UNIVERSITÄT
BERN

TPF

Fabrikstrasse 8
Büro A209
3012 Bern

031 6315513

GazeToMouse: Usage of the **Tobii Eye Tracker 4C** from within a **ztree** program

Tutorial

Simon Maurer - simon.maurer@humdek.unibe.ch
14th February 2018

Contents

Chapter 1

Introduction

This documents describes how to use the Tobii Eye Tracker 4C in order to track the gaze of a subject and feed the gaze data to the mouse device. This allows to use the eye tracker to control the mouse pointer position such that the mouse pointer is placed at the screen coordinates where the subject is gazing at. This documentation comes with a set of tools (executable .exe files) that provide this functionality as well as some auxiliary tools that help to calibrate and test the eye tracker.

The project aims at providing a set of executables which allow to use the Tobii Eye Tracker 4C in conjunction with ztree to perform economic experiments. The following set of executables are provided:

TobiiCalibrate.exe This program is a simple wrapper for the Tobii calibration tool. It launches the calibration GUI where the subject is led through the calibration process. The calibration data is stored in the current profile of the eye tracker engine.

TobiiGuestCalibrate.exe This program is a simple wrapper for the Tobii guest calibration tool. Similar to `TobiiCalibrate.exe` it launches the calibration tool, however, the calibration data is stored in a guest profile.

TobiiTest.exe This program is a simple wrapper for the Tobii eye tracking testing tool. It launches a GUI where the result of the calibration can be verified and a new calibration process can be started if required.

GazeToMouse.exe This program uses the Tobii Core SDK to get the position on the screen where the subject is looking at. The mouse cursor position is updated to this position. As a consequence, the mouse cursor is controlled by the gaze of the subject. This program runs infinitely until it is terminated by an external command. This should **not** be done with a forced kill (e.g. by executing the command `taskkill /F /IM GazeToMouse.exe` or by killing the task with the task manager) because it prevents the program from terminating gracefully. This as several consequences:

- open files are not closed properly and the data stream is cut off. This can lead to corrupt files.
- if the feature of hiding the mouse pointer is used, the mouse will remain hidden.
- memory is not freed properly.

Instead `taskkill /IM GazeToMouse.exe` should be used. This is done in the program `GazeToMouseClose.exe`.

GazeToMouseClose.exe This program requests GazeToMouse.exe to close gracefully and logs these events to the log file.

ShowMouse.exe This program allows to restore the standard mouse pointer. It might be useful if the program `GazeToMouse.exe` crashes or is closed forcefully such that the mouse pointer is not restored after terminating. The subject might end up with a hidden mouse pointer. A good solution for such a case is to install a shortcut to `ShowMouse.exe` on the desktop in order to execute it with the keyboard.

Chapter 2

Quick Start

In order to get started with a quick experiment you need the following things (*server* refers to the machine running `ztree` and *client* refers to the machines running `zleaf`):

- a ztree installation with a server and one or more clients. To install `ztree` download the latest `ztree` version from the download secetion (requires a license and a login, see Section 2.1) and extract the server file `ztree.exe` to the chosen installation path on the server and the client file `zleaf.exe` to the chosen installation path on each client. Throughout this documentation the installation paths of the `ztree` client and server will be referred to as `<zleaf path>` and `<ztree path>`, respectively.
- the Tobii Eye Tracking Core Software installed on each client. To install the software download and execute the installation file. The installation path of this software will be referred to as `<Tobii path>` throughout this documentation.
- the Tobii Eye Tracker 4C mounted and connected on each client. When encountering connection problems follow these instructions which might help to solve the problem.
- the GazeToMouse toolset installed on each client. To install the toolset download the latest version (requires a subscription, see Section 2.2) and extract all files to an installation path of your choosing. The GazeToMouse installation path will be referred to as `<GazeToMouse path>` throughout this document.
- the ztree sample file. The file is located in `<GazeToMouse path>\sample\template.ztt`. Make sure to change the `Path` variable in `Background` of the sample file such that it points to `<GazeToMouse path>`.

The provided sample `ztree` program performs the following stages:

Calibrate Eye Tracker A guest calibration is performed where the Tobii calibration tool is started and the calibration data is stored in a guest profile within the Tobii eye tracker software. The calibration is very intuitive and has a game-like feel to it.

Test Eye Tracker The Tobii eye tracker test tool is started which allows the subject or experimenter to verify the accuracy of the eye tracker. If the accuracy is not sufficient a recalibration can be started from within the test tool.

Gaze To Mouse The gaze of the subject is tracked and transformed to mouse coordinates which allows to control the mouse pointer with the gaze of the subject.

Terminate Terminates the gaze-to-mouse functionality and concludes this simple `ztree` program.

The behaviour of the GazeToMouse toolset can be configured with a configuration file (see Chapter 4 for more details). A sample configuration file is provided in `<GazeToMouse path>\sample\config.json` that can be modified to suit different requirements. Copy the configuration file either to `<GazeToMouse path>` or `<zleaf path>` and modify it there.

2.1 Requesting Access to the Latest `ztree` Release

To download the latest release of the `ztree` software a login must be requested here. Make a careful note of the terms and conditions.

2.2 Requesting a TPF GitLab Account

The necessary code for what is described in this document is contained in a `GitLab` instance running on the TPF server.

Its address is:

`http://phhum-g111-nns.unibe.ch:10012/`

Please send Simon Maurer a mail with a request, specifying your name, the name of the TPF project you are interested in (e.g `GazeToMouse`) and a user name which you would like to have on `GitLab`.

Chapter 3

Use GazeToMouse Toolset in Ztree

This chapter provides instructions of how to use the Tobii Eye Tracker 4C in a `ztree` program. In order to do this there are two main points to understand:

1. How does the Tobii Eye Tracker 4C work
2. How to execute 3rd party applications from within `ztree`

The first point is addressed in Section 3.1 and the second point in Section 3.2.

3.1 Basic Principles of the Tobii Eye Tracker 4C

The eye tracker is mounted on the bottom of the screen such that its cameras are able to capture the eyes of the subject. Several infrared LEDs are visible once the device is connected and properly working (see Figure 3.1). This requires the Tobii software to be installed and running.

Figure 3.1: Picture of the Tobii Eye Tracker 4C in operation.

In order for the eye tracker to being able to track the gaze of the subject it must be calibrated. A user-friendly calibration tool is included in the Tobii software package. For each person different calibration data must be stored. For this reason the Tobii software supports multiple user profiles which can be switched on the fly. Given that in an economic experiment a workstation is used by a subject only once it is suggested to use the *Guest* profile to store the eye tracker calibration data. Once the eye tracker is calibrated for a subject, everything is ready.

3.2 Execute 3rd Party Applications in `ztree`

In a first step, it is useful to define a global variable `Path` in a `ztree` program that points to the location of the toolset.

In order to define a path

1. click on the last table in `Background`
2. choose the menu `Treatment` → `New Program...`
3. define the path of the installation folder of the `GazeToMouse` toolset, i.e. `<GazeToMouse path>` as `Path` variable (e.g. `Path="C:\\Users\\Max Muster\\Documents\\My Experiment\\GazeToMouse\\";`).
   Note that two backslashes are required for each path delimiter because in ztree \ is used as escape character.

When writing the different stages of a `ztree` program, calls to external applications can be made. This can be achieved as follows:

1. click on the position where you want to include the call to the external program
2. choose the menu `Treatment` → `New External Program...`
3. choose `Run on z-Leaf`
4. add the call to the external program to the field `Command Line`
   (e.g a call to the guest calibration tool: `<><Path|-1>TobiiGuestCalibrate.exe`
   Note that the `Path` variable is used to indicate the location of the program. Figure 3.2 shows an example of calling the Tobii guest calibration tool.

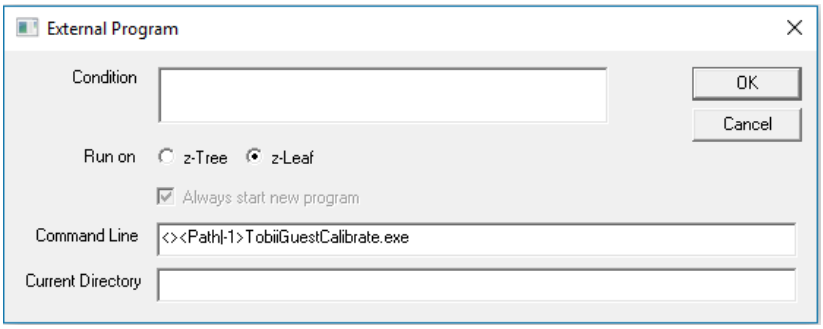

Figure 3.2: External program call definition in `ztree`. The variable `Path` must be globally defined.

Chapter 4

Configuration of the GazeToMouse Toolset

The GazeToMouse toolset can be configured to work with different installations. It allows to specify installation paths and gives some control over implemented features (e.g. mouse hiding or gaze filtering). The Listing 4.1 shows the default configuration values and provides a short explanation for each value.

```
{
    // Defines whether gaze data is written to a log file. If set to false, the configuration items
    // "OuputCount", "OutputPath", and "OutputFormat" are ignored.
    "WriteDataLog": true,

    // Number of output data files to allowed the output path. Oldest files are deleted first. When
    // setting OutputCount to 0, no files are removed. A value of 1 means that only the output of
    // current execution is kept.
    "OutputCount": 200,

    // Defines the location of the output file. It must be the path to a folder (not a file). If empty,
    // the output file is produced in the directory of the caller (e.g the directory of zleaf.exe).
    "OutputPath": "",

    // Allows to define the format of how the gaze data will be logged. Use the .NET syntax to
    // specify the format of individual values:
    // https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
    // Three values are logged:
    // - the timestamp of the gaze point measurement
    // - the x coordinate of the gaze point
    // - the y coordinate of the gaze point
    // The format definition is of the form
    // {0:<format of timestamp>}<delim>{1:<format of x-coordinate>}<delim>{2:<format of y-coordinate>}
    // where all values marked with <> can be customized and where
    // - 0 is replaced by the timestamp
    // - 1 is replaced by the x-coordinate
    // - 2 is replaced by the y-coordinate
    // <delim> can be anything (e.g. \t for tab). Note that the x and y coordinates will be represented
    // differently depending in the localisation settings of the windows installation (e.g. 123,4 for DE_CH
    // or 123.4 for EN_US).
    // When defining <format of timestamp>, special characters (e.g. ':', '.') need to be escaped with '\\'.
    "OutputFormat": "{0:hh\\:mm\\:ss\\.fff}\t{1:0.0}\t{2:0.0}",

    // Defines wheter the mouse cursor shall be controlled by the gaze of the subject during the
    // experiment. If set to true the mouse cursor will be controlled by the gaze of the subject when
    // GazeToMouse.exe is executed and control will be released when GazeToMouseClose.exe is executed.
    // If set to false the configuration item "HideMouse" is ignored.
    "ControlMouse": true,

    // Defines whether the mouse cursor shall be hidden during the experiemnt. If set to true the
    // mouse cursor will be hidden when GazeToMouse.exe is executed and restored when
    // GazeToMousClose.exe is executed.
    "HideMouse": false,
```

```
    // Defines filter settings for the eye tracker (0: unfiltered, 1: lightly filtered).
    "GazeFilter": 0,

    // Defines the Tobii installation path. It must be the path to a folder (not a file).
    "TobiiPath": "C:\\Program Files (x86)\\Tobii",

    // Defines the Path to the standard mouse pointer icon. This is used to restore the mouse pointer.
    "StandardMouseIconPath": "C:\\Windows\\Cursors\\aero_arrow.cur",

    // The Tobii EyeX command to run a calibration.
    "TobiiCalibrate": "Tobii EyeX Config\\Tobii.EyeX.Configuration.exe",

    // The Tobii EyeX parameter to run a calibration.
    "TobiiCalibrateArguments": "--calibrate",

    // The Tobii EyeX command to run a guest calibration.
    "TobiiGuestCalibrate": "Tobii EyeX Config\\Tobii.EyeX.Configuration.exe",

    // The Tobii EyeX parameter to run a guest calibration.
    "TobiiGuestCalibrateArguments": "--guest-calibration",

    // The Tobii EyeX command to run calibration test.
    "TobiiTest": "Tobii EyeX Interaction\\Tobii.EyeX.Interaction.TestEyeTracking.exe"
}
```

Listing 4.1: Default configuartion values

Note that the configuration file follows the `json` syntax which must not be violated. If the following points are respected, no problem should arise:

- the configuration values are enclosed in '{' and '}'.
- each configuration line ends with a ',' except for the last line where it is omitted.
- the Windows path delimiter '\' must be escaped (i.e. write '\\' when describing a path)
- `json` supports standard data types (e.g. integer, boolean, string). Use the same type as the default value.
- everything following a '//' is considered a comment.

Each executable of the toolset uses the same common configuration file. The configuration file must be named `config.json` and is read from the following places with the indicated priority:

1. in the directory of the caller, i.e in `<zleaf path>`
2. in the directory of the executables, i.e. in `<GazeToMouse path>`

If no configuration file can be found, the default values are used.

Warning: A word of warning when using the mouse hiding feature. This feature hides the mouse when running `GazeToMouse.exe`. If `GazeToMouse.exe` is forcefully closed or crashes, the mouse pointer stays hidden. For such cases the `ShowMouse.exe` utility can be used.

4.1 Output File

When running the program `gazeToMouse.exe` an output file is generated which holds the gaze coordinates of the user. The output file is saved in the directory specified by `OutputPath` in `config.json`. The name of the output file follows the form `<yyyyMMddTHHmmss>_<hostName>_gaze.txt` where

- `<yyyyMMddTHHmmss>` is replaced by the timestamp indicating when the file was created (e.g. 20180129T085521 stands for 29.01.2018 08:55:21).
- `<hostName>` is replaced by the name of the machine

4.2 Log File

All executables write continuously to the same log file. This allows to track the eye tracker events that happened throughout a `ztree` session within one log file. The log file is produced at the root directory of the application which is making the calls to the executables (e.g. at the location of zleaf.exe: `<zleaf path>`).