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
15th March 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 (default) or the Tobii Pro 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. The gaze position on the screen (as well as pupil size or eye positions when using Tobii Pro SDK) are further logged to an output file. `GazeToMouse.exe` 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[1] (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[2] 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[3] instructions which might help to solve the problem.
- the GazeToMouse toolset installed on each client. To install the toolset download[4] 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.

[1] https://www.uzh.ch/ztree/ssl-dir/index.php
[2] https://tobiigaming.com/downloadlatest/?bundle=tobii-core
[3] https://help.tobii.com/hc/en-us/articles/115000432589-Is-your-Eye-Tracker-4C-not-connecting-
[4] http://phhum-g111-nns.unibe.ch:10012/TBI/TBI-tobii_eye_tracker_gaze/tree/master/release

2.1 Requesting Access to the Latest `ztree` Release

To download the latest release of the `ztree` software a login must be requested here[5]. 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[6].

Please send Christian Moizzini[7] 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`.

[5] https://www.uzh.ch/ztree/ssl-dir/index.php?action=obtain
[6] http://phhum-g111-nns.unibe.ch:10012/
[7] christian.mozzini@ispw.unibe.ch

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. The guest calibration can be launched by executing the `TobiiGuestCalibrate.exe` program, provided in the toolset. 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 in your `ztree` experiment file 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, gaze filtering, or data logging). The Listing 4.1 shows the default configuration values and provides an explanation for each value. The configuration file contains detailed descriptions for each parameter. It is recommended to go through all the configuration parameters and read the description in order to get an understanding of the available configuration options.

```
{
    // Use this parameter to associate the configuration with an experiment. When "GazeToMouse.exe" is
    // executed, a copy of this configuration file is stored at the "DataLogPath" where the parameter
    // "ConfigName" is postfixed to the filename of the copied config file. E.g., by default the following
    // file will be produced at "DataLogPath": <timestamp>_<computer name>_config_experiment_x.json
    // Note that the following characters are not allowed in a file name: <>:"/\|?*
    "ConfigName": "experiment_x",

    // Allows to define the order and the delimiters between the different values. The definition is
    // of the form
    //  {0}<delim>{1}<delim> ... <delim>{24}
    // where <delim> can be customized (e.g. '\t' for tab, ',' for comma, etc.) and where the numbers are
    // replaced by the following values
    //  - 0: timestamp of the gaze data item (uses DataLogFormatTimeStamp)
    //  - 1: x-coordinate of the gaze point of both eyes (pixel value)
    //  - 2: x-coordinate of the gaze point of the left eye (pixel value) [SDK Pro only]
    //  - 3: x-coordinate of the gaze point of the right eye (pixel value) [SDK Pro only]
    //  - 4: y-coordinate of the gaze point of both eyes (pixel value)
    //  - 5: y-coordinate of the gaze point of the left eye (pixel value) [SDK Pro only]
    //  - 6: y-coordinate of the gaze point of the right eye (pixel value) [SDK Pro only]
    //  - 7: validity of the gaze data of the left eye [SDK Pro only]
    //  - 8: validity of the gaze data of the right eye [SDK Pro only]
    //  - 9: average pupil diameter of both eyes (uses DataLogFormatDiameter) [SDK Pro only]
    //  - 10: pupil diameter of the left eye (uses DataLogFormatDiameter) [SDK Pro only]
    //  - 11: pupil diameter of the right eye (uses DataLogFormatDiameter) [SDK Pro only]
    //  - 12: validity of the pupil data of the left eye [SDK Pro only]
    //  - 13: validity of the pupul data of the right eye [SDK Pro only]
    //  - 14: x-coordinate of the gaze origin of the left eye (uses DataLogFormatOrigin) [SDK Pro only]
    //  - 15: y-coordinate of the gaze origin of the left eye (uses DataLogFormatOrigin) [SDK Pro only]
    //  - 16: z-coordinate of the gaze origin of the left eye (uses DataLogFormatOrigin) [SDK Pro only]
    //  - 17: x-coordinate of the gaze origin of the right eye (uses DataLogFormatOrigin) [SDK Pro only]
    //  - 18: y-coordinate of the gaze origin of the right eye (uses DataLogFormatOrigin) [SDK Pro only]
    //  - 19: z-coordinate of the gaze origin of the right eye (uses DataLogFormatOrigin) [SDK Pro only]
    //  - 20: average distance of the gaze origin of both eyes to the eyetracker (uses DataLogFormatOrigin)
    //        [SDK Pro only]
    //  - 21: distance of the gaze origin of the left eye to the eyetracker (uses DataLogFormatOrigin)
    //        [SDK Pro only]
    //  - 22: distance of the gaze origin of the right eye to the eyetracker (uses DataLogFormatOrigin)
    //        [SDK Pro only]
    //  - 23: validity of the gaze origin data of the left eye [SDK Pro only]
    //  - 24: validity of the gaze origin data of the right eye [SDK Pro only]
    // To log all possible values with a tab (i.e. '\t') as delimiter use the empty string:
```

```
    // "DataLogColumnOrder": "",
    // This configuration value has no effect if "DataLogWriteOutput" is set to false.
    "DataLogColumnOrder": "{0}\t{1}\t{4}",

    // Defines the titles of the data log value columns. A title for all possible columns must be defined.
    // Titles for values that are removed from the "DataLogColumnOrder" parameter will not be logged but
    // must still be defined here. The index of a title must correspond to the value number of the
    // configuration parameter "DataLogColumnOrder".
    // This configuration value has no effect if "DataLogWriteOutput" is set to false.
    "DataLogColumnTitle": [
        "Timestamp",
        "coord-x",
        "coord-x-left",
        "coord-x-right",
        "coord-y",
        "coord-y-left",
        "coord-y-right",
        "coord-valid-left",
        "coord-valid-right",
        "pupil-dia",
        "pupil-dia-left",
        "pupil-dia-right",
        "pupil-valid-left",
        "pupil-valid-right",
        "origin-x-left",
        "origin-y-left",
        "origin-z-left",
        "origin-x-right",
        "origin-y-right",
        "origin-z-right",
        "origin-dist",
        "origin-dist-left",
        "origin-dist-right",
        "origin-valid-left",
        "origin-valid-right"
    ],

    // Number of maximal allowed output data files in the output path. Oldest files are deleted first. To
    // keep all files set the value to 0. A value of 1 means that only the output of the current execution
    // is kept.
    // Note that if multiple clients write to the same folder, this value should be set to at least the
    // number of clients.
    // This configuration value has no effect if "DataLogWriteOutput" is set to false.
    "DataLogCount": 200,

    // Allows to define the format of how the pupil diameter (in millimetres) will be logged. Use the .NET
    // syntax to specify the format:
    //  https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
    // Note that the numbers 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).
    // This configuration value has no effect if "DataLogWriteOutput" is set to false.
    "DataLogFormatDiameter": "0.000",

    // Allows to define the format of how the gaze origin values (in millimetres) will be logged. Use the
    // .NET syntax to specify the format:
    //  https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
    // Note that the numbers 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).
    // This configuration value has no effect if "DataLogWriteOutput" is set to false.
    "DataLogFormatOrigin": "0.000",

    // Allows to define the format of the timestamp. Use the .NET syntax to specify the format:
    //  https://docs.microsoft.com/en-us/dotnet/standard/base-types/formatting-types
    // Note that special characters (e.g. ':', '.') need to be escaped with '\\'.
    // This configuration value has no effect if "DataLogWriteOutput" is set to false.
    "DataLogFormatTimeStamp": "hh\\:mm\\:ss\\.fff",
```

```
    // When set to true, invalid data will be ignored and not logged to the output file. When set to false
    // invalid data will appear as "NaN" or "n. def" (depending on the language of the system) in the output
    // file. A data set is invalid if the tracker is not able to detect the required features of the eye.
    // This parameter might be useful to estimate the number of blinks of the subject or to track when the
    // subject is not gazing at the screen.
    // This configuration value has no effect for the Core SDK where invalid data is always ignored.
    // Note that invlaid data is only removed if it affects all values associated to a timestamp.
    "DataLogIgnoreInvalid": false,

    // 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).
    // This configuration value has no effect if "DataLogWriteOutput" is set to false.
    // To avoid confusion with path locations it is recommended to use absolute paths, e.g.:
    //  C:\\Users\\Subject\\Documents
    "DataLogPath": "",

    // Defines whether gaze data is written to a log file. If set to false, all the configuration items
    // matching the pattern "DataLog*" are ignored.
    "DataLogWriteOutput": true,

    // Defines filter settings for the eye tracker:
    //  - 0: unfiltered, Tobii: "removal of invalid data points and the averaging of the gaze points from
    //       both eyes".
    //  - 1: lightly filtered, Tobii: "an adaptive filter which is weighted based on the age of the gaze
    //       data points GazePointData and the velocity of the eye movements".
    // also see https://tobii.github.io/CoreSDK/articles/streams.html
    // Note that this parameter only has effect for Tobii SDK Core (see parameter "TobiiSDK").
    "GazeFilterCore": 0,

    // Defines the location of the license files. It must be the path to a folder (not a file).
    // This is only required when Tobii SDK Pro is used (see parameter "TobiiSDK")
    // To avoid confusion with path locations it is recommended to use absolute paths, e.g.:
    //  C:\\Users\\Subject\\Documents\\tobii_licenses
    "LicensePath": "tobii_licenses",

    // 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.
    "MouseControl": 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.
    // This parameter is ignored if "MouseControl" is set to false.
    "MouseHide": false,

    // Defines the Path to the standard mouse pointer icon. This is used to restore the mouse pointer.
    // This parameter is ignored if "MouseControl" or "MouseHide" is set to false.
    "MouseStandardIconPath": "C:\\Windows\\Cursors\\aero_arrow.cur",

    // Specifies the amount of time (in milliseconds) to wait for the eyetracker to become ready while it
    // is in any other state. If the eyetracker is not ready within the specified time the subject will
    // be notified with a popup window.
    "ReadyTimer": 5000,

    // Choose the Tobii SDK (0: Tobii Core SDK, 1: Tobii Pro SDK).
    // Note that Tobii SDK Pro requires a license file to work (see parameter "LicesePath").
    "TobiiSDK": 0,

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

    // 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 parameters are enclosed in '{' and '}'.
- all configuration parameters are of the form `"key":value` where `"key"` must not be changed.
- 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 to restore the mouse pointer.

4.1 Output File

When running the program `gazeToMouse.exe` an output file can be generated which holds the gaze data provided by the Tobii engine. 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

An output file is only generated if the parameter `"DataLogWriteOutput"` in the configuration file is set to `true` (which is the default). The presentation of the gaze data is configurable with the help of the parameters `"DataLogColumnOrder"`, `"DataLogColumnTitle"`, `"DataLogFormatDiameter"`, `"DataLogFormatOrigin"`, and `"DataLogFormatTimeStamp"` (see Listing 4.1 for more details).

By default the Tobii Core SDK is used (parameter `"TobiiSDK"` is set to 0) and only the timestamps and the x and the y coordinates of the gaze points are available. A gaze point describes a point on the screen in x and y coordinates (pixel

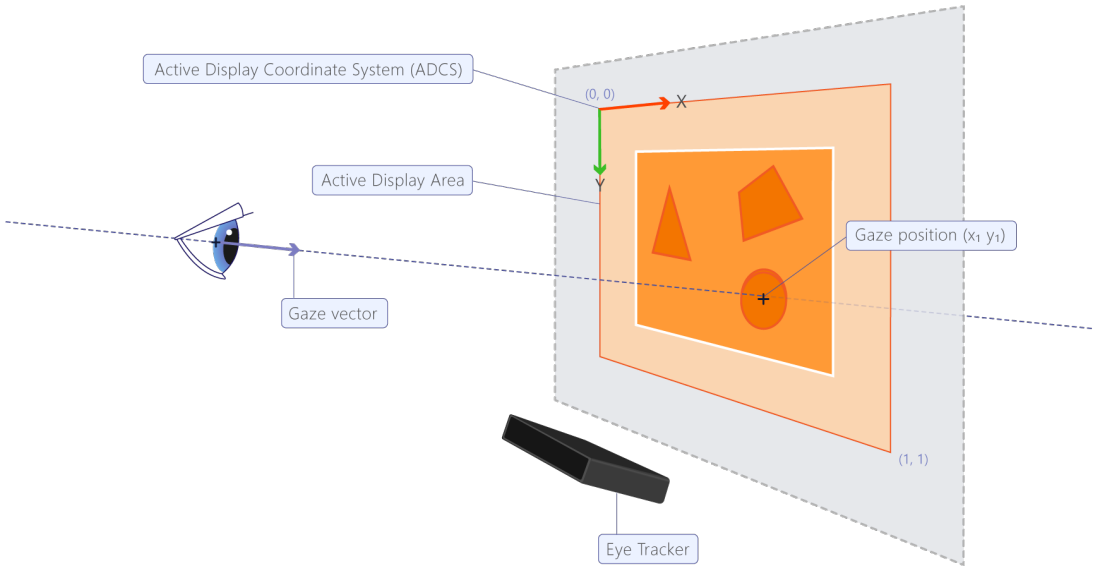

Figure 4.1: The Active Display Coordinate System (ADCS). Figure source: Tobii

values) the user is gazing at. This coordinate system is called *Active Display Coordinate System (ADCS)* and is illustrated in Figure 4.1.

By default, the system is configured to log the timestamp of when the gaze point was captured by the eye tracker (data field number 0) and the x and y coordinates of the gaze point (data field numbers 1 and 4, respectively). Hence, the parameter `"DataLogColumnOrder"` is set to the value `"{0}\t{1}\t{4}"` in order to not bloat the output file with unnecessary header data. This produces an output file that is similar to the following:

```
Timestamp   coord-x coord-y
12:51:55.409    426 342
12:51:55.420    430 341
12:51:55.431    431 341
...
```

When opening the file with a spreadsheet software such as LibreOffice Calc or Microsoft Office Excel two things need to be considered:

1. The column delimiter needs to be set to the delimiter set in the configuration file (`'\t'` by default).
2. The language needs to be set to the language of the windows system where the configuration file was produced. This is important because numbers are represented differently in different languages and depending on the settings, commas might be interpreted as delimiters when they are not.

When using Tobii Pro SDK, much more values are provided and can be logged to the output file. This includes the pupil diameters of each eye as well as the eye positions in space. The latter uses a coordinate system which is called *User Coordinate System (UCS)*. The UCS is illustrated in Figure 4.2.

In order to log additional data fields, the parameter `"DataLogColumnOrder"` must be modified by adding the required numbers, representing different data fields. For example, to log all raw data fields (omitting any data field that is computed from raw data) the following string can be used:

```
"DataLogColumnOrder": "{0}\t{2}\t{3}\t{5}\t{6}\t{7}\t{8}\t{10}\t{11}\t{12}\t{13}\t{14}\t{15}\t{16}\t{17}\t
    {18}\t{19}\t{23}\t{24}"
```

When using the empty string, all possible data fields are logged:

```
"DataLogColumnOrder": ""
```

4.1.1 Data Fields

The comments above the parameter `"DataLogColumnOrder"` in the configuration file (see Listing 4.1) provide a short description for each available data field. In the following some further information is provided.

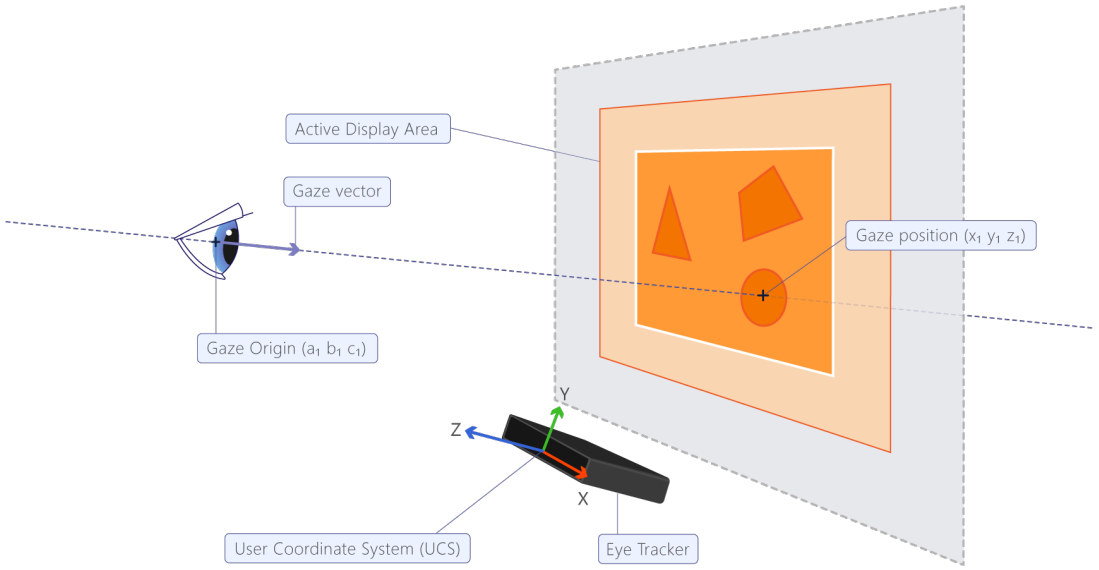

Figure 4.2: The User Coordinate System (UCS). Figure source: Tobii

The following five data field types with individual data fields are provided:

1. timestamps indicating when the data was sampled by the eye tracker (field number: 0)
2. gaze point ADCS coordinates (see Figure 4.1) as pixel values

   Tobii Core SDK

   - the average x and y coordinates where filtering depends on parameter "GazeFilterCore" (field numbers: 1 and 4)

   Tobii Pro SDK

   - raw x and y coordinates of the left and the right eye (field numbers: 2, 3, 5, and 6)
   - the average x and y coordinates which are computed from raw data as defined by Equation 4.1 (field numbers: 1 and 4)

$$\text{val} = \begin{cases} \text{NaN} & \text{if val\_right and val\_left are not valid} \\ \text{val\_left} & \text{if val\_right is not valid} \\ \text{val\_right} & \text{if val\_left is not valid} \\ \frac{\text{val\_left}+\text{val\_right}}{2} & \text{otherwise} \end{cases} \tag{4.1}$$

3. gaze origin UCS coordinates (see Figure 4.2) in millimetres

   Tobii Core SDK not supported

   Tobii Pro SDK

   - raw x, y, and z coordinates of the left and the right eye (field numbers: 14 - 19)
   - the distance from the eye tracker to the left and the right eye, respectively which are computed from raw data as defined by Equation 4.2 (field numbers: 21 and 22)

$$\text{dist} = \sqrt{x^2 + y^2 + z^2} \tag{4.2}$$

   - the average distance which is computed from the distance of the left and the right eye as defined by Equation 4.1 (field number: 20)

4. pupil diameters in millimetres

   Tobii Core SDK not supported

   Tobii Pro SDK

- raw diameters of the left and the right eye
  (field numbers: 10 and 11)
- the average diameter which is computed from raw data as defined by Equation 4.1
  (field number: 9)

5. data validity indicators as true or false

   Tobii Core SDK not supported

   Tobii Pro SDK

   - separate values that indicate whether gaze points, gaze origins, and pupil diameters are valid for the left and the right eye, respectively
     (field numbers: 7, 8, 12, 13, 23, 24)

4.1.2 Configuration File Dump

For each experiment where the utility `GazeToMouse.exe` is executed, a dump of the configuration file is produced. This allows to associate a set of configuration values to an experiment, reuse the same configuration file should the experiment be repeated, and provides transparency of how the output data was produced. Note that in this configuration file all comments are omitted and the formatting (indentations, white spaces, carriage return) is removed. To reformat the file or display the file as a tree structure, online tools, such as the Online JSON Viewer[1], can be used.

The name of the dumped configuration file is of the form `<yyyyMMddTHHmmss>_<hostName>_config_<ConfigName>.json` 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
- `<ConfigName>` is replaced by the configuration value `"ConfigName"` as specified in the configuration file (see Listing 4.1 for more details)

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>`). The name of the log file is of the form `<hostName>_gaze.log` where `<hostName>` is replaced by the name of the machine.

[1] http://jsonviewer.stack.hu/