibration menu using 'Q'. (press twice)
4. Perform noise calibration by pressing spacebar and having the participant fixate the central dot

When calibration is finished, the experiments will start automatically by showing the instruction screen for the first experiment. From an instruction screen, calibration can be restarted by pressing 'R'.

Experiments

Both experiments start at an instruction screen which displays a short instruction about the task to the participant. From this screen, you can press 'R' to recalibrate the eye-tracker or Spacebar to start measuring. While measuring, you can press 'Escape' to return to the instruction screen.

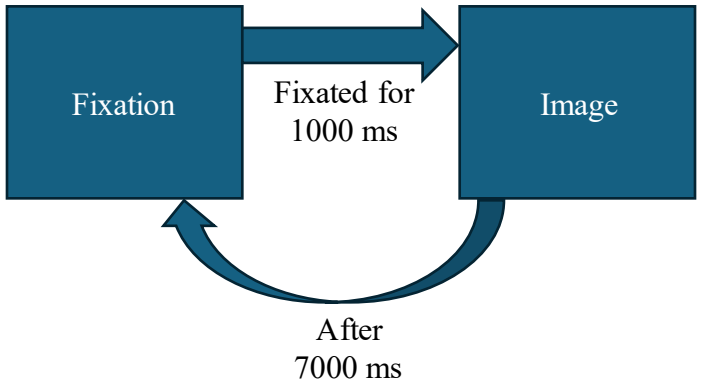
See the figure below for an overview of the possible buttons and where they lead.



Free viewing experiment

The free viewing experiment consists of 10 images that are presented in series, obtained from https://saliency.tuebingen.ai/datasets.html and https://doi.org/10.1007/s00221-024-06823-w . Each image starts after the participant fixates the central fixation point for 1 second. The images can be viewed freely for 7 seconds. During this time, gaze position is recorded. The theory behind this experiment is that patients with neglect will show a strong bias in their gaze position opposite to their neglected visual field hemisphere.

See the figure below for an overview of the free viewing experiment trial loop.

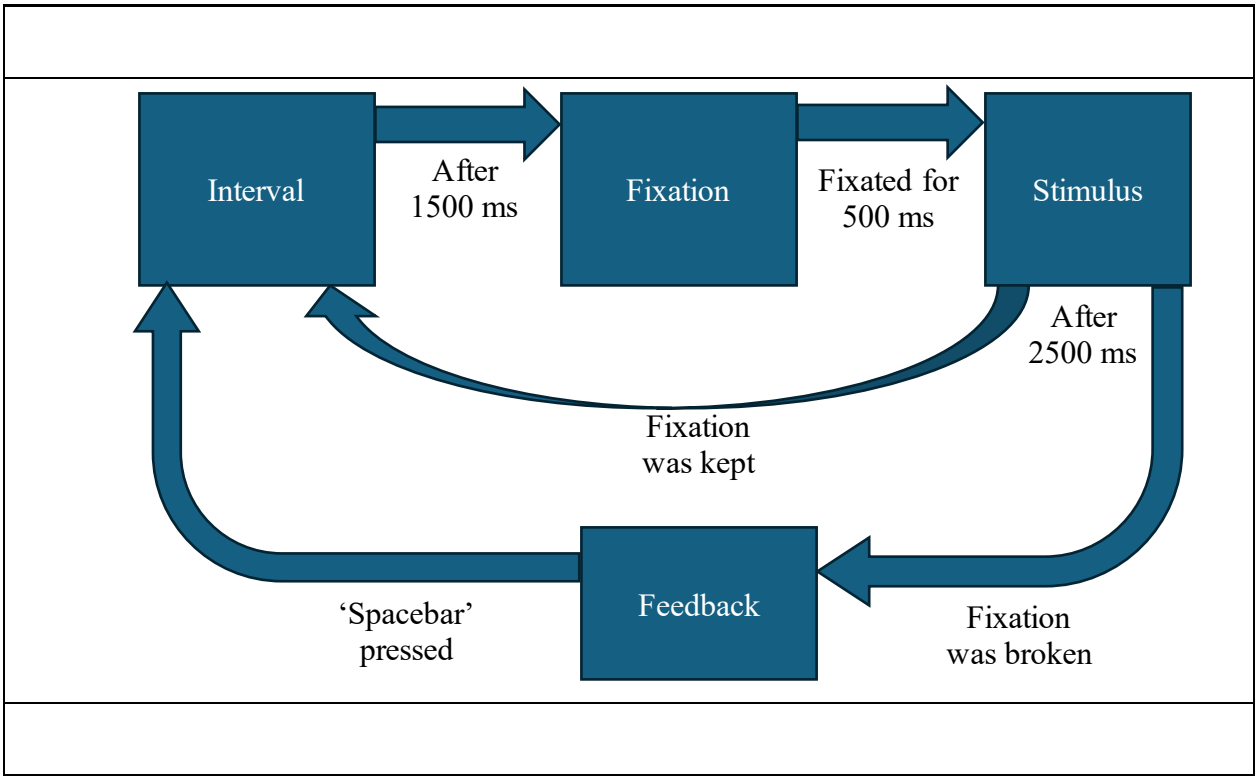


Slideshow experiment

The slideshow experiment consists of four different stimuli (black/white vertical bars) that are presented to the participant. Each stimulus starts after the participant fixates the central fixation point for .5 seconds. Each stimulus is presented for 2.5 seconds. Crucially, the participants are not allowed to freely explore these stimuli, but required to fixate the central fixation point during stimulus presentation. The theory behind this experiment is that patients with neglect will show less pupil constriction when the white bars are presented in the neglected side of their visual field.

As it is crucial that participants keep fixating the central fixation point during stimulus presentation, trials are invalidated and recycled when participants' gaze drifts too far from the central fixation or when participants have blinked for over 500 ms during the stimulus of a trial. If this is the case, a feedback screen is shown when the trial is over. To continue the experiment from a feedback screen, press 'spacebar'. Only six trials per slideshow block will be recycled.

See the figure below for an overview of the slideshow trial loop.



Post-measurement

As of this moment the experiment outputs two files after running. One is a .ses file, which can be viewed from the main .exe program. Additionally, all relevant samples are saved in a .csv file.

The data is also processed automatically at the end of the experiment, and results are printed to the python console. (These numerical results are saved in the .ses file). See the python file DataProcessing.py (found in Import/SessionData) for how the data is processed normally.

Documentation

Overview

The documentation for this project is divided into three distinct sections. Each section contains an overview of the classes that are used. The documentation here only serves as a guide to interpreting the raw code. If you want to view the raw code, please visit our OSF page.

1. TKinter

   This section describes the code behind the interactive widgets that the user can interact with when running the program.

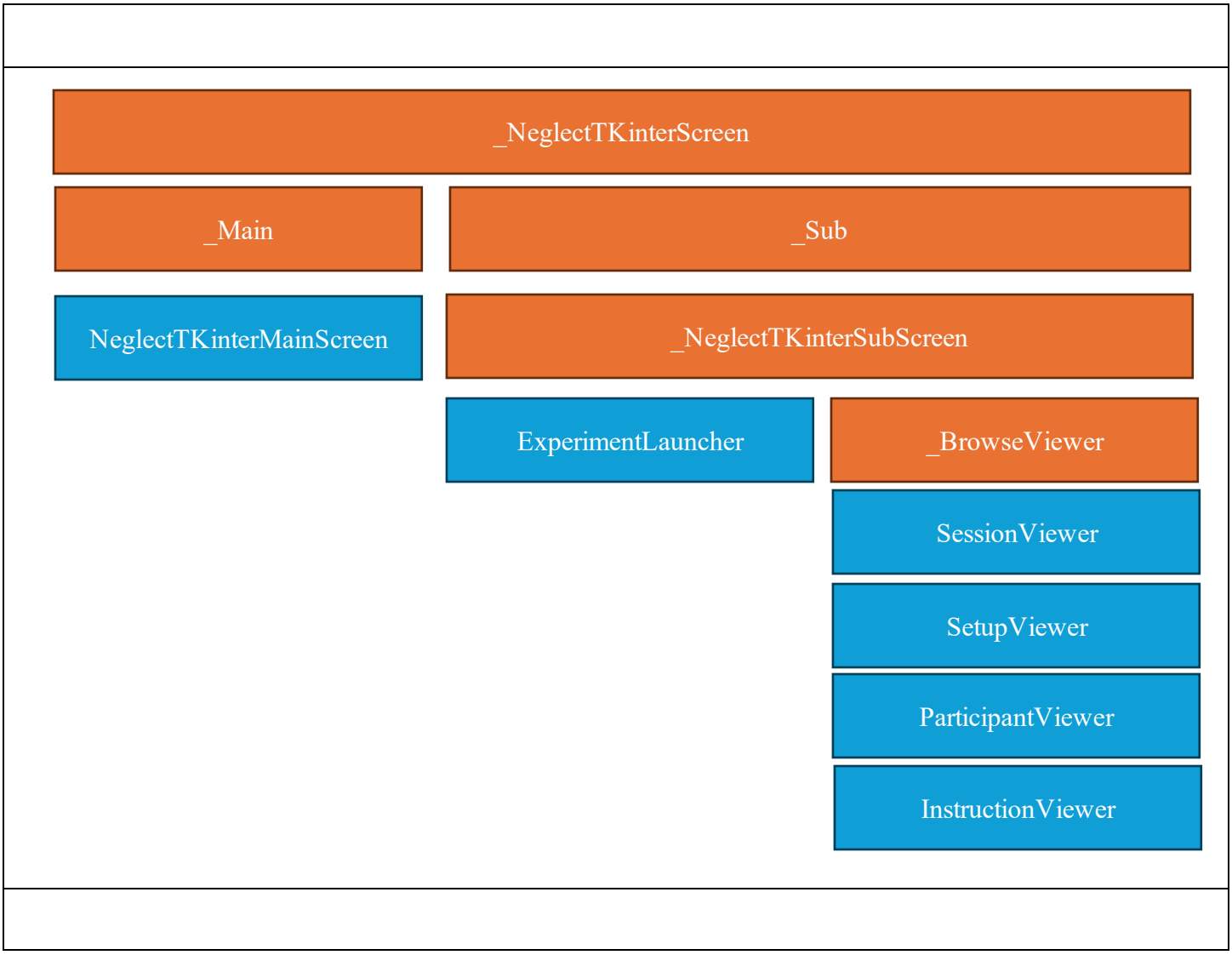
2. I/O

   This section describes the code behind the data formats used to save relevant information about participants, setups, and sessions. Additionally, relevant code used for the loading of other files is described as well.

3. Experiment

   This section describes the code behind the experiments, as well as the pygaze version included with the program (one small change was made to the pygaze library to allow for multiple experiments to be run in succession)

TKinter

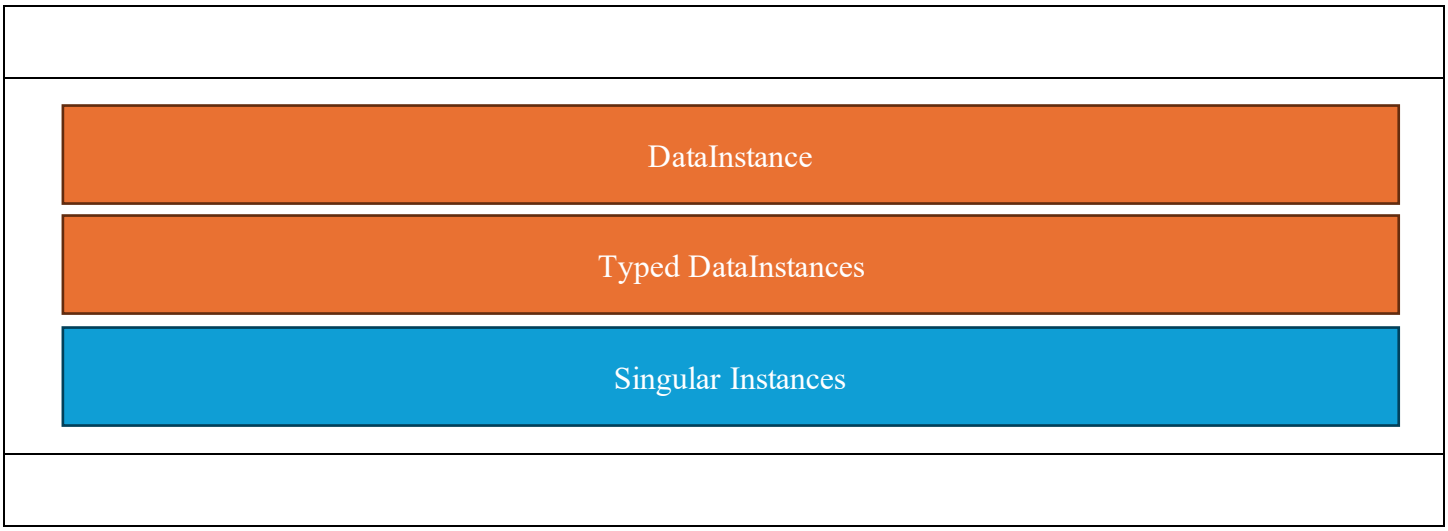

_NeglectTKinterScreen
_Main
_Sub
NeglectTKinterMainScreen
_NeglectTKinterSubScreen
ExperimentLauncher
_BrowseViewer
SessionViewer
SetupViewer
ParticipantViewer
InstructionViewer

I/O

Data Instances

Data Instances are derived from the DataInstance class. They are used for saving, viewing, and editing information about the setup, participant, and session. The file DataInstanceTypes.py holds definitions for the base DataInstance class, as well as the base type classes (Int, Date, Option, etc.). Singular DataInstances are defined in DataInstances.py. For the singular instances, all different instances are defined separately as class (e.g., there are separate object definitions for participant age, screen size, and experiment results).

Singular DataInstances are linked to DataInfo classes in their respective __init__() functions.



DataInfo and TextEditor

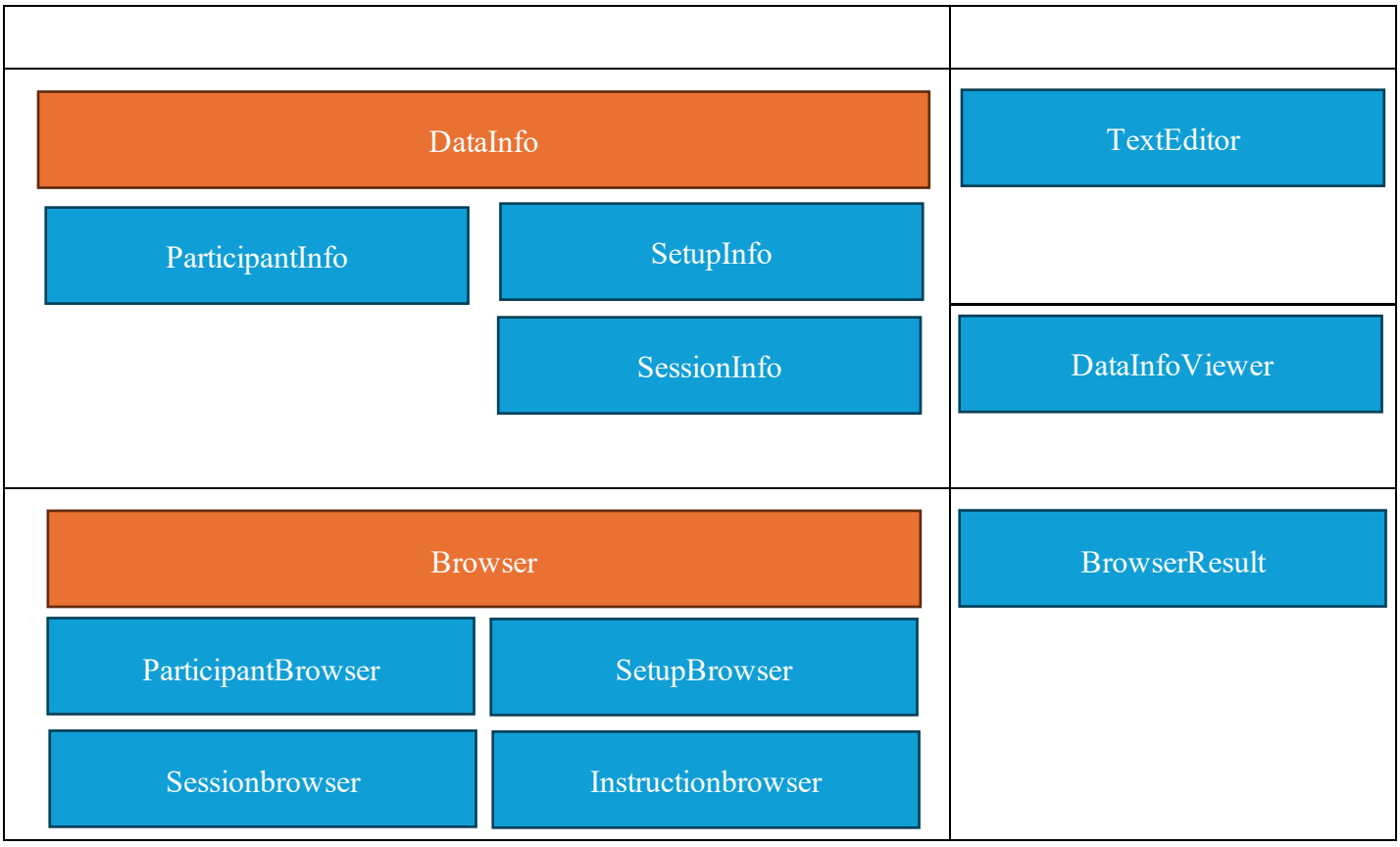
DataInfo objects are a set of DataInstance objects and include some functions for working with the set of DataInstance objects. The sets are bundled into information relating to the participant, setup, and session.

Each DataInfo object is linked to the TKinter class using the DataInfoViewer class. The TextEditor class is different from other DataInfoViewers, as the text is not defined using DataInstances, but with a .txt file. However, the TextEditor does have the same basic functionalities as DataInfoViewer objects.

Browser and BrowserResult

The Browser base class is used to read all files in a related folder, and turn those files into a list of clickable results. For each browsable DataInfo class, a subclass of Browser is used to define the starting folder, file extension, and pictogram. The startPath property of Browser objects returns folders defined in BrowseLocations.py.

For each relevant file found by a browser, a BrowserResult is used as wrapper.



Experiment

OnlineDataCollector

The OnlineDataCollector is used to save eye tracking data during the experiment. This is done by caching a number of samples when the experiment is running. The cached samples are saved whenever a trial is finished. Even though it is not required, baseline values before stimulus are also cached. (for calculating baseline pupil size, the first 200 ms after stimulus presentation are used)

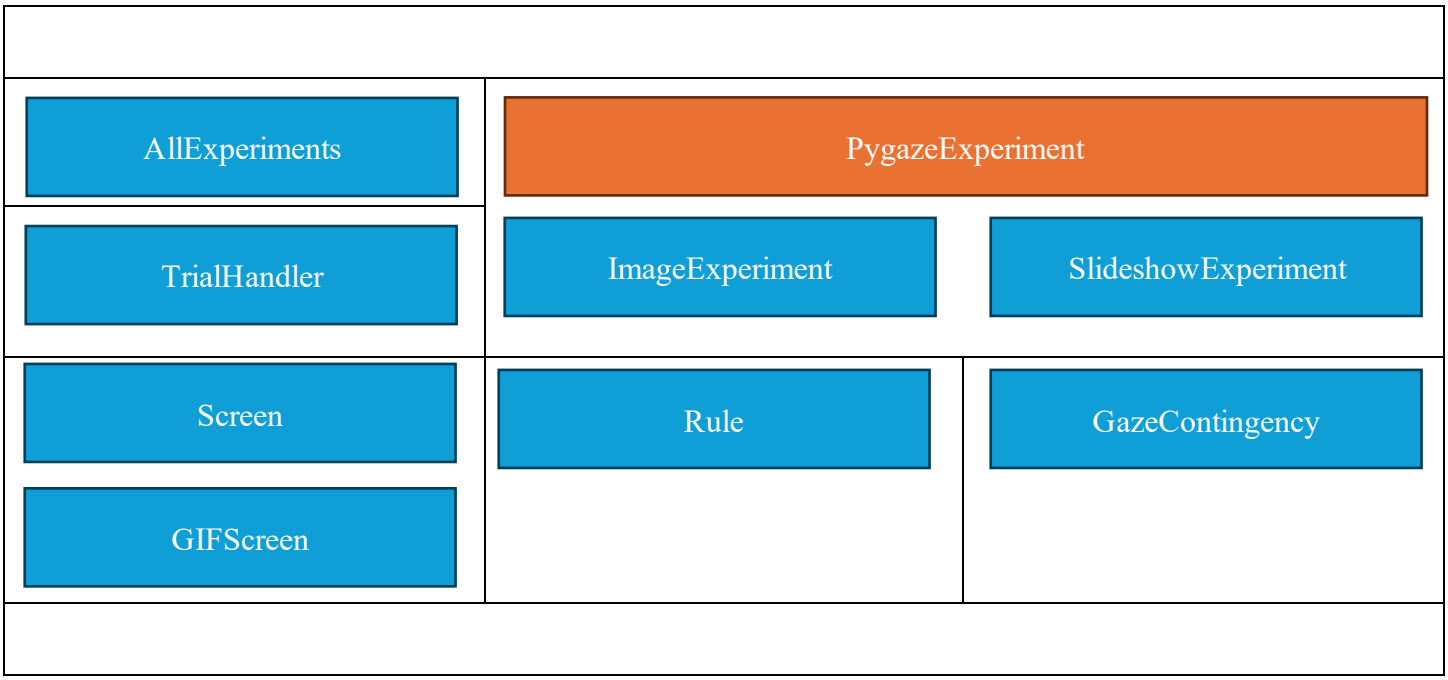
Experiment-related classes and methods

When the experiments are started, the first call is made to the Launch() method in ExperimentLauncher.py. This initialises an AllExperiments object and an OnlineDataCollector.

After this, the AllExperiments object will run each selected experiment. Each experiment is a subclass of PygazeExperiment, with some general behaviour defined in PygazeExperiment, and the experiment-specific behaviour defined in the ImageExperiment or SlideshowExperiment subclass. A TrialHandler object is created by each instance of PygazeExperiment to keep track of trials.

GazeContingency, Rule, and Screen

The GazeContingency class is used to define gaze-contingent behaviour. For this it uses Screen objects, which contain a PyGaze Screen and a list of Rules. The list of rules is evaluated at certain invervals, and when one rule evaluates true, some behaviour is initiated. This behaviour can be the presentation of a different screen, or any custom behaviour defined in-code.


AllExperiments
PygazeExperiment
TrialHandler
ImageExperiment
SlideshowExperiment
Screen
Rule
GazeContingency
GIFScreen