SR Research PyLink

Version: 2.1.762.0

Generated: July 20, 2023

Generated by Doxygen 1.9.1

Chapter 1

Introduction

1.1 Introduction

Performing research with eye-tracking equipment typically requires carefully implemented software tools to collect, process, and analyze data. Much of these tools involve tracker calibration, real-time data collection, and so on.

The EyeLink® eye-tracking system is most powerful when used with the Ethernet link interface, which allows flexible control of data collection and real-time data transfer. The PyLink module implements all core EyeLink functions and classes for EyeLink connection and the EyeLink graphics, such as the display of the camera image, calibration, validation, and drift-check. The EyeLink graphics included in the EyeLink Developer's Kit are currently implemented using the SDL library (www.libsdl.org). However, users can implement custom EyeLink graphics with a library they prefer to use for stimulus presentation (see the PsychoPy and Pygame examples bundled in the EyeLink Developer's Kit).

1.1.1 Organization of This Document

EyeLink Programming Conventions of this document introduces the standard programming convention recommended for an EyeLink experiment, and the standard EyeLink messages that can be used to facilitate data analysis and visualization in Data Viewer. Overview of PyLink module documents the common functions and classes used in the PyLink module in detail. You will rarely need other functions or classes than those listed here.

1.1.2 Getting Started

Please refer to the EyeLink Installation Guide (https://www.sr-support.com/forum-34.html) for instructions on how to set up the stimulus presentation PC (hereby referred to as the "Display PC"). The PyLink library also comes with a Getting Started Guide for Python and PyLink, which clearly explains the installation of Python and PyLink, the example scripts provided by us, the typical structure of an EyeLink experiment, and some troubleshooting tips. For general knowledge about Python programming, you may refer to the tutorials available on the official Python website, https://www.python.org/.

1.2 EyeLink Programming Conventions

The PyLink library contains a set of classes and functions, which are used to program experiments on many different platforms, such as Windows, Linux (e.g., Ubuntu), and macOS. This chapter will outline the programming convention of a typical EyeLink experiment, and the standard messages that would allow seamless integration with analysis tools such as EDF2ASC converter or EyeLink Data Viewers.

1.2.1 Outline of a Typical Windows Experiment

To help programmers understand how to write experiment software, and how to port existing experiments to the EyeLink platform, this section outlines the standard operations involved in an EyeLink experiment. A typical experiment using the EyeLink eye-tracker involves some variation of the following sequence of operations:

- Open a link connection to the EyeLink Host PC.
- Set up an EDF file name, and open an EDF data file on the EyeLink Host PC.
- Send configuration commands to the EyeLink Host PC to prepare it for the experiment.
- Open a full-screen window and configure the calibration graphics routines.
- Record one or more blocks of trials. Each block typically begins with tracker setup (camera setup and calibration), and then several trials are run.
- Close the EDF data file. If desired, retrieve the file via the link to the Host PC.
- Close the link connection to the Host PC and terminate the task.

For each trial, the experiment will do these steps:

- Send a message ("TRIALID") to the Host PC to mark the start of a trial.
- Send a record status message to show on the Host PC screen (optional).
- Draw background graphics (image or landmarks) on the Host PC display (optional).
- Perform a drift-check (or drift-correction).
- Start data recording.
- Present experimental stimuli for the trial, and retrieve real-time eye movement data if needed.
- Stop data recording.
- Send a message ("TRIAL_RESULT") to the Host PC to mark the end of a trial.

This sequence of operations is the core of almost all experiments (see the accompanying "Getting Started with Python and PyLink" doc for an example illustrating the above concept). A real experiment would probably add practice trials, instruction screens, randomization, and so on.

During recording, all eye-tracking data and events are usually written into the EDF file, which is saved on the eye tracker's hard disk, and may be copied to the Display PC at the end of the experiment. Your experiment will also add messages to the EDF file to identify trial conditions and to timestamp important events (such as participant responses and display changes) for use in analysis. The EDF file may be processed directly using the EyeLink Data Viewer (https://www.sr-support.com/forum-7.html), or converted to an ASC file and processed by your own software.

1.2.2 Standard Messages

Experiments should place certain messages into the EDF file, to mark the start and end of trials. These messages will facilitate the SR Research viewing and analysis applications (e.g., Data Viewer) to process the EDF files.

Text messages can be sent to the EyeLink Host PC and added to the EDF file along with the eye movement data. These messages will be time stamped with an accuracy of 1 millisecond from the time sent, and can be used to mark important events such as display changes. Be careful not to send messages too quickly: the eye tracker can handle about 20 messages every 10 milliseconds. Above this rate, some messages may be lost before being written to the EDF file.

To facilitate data analysis in EyeLink Data Viewer, special messages can be added to the EDF file. Examples of these messages include those that specify the overlay image, the interest areas, and the trial variables. Detailed information about the various standard messages that Data Viewer recognizes can be found in the "EyeLink Data Viewer user manual" (https://www.sr-support.com/thread-135.html).

- The "TRIALID" message is sent at the beginning of a trial, before the start of data recording. It is not mandatory, but it is recommended to include a unique trial identifier in the message, for instance, "TRIALID 13".
- The "TRIAL_RESULT" message is sent after the recording ends. Importantly, the variables that would be used for analysis (i.e., the "TRIAL_VAR" messages) should be written in the EDF file before the "TRIAL_↩ RESULT" message. Additional info can be included in the "TRAIL_RESULT" message, e.g., to indicate the result, for instance "TRIAL_RESULT 0".
- The other critical message that worth noting here is the "DISPLAY_COORDS" message. This message is used by Data Viewer to figure out the appropriate screen size (in pixels) for visualizing the eye movement data, e.g., "DISPLAY_COORDS 0 0 1023 767". The four integers in the message mark the left, top, right, and bottom of the screen.
- One or several "!V TRIAL_VAR" messages should be sent to report the experiment condition(s) of a recording trial. These messages should generally be sent at the end of each trial, allowing variables that may get updated during the recording, such as response time, accuracy, etc., to be included. These messages should be sent before the TRIAL_RESULT message, which marks the end of the trial.
- One of several "!V IAREA" messages can be written into the EDF files to allow Data Viewer to display interest areas during analysis.
- One or several "!V IMGLOAD" messages can be sent to the EDF file to specify the background images for fixation/saccade/heatmap data visualization.
- Other Data Viewer integration messages can be used to play back video stimuli, draw positions of target traces, or draw simple graphics such as boxes, lines, or circles to mark object locations on the screen. Please see section "Protocol for EyeLink Data to Viewer Integration" of the EyeLink Data Viewer User Manual.
- Users should also send messages to the EDF file to mark the critical events in a trial (e.g., onset/offset of a critical display, start/end of the audio playing, participant's keyboard/mouse/button responses. These messages can be used in Data Viewer to create interest period or reaction time definition for data filtering.

1.3 Overview of PyLink module

The interaction of the EyeLink tracker and your experiment is fairly sophisticated: for example, in the Setup menu it is possible to perform calibration or validation, display a camera image on the Display PC to aid in participant setup, and record data with the experiment following along by displaying proper graphics. Large files can be transferred over the link, and should the EyeLink tracker software terminate, the experiment application is automatically terminated as well. Keys pressed on the Display PC keyboard are transferred to the EyeLink PC and operate the tracker during setup, and buttons pressed on the tracker button box may be used to control the execution of the experiment on the Display PC.

PyLink, a Python wrapper for the EyeLink core API, implements all of the functions and classes for EyeLink connection and graphics, such as the display of camera image, calibration, validation, and drift correct. Each of the above operations required just one line of code in your program. Almost all of the source code you will need to write is for the experiment itself, e.g., the control of trials, the presentation and generation of visual and auditory stimuli, and error handling.

Please see a detailed description of all functions in the pyLink module and all classes in this module under Classes section.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Module Documentation

3.1 Tracker Data Type Constants

3.1.1 Detailed Description

3.2 Event Type Flags

The following specifies what types of events were written by tracker.

3.2.1 Detailed Description

The following specifies what types of events were written by tracker.

3.3 Event Data Flags

The following specifies what types of data were included in events by tracker.

3.3.1 Detailed Description

The following specifies what types of data were included in events by tracker.

3.4 Special Key values for the tracker

The following specifies special key values for the tracker.

3.4.1 Detailed Description

The following specifies special key values for the tracker.

3.5 Tracker Mode values

Set of bit flags that mark mode function:

3.5.1 Detailed Description

Set of bit flags that mark mode function:

3.6 EyeLink Graphics Functions

Functions

- def openGraphicsEx (eyeCustomDisplay)

  Allow one to configure the graphics with EyeLinkCustomDisplay.
- def openGraphics (*args)

  Opens the graphics if the display mode is not set.
- def setCalibrationColors (*args)

  Passes the colors of the display background and fixation target to the eyelink_core_graphics library.
- def setTargetSize (*args)

  The standard calibration and drift correction target is a disk (for peripheral delectability) with a central "hole" target (for accurate fixation).
- def setCalibrationSounds (*args)

  Selects the sounds to be played during `do_tracker_setup()`, including calibration, validation and drift correction.
- def setDriftCorrectSounds (*args)

  Selects the sounds to be played during `doDriftCorrect()`.
- def setCameraPosition (*args)

  Sets the camera position on the display computer.
- def getDisplayInformation (*args)

  Returns the display configuration.
- def closeGraphics ()

  Notifies the eyelink_core_graphics or the EyeLinkCustomDisplay to close or release the graphics.

3.6.1 Detailed Description

3.6.2 Function Documentation

3.6.2.1 openGraphicsEx()

```
def pylink.openGraphicsEx (
              eyeCustomDisplay )
```

Allow one to configure the graphics with EyeLinkCustomDisplay.

See EyeLinkCustomDisplay for more details.

```
genv = EyeLinkCoreGraphicsPsychoPy(el_tracker, win)
pylink.openGraphicsEx(genv)
```

Parameters

eyeCustomDisplay	instance of EyeLinkCustomDisplay for the desired platform. This cannot be used with eyelink_core_graphics library or openGraphics

3.6.2.2 openGraphics()

```
def pylink.openGraphics (
            * args )
```

Opens the graphics if the display mode is not set.

If the display mode is already set, uses the existing display mode.

Remarks

This is equivalent to the SDL version C API
`INT16 init_expt_graphics(SDL_Surface * s, DISPLAYINFO *info);`

Parameters

dimension	Two-item tuple of display containing width and height information.
bits	Color bits.

Returns

None or run-time error.

Remarks

This function only works with SDL 1.2 in conjunction with eyelink_core_graphics library and cannot be used with EyeLinkCustomDisplay

3.6.2.3 setCalibrationColors()

```
def pylink.setCalibrationColors (
            * args )
```

Passes the colors of the display background and fixation target to the eyelink_core_graphics library.

During calibration, camera image display, and drift correction, the display background should match the brightness of the experimental stimuli as closely as possible, in order to maximize tracking accuracy. This function passes the colors of the display background and fixation target to the eyelink_core_graphics library. This also prevents flickering of the display at the beginning and end of drift correction.

Remarks

This is equivalent to the C API
`void set_calibration_colors(SDL_Color *fg, SDL_Color *bg);`

Parameters

foreground_color	Color for foreground calibration target.
background_color	Color for foreground calibration background.

Both colors must be a three-integer (from 0 to 255) tuple encoding the red, blue, and green color component.

Example:

```
setCalibrationColors((0, 0, 0), (255, 255, 255))
This sets the calibration target in black and calibration background in white.
```

Remarks

This function only works with SDL 1.2 in conjunction with eyelink_core_graphics library and cannot be used with EyeLinkCustomDisplay

3.6.2.4 setTargetSize()

```
def pylink.setTargetSize (
            * args )
```

The standard calibration and drift correction target is a disk (for peripheral delectability) with a central "hole" target (for accurate fixation).

The sizes of these features may be set with this function.

Remarks

This function is equivalent to the C API

```
void set_target_size(UINT16 diameter, UINT16 holesize);
```

Parameters

diameter	Size of outer disk, in pixels.
holesize	Size of central feature, in pixels. If holesize is `0`, no central feature will be drawn. The disk is drawn in the calibration foreground color, and the hole is drawn in the calibration background color.

Remarks

This function only works with SDL 1.2 in conjunction with eyelink_core_graphics library and cannot be used with EyeLinkCustomDisplay

3.6.2.5 setCalibrationSounds()

```
def pylink.setCalibrationSounds (
            * args )
```

Selects the sounds to be played during `do_tracker_setup()`, including calibration, validation and drift correction.

These events are the display or movement of the target, successful conclusion of calibration or good validation, and failure or interruption of calibration or validation.

Remarks

If no sound card is installed, the sounds are produced as "beeps" from the PC speaker. Otherwise, sounds can be selected by passing a string. If the string is "" (empty), the default sounds are played. If the string is "off", no sound will be played for that event. Otherwise, the string should be the name of a .WAV file to play. This function is equivalent to the C API

```
void set_cal_sounds(char *target, char *good, char *error);
```

Parameters

target	Sets sound to play when target moves.
good	Sets sound to play on successful operation.
error	Sets sound to play on failure or interruption.

Remarks

This function only works with SDL 1.2 in conjunction with eyelink_core_graphics library and cannot be used with EyeLinkCustomDisplay

3.6.2.6 setDriftCorrectSounds()

```
def pylink.setDriftCorrectSounds (
            * args )
```

Selects the sounds to be played during `doDriftCorrect()`.

These events are the display or movement of the target, successful conclusion of drift correction, and pressing the 'ESC' key to start the Setup menu.

Remarks

If no sound card is installed, the sounds are produced as "beeps" from the PC speaker. Otherwise, sounds can be selected by passing a string. If the string is "" (empty), the default sounds are played. If the string is "off", no sound will be played for that event. Otherwise, the string should be the name of a .WAV file to play. This function is equivalent to the C API

```
void set_dcorr_sounds(char *target, char *good, char *setup);
```

Parameters

target	Sets sound to play when target moves.
good	Sets sound to play on successful operation.
setup	Sets sound to play on 'ESC' key pressed.

Remarks

This function only works with SDL 1.2 in conjunction with eyelink_core_graphics library and cannot be used with EyeLinkCustomDisplay

3.6.2.7 setCameraPosition()

```
def pylink.setCameraPosition (
            * args )
```

Sets the camera position on the display computer.

Moves the top left hand corner of the camera position to new location.

Parameters

left	X coordinate of upper-left corner of the camera image window.
top	Y coordinate of upper-left corner of the camera image window.
right	X coordinate of lower-right corner of the camera image window.
bottom	Y coordinate of lower-right corner of the camera image window.

Remarks

This function only works with SDL 1.2 in conjunction with eyelink_core_graphics library and cannot be used with EyeLinkCustomDisplay

3.6.2.8 getDisplayInformation()

```
def pylink.getDisplayInformation (
            * args )
```

Returns the display configuration.

Returns

Instance of DisplayInfo class. The width, height, bits, and refresh rate of the display can be accessed from the returned value.

Example:

```
display = getDisplayInformation()
print display.width, display.height, display.bits, display.refresh
```

3.7 EyeLink Utility Functions

Functions

- def currentDoubleUsec ()
- def inRealTimeMode ()
- def endRealTimeMode ()
- def flushGetkeyQueue ()
- def getLastError ()
- def bitmapSave ()
- def getDisplayAPIVersion ()
- def closeMessageFile ()
- def currentUsec ()
- def alert ()
- def beginRealTimeMode ()
- def currentTime ()
- def msecDelay ()
- def openMessageFile ()
- def pumpDelay ()
- def getEYELINK ()

  <> *Returns the EyeLink tracker object*

3.7.1 Detailed Description

3.7.2 Function Documentation

3.7.2.1 currentDoubleUsec()

```
def currentDoubleUsec ( )
```

Returns the current microsecond time (as a double type) since the initialization of the EyeLink library.

Remarks

Same as `currentUsec()` except, this function can return large microseconds. That is the `currentUsec()` can return up to `2147483648` microseconds starting from initialization. The `currentDoubleUsec()` can return up to `36028797018963968` microseconds. This is equivalent to the C API

```
double current_double_usec(void);
```

Returns

A float data for the current microsecond time since the initialization of the EyeLink library.

3.7.2.2 inRealTimeMode()

```
def inRealTimeMode ( )
```

returns whether the current mode is real-time.

Returns

`1` if in realtime mode, else `0`.

3.7.2.3 endRealTimeMode()

```
def endRealTimeMode ( )
```

Returns the application to a priority slightly above normal, to end realtime mode. This function should execute rapidly, but there is the possibility that Windows will allow other tasks to run after this call, causing delays of 1-20 milliseconds.

Remarks

This function is equivalent to the C API
```
void end_realtime_mode(void);
```

3.7.2.4 flushGetkeyQueue()

```
def flushGetkeyQueue ( )
```

Initializes the key queue used by `getkey()`. It may be called at any time to get rid any of old keys from the queue.

Remarks

This is equivalent to the C API
```
void flush_getkey_queue(void);
```

3.7.2.5 getLastError()

```
def getLastError ( )
```

Retrieves a tuple containing the error number and error message generated by last Pylink call to have failed

Remarks

This has no equivalence in C API. This can be used to help identify the Runtime Error last raised by a Pylink call

Returns

(Error-Number, Error-Message) tuple or `(0,'')` if no Runtime Error has been raised.

Example:

```
try:
    getEYELINK().waitForBlockStart(500,1,0)
except RuntimeError:
    if getLastError()[0] == 0: # wait time expired without link data
        print ("ERROR: No link samples received!")
        return TRIAL_ERROR
    else: # for any other status simply re-raise the exception
        raise
```

See also

commandResult()

3.7.2.6 bitmapSave()

```
def bitmapSave ( )
```

This function saves the entire bitmap or selected part of a bitmap in an image file (with an extension of .png, .bmp, .jpg, or .tif). It creates the specified file if this file does not exist.

Parameters

iwidth	Original image width.
iheight	Original image height.
pixels	Pixels of the image in one of two possible formats: pixel=[line1, line2, ... linen] line=[pix1,pix2,...,pixn],pix=(r,g,b). pixel=[line1, line2, ... linen] line=[pix1,pix2,...,pixn],pix=0xAARRGGBB.
xs	Crop x position.
ys	Crop y position.
width	Crop width.
height	Crop height.
fname	File name to save.
path	Path to save.
svoptions	Save options(`SV_NOREPLACE`, `SV_MAKEPATH`). If the file exists, it replaces the file unless SV_NOREPLACE is specified.

This function is equivalent to the C API el_bitmap_save()

3.7.2.7 getDisplayAPIVersion()

```
def getDisplayAPIVersion ( )
```

Returns text associated with last command response: may have error message.

Remarks

This is equivalent to the C API
```
INT16 eyelink_dll_version(char *buf);
```

Returns

Text associated with last command response or None.

3.7.2.8 closeMessageFile()

```
def closeMessageFile ( )
```

Flush and close message file, opened by openMessageFile().

3.7.2.9 currentUsec()

```
def currentUsec ( )
```

Returns the current microsecond time since the initialization of the EyeLink library.

Remarks

This is equivalent to the C API
```
UINT32 current_usec(void);
```

Returns

Long integer for the current microsecond time since the initialization of the EyeLink library.

3.7.2.10 alert()

```
def alert ( )
```

This method is used to give a notification to the user when an error occurs.

Remarks

This function does not allow printf formatting as in c. However you can do a formatted string argument in python. This is equivalent to the C API
```
void alert_printf(char *fmt, ...);
```

Parameters

message	Text message to be displayed.

3.7.2.11 beginRealTimeMode()

```
def beginRealTimeMode ( )
```

Sets the application priority and cleans up pending Windows activity to place the application in realtime mode. This could take up to 100 milliseconds, depending on the operation system, to set the application priority.

Remarks

This function is equivalent to the C API
```
void begin_realtime_mode(UINT32 delay);
```

Parameters

delay	An integer, used to set the minimum time this function takes, so that this function can act as a useful delay.

3.7.2.12 currentTime()

```
def currentTime ( )
```

Returns the current millisecond time since the initialization of the EyeLink library.

Remarks

This function is equivalent to the C API
```
UINT32 current_time(void);
```

Returns

Long integer for the current millisecond time since the initialization of the EyeLink library.

3.7.2.13 msecDelay()

```
def msecDelay ( )
```

Does a unblocked delay using `currentTime()`.

Remarks

This is equivalent to the C API
```
void msec_delay(UINT32 n);
```

Parameters

delay	An integer for number of milliseconds to delay.

3.7.2.14 openMessageFile()

```
def openMessageFile ( )
```

Creates message file, once open call to sendMessageToFile(), will not send messages to tracker. Messages are kept in a queue if the application is in realtime mode, and written to disk on non real-time mode except when closeMessageFile() is called while in real-time mode.

Parameters

in	*fname*	Message file name

3.7.2.15 pumpDelay()

```
def pumpDelay ( )
```

During calls to `msecDelay()`, Windows is not able to handle messages. One result of this is that windows may not appear. This is the preferred delay function when accurate timing is not needed. It calls `pumpMessages()` until the last 20 milliseconds of the delay, allowing Windows to function properly. In rare cases, the delay may be longer than expected. It does not process modeless dialog box messages.

Remarks

This is equivalent to the C API
```
void pump_delay(UINT32 delay);
```

Parameters

delay	An integer, which sets number of milliseconds to delay.

3.7.2.16 getEYELINK()

```
def pylink.getEYELINK ( )
```

<> Returns the EyeLink tracker object

The returned instance had been created when calling `EyeLink()`.

Remarks

This function replaces the previous convention of using `EYELINK`. (`EYELINK` is no longer constant, it is now initialized with `None`)

Chapter 4

Class Documentation

4.1 ButtonEvent Class Reference

A ButtonEvent class, derived from the IOEvent class, is created to handle specifics of button events.

Inherits IOEvent.

Public Member Functions

- def getButtons (self)

  A list of buttons pressed or released.
- def getStates (self)

  The button state (`1` for pressed or `0` for released).

4.1.1 Detailed Description

A ButtonEvent class, derived from the IOEvent class, is created to handle specifics of button events.

For this class, in addition to the generic IOEvent class methods, two additional methods can be used for instances of the ButtonEvent class.

Returned by `getFloatData()` whenever there is a Button Event.

4.1.2 Member Function Documentation

4.1.2.1 getButtons()

```
def getButtons (
              self )
```

A list of buttons pressed or released.

Returns

List of integers.

4.1.2.2 getStates()

```
def getStates (
              self )
```

The button state (`1` for pressed or `0` for released).

Returns

List of integers.

4.2 DisplayInfo Class Reference

The DisplayInfo class is used to contain information on display configurations, including width, height, color bits, and refresh rate.

Public Attributes

- width

  Display width in screen pixels.
- height

  Display height in screen pixels.
- bits

  Color resolution, in bits per pixel, of the display device (for example: 4 bits for 16 colors, 8 bits for 256 colors, or 16 bits for 65,536 colors).
- refresh

  Refresh rate.

4.2.1 Detailed Description

The DisplayInfo class is used to contain information on display configurations, including width, height, color bits, and refresh rate.

The current display configuration can be retrieved by the `getDisplayInformation()` function of the pylink module.

For example:

```
from pylink import *
// Code to open the display
currentDisplay = getDisplayInformation();
   print("Current display settings: ", currentDisplay.width, currentDisplay.height, \
print "Current display settings: ", currentDisplay.width, currentDisplay.height, \
               currentDisplay.bits, currentDisplay.refresh
```

4.2.2 Member Data Documentation

4.2.2.1 refresh

```
refresh
```

Refresh rate.

4.3 EndBlinkEvent Class Reference

Class to represent End Blink event.

Inherits EyeEvent.

Public Member Functions

- def getEndTime (self)
  Timestamp of the last sample of the blink.

4.3.1 Detailed Description

Class to represent End Blink event.

This also contains the Start Blink data. This also inherits all properties from EyeEvent.

4.3.2 Member Function Documentation

4.3.2.1 getEndTime()

```
def getEndTime (
              self )
```

Timestamp of the last sample of the blink.

Returns

Long integer.

4.4 EndFixationEvent Class Reference

Class to represent End Fixation event.

Inherits StartFixationEvent, and EndNonBlinkEvent.

Public Member Functions

- def getAverageGaze (self)

  The average gaze position during the fixation period (in pixel coordinates set by the `screen_pixel_coords` command).
- def getAverageHREF (self)

  Average HEADREF position during the fixation period.
- def getAveragePupilSize (self)

  Average pupil size (in arbitrary units, area or diameter as selected) during a fixation.
- def getEndPupilSize (self)

  Pupil size (in arbitrary units, area or diameter as selected) at the end of a fixation interval.

4.4.1 Detailed Description

Class to represent End Fixation event.

This also contains the Start Fixation data. This also inherits all properties from StartFixationEvent and EndNonBlinkEvent.

4.4.2 Member Function Documentation

4.4.2.1 getAverageGaze()

```
def getAverageGaze (
              self )
```

The average gaze position during the fixation period (in pixel coordinates set by the `screen_pixel_coords` command).

Returns

Two-item tuple in the format of (float, float).

4.4.2.2 getAverageHREF()

```
def getAverageHREF (
              self )
```

Average HEADREF position during the fixation period.

Returns

Two-item tuple in the format of (float, float).

4.4.2.3 getAveragePupilSize()

```
def getAveragePupilSize (
              self )
```

Average pupil size (in arbitrary units, area or diameter as selected) during a fixation.

Returns

Float.

4.4.2.4 getEndPupilSize()

```
def getEndPupilSize (
              self )
```

Pupil size (in arbitrary units, area or diameter as selected) at the end of a fixation interval.

Returns

Float.

4.5 EndNonBlinkEvent Class Reference

This interface is never used as is.

Inherited by EndFixationEvent, EndSaccadeEvent, and FixUpdateEvent.

Public Member Functions

- def getEndTime (self)
  Timestamp of the last sample of the event.
- def getEndGaze (self)
  Gaze position at the end of the event (in pixel coordinates set by the `screen_pixel_coords` *command).*
- def getEndHREF (self)
  HEADREF position at the end of the event.
- def getEndVelocity (self)
  Gaze velocity at the end of the event (in visual degrees per second).
- def getAverageVelocity (self)
  Average gaze velocity during event (in visual degrees per second).
- def getPeakVelocity (self)
  Peak gaze velocity during event (in visual degrees per second).
- def getEndPPD (self)
  Angular resolution at the end of the event (in screen pixels per visual degree).

4.5.1 Detailed Description

This interface is never used as is.

EndFixationEvent, EndSaccadeEvent and FixUpdateEvent types inherit this.

Please note that the `getStartTime()` and `getEndTime()` methods of the event class are the timestamps of the first and last samples in the event. To compute duration, subtract these two values and add 4 msec (even in 500 Hz tracking modes, the internal parser of EyeLink II quantizes event times to 4 milliseconds).

Peak velocity returned by `getPeakVelocity()` for fixations is usually corrupted by terminal segments of the preceding and following saccades.

The `getStartPPD()` and `getEndPPD()` methods contain the angular resolution at the start and end of the saccade or fixation. The average of the start and end angular resolution can be used to compute the size of saccades in degrees. This Python code would compute the true magnitude of a saccade from an `ENDSACC` event in the following way:

```
newEvent = getEYELINK().getFloatData()
if isinstance(newEvent, EndFixationEvent):
    Start_Gaze = newEvent.getStartGaze()
    Start_PPD = newEvent.getStartPPD()
    End_Gaze = newEvent.getEndGaze()
    End_PPD = newEvent. getEndPPD()
dx = (End_Gaze[0] - Start_Gaze[0]) / ((End_PPD[0] + Start_PPD[0])/2.0);
dy = (End_Gaze[1] - Start_Gaze[1]) / ((End_PPD[1] + Start_PPD[1])/2.0);
dist = math.sqrt(dx*dx + dy*dy);
```

Please note that the average velocity for saccades may be larger than the saccade magnitude divided by its duration because of overshoots and returns.

4.5.2 Member Function Documentation

4.5.2.1 getEndTime()

```
def getEndTime (
              self )
```

Timestamp of the last sample of the event.

Returns

Long integer.

4.5.2.2 getEndGaze()

```
def getEndGaze (
              self )
```

Gaze position at the end of the event (in pixel coordinates set by the `screen_pixel_coords` command).

The first and second items of the returned tuple store the x- and y- gaze position respectively.

Returns

Two-item tuple in the format of (float, float).

4.5.2.3 getEndHREF()

```
def getEndHREF (
              self )
```

HEADREF position at the end of the event.

The first and second items of the returned tuple store the x- and y- coordinate HREF data respectively.

Returns

Two-item tuple in the format of (float, float).

4.5.2.4 getEndVelocity()

```
def getEndVelocity (
              self )
```

Gaze velocity at the end of the event (in visual degrees per second).

Returns

Float.

4.5.2.5 getAverageVelocity()

```
def getAverageVelocity (
              self )
```

Average gaze velocity during event (in visual degrees per second).

Returns

Float.

4.5.2.6 getPeakVelocity()

```
def getPeakVelocity (
              self )
```

Peak gaze velocity during event (in visual degrees per second).

Returns

Float.

4.5.2.7 getEndPPD()

```
def getEndPPD (
              self )
```

Angular resolution at the end of the event (in screen pixels per visual degree).

Returns

Two-item tuple in the format of (float, float).

4.6 EndSaccadeEvent Class Reference

Class to represent End Saccade event.

Inherits StartSaccadeEvent, and EndNonBlinkEvent.

Public Member Functions

- def getAmplitude (self)

  Returns the amplitude between the start and end of the event.
- def getAngle (self)

  Returns the angle between the start and end of the event.

4.6.1 Detailed Description

Class to represent End Saccade event.

This also contains the Start Saccade data. This also inherits all properties from StartSaccadeEvent and EndNonBlinkEvent.

4.6.2 Member Function Documentation

4.6.2.1 getAmplitude()

```
def getAmplitude (
              self )
```

Returns the amplitude between the start and end of the event.

Returns

Amplitude between the start and end of the event.

4.6.2.2 getAngle()

```
def getAngle (
              self )
```

Returns the angle between the start and end of the event.

Returns

Angle between the start and end of the event.

4.7 EyeEvent Class Reference

The EyeLink tracker simplifies data analysis by detecting important changes in the sample data and placing corresponding events into the data stream.

Inherited by EndBlinkEvent, StartBlinkEvent, and StartNonBlinkEvent.

Public Member Functions

- def getTime (self)

  Timestamp of the sample causing event (in milliseconds since EyeLink tracker was activated).
- def getType (self)

  The event code.
- def getEye (self)

  Which eye produced the event: `0` (`LEFT_EYE`) or `1` (`RIGHT_EYE`).
- def getRead (self)

  Bits indicating which data fields contain valid data (see eye_data.h file for details of the bits information).
- def getStartTime (self)

  Timestamp of the first sample of the event.
- def getStatus (self)

  Errors and warnings of the event.

4.7.1 Detailed Description

The EyeLink tracker simplifies data analysis by detecting important changes in the sample data and placing corresponding events into the data stream.

These include eye-data events (blinks, saccades, and fixations), button events, input-port events, and messages. Several classes have been created to holds eye event data (start/end of fixation, start/end of saccade, start/end of blink, fixation update) information. Start events contain only the start time, and optionally the start eye or gaze position.
End events contain the start and end time, plus summary data on saccades and fixations.

It is important to remember that data sent over the link does not arrive in strict time sequence.
Typically, eye events (such as `STARTSACC` and `ENDFIX`) arrive up to 32 milliseconds after the corresponding samples, and messages and buttons may arrive before a sample with the same time code. This differs from the order seen in an ASC file, where the events and samples have been sorted into a consistent order by their timestamps.

The `LOST_DATA_EVENT` is a new event, introduced for EyeLink tracker version 2.1 and later, and produced within the DLL to mark the location of lost data. It is possible that data may be lost, either during recording with real-time data enabled, or during playback. This might happen because of a lost link packet or because data was not read fast enough (data is stored in a large queue that can hold 2 to 10 seconds of data, and once it is full the oldest data is discarded to make room for new data). This event has no data or time associated with it.

Event data returned by the `getFloatData()` method of the EyeLink class.

For example,
```
newEvent = getEYELINK().getFloatData()
```

Right now, the EyeLink Core implements the following eye events:

Constant Name	Value	Description
STARTBLINK	3	Pupil disappeared (time only)
ENDBLINK	4	Pupil reappeared (duration data)
STARTSACC	5	Start of saccade (with time only)
ENDSACC	6	End of saccade (with summary data)
STARTFIX	7	Start of fixation (with time only)
ENDFIX	8	End of fixation (with summary data)
FIXUPDATE	9	Update within fixation (summary data for interval)
MESSAGEEVENT	24	User-definable text (IMESSAGE structure)
BUTTONEVENT	25	Button state change (IOEVENT structure)
INPUTEVENT	28	Change of input port (IOEVENT structure)
SAMPLE_TYPE	200	Event flags gap in data stream

Please note that due to the tracker configuration, some of the property information returned may be a missing value `MISSING_DATA` (or `0`, depending on the field). So make sure you check for the validity of the data before trying to use them. To do the tracker configuration, the user can use the `setLinkEventFilter()` and `setLinkEventData()` methods of the EyeLink class to send commands at the start of the experiment or modify the DATA.INI file on the tracker PC.

When both eyes are being tracked, left and right eye events are produced. The eye from which data was produced can be retrieved by the `getEye()` method.

4.7.2 Member Function Documentation

4.7.2.1 getTime()

```
def getTime (
              self )
```

Timestamp of the sample causing event (in milliseconds since EyeLink tracker was activated).

Returns

Long integer.

4.7.2.2 getType()

```
def getType (
              self )
```

The event code.

Returns

Integer.

4.7.2.3 getEye()

```
def getEye (
              self )
```

Which eye produced the event: `0` (`LEFT_EYE`) or `1` (`RIGHT_EYE`).

Returns

Integer.

4.7.2.4 getRead()

```
def getRead (
              self )
```

Bits indicating which data fields contain valid data (see eye_data.h file for details of the bits information).

Returns

Integer.

4.7.2.5 getStartTime()

```
def getStartTime (
              self )
```

Timestamp of the first sample of the event.

Returns

Long integer.

4.8 EyeLink Class Reference

The EyeLink class is an extension of the EyeLinkListener class with additional utility functions.

Inherits EyeLinkListener.

Public Member Functions

- def __init__ (self, trackeraddress="100.1.1.1")
  Constructor.
- def progressUpdate (self, size, received)
  prints out the file transfer progress to the console.
- def progressSendDataUpdate (self, size, sent)
  prints out the sending file progress to the console.
- def setSampleSizeForVelAndAcceleration (self, sm)
  Sets the sample model to be used for velocity and acceleration calculation.
- def getSampleSizeForVelAndAcceleration (self)
  Returns the sample model used for velocity and acceleration calculation.
- def setVelocityAccelerationModel (self, sm)
  Sets the sample model to be used for velocity and acceleration calculation.
- def getVelocityAccelerationModel (self)
  Returns the sample model used for velocity and acceleration calculation in text form.
- def getTrackerAddress (self)
  Returns the tracker address.
- def getDummyMode (self)
  Returns whether in dummy mode or not.
- def doTrackerSetup (self, width=None, height=None)
  Switches the EyeLink tracker to the Setup menu, from which camera setup, calibration, validation, drift correction, and configuration may be performed.
- def setAcceptTargetFixationButton (self, button)
  This programs a specific button for use in drift correction.
- def setCalibrationType (self, type)
  This command sets the calibration type, and recomputed the calibration targets after a display resolution change.
- def setXGazeConstraint (self, x_position="AUTO")
  Locks the x part of gaze position data.
- def setYGazeConstraint (self, y_position="AUTO")
  Locks the y part of gaze position data.
- def enableAutoCalibration (self)

Enables the auto calibration mechanism.

- def disableAutoCalibration (self)

  Disables the auto calibration mechanism.
- def setAutoCalibrationPacing (self, pace)

  Sets automatic calibration pacing.
- def readIOPort (self, ioport)

  Sends a command to the tracker to read the specified io port.
- def writeIOPort (self, ioport, data)

  Sends a command to the tracker to write the specified io port.
- def setHeuristicLinkAndFileFilter (self, linkfilter, filefilter=-1)

  EyeLink II only: Can be used to set level of filtering on the link and analog output, and on file data.
- def setHeuristicFilterOn (self)

  EyeLink 1 Only: Can be used to enable filtering, increases system delay by 4 msec if the filter was originally off.
- def setHeuristicFilterOff (self)

  EyeLink 1 Only: Can be used to disable filtering, reduces system delay by 4 msec.
- def setPupilSizeDiameter (self, value)

  Can be used to determine pupil size information to be recorded.
- def setSimulationMode (self, value)

  Can be used to turn off head tracking if not used.
- def setScreenSimulationDistance (self, distance)

  Used to compute correct visual angles and velocities when head tracking not used.
- def markPlayBackStart (self)

  Marks the location in the data file from which playback will begin at the next call to `EYEYLINK.startPlayBack()`.
- def setNoRecordEvents (self, message=False, button=False, inputevent=False)

  Selects what types of events can be sent over the link while not recording (e.g between trials).
- def setFileSampleFilter (self, list)

  Sets data in samples written to EDF file.
- def setFileEventData (self, list)

  Sets data in events written to EDF file.
- def setFileEventFilter (self, list)

  Sets which types of events will be written to EDF file.
- def setLinkSampleFilter (self, list)

  Sets data in samples sent through link.
- def setLinkEventData (self, list)

  Sets data in events sent through link.
- def setLinkEventFilter (self, list)

  Sets which types of events will be sent through link.
- def setSaccadeVelocityThreshold (self, vel)

  Sets velocity threshold of saccade detector: usually `30` for cognitive research, `22` for pursuit and neurological work.
- def setAccelerationThreshold (self, accel)

  Sets acceleration threshold of saccade detector: usually `9500` for cognitive research, `5000` for pursuit and neurological work.
- def setMotionThreshold (self, deg)

  Sets a spatial threshold to shorten saccades.
- def setPursuitFixup (self, maxvel)

  Sets the maximum pursuit velocity accommodation by the saccade detector.
- def setUpdateInterval (self, time)

  Normally set to `0` to disable fixation update events.
- def setFixationUpdateAccumulate (self, time)

  Normally set to `0` to disable fixation update events.

- def setRecordingParseType (self, rtype="GAZE")

  Sets how velocity information for saccade detection is computed.
- def drawText (self, text, pos=(-1,-1))

  Draws text, coordinates are gaze-position display coordinates.
- def clearScreen (self, color)

  Clear tracker screen for drawing background graphics or messages.
- def drawLine (self, firstPoint, secondPoint, color)

  Draws line, coordinates are gaze-position display coordinates.
- def drawBox (self, x, y, width, height, color)

  Draws an empty box, coordinates are gaze-position display coordinates.
- def drawFilledBox (self, x, y, width, height, color)

  Draws a solid block of color, coordinates are gaze-position display coordinates.
- def getFixationUpdateInterval (self)

  Returns the fixation update interval value.
- def getFixationUpdateAccumulate (self)

  Returns the fixation update accumulate value.
- def setFixationUpdateInterval (self, interval)

  Sends a command to the tracker to update the FixationUpdateInterval.
- def setFixationUpdateAccumulate (self, accumulate)

  Sends a command to the tracker to update the FixationUpdateAccumulate.
- def echo (self, text, pos=(-1,-1))

  Prints text at current print position to tracker screen, gray on black only.
- def drawCross (self, x, y, color)

  Draws a small "+" to mark a target point.

4.8.1 Detailed Description

The EyeLink class is an extension of the EyeLinkListener class with additional utility functions.

Most of these functions are used to perform tracker setups (For current information on the EyeLink tracker configuration, examine the ∗.INI files in the EYELINK2\EXE\ or ELCL\EXE\ directory of the eye tracker computer). An instance of the EyeLink class can be created by using the class constructor function. For example,

```
try:
    EYELINK = EyeLink()
except:
    EYELINK = None
```

An instance of EyeLink class can directly use all of the EyeLinkListener methods. In addition, it has its own methods as listed in the following. All of the methods should be called in the format of: EYELINK.functionName(parameters), where EYELINK is an instance of EyeLink class.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 __init__()

```
def __init__ (
              self,
              trackeraddress = "100.1.1.1" )
```

Constructor.

Parameters

trackeraddress	optional tracker address. If no parameters passed in, default address of 100.1.1.1 is used. If None is passed as the address, the connection is opened in dummy mode.

4.8.3 Member Function Documentation

4.8.3.1 progressUpdate()

```
def progressUpdate (
              self,
              size,
              received )
```

prints out the file transfer progress to the console.

This is called after the call to receiveDataFile

Parameters

size	Size of the file.
received	Size received so far.

4.8.3.2 progressSendDataUpdate()

```
def progressSendDataUpdate (
              self,
              size,
              sent )
```

prints out the sending file progress to the console.

This is called after the call to sendDataFile

Parameters

size	Size of the file.
sent	Size sent so far.

4.8.3.3 setSampleSizeForVelAndAcceleration()

```
def setSampleSizeForVelAndAcceleration (
```

```
                self,
                sm )
```

Sets the sample model to be used for velocity and acceleration calculation.

Parameters

sm	sample model to be used. Valid values are `FIVE_SAMPLE_MODEL`, `NINE_SAMPLE_MODEL`,`SEVENTEEN_SAMPLE_MODEL`, and `EL1000_TRACKER_MODEL`

4.8.3.4 setVelocityAccelerationModel()

```
def setVelocityAccelerationModel (
                self,
                sm )
```

Sets the sample model to be used for velocity and acceleration calculation.

Same as setSampleSizeForVelAndAcceleration excepts this interprets string message.

Parameters

sm	sample model to be used. Valid values are `5-sample Model`, `9-sample Model`,`17-sample Model`, and `EL1000 Tracker Model`

4.8.3.5 doTrackerSetup()

```
def doTrackerSetup (
                self,
                width = None,
                height = None )
```

Switches the EyeLink tracker to the Setup menu, from which camera setup, calibration, validation, drift correction, and configuration may be performed.

Pressing the 'Esc' key on the tracker keyboard will exit the Setup menu and return from this function. Calling `exitCalibration()` from an event handler will cause any call to `do_tracker_setup()` in progress to return immediately.

Parameters

width	Width of the screen.
height	Height of he screen.

4.8.3.6 setAcceptTargetFixationButton()

```
def setAcceptTargetFixationButton (
              self,
              button )
```

This programs a specific button for use in drift correction.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("button_function %d 'accept_target_fixation'"%button);`

Parameters

button	Id of the button that is used to accept target fixation.

4.8.3.7 setCalibrationType()

```
def setCalibrationType (
              self,
              type )
```

This command sets the calibration type, and recomputed the calibration targets after a display resolution change.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("calibration_type=%s"%caltype);`

Parameters

type	One of these calibration type codes listed below:

H3	horizontal 3-point calibration
HV3	3-point calibration, poor linearization
HV5	5-point calibration, poor at corners
HV9	9-point grid calibration, best overall
HV13	13-point calibration for large calibration region (EyeLink II version 2.0 or later; EyeLink 1000)

4.8.3.8 setXGazeConstraint()

```
def setXGazeConstraint (
```

```
                self,
                x_position = "AUTO" )
```

Locks the x part of gaze position data.

Usually set to `AUTO`: this will use the last drift-correction target position when in `H3` mode.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("x_gaze_constraint=%s"%(str(value)));
```

Parameters

x_position	x gaze coordinate, or `AUTO`.

4.8.3.9 setYGazeConstraint()

```
def setYGazeConstraint (
                self,
                y_position = "AUTO" )
```

Locks the y part of gaze position data.

Usually set to `AUTO`: this will use the last drift-correction target position when in `H3` mode.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("y_gaze_constraint=%s"%(str(value)));
```

Parameters

y_position	y gaze coordinate, or `AUTO`.

4.8.3.10 enableAutoCalibration()

```
def enableAutoCalibration (
                self )
```

Enables the auto calibration mechanism.

By default, this mechanism is turned off.

Remarks

This function is equivalent to
```
if(getEYELINK().isConnected()):
    getEYELINK().sendCommand("enable_automatic_calibration=YES")
```

4.8.3.11 disableAutoCalibration()

```
def disableAutoCalibration (
              self )
```

Disables the auto calibration mechanism.

By default, this mechanism is turned off.

Remarks

This function is equivalent to
```
if(getEYELINK().isConnected()):
    getEYELINK().sendCommand("enable_automatic_calibration=NO")
```

4.8.3.12 setAutoCalibrationPacing()

```
def setAutoCalibrationPacing (
              self,
              pace )
```

Sets automatic calibration pacing.

`1000` is a good value for most subjects, `1500` for slow subjects and when interocular data is required.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("automatic_calibration_pacing=%d"%(time))
```

Parameters

pace	shortest delay.

4.8.3.13 readIOPort()

```
def readIOPort (
              self,
              ioport )
```

Sends a command to the tracker to read the specified io port.

Parameters

ioport	port id of the io port.

4.8.3.14 writeIOPort()

```
def writeIOPort (
              self,
              ioport,
              data )
```

Sends a command to the tracker to write the specified io port.

Parameters

ioport	byte hardware I/O port address. The port address for the C and D ports on the EyeLink analog output card are 4 and 5, respectively; the print port address will typically be 0x378 (please see the buttons.ini settings).
data	data to write

4.8.3.15 setHeuristicLinkAndFileFilter()

```
def setHeuristicLinkAndFileFilter (
              self,
              linkfilter,
              filefilter = -1 )
```

EyeLink II only: Can be used to set level of filtering on the link and analog output, and on file data.

An additional delay of 1 sample is added to link or analog data for each filter level.
If an argument of <on> is used, link filter level is set to `1` to match EyeLink I delays.
The file filter level is not changed unless two arguments are supplied.
The default file filter level is `2`.

Remarks

This function is equivalent to

```
if(getEYELINK().getTrackerVersion() >=2):
    if(filefilter == -1):
        getEYELINK().sendCommand("heuristic_filter %d"%(linkfilter))
    else:
        getEYELINK().sendCommand(" %d %d"%(linkfilter, filefilter));
```

Parameters

linkfilter	Filter level of the link data. `0` or `OFF` disables link filter. `1` or `ON` sets filter to 1 (moderate filtering, 1 sample delay). `2` applies an extra level of filtering (2 sample delay).
filefilter	Filter level of the data written to EDF file. `0` or `OFF` disables link filter. `1` or `ON` sets filter to 1 (moderate filtering, 1 sample delay). `2` applies an extra level of filtering (2 sample delay).

4.8.3.16 setHeuristicFilterOn()

```
def setHeuristicFilterOn (
              self )
```

EyeLink 1 Only: Can be used to enable filtering, increases system delay by 4 msec if the filter was originally off.

NEVER TURN OFF THE FILTER WHEN ANTIREFLECTION IS TURNED ON.
For EyeLink II and newer eye tracker models, you should use the `setHuresticFileAndLinkFilter()` method instead.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("heuristic_filter=ON");`

4.8.3.17 setHeuristicFilterOff()

```
def setHeuristicFilterOff (
              self )
```

EyeLink 1 Only: Can be used to disable filtering, reduces system delay by 4 msec.

NEVER TURN OFF THE FILTER WHEN ANTIREFLECTION IS TURNED ON. For EyeLink II and newer eye tracker models, you should use the following `setHuresticFileAndLinkFilter()` method instead.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("heuristic_filter = OFF");`

4.8.3.18 setPupilSizeDiameter()

```
def setPupilSizeDiameter (
              self,
              value )
```

Can be used to determine pupil size information to be recorded.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("pupil_size_diameter = %s"%(value));`

Parameters

value	`YES` to convert pupil area to diameter; `NO` to output pupil area data.

4.8.3.19 setSimulationMode()

```
def setSimulationMode (
              self,
              value )
```

Can be used to turn off head tracking if not used.

Do this before calibration.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("simulate_head_camera = %s"%(value));`

Parameters

value	YES to disable head tracking; NO to enable head tracking.

4.8.3.20 setScreenSimulationDistance()

```
def setScreenSimulationDistance (
              self,
              distance )
```

Used to compute correct visual angles and velocities when head tracking not used.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("simulation_screen_distance = %s"%(distance));`

Parameters

distance	simulated distance from display to subject in millimeters.

4.8.3.21 markPlayBackStart()

```
def markPlayBackStart (
              self )
```

Marks the location in the data file from which playback will begin at the next call to EYEYLINK.startPlay↩Back().

When this command is not used (or on older tracker versions), playback starts from the beginning of the previous recording block. This default behavior is suppressed after this command is used, until the tracker application is shut down.

Remarks

This function is equivalent to

```
getEYELINK().sendCommand("mark_playback_start");
```

4.8.3.22 setNoRecordEvents()

```
def setNoRecordEvents (
              self,
              message = False,
              button = False,
              inputevent = False )
```

Selects what types of events can be sent over the link while not recording (e.g between trials).

This command has no effect for EyeLink II, and messages cannot be enabled for versions of EyeLink I before v2.1.

Remarks

This function is equivalent to

```
re = []
if(message):
    re.append("MESSAGE ")
if(button):
    re.append("BUTTON ")
if(inputevent):
    re.append("INPUT ")
getEYELINK().sendCommand("link_nonrecord_events = %s"%"".join(re));
```

Parameters

message	1 to enable the recording of EyeLink messages.
button	1 to enable recording of buttons (1..8 press or release).
inputevent	1 to enable recording of changes in input port lines.

4.8.3.23 setFileSampleFilter()

```
def setFileSampleFilter (
              self,
              list )
```

Sets data in samples written to EDF file.

See tracker file "DATA.INI" for types.

Remarks

This function is equivalent to

```
getEYELINK().sendCommand("file_sample_data = %s"%list)
```

Parameters

list	list of the following data types, separated by spaces or commas.

GAZE	screen x/y (gaze) position
GAZERES	units-per-degree screen resolution
HREF	head-referenced gaze
PUPIL	raw eye camera pupil coordinates
AREA	pupil area
STATUS	warning and error flags
BUTTON	button state and change flags
INPUT	input port data lines

4.8.3.24 setFileEventData()

```
def setFileEventData (
              self,
              list )
```

Sets data in events written to EDF file.

See tracker file "DATA.INI" for types.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("file_event_data = %s"%list);`

Parameters

list	list of the following event data types, separated by spaces or commas.

GAZE	screen xy (gaze) position
GAZERES	units-per-degree angular resolution
HREF	HREF gaze position
AREA	pupil area or diameter
VELOCITY	velocity of eye motion (avg, peak)
STATUS	warning and error flags for event
FIXAVG	include ONLY average data in ENDFIX events
NOSTART	start events have no data, just time stamp

4.8.3.25 setFileEventFilter()

```
def setFileEventFilter (
              self,
              list )
```

Sets which types of events will be written to EDF file.

See tracker file "DATA.INI" for types.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("file_event_filter = %s"%list);
```

Parameters

list	list of the following event types, separated by spaces or commas.

LEFT, RIGHT	events for one or both eyes
FIXATION	fixation start and end events
FIXUPDATE	fixation (pursuit) state updates
SACCADE	saccade start and end
BLINK	blink start an end
MESSAGE	messages (user notes in file)
BUTTON	button 1..8 press or release
INPUT	changes in input port lines;

4.8.3.26 setLinkSampleFilter()

```
def setLinkSampleFilter (
              self,
              list )
```

Sets data in samples sent through link.

See tracker file "DATA.INI" for types.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("link_sample_data = %s"%list)
```

Parameters

list	list of data types, separated by spaces or commas.

GAZE	screen xy (gaze) position
GAZERES	units-per-degree screen resolution
HREF	head-referenced gaze
PUPIL	raw eye camera pupil coordinates
AREA	pupil area
STATUS	warning and error flags
BUTTON	button state and change flags
INPUT	input port data lines

4.8.3.27 setLinkEventData()

```
def setLinkEventData (
              self,
              list )
```

Sets data in events sent through link.

See tracker file "DATA.INI" for types.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("link_event_data = %s"%list);`

Parameters

list	list of data types, separated by spaces or commas.

GAZE	screen xy (gaze) position
GAZERES	units-per-degree angular resolution
HREF	HREF gaze position
AREA	pupil area or diameter
VELOCITY	velocity of eye motion (avg, peak)
STATUS	warning and error flags for event
FIXAVG	include ONLY average data in ENDFIX events
NOSTART	start events have no data, just time stamp

4.8.3.28 setLinkEventFilter()

```
def setLinkEventFilter (
              self,
              list )
```

Sets which types of events will be sent through link.

See tracker file "DATA.INI" for types.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("link_event_filter = %s"%list);`

Parameters

list	list of event types.

LEFT, RIGHT

events for one or both eyes

FIXATION

fixation start and end events

FIXUPDATE

fixation (pursuit) state updates

SACCADE

saccade start and end

BLINK

blink start an end

MESSAGE

messages (user notes in file)

BUTTON

button 1-8 press or release

INPUT

changes in input port lines;

4.8.3.29 setSaccadeVelocityThreshold()

```
def setSaccadeVelocityThreshold (
              self,
              vel )
```

Sets velocity threshold of saccade detector: usually 30 for cognitive research, 22 for pursuit and neurological work.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("saccade_velocity_threshold =%d"%(vel));`

Parameters

vel	minimum velocity (°/sec) for saccades.

4.8.3.30 setAccelerationThreshold()

```
def setAccelerationThreshold (
              self,
              accel )
```

Sets acceleration threshold of saccade detector: usually `9500` for cognitive research, `5000` for pursuit and neurological work.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("saccade_acceleration_threshold  =%d"%(accl));`

Parameters

accel	minimum acceleration (°/sec/sec) for saccades.

4.8.3.31 setMotionThreshold()

```
def setMotionThreshold (
              self,
              deg )
```

Sets a spatial threshold to shorten saccades.

Usually `0.15` for cognitive research, `0` for pursuit and neurological work.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("saccade_motion_threshold  =%d"%(deg));`

Parameters

deg	minimum motion (degrees) out of fixation before saccade onset allowed.

4.8.3.32 setPursuitFixup()

```
def setPursuitFixup (
              self,
              maxvel )
```

Sets the maximum pursuit velocity accommodation by the saccade detector.

Usually set to `60`.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("saccade_pursuit_fixup =  %d"%(v));
```

Parameters

maxvel	maximum pursuit velocity fixup (°/sec).

4.8.3.33 setUpdateInterval()

```
def setUpdateInterval (
              self,
              time )
```

Normally set to `0` to disable fixation update events.

Set to `50` or `100` milliseconds to produce updates for gaze-controlled interface applications.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("fixation_update_interval =  %d"%(time));
```

Parameters

time	milliseconds between fixation updates, `0` turns off.

4.8.3.34 setFixationUpdateAccumulate() [1/2]

```
def setFixationUpdateAccumulate (
              self,
              time )
```

Normally set to `0` to disable fixation update events.

Set to `50` or `100` milliseconds to produce updates for gaze-controlled interface applications. Set to `4` to collect single sample rather than average position.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("fixation_update_accumulate =  %d"%(time));
```

Parameters

time	milliseconds to collect data before fixation update for average gaze position.

4.8.3.35 setRecordingParseType()

```
def setRecordingParseType (
              self,
              rtype = "GAZE" )
```

Sets how velocity information for saccade detection is computed.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("recording_parse_type %s"%(rtype));
```

Parameters

rtype	GAZE or HREF; Almost always left to GAZE.

4.8.3.36 drawText()

```
def drawText (
              self,
              text,
              pos = (-1,-1) )
```

Draws text, coordinates are gaze-position display coordinates.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("print_position= %d %d"%pos)
getEYELINK().sendCommand("echo %s"%(text))
```

Parameters

text	text to print in quotes.
pos	Center point of text; Default position is (−1, −1).

4.8.3.37 clearScreen()

```
def clearScreen (
              self,
              color )
```

Clear tracker screen for drawing background graphics or messages.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("clear_screen %d"%(color));
```

Parameters

color	0 to 15.

4.8.3.38 drawLine()

```
def drawLine (
              self,
              firstPoint,
              secondPoint,
              color )
```

Draws line, coordinates are gaze-position display coordinates.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("draw_line %d %d %d %d %d"%
    (firstPoint[0],firstPoint[1],secondPoint[0],secondPoint[1], color));
```

Parameters

firstPoint	a two-item tuple, containing the x, y coordinates of the start point.
secondPoint	a two-item tuple, containing the x, y coordinates of the end point.
color	0 to 15.

4.8.3.39 drawBox()

```
def drawBox (
              self,
              x,
              y,
              width,
```

```
                height,
                color )
```

Draws an empty box, coordinates are gaze-position display coordinates.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("draw_box %d %d %d %d %d"%(x,y,x+width,y+height,color));`

Parameters

x	x coordinates for the top-left corner of the rectangle.
y	y coordinates for the top-left corner of the rectangle.
width	width of the filled rectangle.
height	height of the filled rectangle.
color	0 to 15.

4.8.3.40 drawFilledBox()

```
def drawFilledBox (
                self,
                x,
                y,
                width,
                height,
                color )
```

Draws a solid block of color, coordinates are gaze-position display coordinates.

Remarks

This function is equivalent to
`getEYELINK().sendCommand("draw_filled_box %d %d %d %d %d"%(x,y,x+width,y+height,color));`

Parameters

x	x coordinates for the top-left corner of the rectangle.
y	y coordinates for the top-left corner of the rectangle.
width	width of the filled rectangle.
height	height of the filled rectangle.
color	0 to 15.

4.8.3.41 getFixationUpdateInterval()

```
def getFixationUpdateInterval (
                self )
```

Returns the fixation update interval value.

This does not query the tracker, only valid if setFixationUpdateInterval is called prior to calling this function.

4.8.3.42 getFixationUpdateAccumulate()

```
def getFixationUpdateAccumulate (
              self )
```

Returns the fixation update accumulate value.

This does not query the tracker, only valid if setFixationUpdateAccumulate is called prior to calling this function.

4.8.3.43 setFixationUpdateInterval()

```
def setFixationUpdateInterval (
              self,
              interval )
```

Sends a command to the tracker to update the FixationUpdateInterval.

Parameters

interval	value for fixation update interval

4.8.3.44 setFixationUpdateAccumulate() [2/2]

```
def setFixationUpdateAccumulate (
              self,
              accumulate )
```

Sends a command to the tracker to update the FixationUpdateAccumulate.

Parameters

accumulate	value for fixation update accumulate

4.8.3.45 echo()

```
def echo (
              self,
              text,
              pos = (-1,-1) )
```

Prints text at current print position to tracker screen, gray on black only.

Parameters

text	text to print in quotes.
pos	position of the text to display

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("echo %s"%text)
```

4.8.3.46 drawCross()

```
def drawCross (
              self,
              x,
              y,
              color )
```

Draws a small "+" to mark a target point.

Remarks

This function is equivalent to
```
getEYELINK().sendCommand("draw_cross %d %d %d"%(x,y, color));
```

Parameters

x	x coordinates for the center point of cross.
y	y coordinates for the center point of cross.
color	0 to 15 (0 for black; 1 for blue; 2 for green; 3 for cyan; 4 for red; 5 for magenta; 6 for brown; 7 for light gray; 8 for dark gray; 9 for light blue; 10 for light green; 11 for light cyan; 12 for light red; 13 for bright magenta; 14 for yellow; 15 for bright white).

4.9 EyeLinkAddress Class Reference

The EyeLinkAddress class is used to hold addresses to EyeLink nodes.

Inherited by EyelinkMessage.

Public Member Functions

- def __init__ (self, ip=(100, 1, 1, 1), port=4000)

  Constructor.
- def getIP (self)

  Returns the IP address of the EyeLink node.
- def getPort (self)

  Returns the port number of the EyeLink node.

4.9.1 Detailed Description

The EyeLinkAddress class is used to hold addresses to EyeLink nodes.

An instance of EyeLinkAddress class can be initialized with the class constructor:
```
EyeLinkAddress(ip = (100,1,1,1), port = 4000)
```

where *ip* is a four-item tuple containing the IP address of the EyeLink node and *port* is the port number of the connection.

For example,
```
myAddress = EyeLinkAddress((100, 1, 1, 1), 4000)
```

4.9.2 Constructor & Destructor Documentation

4.9.2.1 __init__()

```
def __init__ (
              self,
              ip = (100,1,1,1),
              port = 4000 )
```

Constructor.

Parameters

ip	optional ipaddress in tuple form. eg. if the ip address is 192.168.25.48, the tuple form is (192,168,25,48) The default ip address is 100.1.1.1
port	optional port value as integer. The default value is 4000.

4.9.3 Member Function Documentation

4.9.3.1 getIP()

```
def getIP (
              self )
```

Returns the IP address of the EyeLink node.

Returns

A four-item tuple (integer) containing the IP address of the EyeLink node.

4.9.3.2 getPort()

```
def getPort (
              self )
```

Returns the port number of the EyeLink node.

Returns

An integer for the port number of the connection.

4.10 EyeLinkCBind Class Reference

Inherited by EyeLinkListener.

Public Member Functions

- def calculateOverallVelocityAndAcceleration ()
- def getTrackerVersion ()
- def nodeSendMessage ()
- def getkey ()
- def trackerTimeUsecOffset ()
- def startPlayBack ()
- def doTrackerSetup ()
- def inSetup ()
- def stopData ()
- def receiveDataFile ()
- def readKeyQueue ()
- def sendCommand ()
- def reset ()
- def openDataFile ()
- def sendTimedCommandEx ()
- def eyeAvailable ()
- def userMenuSelection ()
- def dummy_open ()
- def calculateVelocityXY ()
- def imageModeDisplay ()
- def requestTime ()
- def calculateVelocity ()
- def waitForBlockStart ()
- def echo_key ()
- def sendMessage ()
- def getNode ()
- def readKeyButton ()
- def bitmapBackdrop ()
- def pollTrackers ()
- def close ()
- def key_message_pump ()
- def closeDataFile ()
- def getNextData ()
- def acceptTrigger ()

- def startRecording ()
- def getLastMessage ()
- def readReply ()
- def trackerTimeUsec ()
- def getEventDataFlags ()
- def dataSwitch ()
- def isRecording ()
- def readRequest ()
- def getTrackerMode ()
- def sendKeybutton ()
- def startSetup ()
- def quietMode ()
- def getModeData ()
- def nodeSend ()
- def terminalBreak ()
- def startData ()
- def pollResponses ()
- def getLastButtonStates ()
- def isConnected ()
- def waitForData ()
- def broadcastOpen ()
- def getRecordingStatus ()
- def open ()
- def targetModeDisplay ()
- def getCalibrationResult ()
- def nodeReceive ()
- def getDataCount ()
- def getLastData ()
- def getSample ()
- def doDriftCorrect ()
- def setOfflineMode ()
- def getNewestSample ()
- def breakPressed ()
- def setName ()
- def getCurrentMode ()
- def resetData ()
- def getCalibrationMessage ()
- def pollRemotes ()
- def sendDataFile ()
- def getTrackerVersionString ()
- def escapePressed ()
- def pumpMessages ()
- def getSampleDataFlags ()
- def stopRecording ()
- def getButtonStates ()
- def abort ()
- def stopPlayBack ()
- def sendTimedCommand ()
- def getLastButtonPress ()
- def getkeyEx ()
- def isInDataBlock ()
- def nodeRequestTime ()
- def applyDriftCorrect ()
- def setAddress ()
- def readTime ()

- def getImageCrossHairData ()
- def bitmapSaveAndBackdrop ()
- def getEventTypeFlags ()
- def trackerTimeOffset ()
- def getPositionScalar ()
- def waitForModeReady ()
- def exitCalibration ()
- def openNode ()
- def flushKeybuttons ()
- def commandResult ()
- def getFloatData ()
- def getTargetPositionAndState ()
- def startDriftCorrect ()
- def trackerTime ()

4.10.1 Detailed Description

C Implementation of EyeLinkListener.

4.10.2 Member Function Documentation

4.10.2.1 calculateOverallVelocityAndAcceleration()

```
def calculateOverallVelocityAndAcceleration ( )
```

Calculates overall velocity and acceleration for left and right eyes separately.

Parameters

in	*slen*	Sample model to use for velocity calculation. Acceptable models are FIVE_SAMPLE_MODEL, NINE_SAMPLE_MODEL, SEVENTEEN_SAMPLE_MODEL and EL1000_TRACKER_MODEL.

Returns

A list with 3 elements

- The first element of the list:
  - overall velocity for left and right eye. Upon return of this function,vel[0] will contain overall velocity for left eye and vel[1] will contain overall velocity for right eye. If velocity cannot be calculated for any reason(eg. insufficient samples, no data) MISSING_DATA is filled for the given velocity.
- The second element of the list:
  - overall acceleration for left and right eye. Upon return of this function, acc[0] will contain overall acceleration for left eye and acc[1] will contain overall acceleration for right eye. If acceleration cannot be calculated for any reason(eg. insufficient samples, no data) MISSING_DATA is filled for the given acceleration.
- The third element of the list:
  - vel_sample Velocity for sample.

4.10.2.2 getTrackerVersion()

```
def getTrackerVersion ( )
```

After connection, determines if the connected tracker is an EyeLink I or II. Use `getTrackerVersionString()` to get the string value.

Remarks

This is equivalent to the C API
```
INT16 eyelink_get_tracker_version(char *c);
```

Returns

The returned value is a number (`0` if not connected, `1` for EyeLink I, `2` for EyeLink II).

4.10.2.3 nodeSendMessage()

```
def nodeSendMessage ( )
```

Sends a text message the connected eye tracker. The text will be added to the EDF file.

Remarks

If the link is initialized but not connected to a tracker, the message will be sent to the tracker set by `setAddress()` of the pylink module. This function is equivalent to the C API
```
INT16 eyelink_node_send_message(ELINKADDR node, char *msg);
```

Parameters

address	Address of a specific tracker.
message	Text to send to the tracker.

Returns

`0` if no error, else link error code.

4.10.2.4 getkey()

```
def getkey ( )
```

Returns the key pressed.

Remarks

Warning: This function processes and dispatches any waiting messages. This will allow Windows to perform disk access and negates the purpose of realtime mode. Usually these delays will be only a few milliseconds, but delays over 20 milliseconds have been observed. You may wish to call `escapePressed()` or `breakPressed()` in recording loops instead of `getkey()` if timing is critical, for example in a gaze-contingent display. Under Windows XP, these calls will not work in realtime mode at all (although these do work under Windows 2000). Under Windows 95/98/Me, realtime performance is impossible even with this strategy. Some useful keys are:

- `CURS_UP`
- `CURS_DOWN`
- `CURS_LEFT`
- `CURS_RIGHT`
- `ESC_KEY`
- `ENTER_KEY`
- `TERMINATE_KEY`
- `JUNK_KEY`

This function is equivalent to the C API

```
unsigned getkey(void);
```

Returns

`0` if no key pressed, else key code. `TERMINATE_KEY` if CTRL-C held down or program has been terminated.

4.10.2.5 trackerTimeUsecOffset()

```
def trackerTimeUsecOffset ( )
```

Returns the time difference between the tracker time and display pc time.

Remarks

This is equivalent to the C API

```
double eyelink_time_usec_offset();
```

Returns

A double precision data for the time difference (in microseconds) between the tracker time and display pc time.

4.10.2.6 startPlayBack()

```
def startPlayBack ( )
```

Flushes data from queue and starts data playback. An EDF file must be open and have at least one recorded trial. Use `waitForData()` method to wait for data: this will time out if the playback failed. Playback begins from start of file or from just after the end of the next-but-last recording block. Link data is determined by file contents, not by link sample and event settings.

Remarks

This function is equivalent to the C API
`INT16 eyelink_playback_start(void);`

Returns

`0` if command sent fine, else link error.

4.10.2.7 doTrackerSetup()

```
def doTrackerSetup ( )
```

Switches the EyeLink tracker to the Setup menu, from which camera setup, calibration, validation, drift correction, and configuration may be performed. Pressing the 'ESC' key on the tracker keyboard will exit the Setup menu and return from this function. Calling `exitCalibration()` from an event handler will cause any call to `do_tracker_setup()` in progress to return immediately.

Parameters

width	Width of the screen.
height	Height of he screen.

4.10.2.8 inSetup()

```
def inSetup ( )
```

Checks if tracker is still in a Setup menu activity (includes camera image view, calibration, and validation). Used to terminate the subject setup loop.

Remarks

This function is equivalent to the C API
`INT16 eyelink_in_setup(void);`

Returns

`0` if no longer in setup mode.

4.10.2.9 stopData()

```
def stopData ( )
```

Places tracker in idle (off-line) mode, does not flush data from queue.

Remarks

Should be followed by a call to `waitForModeReady()` method. This function is equivalent to the C API
`INT16 eyelink_data_stop(void);`

Returns

`0` if command sent fine, else link error.

4.10.2.10 receiveDataFile()

```
def receiveDataFile ( )
```

This receives a data file from the EyeLink tracker PC. Source file name and destination file name should be given.

Remarks

This function is equivalent to the C API
`int receive_data_file(char *src, char *dest, int is_path=0);`

Parameters

src	Name of eye tracker file (including extension).
dest	Name of local file to write to (including extension).

Returns

Size of file if successful, Otherwise Runtime Exception is raised.

4.10.2.11 readKeyQueue()

```
def readKeyQueue ( )
```

Read keys from the key queue. It is similar to `getkey()`, but does not process Windows messages. This can be used to build key-message handlers in languages other than C.

Remarks

This function is equivalent to the C API
`UINT16 read_getkey_queue(void);`

Returns

`0` if no key pressed.
`JUNK_KEY (1)` if untranslatable key.
`TERMINATE_KEY (0x7FFF)` if CTRL-C is pressed, `terminal_break()` was called, or the program has been terminated with ALT-F4.
or code of key if any key pressed.

4.10.2.12 sendCommand()

```
def sendCommand ( )
```

Sends the given command to connected eyelink tracker and returns the command result.

Remarks

This is equivalent to the C API
```
int eyecmd_printf(char *fmt, ...); // without any formatting.
```

Parameters

command_text	Text command to be sent. It does not support `printf()` kind of formatting.

Returns

Command result. If there is any problem sending the command, a runtime exception is raised.

4.10.2.13 reset()

```
def reset ( )
```

Sends a reset message to the EyeLink tracker.

Remarks

This function is equivalent to the C API
```
INT16 eyelink_close(int send_msg);  // with send_msg parameter 0.
```

Returns

0 if successful, otherwise link error.

4.10.2.14 openDataFile()

```
def openDataFile ( )
```

Opens a new EDF file on the EyeLink tracker computer's hard disk. By calling this function will close any currently opened file. This may take several seconds to complete. The file name should be formatted for MS-DOS, usually 8 or less characters with only 0-9, A-Z, and '_' allowed.

Remarks

This function is equivalent to the C API
`int open_data_file(char *name);`

Parameters

name	Name of eye tracker file, 8 characters or less.

Returns

`0` if file was opened successfully else error code.

4.10.2.15 sendTimedCommandEx()

```
def sendTimedCommandEx ( )
```

Sends a command to the connected eye tracker, wait for reply.

Returns

List of 2 items. The first item contains the return value of `eyelink_timed_command()`. The second item contains a string description of the error message.

Remarks

If there is an error, no exception is raised.

See also

`sendTimedCommand()`

4.10.2.16 eyeAvailable()

```
def eyeAvailable ( )
```

After calling the `waitForBlockStart()` method, or after at least one sample or eye event has been read, this function can be used to check which eyes data is available for.

Remarks

This is equivalent to the C API
`INT16 eyelink_eye_available(void);`

Returns

`LEFT_EYE (0)` if left eye data.
`RIGHT_EYE (1)` if right eye data.
`BINOCULAR (2)` if both left and right eye data.
`-1` if no eye data is available.

4.10.2.17 userMenuSelection()

```
def userMenuSelection ( )
```

Checks for a user-menu selection, clears response for next call.

Remarks

This function is equivalent to the C API
`INT16 eyelink_user_menu_selection(void);`

Returns

`0` if no selection made since last call, else code of selection.

4.10.2.18 dummy_open()

```
def dummy_open ( )
```

Sets the EyeLink library to simulate an eyetracker connection. Functions will return plausible values, but no data.

Remarks

The function `isConnected()` will return `-1` to indicate a simulated connection.

4.10.2.19 calculateVelocityXY()

```
def calculateVelocityXY ( )
```

Calculates left x velocity, left y velocity, right x velocity and right y velocity from queue of samples.

Parameters

in	*slen*	Sample model to use for velocity calculation. Acceptable models are FIVE_SAMPLE_MODEL, NINE_SAMPLE_MODEL, SEVENTEEN_SAMPLE_MODEL and EL1000_TRACKER_MODEL.

Returns

A list with 3 elements

- The first element of the list:
  - leftvel x and y velocity for left eye. The float tuple of 2 filled with x and y velocity values. Upon return of this function leftvel[0] contains the left x velocity data and leftvel[1] contains left y velocity data. If velocity cannot be calculated for any reason (eg. insufficient samples, no data) MISSING_DATA is filled for the given velocity.
- The second element of the list:
  - rightvel x and y velocity for right eye. The float tuple of 2 filled with x and y velocity values. Upon return of this function rightvel[0] contains the right x velocity data and rightvel[1] contains right y velocity data. If velocity cannot be calculated for any reason (eg. insufficient samples, no data) MISSING_DATA is filled for the given velocity.
- The third element of the list:
  - vel_sample Velocity for sample.

4.10.2.20 imageModeDisplay()

```
def imageModeDisplay ( )
```

This handles display of the EyeLink camera images. While in imaging mode, it continuously requests and displays the current camera image. It also displays the camera name and threshold setting. Keys on the subject PC keyboard are sent to the tracker, so the experimenter can use it during setup. It will exit when the tracker leaves imaging mode or disconnects.

Returns

0 if OK, TERMINATE_KEY if pressed, −1 if disconnect.

Remarks

This function not normally used externally. If you need camera setup use doTrackerSetup() or if you need drift correction use doDriftCorrect()

4.10.2.21 requestTime()

```
def requestTime ( )
```

Sends a request the connected eye tracker to return its current time.

Remarks

The time reply can be read with readTime().

Returns

0 if no error, else link error code.

See also

rackerTime()

4.10.2.22 calculateVelocity()

```
def calculateVelocity ( )
```

Calculates overall velocity for left and right eyes separately.

Parameters

in	*slen*	Sample model to use for velocity calculation. Acceptable models are FIVE_SAMPLE_MODEL, NINE_SAMPLE_MODEL, SEVENTEEN_SAMPLE_MODEL and EL1000_TRACKER_MODEL.

Returns

A list with 3 elements:

- First two elements of the list:
  - Upon return of this function, vel[0] will contain overall velocity for left eye and vel[1] will contain overall velocity for right eye. If velocity cannot be calculated for any reason(eg. insufficient samples, no data) MISSING_DATA is filled for the given velocity.
- Third element of the list:
  - vel_sample Velocity for sample.

4.10.2.23 waitForBlockStart()

```
def waitForBlockStart ( )
```

Reads and discards events in data queue until in a recording block. Waits for up to <timeout> milliseconds for a block containing samples, events, or both to be opened. Items in the queue are discarded until the block start events are found and processed. This function will fail if both samples and events are selected but only one of link samples and events were enabled by startRecording().

Remarks

This function did not work in versions previous to 2.0. This function is equivalent to the C API
`INT16 eyelink_wait_for_block_start(UINT32 maxwait,INT16 samples, INT16 events);`

Parameters

timeout	Time in milliseconds to wait.
samples	If non-zero, check if in a block with samples.
events	If non-zero, check if in a block with events.

Exceptions

Runtime	exception is raised if time expires and no data of masked types is available. **handling:** `getLastError()[0]==0`

Returns

`true` if data is available

4.10.2.24 echo_key()

```
def echo_key ( )
```

Checks for Windows keystroke events and dispatches messages; similar to `getkey()`, but also sends keystroke to tracker.

Remarks

Warning: Under Windows XP, this call will not work in realtime mode at all, and will take several seconds to respond if graphics are being drawn continuously. This function works well in realtime mode under Windows 2000. This function is equivalent to the C API
`unsigned echo_key(void);`

Returns

`0` if no key pressed, else key code `TERMINATE_KEY` if CTRL-C held down or program has been terminated.

4.10.2.25 sendMessage()

```
def sendMessage ( )
```

Sends the given message to the connected eyelink tracker. The message will be written to the eyelink tracker.

Remarks

This is equivalent to the C API
`int eyemsg_printf(char *fmt, ...);`

Parameters

message_text	Text message to be sent. It does not support `printf()` kind of formatting.

Returns

If there is any problem sending the message, a runtime exception is raised.

4.10.2.26 getNode()

```
def getNode ( )
```

Reads the responses returned by other trackers or remotes in response to `pollTrackers()` or `pollRemotes()`. It can also read the tracker broadcast address and remote broadcast addresses.

Remarks

This function is equivalent to the C API
```
INT16 eyelink_get_node(INT16 resp, void *data);
```

Parameters

resp	Number of responses to read: `0` gets our data, `1` get first response, `2` gets the second response, etc. `-1` to read the tracker broadcast address. `-2` to read remote broadcast addresses.

Returns

If successful, an instance of EyeLinkMessage class returned.

4.10.2.27 readKeyButton()

```
def readKeyButton ( )
```

Reads any queued key or button events from tracker.

Remarks

This function is equivalent to the C API
```
UINT16 eyelink_read_keybutton(INT16 *mods,INT16 *state, UINT16 *kcode, UINT32 *time);
```

Returns

A five-item tuple, recording (in the following order):

- Key character if key press/release/repeat, `KB_BUTTON (0xFF00)` if button press or release,
- Button number or key modifier (Shift, Alt and Ctrl key states),
- Key or button change (`KB_PRESS`, `KB_RELEASE`, or `KB_REPEAT`),
- Key scan code,
- Tracker time of the key or button change.

4.10.2.28 bitmapBackdrop()

```
def bitmapBackdrop ( )
```

This function transfers the bitmap to the tracker PC as backdrop for gaze cursors.

Parameters

iwidth	Original image width.
iheight	Original image height.
pixels	Pixels of the image in one of two possible formats: pixel=[line1, line2, ... linen] line=[pix1,pix2,...,pixn],pix=(r,g,b). pixel=[line1, line2, ... linen] line=[pix1,pix2,...,pixn],pix=0xAARRGGBB.
xs	Crop x position.
ys	Crop y position.
width	Crop width.
height	Crop height.
xd	X position - transfer.
yd	Y position - transfer.
xferoptions	Transfer options(BX_AVERAGE, BX_DARKEN, BX_LIGHTEN, BX_MAXCONTRAST, BX_NODITHER, BX_GRAYSCALE). Transfer options set with bitwise OR of the following constants, determines how bitmap is processed: • BX_AVERAGE Averaging combined pixels • BX_DARKEN Choosing darkest and keep thin dark lines. • BX_LIGHTEN Choosing darkest and keep thin white lines and control how bitmap size is reduced to fit tracker display. • BX_MAXCONTRAST Maximizes contrast for clearest image. • BX_NODITHER Disables the dithering of the image. • BX_GREYSCALE Converts the image to grayscale (grayscale works best for EyeLink I, text, etc.).

Remarks

This function should not be called when timing is critical, as this might take very long to return.

4.10.2.29 pollTrackers()

```
def pollTrackers ( )
```

Asks all trackers (with EyeLink software running) on the network to send their names and node address.

Remarks

This function is equivalent to the C API
```
INT16 eyelink_poll_trackers(void);
```

Returns

0 if successful, otherwise link error.

4.10.2.30 close()

```
def close ( )
```

Sends a disconnect message to the EyeLink tracker.

Remarks

This function is equivalent to the C API
```
INT16 eyelink_close(int send_msg);  // with send_msg parameter 1.
```

Returns

0 if successful, otherwise link error.

4.10.2.31 key_message_pump()

```
def key_message_pump ( )
```

Similar to `pumpMessages()`, but only processes keypresses. This may help reduce latency.

4.10.2.32 closeDataFile()

```
def closeDataFile ( )
```

Closes any currently opened EDF file on the EyeLink tracker computer's hard disk. This may take several seconds to complete.

Remarks

This function is equivalent to the C API
```
int close_data_file(void);
```

Returns

`0` if command executed successfully else error code.

4.10.2.33 getNextData()

```
def getNextData ( )
```

Fetches next data item from link buffer and returns the data item type. If the item is not wanted, simply ignore it. Otherwise, call `getFloatData()` to read it into a buffer.

Returns

`0` if no data, `SAMPLE_TYPE` if sample, else event type.

4.10.2.34 acceptTrigger()

```
def acceptTrigger ( )
```

Triggers the EyeLink tracker to accept a fixation on a target, similar to the 'Enter' key or spacebar on the tracker.

Remarks

This function is equivalent to the C API
`INT16 eyelink_accept_trigger(void);`

Returns

`NO_REPLY` if drift correction not completed yet.
`OK_RESULT (0)` if success.
`ABORT_REPLY (27)` if 'ESC' key aborted operation.
`-1` if operation failed.
`1` if poor calibration or excessive validation error.

4.10.2.35 startRecording()

```
def startRecording ( )
```

Starts the EyeLink tracker recording, sets up link for data reception if enabled.

Remarks

Recording may take 10 to 30 milliseconds to begin from this command. The function also waits until at least one of all requested link data types have been received. If the return value is not zero, return the result as the trial result code. This is equivalent to the C API
`INT16 start_recording(INT16 file_samples, INT16 file_events, INT16 link_samples, INT16 link_events);`

Parameters

file_samples	If `1`, writes samples to EDF file. If `0`, disables sample recording.
file_events	If `1`, writes events to EDF file. If `0`, disables event recording.
link_samples	If `1`, sends samples through link. If `0`, disables link sample access.
link_events	If `1`, sends events through link. If `0`, disables link event access.

Returns

`0` if successful, else trial return code.

4.10.2.36 getLastMessage()

```
def getLastMessage ( )
```

Returns text associated with last command response: may have error message.

Remarks

This is equivalent to the C API
`INT16 eyelink_last_message(char *buf);`

Returns

Text associated with last command response or None.

4.10.2.37 readReply()

```
def readReply ( )
```

Returns text with reply to last read request.

Remarks

This is equivalent to the C API
`INT16 eyelink_read_reply(char *buf);`

Returns

String to contain text or None.

4.10.2.38 trackerTimeUsec()

```
def trackerTimeUsec ( )
```

Returns the current tracker time (in microseconds) since the tracker application started.

Remarks

This is equivalent to the C API
`UINT32 eyelink_tracker_time();`

Returns

A double precision data for the current tracker time (in microseconds) since tracker initialization.

4.10.2.39 getEventDataFlags()

```
def getEventDataFlags ( )
```

Returns the event data content flags.

Remarks

This is equivalent to the C API
`UINT16 eyelink_event_data_flags(void);`

Returns

Possible return values are a set of the following bit flags:

Constant Name	Value	Description
EVENT_VELOCITY	0x8000	Has velocity data
EVENT_PUPILSIZE	0x4000	Has pupil size data
EVENT_GAZERES	0x2000	Has gaze resolution
EVENT_STATUS	0x1000	Has status flags
EVENT_GAZEXY	0x0400	Has gaze x, y position
EVENT_HREFXY	0x0200	Has head-ref x, y position
EVENT_PUPILXY	0x0100	Has pupil x, y position
FIX_AVG_ONLY	0x0008	Only average data to fixation events
START_TIME_ONLY	0x0004	Only start-time in start events
PARSEDBY_GAZE	0x00C0	Events were generated by GAZE data
PARSEDBY_HREF	0x0080	Events were generated by HREF data
PARSEDBY_PUPIL	0x0040	Events were generated by PUPIL data

4.10.2.40 dataSwitch()

```
def dataSwitch ( )
```

Sets what data from tracker will be accepted and placed in queue.

Remarks

This does not start the tracker recording, and so can be used with `broadcastOpen()`. It also does not clear old data from the queue. This function is equivalent to the C API
```
INT16 eyelink_data_switch(UINT16 flags);
```

Parameters

flags	Bitwise OR of the following flags: • RECORD_LINK_SAMPLES - send samples on link. • RECORD_LINK_EVENTS - send events on link.

Returns

0 if no error, else link error code.

4.10.2.41 isRecording()

```
def isRecording ( )
```

Check if we are recording: if not, report an error. Call this function while recording. It will return true if recording is still in progress, otherwise it will throw an exception. It will also handle the EyeLink Abort menu. Any errors returned by this function should be returned by the trial function. On error, this will disable realtime mode and restore the heuristic.

Remarks

This function is equivalent to the C API
`int check_recording(void);`

Returns

`TRIAL_OK (0)` if no error.
`REPEAT_TRIAL, SKIP_TRIAL, ABORT_EXPT, TRIAL_ERROR` if recording aborted.

4.10.2.42 readRequest()

```
def readRequest ( )
```

Sends a text variable name whose value is read and returned by the tracker as a text string.

Remarks

If the link is initialized but not connected to a tracker, the message will be sent to the tracker set by `setAddress()`. However, these requests will be ignored by tracker versions older than EyeLink I v2.1 and EyeLink II v1.1. This is equivalent to the C API
`INT16 eyelink_read_request(char *text);`

Parameters

text	String with message to send.

Returns

`0` if success, otherwise link error code.

4.10.2.43 getTrackerMode()

```
def getTrackerMode ( )
```

Returns raw EyeLink mode numbers.

Remarks

This function is equivalent to the C API
`INT16 eyelink_tracker_mode(void);`

Returns

Raw EyeLink mode, -1 if link disconnected.

Constant	Value
EL_IDLE_MODE	1
EL_IMAGE_MODE	2
EL_SETUP_MENU_MODE	3
EL_USER_MENU_1	5
EL_USER_MENU_2	6
EL_USER_MENU_3	7
EL_OPTIONS_MENU_MODE	8
EL_OUTPUT_MENU_MODE	9
EL_DEMO_MENU_MODE	10
EL_CALIBRATE_MODE	11
EL_VALIDATE_MODE	12
EL_DRIFT_CORR_MODE	13
EL_RECORD_MODE	14

```
USER_MENU_NUMBER(mode) ((mode)-4)
```

4.10.2.44 sendKeybutton()

```
def sendKeybutton ( )
```

Sends a key or button event to tracker. Only key events are handled for remote control.

Remarks

This function is equivalent to the C API
`INT16 eyelink_send_keybutton(UINT16 code, UINT16 mods, INT16 state);`

Parameters

code	Key character, or KB_BUTTON (0xFF00) if sending button event.
mods	Button number, or key modifier (Shift, Alt and Ctrl key states).
state	Key or button change (KB_PRESS or KB_RELEASE).

Returns

0 if OK, else send link error.

4.10.2.45 startSetup()

```
def startSetup ( )
```

Switches the EyeLink tracker to the setup menu, for calibration, validation, and camera setup. Should be followed by a call to waitForModeReady().

Remarks

This is equivalent to the C API
`INT16 eyelink_start_setup(void);`

Returns

`0` if command send fine.

4.10.2.46 quietMode()

```
def quietMode ( )
```

Controls the level of control an application has over the tracker.

Remarks

This function is equivalent to the C API
`INT16 eyelink_quiet_mode(INT16 mode);`

Parameters

mode	`0` to allow all communication; `1` to block commands (allows only key presses, messages, and time or variable read requests); `2` to disable all commands, requests and messages; `-1` to just return current setting.

Returns

Returns the previous mode settings.

4.10.2.47 getModeData()

```
def getModeData ( )
```

After calling `waitForBlockStart()`, or after at least one sample or eye event has been read, returns EyeLink II extended mode data.

Remarks

This function is equivalent to the C API
`INT16 eyelink2_mode_data(INT16 *sample_rate, INT16 *crmode, INT16 *file_filter, INT16 *link_filter);`

Returns

A five-item tuple holding (in the following order):

- success value of call to eyelink2_mode_data(). 0 if link state data available, -1 otherwise.
- sampling rate (samples per second),
- CR mode flag (0 if pupil-only mode, else pupil-CR mode),
- filter level applied to file samples (0 = off, 1 = std, 2 = extra),
- filter level applied to link and analog output samples (0 = off, 1 = std, 2 = extra).

4.10.2.48 nodeSend()

```
def nodeSend ( )
```

Sends a given data to the given node.

Remarks

This function is equivalent to the C API
`INT16 eyelink_node_send(ELINKADDR node, void *data, UINT16 dsize);`

Parameters

addr	the address of the node.
data	Pointer to buffer containing data to send.
length	Number of bytes of data.

Returns

0 if successful, otherwise link error.

4.10.2.49 terminalBreak()

```
def terminalBreak ( )
```

This function can be called in an event handler to signal that the program is terminating. Calling this function with an argument of 1 will cause `breakPressed()` to return 1, and `getkey()` to return TERMINATE_KEY. These functions can be re-enabled by calling `terminalBreak()` with an argument of 0.

Remarks

This function is equivalent to the C API
`void terminal_break(INT16 assert);`

Parameters

assert	1 to signal a program break, 0 to reset break.

4.10.2.50 startData()

```
def startData ( )
```

Switches tracker to Record mode, enables data types for recording to EDF file or sending to link. These types are set with a bit wise OR of these flags:

Constant Name	Value	Description
RECORD_FILE_SAMPLES	1	Enables sample recording to EDF file
RECORD_FILE_EVENTS	2	Enables event recording to EDF file
RECORD_LINK_SAMPLES	4	Enables sending samples to the link
RECORD_LINK_EVENTS	8	Enables sending events to the link

Remarks

If <lock> is nonzero, the recording may only be terminated through stopRecording() or stopData() method of the EyeLinkListener class, or by the Abort menu ('Ctrl' 'Alt' 'A' keys on the eye tracker). If zero, the tracker 'ESC' key may be used to halt recording. This function is equivalent to the C API

```
INT16 eyelink_data_start(UINT16 flags, INT16 lock);
```

Parameters

flags	Bitwise OR of flags to control what data is recorded. If 0, recording will be stopped.
lock	If nonzero, prevents 'ESC' key from ending recording.

Returns

0 if command sent fine, else link error.

4.10.2.51 pollResponses()

```
def pollResponses ( )
```

Returns the count of node addresses received so far following the call of pollRemotes() or pollTrackers().

Remarks

You should allow about 100 milliseconds for all nodes to respond. Up to 4 node responses are saved. This function is equivalent to the C API

```
INT16 eyelink_poll_responses(void);
```

Returns

Number of nodes responded. 0 if no responses.

4.10.2.52 getLastButtonStates()

```
def getLastButtonStates ( )
```

Returns a flag word with bits set to indicate which tracker buttons are currently pressed. This is button 1 for the LSB, up to button 16 for the MSB. Same as getButtonStates() except, the time is also returned.

Returns

a list of with values time and states of 7 buttons. Example:

```
v = eyelink.getLastButtonStates()
print "time of last button states ", v[0]
print "Button states"
button_states = v[1:]
button =0
for x in button_states:
    button = button +1
    if x!=0:
        print "Button ",button," Pressed "
```

See also

`eyelink_send_keybutton()`

4.10.2.53 isConnected()

```
def isConnected ( )
```

Checks whether the connection to the tracker is alive.

Remarks

This is equivalent to the C API

```
INT16 eyelink_is_connected(void);
```

Returns

`0` if link closed.
`-1` if simulating connection.
`1` for normal connection.
`2` for broadcast connection.

4.10.2.54 waitForData()

```
def waitForData ( )
```

Waits for data to be received from the eye tracker. Can wait for an event, a sample, or either. Typically used after record start to check if data is being sent.

Remarks

This function is equivalent to the C API

```
INT16 eyelink_wait_for_data (UINT32 maxwait, INT16 samples, INT16 events);
```

Parameters

maxwait	Time in milliseconds to wait for data.
samples	If `1`, return when first sample available.
events	If `1`, return when first event available.

Exceptions

RuntimeError	if time expires and no data of masked types is available. **handling:** getLastError()[0] == 0

Returns

`1` if data is available.

4.10.2.55 broadcastOpen()

```
def broadcastOpen ( )
```

Allows a third computer to listen in on a session between the eye tracker and a controlling remote machine. This allows it to receive data during recording and playback, and to monitor the eye tracker mode. The local computer will not be able to send commands to the eye tracker, but may be able to send messages or request the tracker time.

Remarks

This may not function properly, if there are more than one Ethernet cards installed. This function is equivalent to the C API

```
INT16  eyelink_broadcast_open(void);
```

Returns

`0` if successful.
`LINK_INITIALIZE_FAILED` if link could not be established.
`CONNECT_TIMEOUT_FAILED` if tracker did not respond.
`WRONG_LINK_VERSION` if the versions of the EyeLink library and tracker are incompatible.

4.10.2.56 getRecordingStatus()

```
def getRecordingStatus ( )
```

Checks if we are in Abort menu after recording stopped and returns trial exit code. Call this function on leaving a trial. It checks if the EyeLink tracker is displaying the Abort menu, and handles it if required. The return value from this function should be returned as the trial result code.

Remarks

This function is equivalent to the C API

```
INT16 check_record_exit(void);
```

Returns

`TRIAL_OK` if no error.
`REPEAT_TRIAL, SKIP_TRIAL, ABORT_EXPT` if Abort menu activated.

4.10.2.57 open()

```
def open ( )
```

Opens connection to single tracker. If no parameters are given, it tries to open connection to the default host (100.1.1.1).

Remarks

This is equivalent to the C API
`INT16 eyelink_open_node(ELINKADDR node, INT16 busytest);`

Parameters

eyelink_address	Optional argument. Text IP address of the host PC (the default value is, "100.1.1.1").
busytest	Optional argument. If non-zero the call to `eyelink_open_node()` will not disconnect an existing connection.

Returns

Throws Runtime error exception if it cannot open the connection.

4.10.2.58 targetModeDisplay()

```
def targetModeDisplay ( )
```

This function needs some "helper" graphics to clear the scren and draw the fixation targets. Since C graphics are compiler-dependent, these are found in other C source files.

While tracker is in any mode with fixation targets... Reproduce targets tracker needs. (if local_trigger) Local Space-bar acts as trigger. (if local_control) Local keys echoes to tracker.

Returns

`0` if OK, `27` if aborted, `TERMINATE_KEY` if pressed..

4.10.2.59 getCalibrationResult()

```
def getCalibrationResult ( )
```

Checks for a numeric result code returned by calibration, validation, or drift correction.

Remarks

This function is equivalent to the C API
`INT16 eyelink_cal_result(void);`

Returns

`NO_REPLY` if drift correction not completed yet.
`OK_RESULT (0)` if success.
`ABORT_REPLY (27)` if 'ESC' key aborted operation.
`-1` if operation failed.
`1` if poor calibration or excessive validation error.

4.10.2.60 nodeReceive()

```
def nodeReceive ( )
```

Checks for and gets the last packet received, stores the data and the node address sent from.

Remarks

Data can only be read once, and is overwritten if a new packet arrives before the last packet has been read.
This function is equivalent to the C API
`INT16 eyelink_node_receive(ELINKADDR node, void *data);`

Returns

An instance of EyeLinkMessage class is returned, if successful.

4.10.2.61 getDataCount()

```
def getDataCount ( )
```

Counts total items in queue: samples, events, or both.

Remarks

This function is equivalent to the C API
`INT16 eyelink_data_count(INT16 samples, INT16 events);`

Parameters

samples	if non-zero count the samples.
events	if non-zero count the events.

Returns

Total number of samples and events is in the queue.

4.10.2.62 getLastData()

```
def getLastData ( )
```

Gets an integer (unconverted) copy of the last/newest link data (sample or event) seen by `getNextData()`.

Remarks

This function is equivalent to the C API
`INT16 eyelink_get_last_data(void *buf);`

Returns

Object of type Sample or Event.

4.10.2.63 getSample()

```
def getSample ( )
```

Gets an integer (unconverted) sample from end of queue, discards any events encountered.

Remarks

This is equivalent to the C API
```
INT16 eyelink_get_sample(void *sample);
```

Returns

Object of type Sample.

4.10.2.64 doDriftCorrect()

```
def doDriftCorrect ( )
```

Performs a drift correction before a trial.

Remarks

This is equivalent to the C API
```
int do_drift_correct(int x, int y, int draw, int allow_setup);
```

Parameters

x	X Position (in pixels) of drift correction target.
y	Y Position (in pixels) of drift correction target.
draw	If `1`, the drift correction will clear the screen to the target background color, draw the target, and clear the screen again when the drift correction is done. If `0`, the fixation target must be drawn by the user.
allow_setup	If `1`, accesses Setup menu before returning, else aborts drift correction.

Returns

`0` if successful, `27` if 'ESC' key was pressed to enter Setup menu or abort.

4.10.2.65 setOfflineMode()

```
def setOfflineMode ( )
```

Places EyeLink tracker in off-line (idle) mode. Wait till the tracker has finished the mode transition.

Remarks

This is equivalent to the C API
```
INT16 set_offline_mode(void);
```

4.10.2.66 getNewestSample()

```
def getNewestSample ( )
```

Check if a new sample has arrived from the link. This is the latest sample, not the oldest sample that is read by `getNextData()`, and is intended to drive gaze cursors and gaze-contingent displays.

Remarks

This function is equivalent to the C API
```
INT16 CALLTYPE eyelink_newest_float_sample(void FARTYPE *buf);
```

Returns

None if there is no sample, instance of Sample type otherwise.

4.10.2.67 breakPressed()

```
def breakPressed ( )
```

Tests if the program is being interrupted. You should break out of loops immediately if this function does not return `0`, if `getkey()` return `TERMINATE_KEY`, or if `isConnected()` method of the class returns `0`.

Remarks

Under Windows XP, this call will not work in realtime mode at all, and will take several seconds to respond if graphics are being drawn continuously. This function works well in realtime mode under Windows 2000. This function is equivalent to the C API
```
INT16 break_pressed(void);
```

Returns

`1` if CTRL-C is pressed, `terminalBreak()` was called, or the program has been terminated with ALT-F4; `0` otherwise

4.10.2.68 setName()

```
def setName ( )
```

Sets the node name of this computer (up to 35 characters).

Remarks

This function is equivalent to the C API
```
INT16 eyelink_set_name(char *name);
```

Parameters

name	String to become new name.

4.10.2.69 getCurrentMode()

```
def getCurrentMode ( )
```

This function tests the current tracker mode, and returns a set of flags based of what the mode is doing. The most useful flag using the EyeLink experiment toolkit is `IN_USER_MENU` to test if the EyeLink Abort menu has been activated.

Remarks

This is equivalent to the C API
```
INT16 eyelink_current_mode(void);
```

Returns

Set of bit flags that mark mode function:

`IN_DISCONNECT_MODE`	if disconnected
`IN_IDLE_MODE`	if off-line (Idle mode)
`IN_SETUP_MODE`	if in Setup-menu related mode
`IN_RECORD_MODE`	if tracking is in progress
`IN_PLAYBACK_MODE`	if currently playing back data
`IN_TARGET_MODE`	if in mode that requires a fixation target
`IN_DRIFTCORR_MODE`	if in drift-correction
`IN_IMAGE_MODE`	if displaying grayscale camera image
`IN_USER_MENU`	if displaying Abort or user-defined menu

4.10.2.70 resetData()

```
def resetData ( )
```

Prepares link buffers to receive new data and removes old data from buffer.

4.10.2.71 getCalibrationMessage()

```
def getCalibrationMessage ( )
```

Returns text associated with result of last calibration, validation, or drift correction. This usually specifies errors or other statistics.

Remarks

This function is equivalent to the C API
`INT16 eyelink_cal_message(char *msg);`

Returns

Message string associated with result of last calibration, validation, or drift correction.

4.10.2.72 pollRemotes()

`def pollRemotes ( )`

Asks all non-tracker computers (with EyeLink software running) on the network to send their names and node address.

Remarks

This function is equivalent to the C API
`INT16 eyelink_poll_ remotes(void);`

Returns

`0` if successful, otherwise link error.

4.10.2.73 sendDataFile()

`def sendDataFile ( )`

This sends a file to the EyeLink tracker PC. Source file name and destination file name should be given. Using this function, an image or video can be uploaded from the Display PC to the Tracker PC. The image can later be used as a Gaze Cursor Backdrop via a call to SendCommand("draw_image imagename.ext")

Remarks

This function is equivalent to the C API
`int send_data_file(char *src, char *dest, int is_path=0)`

Parameters

src	Name of local file (including extension).
dest	Short Name of eye tracker file to write to (including extension).

Returns

Size of file if successful, Otherwise Runtime Exception is raised.

Example:

```
getEYELINK().sendDataFile("ur.jpg", "ur.jpg")             # transfer images as files
getEYELINK().sendDataFile("small.wmv","small.wmv")        # transfer images as files
getEYELINK().sendCommand("draw_image ur.jpg")             # draw_image <name>
getEYELINK().sendCommand("draw_video small.wmv")          # draw_video <name> <-- webui only command
```

See also

bitmapSaveAndBackdrop(), bitmapSave(), bitmapBackdrop(), sendCommand()

4.10.2.74 getTrackerVersionString()

```
def getTrackerVersionString ( )
```

After connection, determines if the connected tracker is an EyeLink I or II (use getTrackerVersion) to get number value.

Remarks

This is equivalent to the C API

```
INT16 eyelink_get_tracker_version(char *c);
```

Returns

A string indicating EyeLink tracker version.

4.10.2.75 escapePressed()

```
def escapePressed ( )
```

This function tests if the 'ESC' key is held down, and is usually used to break out of nested loops.

Remarks

Under Windows XP, this call will not work in realtime mode at all, and will take several seconds to respond if graphics are being drawn continuously. This function works well in realtime mode under Windows 2000. This function is equivalent to the C API

```
INT16 escape_pressed(void);
```

Returns

`1` if 'ESC' key held down `0` if not.

4.10.2.76 pumpMessages()

```
def pumpMessages ( )
```

Forces the graphical environment to process any pending key or mouse events.

Remarks

This function is equivalent to the C API
`INT16 message_pump(HWND dialog_hook);`

4.10.2.77 getSampleDataFlags()

```
def getSampleDataFlags ( )
```

After calling `waitForBlockStart()`, or after at least one sample or eye event has been read, returns sample data content flag (`0` if not in sample block).

Remarks

This function is equivalent to the C API
`INT16 eyelink_sample_data_flags(void);`

Returns

Possible return values are a set of the following bit flags:

Constant Name	Value	Description
SAMPLE_LEFT	0x8000	Data for left eye
SAMPLE_RIGHT	0x4000	Data for right eye
SAMPLE_TIMESTAMP	0x2000	always for link, used to compress files
SAMPLE_PUPILXY	0x1000	pupil x,y pair
SAMPLE_HREFXY	0x0800	head-referenced x,y pair
SAMPLE_GAZEXY	0x0400	gaze x,y pair
SAMPLE_GAZERES	0x0200	gaze res (x,y pixels per degree) pair
SAMPLE_PUPILSIZE	0x0100	pupil size
SAMPLE_STATUS	0x0080	error flags
SAMPLE_INPUTS	0x0040	input data port
SAMPLE_BUTTONS	0x0020	button state: LSBy state, MSBy changes
SAMPLE_HEADPOS	0x0010	head-position: byte tells # words
SAMPLE_TAGGED	0x0008	reserved variable-length tagged
SAMPLE_UTAGGED	0x0004	user-definable variable-length tagged

4.10.2.78 stopRecording()

```
def stopRecording ( )
```

Stops recording, resets EyeLink data mode. Call 50 to 100 msec after an event occurs that ends the trial. This function waits for mode switch before returning.

Remarks

This is equivalent to the C API
`void stop_recording(void);`

4.10.2.79 getButtonStates()

```
def getButtonStates ( )
```

Returns a flag word with bits set to indicate which tracker buttons are currently pressed. This is button `1` for the LSB, up to button `16` for the MSB. Buttons above `8` are not realized on the EyeLink tracker.

Remarks

This function is equivalent to the C API
`UINT16 eyelink_button_states(void);`

Returns

Flag bits for buttons currently pressed.

4.10.2.80 abort()

```
def abort ( )
```

Places EyeLink tracker in off-line (idle) mode.

Remarks

Use before attempting to draw graphics on the tracker display, transferring files, or closing link. Always call `waitForModeReady()` afterwards to ensure tracker has finished the mode transition. This function pair is implemented by the EyeLink toolkit library function `setOfflineMode()`. This function is equivalent to the C API
`INT16 eyelink_abort(void);`

Returns

`0` if mode switch begun, else link error.

4.10.2.81 stopPlayBack()

```
def stopPlayBack ( )
```

Stops playback if in progress. Flushes any data in queue.

Remarks

This function is equivalent to the C API
`INT16 eyelink_playback_stop(void);`

4.10.2.82 sendTimedCommand()

```
def sendTimedCommand ( )
```

Sends a command to the connected eye tracker, wait for reply.

Exceptions

If	there is an error, Runtime exception is raised.

See also

sendTimedCommandEx()

4.10.2.83 getLastButtonPress()

```
def getLastButtonPress ( )
```

Reads the number of the last button detected by the EyeLink tracker. This is 0 if no buttons were pressed since the last call, or since the buttons were flushed. If a pointer to a variable is supplied the eye-tracker timestamp of the button may be read. This could be used to see if a new button has been pressed since the last read. If multiple buttons were pressed since the last call, only the last button is reported.

Remarks

This function is equivalent to the C API
```
UINT16 eyelink_last_button_press(UINT32 *time);
```

Returns

Two-item tuple, recording the button last pressed (0 if no button pressed since last read) and the time of the button press.

4.10.2.84 getkeyEx()

```
def getkeyEx ( )
```

Returns the key pressed. Same as getkey() except, this returns a tuple with the first value contains the key and the second contains value contains the modifier.

Remarks

Warning: This function processes and dispatches any waiting messages. This will allow Windows to perform disk access and negates the purpose of realtime mode. Usually these delays will be only a few milliseconds, but delays over 20 milliseconds have been observed. You may wish to call `escapePressed()` or `breakPressed()` in recording loops instead of `getkey()` if timing is critical, for example in a gaze-contingent display. Under Windows XP, these calls will not work in realtime mode at all (although these do work under Windows 2000). Under Windows 95/98/Me, realtime performance is impossible even with this strategy. Some useful keys are:

- CURS_UP
- CURS_DOWN
- CURS_LEFT
- CURS_RIGHT
- ESC_KEY
- ENTER_KEY
- TERMINATE_KEY
- JUNK_KEY

This function is equivalent to the C API
```
unsigned getkey(void);
```

Returns

0 if no key pressed, else key code. TERMINATE_KEY if CTRL-C held down or program has been terminated.

4.10.2.85 isInDataBlock()

```
def isInDataBlock ( )
```

Checks to see if framing events read from queue indicate that the data is in a block containing samples, events, or both.

Remarks

The first item in queue may not be a block start even, so this should be used in a loop while discarding items using eyelink_get_next_data(NULL). NOTE: this function did not work reliably in versions of the DLL before v2.0 (did not detect end of blocks). This function is equivalent to the C API
```
INT16 eyelink_in_data_block(INT16 samples, INT16 events);
```

Parameters

samples	if non-zero, check if in a block with samples.
events	if non-zero, check if in a block with events.

Returns

0 if no data of either masked type is being sent.

4.10.2.86 nodeRequestTime()

```
def nodeRequestTime ( )
```

Sends a request the connected eye tracker to return its current time.

Remarks

The time reply can be read with getTrackerTime(). This function is equivalent to the C API
```
UINT32 eyelink_node_request_time(ELINKADDR node);
```

Parameters

address	Text IP address (for example, "100.1.1.1") for a specific tracker.

Returns

0 if no error, else link error code.

4.10.2.87 applyDriftCorrect()

```
def applyDriftCorrect ( )
```

Applies the results of the last drift correction. This is not done automatically after a drift correction, allowing the message returned by `getCalibrationMessage()` to be examined first.

Remarks

This function is equivalent to the C API
`INT16 eyelink_apply_driftcorr(void);`

Returns

`0` if command sent fine, else link error.

4.10.2.88 setAddress()

```
def setAddress ( )
```

Sets the IP address used for connection to the EyeLink tracker. This is set to "100.1.1.1" in the DLL, but may need to be changed for some network configurations. This must be set before attempting to open a connection to the tracker.

A "broadcast" address ("255.255.255.255") may be used if the tracker address is not known - this will work only if a single Ethernet card is installed, or if DLL version 2.1 or higher, and the latest tracker software versions (EyeLink I v2.1 or higher, and EyeLink II v1.1 or higher) are installed.

Remarks

This is equivalent to the C API
`INT16 set_eyelink_address(char *addr);`

Parameters

text_IP_address	Pointer to a string containing a "dotted" 4-digit IP address;

Returns

`0` if success, `-1` if could not parse address string.

4.10.2.89 readTime()

```
def readTime ( )
```

Returns the tracker time requested by `eyelink_request_time()` or `eyelink_node_request_time()`.

Returns

0 if no response yet, else timestamp in millisecond.

See also

trackerTime()

4.10.2.90 getImageCrossHairData()

```
def getImageCrossHairData ( )
```

4.10.2.91 bitmapSaveAndBackdrop()

```
def bitmapSaveAndBackdrop ( )
```

This function saves the entire bitmap as a .BMP, .JPG, .PNG, or .TIF file, and transfers the image to tracker as backdrop for gaze cursors.

Parameters

iwidth	Original image width.
iheight	Original image height.
pixels	Pixels of the image in one of two possible formats: pixel=[line1, line2, ... linen] line=[pix1,pix2,...,pixn],pix=(r,g,b). pixel=[line1, line2, ... linen] line=[pix1,pix2,...,pixn],pix=0xAARRGGBB.
xs	Crop x position.
ys	Crop y position.
width	Crop width.
height	Crop height.
fname	File name to save.
path	Path to save.
svoptions	Save options(SV_NOREPLACE, SV_MAKEPATH). If the file exists, it replaces the file unless SV_NOREPLACE is specified.
xd	X position - transfer.
yd	Y position - transfer.
xferoptions	Transfer options(BX_AVERAGE, BX_DARKEN, BX_LIGHTEN, BX_MAXCONTRAST, BX_NODITHER, BX_GRAYSCALE). Transfer options set with bitwise OR of the following constants, determines how bitmap is processed: • BX_AVERAGE Averaging combined pixels • BX_DARKEN Choosing darkest and keep thin dark lines. • BX_LIGHTEN Choosing darkest and keep thin white lines and control how bitmap size is reduced to fit tracker display. • BX_MAXCONTRAST Maximizes contrast for clearest image. • BX_NODITHER Disables the dithering of the image.
	• BX_GREYSCALE Converts the image to grayscale (grayscale works best for EyeLink I, text, etc.).

See also

bitmapBackdrop(),bitmapSave()

4.10.2.92 getEventTypeFlags()

```
def getEventTypeFlags ( )
```

After at least one button or eye event has been read, can be used to check what type of events will be available.

Remarks

This is equivalent to the C API
```
UINT16 eyelink_event_type_flags(void);
```

Returns

Possible return values are a set of the following bit flags:

Constant Name	Value	Description
LEFTEYE_EVENTS	0x8000	Has left eye events
RIGHTEYE_EVENTS	0x4000	Has right eye events
BLINK_EVENTS	0x2000	Has blink events
FIXATION_EVENTS	0x1000	Has fixation events
FIXUPDATE_EVENTS	0x0800	Has fixation updates
SACCADE_EVENTS	0x0400	Has saccade events
MESSAGE_EVENTS	0x0200	Has message events
BUTTON_EVENTS	0x0040	Has button events
INPUT_EVENTS	0x0020	Has input port events

4.10.2.93 trackerTimeOffset()

```
def trackerTimeOffset ( )
```

Returns the time difference between the tracker time and display pc time.

Remarks

This is equivalent to the C API
```
UINT32 eyelink_time_offset();
```

Returns

An integer data for the time difference (in milliseconds) between the tracker time and display pc time.

4.10.2.94 getPositionScalar()

```
def getPositionScalar ( )
```

Returns the divisor used to convert integer eye data to floating point data.

Remarks

This function is equivalent to the C API
`INT16 eyelink_position_prescaler(void);`

Returns

Integer for the divisor (usually 10).

4.10.2.95 waitForModeReady()

```
def waitForModeReady ( )
```

After a mode-change command is given to the EyeLink tracker, an additional 5 to 30 milliseconds may be needed to complete mode setup. Call this function after mode change functions.

Remarks

If it does not return `0`, assume a tracker error has occurred. This function is equivalent to the C API
`INT16 eyelink_wait_for_mode_ready(UINT32 maxwait);`

Parameters

maxwait	Maximum milliseconds to wait for the mode to change.

Returns

`0` if mode switching is done, else still waiting.

4.10.2.96 exitCalibration()

```
def exitCalibration ( )
```

This function should be called from a message or event handler if an ongoing call to `doDriftCorrect()` or `doTrackerSetup()` should return immediately.

Remarks

This function is equivalent to the C API
`void exit_calibration(void);`

4.10.2.97 openNode()

```
def openNode ( )
```

Allows the computer to connect to tracker, where the tracker is on the same network.

Remarks

This is equivalent to the C API
`INT16 eyelink_open_node(ELINKADDR node, INT16 busytest);`
with node parameter converted from text to ELINKADDR.

Parameters

eyelink_address	Text IP address of the host PC (the default value is, "100.1.1.1").
busytest	If non-zero the call to `openNode()` will not disconnect an existing connection.

Returns

Throws Runtime Exception if it connects to the remote host.

4.10.2.98 flushKeybuttons()

```
def flushKeybuttons ( )
```

Causes the EyeLink tracker and the EyeLink library to flush any stored button or key events. This should be used before a trial to get rid of old button responses. The `<enable_buttons>` argument controls whether the EyeLink library will store button press and release events. It always stores tracker key events. Even if disabled, the last button pressed and button flag bits are updated.

Remarks

This is equivalent to the C API
`INT16 eyelink_flush_keybuttons(INT16 enable_buttons);`

Parameters

enable_buttons	Sets to `0` to monitor last button press only, `1` to queue button events.

Returns

Always `0`.

4.10.2.99 commandResult()

```
def commandResult ( )
```

Check for and retrieves the numeric result code sent by the tracker from the last command.

Remarks

This function is equivalent to the C API
`INT16 eyelink_command_result(void);`

Returns

`NO_REPLY` if no reply to last command.
`OK_RESULT (0)` if OK.
Other error codes represent tracker execution error.

4.10.2.100 getFloatData()

```
def getFloatData ( )
```

Reads data of a specific type returned by `getNextData()`. If this function called multiple times without calling `getNextData()`, the same data is returned.

Remarks

This function is equivalent to the C API
`INT16 eyelink_get_next_data(void *buf);`

Returns

None if no data available. Otherwise, a valid data is returned. The returned data type can be:

- Sample
- StartBlinkEvent
- EndBlinkEvent
- StartSacadeEvent
- EndSacadeEvent
- StartFixationEvent
- EndFixationEvent
- FixUpdateEvent
- IOEvent
- MessageEvent

4.10.2.101 getTargetPositionAndState()

```
def getTargetPositionAndState ( )
```

Returns the current target position and state.

Remarks

This function is equivalent to the C API
`INT16 eyelink_target_check(INT16 *x, INT16 *y);`

Returns

A three-item tuple holding (in the following order):

- the target visibility (`1` if visible, 0 if not),
- x position of the target,
- and y position of the target.

4.10.2.102 startDriftCorrect()

```
def startDriftCorrect ( )
```

Sets the position of the drift correction target, and switches the tracker to drift-correction mode. Should be followed by a call to `waitForModeReady()` method.

Remarks

This function is equivalent to the C API
`INT16 eyelink_driftcorr_start(INT16 x, INT16 y);`

Parameters

x	x position of the target.
y	y position of the target.

Returns

`0` if command sent fine, else link error.

4.10.2.103 trackerTime()

```
def trackerTime ( )
```

Returns the current tracker time (in milliseconds) since the tracker application started.

Remarks

This is equivalent to the C API
`UINT32 eyelink_tracker_time();`

Returns

An integer data for the current tracker time (in milliseconds) since tracker initialization.

4.11 EyeLinkCustomDisplay Class Reference

EyeLinkCustomDisplay is an abstract class, that one would implement to present calibration/validation/drift correction targets and camera images.

Public Member Functions

- def __init__ (self)
  Constructor takes no parameters.
- def __updateimgsize__ (self, width, height)
  Internal function to update the mage size.
- def setup_cal_display (self)
  This function is called to setup calibration/validation display.
- def exit_cal_display (self)
  This function is called just before exiting calibration/validation display.
- def record_abort_hide (self)
  This function is called if abort of record.
- def setup_image_display (self, width, height)
  This function is used to setup image display.
- def image_title (self, title)
  This function displays the camera image title.
- def draw_image_line (self, width, line, totlines, buff)
  This function is used to display an image line.
- def set_image_palette (self, red, green, blue)
  This function is called to setup the image palettes.
- def exit_image_display (self)
  Called to end the image display.
- def clear_cal_display (self)
  Called to clear the calibration display.
- def erase_cal_target (self)
  Called to erase the calibration or validation target.
- def draw_cal_target (self, x, y)
  Called to draw the calibration or validation target.
- def play_beep (self, beepid)
  Called to play target beeps.
- def get_input_key (self)
  This function should return list of KeyInput.
- def alert_printf (self, msg)
  Called to notify any error message to display or print.
- def draw_line (self, x1, y1, x2, y2, colorindex)
  Called to draw the cross hair, in response to the call to `draw_cross_hair()`.

- def draw_lozenge (self, x, y, width, height, colorindex)

  Called to draw the cross hair, in response to the call to `draw_cross_hair()`.
- def get_mouse_state (self)

  Called to get the mouse location.
- def draw_cross_hair (self)

  User call this function to request draw cross hair.

4.11.1 Detailed Description

EyeLinkCustomDisplay is an abstract class, that one would implement to present calibration/validation/drift correction targets and camera images.

In addition, EyeLinkCustomDisplay can also play target beeps and display error messages.

To use custom display do the following.

- 1. Implement EyeLinkCustomDisplay
- 2. Create an instance of the custom display object
- 3. Use pylink.openGraphicsEx to let pylink know to use the custom display object

Example:

```
genv = EyeLinkCoreGraphicsPyGame(800,600,eyelinktracker)
openGraphicsEx(genv)
```

Example implementation of EyeLinkCustomDisplay Code to implement EyeLinkCustomDisplay using pygame.

4.11.2 Member Function Documentation

4.11.2.1 __updateimgsize__()

```
def __updateimgsize__ (
              self,
              width,
              height )
```

Internal function to update the mage size.

The size set by this function is used to draw cross hair when `draw_cross_hair()` is called This function should **not** be overridden and should **not** be called other than the display mechanism.

Parameters

width	Width of the image.
height	Height of the image.

4.11.2.2 setup_cal_display()

```
def setup_cal_display (
              self )
```

This function is called to setup calibration/validation display.

This will be called just before we enter into the calibration or validation or drift correction mode. Any allocation per calibration or validation drift correction can be done here. Also, it is normal to clear the display in this call.

4.11.2.3 exit_cal_display()

```
def exit_cal_display (
              self )
```

This function is called just before exiting calibration/validation display.

Any resource allocation done in `setup_cal_display()` can be cleared.

4.11.2.4 record_abort_hide()

```
def record_abort_hide (
              self )
```

This function is called if abort of record.

It is used to hide display from subject.

4.11.2.5 setup_image_display()

```
def setup_image_display (
              self,
              width,
              height )
```

This function is used to setup image display.

It takes expected image size of the source image. This may be called repeatedly for same display. If this fails, It should return `1` if success and `0` otherwise.

Parameters

width	Width of the incoming image.
height	Height of the incoming image.

Returns

`1` if success, `0` otherwise.

4.11.2.6 image_title()

```
def image_title (
               self,
               title )
```

This function displays the camera image title.

This is called whenever the title changes

Parameters

title	title change

4.11.2.7 draw_image_line()

```
def draw_image_line (
               self,
               width,
               line,
               totlines,
               buff )
```

This function is used to display an image line.

This function is called with an array of bytes containing picture colors. The byte on pixels are just palette indexes. This index should be used against the palette created on the call to `set_image_palette_hook()`. The image is given line by line from top to bottom. It may be efficient to collect one full image and do a full blit of the entire image.

```
i =0
imline = self.imagebuffer[line-1]
while i <width:
        imline[i] = self.pal[buff[i]]
i= i+1
```

4.11.2.8 set_image_palette()

```
def set_image_palette (
               self,
               red,
               green,
               blue )
```

This function is called to setup the image palettes.

The function is called with a set of RGB colors to set up for next image.

```
self.pal = []
while i < sz:
    rf = int(b[i])
    gf = int(g[i])
    bf = int(r[i])
    self.pal.append((rf«16) | (gf«8) | (bf))
    i = i+1
```

4.11.2.9 erase_cal_target()

```
def erase_cal_target (
              self )
```

Called to erase the calibration or validation target.

Erase the target drawn by the previous call to `draw_cal_target()`.

4.11.2.10 draw_cal_target()

```
def draw_cal_target (
              self,
              x,
              y )
```

Called to draw the calibration or validation target.

Draw a target at x,y and display it.

Remarks

x and y values are relative to the active `screen_pixel_coords` command.

Parameters

x	X location to draw the target.
y	Y location to draw the target.

4.11.2.11 play_beep()

```
def play_beep (
              self,
              beepid )
```

Called to play target beeps.

Parameters

beepid	Id of the beep to be played. Possible values for beepid are: • CAL_ERR_BEEP =-1 • DC_ERR_BEEP =-2 • CAL_GOOD_BEEP = 0 • CAL_TARG_BEEP = 1 • DC_GOOD_BEEP = 2 • DC_TARG_BEEP = 3

4.11.2.12 get_input_key()

```
def get_input_key (
                self )
```

This function should return list of KeyInput.

If there are not keys, return an empty list or None.

4.11.2.13 draw_line()

```
def draw_line (
                self,
                x1,
                y1,
                x2,
                y2,
                colorindex )
```

Called to draw the cross hair, in response to the call to `draw_cross_hair()`.

This function should draw a line from (x1,y1) to (x2,y2). The x and y values are relative to the width and height of the image, given at `setup_image_display()`.

Parameters

x1	Starting x position.
y1	Starting y position.
x2	Ending x position.
y2	Ending y position.
colorindex	Color id of the line. Possible value for colorindex are: • CR_HAIR_COLOR=1 • PUPIL_HAIR_COLOR=2 • PUPIL_BOX_COLOR=3 • SEARCH_LIMIT_BOX_COLOR=4 • MOUSE_CURSOR_COLOR=5

4.11.2.14 draw_lozenge()

```
def draw_lozenge (
              self,
              x,
              y,
              width,
              height,
              colorindex )
```

Called to draw the cross hair, in response to the call to `draw_cross_hair()`.

This function should draw an lozenge bounded by the box (x,y),(width,height). The x and y values are relative to the width and height of the image, given at `setup_image_display()`.

Remarks

This function is not used at the moment.

Parameters

x	Starting x position.
y	Starting y position.
width	bounding width
height	bounding height
colorindex	Color id of the ellipse. Possible value for colorindex are: • CR_HAIR_COLOR=1 • PUPIL_HAIR_COLOR=2 • PUPIL_BOX_COLOR=3 • SEARCH_LIMIT_BOX_COLOR=4 • MOUSE_CURSOR_COLOR=5

4.11.2.15 get_mouse_state()

```
def get_mouse_state (
              self )
```

Called to get the mouse location.

This function should return the mouse location and the state at the time of call. ((x,y),state). At the moment we only care if the mouse is clicked or not. So, if clicked the state = 1, 0 otherwise. This function is only useful for EyeLink1000.

4.11.2.16 draw_cross_hair()

```
def draw_cross_hair (
              self )
```

User call this function to request draw cross hair.

After completion of filling in the camera image, call this function to draw the cross hair on the camera image.

4.12 EyeLinkListener Class Reference

EyeLinkListener class implements most of the core EyeLink interface.

Inherits EyeLinkCBind.

Inherited by EyeLink.

Public Member Functions

- def getTrackerInfo (self)
  Returns the current tracker information.
- def drawCalTarget (self, position)
  Allow the normal calibration target drawing to proceed at different locations.
- def getCurrentTime (self)
  returns the current tracker time.
- def getSampleRate (self)
  returns the current sample rate.
- def getCRMode (self)
  returns the current mode data, either PUPIL_ONLY or PUPIL_CR.
- def getLinkFilter (self)
  returns the Link Filter Level.
- def getFileFilter (self)
  returns the File Filter Level.
- def getEyeUsed (self)
  returns the eye used.
- def sendMessage (self, message_text, offset=0)
  Sends the given message to the connected EyeLink tracker.
- def imageBackdrop (self, filename, Xs, Ys, width, height, Xd, Yd, xferoptions)
  Sends the given image file backdrop to the connected EyeLink tracker.

4.12.1 Detailed Description

EyeLinkListener class implements most of the core EyeLink interface.

This includes the simple connection to the eye tracker, sending commands and messages to the tracker, opening and saving a recording file, performing calibration and drift correction, real-time access to tracker data and eye movement events (such as fixations, blinks, and saccades), as well as other important operations.

An instance of the EyeLinkListener class can be created by using the class constructor function. For example,

```
try:
    EYELINK = EyeLinkListener()
except:
    EYELINK = None
```

All of the methods should be called in the format of: EYELINK.functionName(parameters), where EYELINK is an instance of the EyeLinkListener class.

4.12.2 Member Function Documentation

4.12.2.1 getTrackerInfo()

```
def getTrackerInfo (
              self )
```

Returns the current tracker information.

Returns

An instance of the ILinkData class.

4.12.2.2 drawCalTarget()

```
def drawCalTarget (
              self,
              position )
```

Allow the normal calibration target drawing to proceed at different locations.

This is equivalent to the C API
`INT16 CALLTYPE set_draw_cal_target_hook(INT16 (CALLBACK * erase_cal_target_hook)(HDC hdc), INT16 options);`

Parameters

position	A tuple in the format of (x, y), passing along the position of drift correction target. X and y are in screen pixels.

4.12.2.3 sendMessage()

```
def sendMessage (
              self,
              message_text,
              offset = 0 )
```

Sends the given message to the connected EyeLink tracker.

The message will be written to the EyeLink tracker.

Remarks

This is equivalent to the C API
`int eyecmd_printf(char *fmt, ...);`

The maximum text length is 130 characters. If the given string has more than 130 characters, the first 130 characters will be sent and if the send passes this function will return 1. If the text is not truncated, 0 will be returned on a successful message send.

Parameters

message_text	text message to be sent. It does not support printf() kind of formatting.
offset	time offset in millisencond for the message.

Returns

If there is any problem sending the message, a runtime exception is raised.

4.12.2.4 imageBackdrop()

```
def imageBackdrop (
              self,
              filename,
              Xs,
              Ys,
              width,
              height,
              Xd,
              Yd,
              xferoptions )
```

Sends the given image file backdrop to the connected EyeLink tracker.

Remarks

This will open the image file, convert to bitmap using PIL.Image and call bitmapBackdrop to send the image to host

Parameters

filename	- full or relative path of the image file name
Xs	- crop x position
Ys	- crop y position
width	- crop width
height	- crop height
Xd	- xposition - transfer
Yd	- yposition - transfer
xferoptions	- transfer options(BX_AVERAGE,BX_DARKEN,BX_LIGHTEN,BX_MAXCONTRAST,BX_↩ NODITHER,BX_GRAYSCALE)

Returns

if PIL couldn't load, it will return None, otherwise return value of bitmapBackdrop

4.13 EyelinkMessage Class Reference

EyelinkMessage class, derived from EyeLinkAddress class, is used to send and receive messages between EyeLink nodes.

Inherits EyeLinkAddress.

Public Member Functions

- def __init__ (self, ip=(100, 1, 1, 1), port=4000, msg="")

  Constructor.
- def getText ()

  Returns the message to be sent to or received from the node.

4.13.1 Detailed Description

EyelinkMessage class, derived from EyeLinkAddress class, is used to send and receive messages between EyeLink nodes.

Instances of this class are commonly used as the return values of the `nodeReceive()` and `getNode()` methods of the EyeLinkListener or EyeLink class. An instance of EyelinkMessage class can be initialized with the class constructor:

```
EyelinkMessage(ip = (100,1,1,1), port = 4000, msg = "")
```

where *ip* is a four-item tuple containing the IP address of the EyeLink node, *port* is the port number of the connection, *msg* is the message to be sent to or received from the node.

For example,

```
myMessage = EyelinkMessage((100, 1, 1, 1), 4000, "test")
```

4.13.2 Constructor & Destructor Documentation

4.13.2.1 __init__()

```
def __init__ (
              self,
              ip = (100,1,1,1),
              port = 4000,
              msg = "" )
```

Constructor.

Parameters

ip	optional ipaddress in tuple form. eg. if the ip address is 192.168.25.48, the tuple form is (192,168,25,48) The default ip address is 100.1.1.1
port	optional port value as integer. The default value is 4000.
msg	text message.

4.13.3 Member Function Documentation

4.13.3.1 getText()

```
def getText ( )
```

Returns the message to be sent to or received from the node.

Returns

Text message.

4.14 FixUpdateEvent Class Reference

Class to represent the Fix Update event.

Inherits StartNonBlinkEvent, and EndNonBlinkEvent.

Public Member Functions

- def getStartPupilSize (self)

  Pupil size (in arbitrary units, area or diameter as selected) at the start of a fixation interval.
- def getAverageGaze (self)

  The average gaze position during the fixation period (in pixel coordinates set by the `screen_pixel_coords` command).
- def getAverageHREF (self)

  Average HEADREF position during the fixation period.
- def getAveragePupilSize (self)

  Average pupil size (in arbitrary units, area or diameter as selected) during a fixation.
- def getEndPupilSize (self)

  Pupil size (in arbitrary units, area or diameter as selected) at the end of a fixation interval.

4.14.1 Detailed Description

Class to represent the Fix Update event.

This also inherits all properties from StartNonBlinkEvent and EndNonBlinkEvent.

4.14.2 Member Function Documentation

4.14.2.1 getStartPupilSize()

```
def getStartPupilSize (
              self )
```

Pupil size (in arbitrary units, area or diameter as selected) at the start of a fixation interval.

Returns

Float.

4.14.2.2 getAverageGaze()

```
def getAverageGaze (
              self )
```

The average gaze position during the fixation period (in pixel coordinates set by the `screen_pixel_coords` command).

Returns

Two-item tuple in the format of (float, float).

4.14.2.3 getAverageHREF()

```
def getAverageHREF (
              self )
```

Average HEADREF position during the fixation period.

Returns

Two-item tuple in the format of (float, float).

4.14.2.4 getAveragePupilSize()

```
def getAveragePupilSize (
              self )
```

Average pupil size (in arbitrary units, area or diameter as selected) during a fixation.

Returns

Float.

4.14.2.5 getEndPupilSize()

```
def getEndPupilSize (
              self )
```

Pupil size (in arbitrary units, area or diameter as selected) at the end of a fixation interval.

Returns

Float.

4.15 ILinkData Class Reference

Class to represent tracker status information such as time stamps, flags, tracker addresses and so on.

Public Member Functions

- def getTime (self)
  Time of last control event.
- def getSampleRate (self)
  10∗sample rate (0 if no samples, 1 if nonconstant).
- def getSampleDivisor (self)
  Sample "divisor" (min msec between samples).
- def getPrescaler (self)
  Amount to divide gaze x,y,res by.
- def getVelocityPrescaler (self)
  Amount to divide velocity by.
- def getPupilPrescaler (self)
  Pupil prescale (1 if area, greater if diameter).
- def getHeadDistancePrescaler (self)
  Head-distance prescale (to mm).
- def getSampleDataFlags (self)
  0 if off, else all flags.
- def getEventDataFlags (self)
  0 if off, else all flags.
- def getEventTypeFlags (self)

0 if off, else event-type flags.

- def isInBlockWithSamples (self)

  Set if in block with samples.
- def isInBlockWithEvents (self)

  Set if in block with events.
- def haveLeftEye (self)

  Set if any left-eye data expected.
- def haveRightEye (self)

  Set if any right-eye data expected.
- def getLostDataTypes (self)

  Flags what we lost before last item.
- def getLastBufferType (self)

  Buffer-type code.
- def getLastBufferSize (self)

  Buffer size of last item.
- def isControlEvent (self)

  Set if control event read with last data.
- def isNewBlock (self)

  Set if control event started new block.
- def getLastItemTimeStamp (self)

  Time field of item.
- def getLastItemType (self)

  Type: 100 = sample, 0 = none, else event type.
- def getLastItemContent (self)

  Content: <read> (IEVENT), <flags> (ISAMPLE).
- def getBlockNumber (self)

  Block in file.
- def getSamplesInBlock (self)

  Samples read in block so far.
- def getEventsInBlock (self)

  Events (excl.
- def getLastResX (self)

  Updated by samples only.
- def getLastResY (self)

  Updated by samples only.
- def getLastPupil (self)

  Updated by samples only.
- def getLastItemStatus (self)

  Updated by samples, events.
- def getSampleQueueLength (self)

  Number of items in queue.
- def getEventQueueLength (self)

  Includes control events.
- def getQueueSize (self)

  Total queue buffer size.
- def getFreeQueueLength (self)

  Unused bytes in queue.
- def getLastReceiveTime (self)

  Time tracker last sent packet.
- def isSamplesEnabled (self)

  Data type rcve enable (switch).

- def isEventsEnabled (self)

  Data type rcve enable (switch).
- def getPacketFlags (self)

  Status flags from data packet.
- def getLinkFlags (self)

  Status flags from link packet header.
- def getStateFlags (self)

  Tracker error state flags.
- def getTrackerDataOutputState (self)

  Tracker data output state.
- def getPendingCommands (self)

  Tracker commands pending.
- def isPoolingRemote (self)

  `1` *if polling remotes, else polling trackers.*
- def getPoolResponse (self)

  Total nodes responding to polling.
- def getReserved (self)

  `0` *for EyeLink I or original EyeLink API DLL.*
- def getName (self)

  A name for our machine.
- def getTrackerName (self)

  Name of tracker connected to.
- def getNodes (self)

  Data on nodes.
- def getLastItem (self)

  Buffer containing last item.
- def getAddress (self)

  Address of our machine.
- def getTrackerAddress (self)

  Address of the connected tracker.
- def getTrackerBroadcastAddress (self)

  Broadcast address for eye trackers.
- def getRemoteBroadcastAddress (self)

  Broadcast address for remotes Equivalent field in ILINKDATA "C": rbroadcast_address.

4.15.1 Detailed Description

Class to represent tracker status information such as time stamps, flags, tracker addresses and so on.

A valid reference to this object can be obtained by calling the function `getEYELINK()`.getTrackerInfo().

4.15.2 Member Function Documentation

4.15.2.1 getTime()

```
def getTime (
              self )
```

Time of last control event.

Equivalent field in ILINKDATA "C": Time.

4.15.2.2 getSampleRate()

```
def getSampleRate (
              self )
```

10∗sample rate (0 if no samples, 1 if nonconstant).

Equivalent field in ILINKDATA "C": samrate.

4.15.2.3 getSampleDivisor()

```
def getSampleDivisor (
              self )
```

Sample "divisor" (min msec between samples).

Equivalent field in ILINKDATA "C": samdiv.

4.15.2.4 getPrescaler()

```
def getPrescaler (
              self )
```

Amount to divide gaze x,y,res by.

Equivalent field in ILINKDATA "C": prescaler.

4.15.2.5 getVelocityPrescaler()

```
def getVelocityPrescaler (
              self )
```

Amount to divide velocity by.

Equivalent field in ILINKDATA "C": vprescaler.

4.15.2.6 getPupilPrescaler()

```
def getPupilPrescaler (
              self )
```

Pupil prescale (1 if area, greater if diameter).

Equivalent field in ILINKDATA "C": pprescaler.

4.15.2.7 getHeadDistancePrescaler()

```
def getHeadDistancePrescaler (
              self )
```

Head-distance prescale (to mm).

Equivalent field in ILINKDATA "C": hprescaler.

4.15.2.8 getSampleDataFlags()

```
def getSampleDataFlags (
              self )
```

0 if off, else all flags.

Equivalent field in ILINKDATA "C": sample_data.

4.15.2.9 getEventDataFlags()

```
def getEventDataFlags (
              self )
```

0 if off, else all flags.

Equivalent field in ILINKDATA "C": event_data.

4.15.2.10 getEventTypeFlags()

```
def getEventTypeFlags (
              self )
```

0 if off, else event-type flags.

Equivalent field in ILINKDATA "C": event_types.

4.15.2.11 isInBlockWithSamples()

```
def isInBlockWithSamples (
              self )
```

Set if in block with samples.

Equivalent field in ILINKDATA "C": in_sample_block.

4.15.2.12 isInBlockWithEvents()

```
def isInBlockWithEvents (
              self )
```

Set if in block with events.

Equivalent field in ILINKDATA "C": in_event_block.

4.15.2.13 haveLeftEye()

```
def haveLeftEye (
              self )
```

Set if any left-eye data expected.

Equivalent field in ILINKDATA "C": have_left_eye.

4.15.2.14 haveRightEye()

```
def haveRightEye (
              self )
```

Set if any right-eye data expected.

Equivalent field in ILINKDATA "C": have_right_eye.

4.15.2.15 getLostDataTypes()

```
def getLostDataTypes (
              self )
```

Flags what we lost before last item.

Equivalent field in ILINKDATA "C": last_data_gap_types.

4.15.2.16 getLastBufferType()

```
def getLastBufferType (
              self )
```

Buffer-type code.

Equivalent field in ILINKDATA "C": last_data_buffer_type.

4.15.2.17 getLastBufferSize()

```
def getLastBufferSize (
              self )
```

Buffer size of last item.

Equivalent field in ILINKDATA "C": last_data_buffer_size.

4.15.2.18 isControlEvent()

```
def isControlEvent (
              self )
```

Set if control event read with last data.

Equivalent field in ILINKDATA "C": control_read.

4.15.2.19 isNewBlock()

```
def isNewBlock (
              self )
```

Set if control event started new block.

Equivalent field in ILINKDATA "C": first_in_block.

4.15.2.20 getLastItemTimeStamp()

```
def getLastItemTimeStamp (
              self )
```

Time field of item.

Equivalent field in ILINKDATA "C": last_data_item_time.

4.15.2.21 getLastItemType()

```
def getLastItemType (
              self )
```

Type: `100` = sample, `0` = none, else event type.

Equivalent field in ILINKDATA "C": last_data_item_type.

4.15.2.22 getLastItemContent()

```
def getLastItemContent (
              self )
```

Content: <read> (`IEVENT`), <flags> (`ISAMPLE`).

Equivalent field in ILINKDATA "C": last_data_item_contents.

4.15.2.23 getBlockNumber()

```
def getBlockNumber (
              self )
```

Block in file.

Equivalent field in ILINKDATA "C": block_number.

4.15.2.24 getSamplesInBlock()

```
def getSamplesInBlock (
              self )
```

Samples read in block so far.

Equivalent field in ILINKDATA "C": block_sample.

4.15.2.25 getEventsInBlock()

```
def getEventsInBlock (
              self )
```

Events (excl.

control read in block). Equivalent field in ILINKDATA "C": block_event.

4.15.2.26 getLastResX()

```
def getLastResX (
              self )
```

Updated by samples only.

Equivalent field in ILINKDATA "C": last_resx.

4.15.2.27 getLastResY()

```
def getLastResY (
              self )
```

Updated by samples only.

Equivalent field in ILINKDATA "C": last_resy.

4.15.2.28 getLastPupil()

```
def getLastPupil (
              self )
```

Updated by samples only.

Equivalent field in ILINKDATA "C": last_pupil.

4.15.2.29 getLastItemStatus()

```
def getLastItemStatus (
              self )
```

Updated by samples, events.

Equivalent field in ILINKDATA "C": last_status.

4.15.2.30 getSampleQueueLength()

```
def getSampleQueueLength (
              self )
```

Number of items in queue.

Equivalent field in ILINKDATA "C": queued_samples.

4.15.2.31 getEventQueueLength()

```
def getEventQueueLength (
              self )
```

Includes control events.

Equivalent field in ILINKDATA "C": queued_events.

4.15.2.32 getQueueSize()

```
def getQueueSize (
              self )
```

Total queue buffer size.

Equivalent field in ILINKDATA "C": queue_size.

4.15.2.33 getFreeQueueLength()

```
def getFreeQueueLength (
              self )
```

Unused bytes in queue.

Equivalent field in ILINKDATA "C": queue_free.

4.15.2.34 getLastReceiveTime()

```
def getLastReceiveTime (
              self )
```

Time tracker last sent packet.

Equivalent field in ILINKDATA "C": last_rcve_time.

4.15.2.35 isSamplesEnabled()

```
def isSamplesEnabled (
              self )
```

Data type rcve enable (switch).

Equivalent field in ILINKDATA "C": samples_on.

4.15.2.36 isEventsEnabled()

```
def isEventsEnabled (
              self )
```

Data type rcve enable (switch).

Equivalent field in ILINKDATA "C": events_on.

4.15.2.37 getPacketFlags()

```
def getPacketFlags (
              self )
```

Status flags from data packet.

Equivalent field in ILINKDATA "C": packet_flags.

4.15.2.38 getLinkFlags()

```
def getLinkFlags (
              self )
```

Status flags from link packet header.

Equivalent field in ILINKDATA "C": link_flags.

4.15.2.39 getStateFlags()

```
def getStateFlags (
              self )
```

Tracker error state flags.

Equivalent field in ILINKDATA "C": state_flags.

4.15.2.40 getTrackerDataOutputState()

```
def getTrackerDataOutputState (
              self )
```

Tracker data output state.

Equivalent field in ILINKDATA "C": link_dstatus.

4.15.2.41 getPendingCommands()

```
def getPendingCommands (
              self )
```

Tracker commands pending.

Equivalent field in ILINKDATA "C": link_pendcmd.

4.15.2.42 isPoolingRemote()

```
def isPoolingRemote (
              self )
```

`1` if polling remotes, else polling trackers.

Equivalent field in ILINKDATA "C": polling_remotes.

4.15.2.43 getPoolResponse()

```
def getPoolResponse (
              self )
```

Total nodes responding to polling.

Equivalent field in ILINKDATA "C": poll_responses.

4.15.2.44 getReserved()

```
def getReserved (
              self )
```

0 for EyeLink I or original EyeLink API DLL.

Equivalent field in ILINKDATA "C": reserved.

4.15.2.45 getName()

```
def getName (
              self )
```

A name for our machine.

Equivalent field in ILINKDATA "C": our_name.

4.15.2.46 getTrackerName()

```
def getTrackerName (
              self )
```

Name of tracker connected to.

Equivalent field in ILINKDATA "C": eye_name.

4.15.2.47 getNodes()

```
def getNodes (
              self )
```

Data on nodes.

Equivalent field in ILINKDATA "C": nodes.

4.15.2.48 getLastItem()

```
def getLastItem (
              self )
```

Buffer containing last item.

Equivalent field in ILINKDATA "C": last_data_item.

4.15.2.49 getAddress()

```
def getAddress (
              self )
```

Address of our machine.

Equivalent field in ILINKDATA "C": our_address.

4.15.2.50 getTrackerAddress()

```
def getTrackerAddress (
              self )
```

Address of the connected tracker.

Equivalent field in ILINKDATA "C": eye_address.

4.15.2.51 getTrackerBroadcastAddress()

```
def getTrackerBroadcastAddress (
              self )
```

Broadcast address for eye trackers.

Equivalent field in ILINKDATA "C": ebroadcast_address.

4.16 IOEvent Class Reference

IOEvent class is used to handle `BUTTONEVENT` and `INPUTEVENT` types, which report changes in button status or in the input port data.

Inherited by ButtonEvent.

Public Member Functions

- def getTime (self)

  Timestamp of the sample causing event (in milliseconds since EyeLink tracker was activated).
- def getType (self)

  The event code.
- def getData (self)

  Coded event data.

4.16.1 Detailed Description

IOEvent class is used to handle BUTTONEVENT and INPUTEVENT types, which report changes in button status or in the input port data.

The getTime() method records the timestamp of the eye-data sample where the change occurred, although the event itself is usually sent before that sample. Button events from the link are rarely used; monitoring buttons with one of readKeybutton(), lastButtonPress(), or buttonStates() of the EyeLink class methods is preferable, since these can report button states at any time, not just during recording.

Returned by getFloatData() whenever there is an IOEvent.

4.16.2 Member Function Documentation

4.16.2.1 getTime()

```
def getTime (
              self )
```

Timestamp of the sample causing event (in milliseconds since EyeLink tracker was activated).

Returns

Long integer.

4.16.2.2 getType()

```
def getType (
              self )
```

The event code.

This should be BUTTONEVENT (i.e., 25) or INPUTEVENT (i.e., 28).

Returns

Integer.

4.16.2.3 getData()

```
def getData (
              self )
```

Coded event data.

Returns

Long integer.

4.17 KeyInput Class Reference

This represents a key input.

4.17.1 Detailed Description

This represents a key input.

This is used with EyeLinkCustomDisplay to notify the eyelink_core.dll that a key input is available.

4.18 MessageEvent Class Reference

A message event is created by your experiment program and placed in the EDF file.

Public Member Functions

- def getTime (self)

  Timestamp of the sample causing event (when camera imaged eye, in milliseconds since EyeLink tracker was activated).
- def getType (self)

  The event code.
- def getText (self)

  Message contents (max length 255 characters).

4.18.1 Detailed Description

A message event is created by your experiment program and placed in the EDF file.

It is possible to enable the sending of these messages back through the link, although there is rarely a reason to do this. The MessageEvent class is designed to hold information on EyeLink message events retrieved from the link. Please note that all methods for the MessageEvent class do not take a parameter.

Returned by `getFloatData()` whenever there is a Message Event.

4.18.2 Member Function Documentation

4.18.2.1 getTime()

```
def getTime (
              self )
```

Timestamp of the sample causing event (when camera imaged eye, in milliseconds since EyeLink tracker was activated).

Returns

Long integer.

4.18.2.2 getType()

```
def getType (
              self )
```

The event code.

This should be `MESSAGEEVENT` (i.e., 24).

Returns

Integer.

4.18.2.3 getText()

```
def getText (
              self )
```

Message contents (max length 255 characters).

Returns

String.

4.19 Sample Class Reference

The EyeLink toolkit library defines special data classes that allow the same programming calls to be used on different platforms such as Windows, Linux and macOS.

Public Member Functions

- def initFromSample (self, sample)

  Convenient method to clone a sample.
- def isLeftSample (self)

  1 if the sample contains the left eye data; 0 if not.
- def isRightSample (self)

  1 if the sample contains the right eye data; 0 if not.
- def isBinocular (self)

  1 if the sample contains data from both eyes; 0 if not.
- def getTime (self)

  Timestamp when camera imaged eye (in milliseconds since EyeLink tracker was activated).
- def getType (self)

  Always SAMPLE_TYPE.
- def getPPD (self)

  Angular resolution at current gaze position in screen pixels per visual degree.
- def getStatus (self)

  Error and status flags (only useful for EyeLink II and newer hardware, report CR status and tracking error).
- def getInput (self)

  Data from input port(s).
- def getFlags (self)

  Bits indicating what types of data are present, and for which eye(s).
- def getButtons (self)

  Button input data: high 8 bits indicate changes from last sample, low 8 bits indicate current state of buttons 8 (MSB) to 1 (LSB).
- def getRightEye (self)

  Returns the sample data information from the desired eye.
- def getLeftEye (self)

  Returns the sample data information from the desired eye.
- def getHData (self)

  Returns the href data.
- def getEye (self)

  Returns Eye data status.
- def getTargetDistance (self)

  Target Distance.
- def getTargetX (self)

  Target X.
- def getTargetY (self)

  Target Y.
- def getTargetFlags (self)

  Target Flags.

4.19.1 Detailed Description

The EyeLink toolkit library defines special data classes that allow the same programming calls to be used on different platforms such as Windows, Linux and macOS.

You will need to know these classes to read the examples and to write your own experiments. In this documentation, the common data classes are: Sample class, Eye Event Classes, MessageEvent Class, and IOEvent Class. You only need to read this section if you are planning to use real-time link data for gaze-contingent displays or gaze-controlled interfaces, or to use data playback.

The EyeLink tracker measures eye position 250 or 2000 times per second depending on the tracking hardware and the tracker mode you are working with, and computes true gaze position on the display using the head camera data. This data is stored in the EDF file, and made available through the link in as little as 3 milliseconds after a physical eye movement.

Samples can be read from the link by `getFloatData()` or `getNewestSample()` method of the EyeLink/↩ EyeLinkLisenter class. These functions can return instances of Sample class. For example,

```
newSample = getEYELINK().getFloatData()
```

The following methods can be used to retrieve properties of a Sample class instance. For example, the timestamp of the sample can be retrieved as `newSample.getTime()`. Please note that all methods for the Sample class do not take a parameter whereas the return values are noted.

4.19.2 Member Function Documentation

4.19.2.1 isLeftSample()

```
def isLeftSample (
              self )
```

`1` if the sample contains the left eye data; `0` if not.

Returns

Integer.

4.19.2.2 isRightSample()

```
def isRightSample (
              self )
```

`1` if the sample contains the right eye data; `0` if not.

Returns

Integer.

4.19.2.3 isBinocular()

```
def isBinocular (
              self )
```

`1` if the sample contains data from both eyes; `0` if not.

Returns

Integer.

4.19.2.4 getTime()

```
def getTime (
              self )
```

Timestamp when camera imaged eye (in milliseconds since EyeLink tracker was activated).

Returns

Long integer.

4.19.2.5 getType()

```
def getType (
              self )
```

Always SAMPLE_TYPE.

Returns

Integer.

4.19.2.6 getPPD()

```
def getPPD (
              self )
```

Angular resolution at current gaze position in screen pixels per visual degree.

The first item of the tuple stores the x-coordinate resolution and the second item of the tuple stores the y-coordinate resolution.

Returns

Two-item tuple in the format of (float, float).

4.19.2.7 getStatus()

```
def getStatus (
              self )
```

Error and status flags (only useful for EyeLink II and newer hardware, report CR status and tracking error).

See eye_data.h for useful bits.

Returns

Integer.

4.19.2.8 getInput()

```
def getInput (
              self )
```

Data from input port(s).

Returns

Integer.

4.19.2.9 getFlags()

```
def getFlags (
              self )
```

Bits indicating what types of data are present, and for which eye(s).

See eye_data.h for useful bits.

Returns

Integer.

4.19.2.10 getButtons()

```
def getButtons (
              self )
```

Button input data: high 8 bits indicate changes from last sample, low 8 bits indicate current state of buttons 8 (MSB) to 1 (LSB).

Returns

Integer.

4.19.2.11 getRightEye()

```
def getRightEye (
              self )
```

Returns the sample data information from the desired eye.

Returns

Instance of sample data class.

4.19.2.12 getLeftEye()

```
def getLeftEye (
              self )
```

Returns the sample data information from the desired eye.

Returns

Instance of sample data class.

4.19.2.13 getEye()

```
def getEye (
              self )
```

Returns Eye data status.

Returns

- 2 if both left and right eye data are present
- 1 if right eye data present
- 0 if left eye data present

4.19.2.14 getTargetDistance()

```
def getTargetDistance (
              self )
```

Target Distance.

Returns

Float.

4.19.2.15 getTargetX()

```
def getTargetX (
              self )
```

Target X.

Returns

Float.

4.19.2.16 getTargetY()

```
def getTargetY (
              self )
```

Target Y.

Returns

Float.

4.19.2.17 getTargetFlags()

```
def getTargetFlags (
              self )
```

Target Flags.

Returns

Float.

4.20 SampleData Class Reference

Sample data for left and right eye.

Public Member Functions

- def getGaze (self)

  Display gaze position (in pixel coordinates set by the `screen_pixel_coords` command).
- def getHREF (self)

  HREF angular coordinates.
- def getRawPupil (self)

  Camera x, y of pupil center.
- def getPupilSize (self)

  Pupil size (in arbitrary units, area or diameter as selected).

4.20.1 Detailed Description

Sample data for left and right eye.

The `getRightEye()` or `getLeftEye()` functions returns an instance of SampleData class, which contains the current sample position (raw, HREF, or gaze) and pupil size information of the desired eye. The following methods can be used to retrieve the attributes of an instance of the SampleData class.

For example, the x gaze position of the left eye for a given sample can be retrieved as:

```
newSample = getEYELINK().getFloatData()
gaze = newSample.getLeftEye().getGaze()
left_eye_gaze_x = gaze[0]
```

If certain property information not sent for this sample, the value `MISSING_DATA` (or `0`, depending on the field) will be returned, and the corresponding bit in the flags field will be zero (see eye_data.h for a list of bits).
Data may be missing because of the tracker configuration (set by commands sent at the start of the experiment, from the Set Options screen of the EyeLink II tracker and newer eye tracker models, or from the default configuration set by the DATA.INI file for the EyeLink I tracker). Eye position data may also be set to `MISSING_VALUE` during a blink.

4.20.2 Member Function Documentation

4.20.2.1 getGaze()

```
def getGaze (
              self )
```

Display gaze position (in pixel coordinates set by the `screen_pixel_coords` command).

The first and second item of the tuple store the x- and y- coordinate gaze position respectively.

Returns

Two-item tuple in the format of (float, float).

4.20.2.2 getHREF()

```
def getHREF (
              self )
```

HREF angular coordinates.

The first and second items of the tuple are for the x and y coordinates, respectively.

Returns

Two-item tuple in the format of (float, float).

4.20.2.3 getRawPupil()

```
def getRawPupil (
              self )
```

Camera x, y of pupil center.

The first and second items of the tuple store pupil center in the x- and y- coordinate respectively.

Returns

Two-item tuple in the format of (float, float).

4.20.2.4 getPupilSize()

```
def getPupilSize (
              self )
```

Pupil size (in arbitrary units, area or diameter as selected).

Returns

Float.

4.21 StartBlinkEvent Class Reference

Class to represent Start Blink event.

Inherits EyeEvent.

Additional Inherited Members

4.21.1 Detailed Description

Class to represent Start Blink event.

There are no direct properties for this interface. All properties are inherited from EyeEvent.

4.22 StartFixationEvent Class Reference

Class to represent Start Fixation event.

Inherits StartNonBlinkEvent.

Inherited by EndFixationEvent.

Public Member Functions

- def getStartPupilSize (self)

  Pupil size (in arbitrary units, area or diameter as selected) at the start of a fixation interval.

4.22.1 Detailed Description

Class to represent Start Fixation event.

This also inherits all properties from StartNonBlinkEvent.

4.22.2 Member Function Documentation

4.22.2.1 getStartPupilSize()

```
def getStartPupilSize (
              self )
```

Pupil size (in arbitrary units, area or diameter as selected) at the start of a fixation interval.

Returns

Float.

4.23 StartNonBlinkEvent Class Reference

This interface is never used as is.

Inherits EyeEvent.

Inherited by FixUpdateEvent, StartFixationEvent, and StartSaccadeEvent.

Public Member Functions

- def getStartGaze (self)

  Gaze position at the start of the event (in pixel coordinates set by the `screen_pixel_coords` command).
- def getStartHREF (self)

  HEADREF position at the start of the event.
- def getStartVelocity (self)

  Gaze velocity at the start of the event (in visual degrees per second).
- def getStartPPD (self)

  Angular resolution at the start of the event (in screen pixels per visual degree, PPD).

4.23.1 Detailed Description

This interface is never used as is.

FixUpdateEvent, StartFixationEvent and StartSaccadeEvent types inherit this. This also inherits all properties from EyeEvent.

4.23.2 Member Function Documentation

4.23.2.1 getStartGaze()

```
def getStartGaze (
              self )
```

Gaze position at the start of the event (in pixel coordinates set by the `screen_pixel_coords` command).

The first and second items of the tuple store the x- and y- gaze position respectively.

Returns

Two-item tuple in the format of (float, float).

4.23.2.2 getStartHREF()

```
def getStartHREF (
              self )
```

HEADREF position at the start of the event.

The first and second items of the tuple store the x- and y- HREF data respectively.

Returns

Two-item tuple in the format of (float, float).

4.23.2.3 getStartVelocity()

```
def getStartVelocity (
              self )
```

Gaze velocity at the start of the event (in visual degrees per second).

Returns

Float.

4.23.2.4 getStartPPD()

```
def getStartPPD (
              self )
```

Angular resolution at the start of the event (in screen pixels per visual degree, PPD).

The first item of the tuple stores the x-coordinate PPD resolution and the second item of the tuple stores the y-coordinate PPD resolution.

Returns

Two-item tuple in the format of (float, float).

4.24 StartSaccadeEvent Class Reference

Class to represent Start Saccade event.

Inherits StartNonBlinkEvent.

Inherited by EndSaccadeEvent.

Additional Inherited Members

4.24.1 Detailed Description

Class to represent Start Saccade event.

There are no direct properties for this interface. All properties are inherited from StartNonBlinkEvent.

Chapter 5

Example implementation of EyeLinkCustomDisplay

```
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
# IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
# TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
# PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR
# CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
# EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
# PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES LOSS OF USE, DATA, OR
# PROFITS OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
# LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
# NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
# SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# Redistributions in binary form must reproduce the above copyright
# notice, this list of conditions and the following disclaimer in
# the documentation and/or other materials provided with the distribution.
#
# Neither name of SR Research Ltd nor the name of contributors may be used
# to endorse or promote products derived from this software without
# specific prior written permission.
#
# Last updated on 3/18/2021
import pygame
from pygame.locals import *
from math import pi
import array
import pylink
import platform
import sys
import os
#allow to disable sound, or if we failed to initialize pygame.mixer or failed to load audio file
#continue experiment without sound.
DISABLE_AUDIO=False
class PygameEyeLinkCustomDisplay(pylink.EyeLinkCustomDisplay):
    def __init__(self, tracker, win):
        global DISABLE_AUDIO
        pylink.EyeLinkCustomDisplay.__init__(self)
        self._disp = win  # screen to use for calibration
        self._tracker = tracker  # connection to the tracker
        self._version = '2021.3.16'
        self._last_updated = '3/16/2021'
        pygame.mouse.set_visible(False)  # hide mouse cursor
        self._bgColor = (128, 128, 128)  # target color (foreground)
        self._fgColor = (0, 0, 0)  # target color (background)
        self._targetSize = 32  # diameter of the target
        self._targetType = 'circle'  # could be 'circle' or 'picture'
        self._pictureTarget = None  # picture target
        self._target_beep = None
        self._done_beep = None
        self._error_beep = None
        if not DISABLE_AUDIO:
            try:
                self._target_beep = pygame.mixer.Sound("type.wav")
                self._done_beep = pygame.mixer.Sound("qbeep.wav")
                self._error_beep = pygame.mixer.Sound("error.wav")
            except Exception as e:
                print ('Failed to load audio: '+ str(e))
                #we failed to load audio, so disable it
                #if the experiment is run with sudo/root user in Ubuntu, then audio will
                #fail. The work around is either allow audio playback permission
                #for root user or,  run the experiment with non root user.
                DISABLE_AUDIO=True
        self._size = (384, 320)  # size of the camera image
```

```
        self._imagebuffer = array.array('I')  # buffer to store camera image
        self._resizedImg = None
        self.surf = pygame.display.get_surface()
        # image palette; its indices are used to reconstruct the camera image
        self._pal = []
        # we will use this for text messages
        self._fnt = pygame.font.SysFont('Arial', 26)
        self._w, self._h = self._disp.get_size()
        self._cam_region = pygame.Rect((0, 0), (0, 0))
        # cache the camera title
        self._title = ''
        # keep track of mouse states
        self.mouse_pos = (self._w/2, self._h/2)
        self.last_mouse_state = -1
    def __str__(self):
        """ overwrite __str__ to show some information about the
        CoreGraphicsPsychoPy library
        """
        return "Using the CalibrationGraphicsPygame library, " + \
            "version %s, " % self._version + \
            "last updated on %s" % self._last_updated
    def getForegroundColor(self):
        """ get the foreground color """
        return self._fgColor
    def getBackgroundColor(self):
        """ get the foreground color """
        return self._bgColor
    def setCalibrationColors(self, foreground_color, background_color):
        """ Set calibration background and foreground colors
        Parameters:
            foreground_color--foreground color for the calibration target
            background_color--calibration background.
            """
        self._fgColor = foreground_color
        self._bgColor = background_color
    def setTargetType(self, type):
        """ Set calibration target size in pixels
        Parameters:
            type: "circle" (default) or "picture"
        """
        self._targetType = type
    def setTargetSize(self, size):
        """ Set calibration target size in pixels"""
        self._targetSize = size
    def setPictureTarget(self, picture_target):
        """ set the movie file to use as the calibration target """
        self._pictureTarget = picture_target
    def setCalibrationSounds(self, target_beep, done_beep, error_beep):
        """ Provide three wav files as the warning beeps
        Parameters:
            target_beep -- sound to play when the target comes up
            done_beep -- calibration is done successfully
            error_beep -- calibration/drift-correction error.
        """
        # target beep
        if target_beep == '':
            self._target_beep = pygame.mixer.Sound("type.wav")
        elif target_beep == 'off':
            self._target_beep = None
        else:
            self._target_beep = pygame.mixer.Sound(target_beep)
        # done beep
        if done_beep == '':
            self._done_beep = pygame.mixer.Sound("qbeep.wav")
        elif done_beep == 'off':
            self._done_beep = None
        else:
            self._done_beep = pygame.mixer.Sound(done_beep)
        # error beep
        if error_beep == '':
            self._error_beep = pygame.mixer.Sound("error.wav")
        elif error_beep == 'off':
            self._error_beep = None
        else:
            self._error_beep = pygame.mixer.Sound(error_beep)
    def setup_cal_display(self):
        """ setup calibration/validation display"""
        self.clear_cal_display()
    def exit_cal_display(self):
        """ exit calibration/validation display"""
        self.clear_cal_display()
    def record_abort_hide(self):
        pass
    def clear_cal_display(self):
        self._disp.fill(self._bgColor)
        pygame.display.flip()
        self._disp.fill(self._bgColor)
```

```
    def erase_cal_target(self):
        self.clear_cal_display()
    def draw_cal_target(self, x, y):
        """  draw the calibration target, i.e., a bull's eye"""
        if self._targetType == 'picture':
            if self._pictureTarget is None:
                print('ERROR: Provide a picture as the calibration target')
                pygame.quit()
                sys.exit()
            elif not os.path.exists(self._pictureTarget):
                print('ERROR: Picture %s not found' % self._pictureTarget)
                pygame.quit()
                sys.exit()
            else:
                cal_pic = pygame.image.load(self._pictureTarget)
                w, h = cal_pic.get_size()
                self._disp.blit(cal_pic, (x - int(w/2.0), y - int(h/2.0)))
        else:
            pygame.draw.circle(self._disp, self._fgColor, (x, y),
                               int(self._targetSize / 2.))
            pygame.draw.circle(self._disp, self._bgColor, (x, y),
                               int(self._targetSize / 4.))
        pygame.display.flip()
    def play_beep(self, beepid):
        """ play warning beeps if being requested"""
        global DISABLE_AUDIO
        # if sound is disabled, don't play
        if DISABLE_AUDIO:
            pass
        else:
            if beepid in [pylink.DC_TARG_BEEP, pylink.CAL_TARG_BEEP]:
                if self._target_beep is not None:
                    self._target_beep.play()
                    pygame.time.wait(50)
            if beepid in [pylink.CAL_ERR_BEEP, pylink.DC_ERR_BEEP]:
                if self._error_beep is not None:
                    self._error_beep.play()
                    pygame.time.wait(300)
            if beepid in [pylink.CAL_GOOD_BEEP, pylink.DC_GOOD_BEEP]:
                if self._done_beep is not None:
                    self._done_beep.play()
                    pygame.time.wait(100)
    def getColorFromIndex(self, colorindex):
        """  color scheme for different elements """
        if colorindex == pylink.CR_HAIR_COLOR:
            return (255, 255, 255, 255)
        elif colorindex == pylink.PUPIL_HAIR_COLOR:
            return (255, 255, 255, 255)
        elif colorindex == pylink.PUPIL_BOX_COLOR:
            return (0, 255, 0, 255)
        elif colorindex == pylink.SEARCH_LIMIT_BOX_COLOR:
            return (255, 0, 0, 255)
        elif colorindex == pylink.MOUSE_CURSOR_COLOR:
            return (255, 0, 0, 255)
        else:
            return (0, 0, 0, 0)
    def draw_line(self, x1, y1, x2, y2, colorindex):
        """  draw lines"""
        color = self.getColorFromIndex(colorindex)
        # get the camera image rect, then scale
        if self._size[0] > 192:
            imr = self._img.get_rect()
            x1 = int((float(x1) / 192) * imr.w)
            x2 = int((float(x2) / 192) * imr.w)
            y1 = int((float(y1) / 160) * imr.h)
            y2 = int((float(y2) / 160) * imr.h)
        # draw the line
        if True not in [x < 0 for x in [x1, x2, y1, y2]]:
            pygame.draw.line(self._img, color, (x1, y1), (x2, y2))
    def draw_lozenge(self, x, y, width, height, colorindex):
        """  draw the search limits with two lines and two arcs"""
        color = self.getColorFromIndex(colorindex)
        if self._size[0] > 192:
            imr = self._img.get_rect()
            x = int((float(x) / 192) * imr.w)
            y = int((float(y) / 160) * imr.h)
            width = int((float(width) / 192) * imr.w)
            height = int((float(height) / 160) * imr.h)
        if width > height:
            rad = int(height / 2.)
            if rad == 0:
                return
            else:
                pygame.draw.line(self._img,
                                 color,
                                 (x + rad, y),
                                 (x + width - rad, y))
```

```
            pygame.draw.line(self._img,
                             color,
                             (x + rad, y + height),
                             (x + width - rad, y + height))
            pygame.draw.arc(self._img,
                            color,
                            [x, y, rad*2, rad*2],
                            pi/2, pi*3/2, 1)
            pygame.draw.arc(self._img,
                            color,
                            [x+width-rad*2, y, rad*2, height],
                            pi*3/2, pi/2 + 2*pi, 1)
        else:
            rad = int(width / 2.)
            if rad == 0:
                return
            else:
                pygame.draw.line(self._img,
                                 color,
                                 (x, y + rad),
                                 (x, y + height - rad))
                pygame.draw.line(self._img,
                                 color,
                                 (x + width, y + rad),
                                 (x + width, y + height - rad))
                pygame.draw.arc(self._img,
                                color,
                                [x, y, rad*2, rad*2],
                                0, pi, 1)
                pygame.draw.arc(self._img,
                                color,
                                [x, y+height-rad*2, rad*2, rad*2],
                                pi, 2*pi, 1)
    def get_mouse_state(self):
        """  get mouse position and states"""
        x, y = pygame.mouse.get_pos()
        state = pygame.mouse.get_pressed()
        x = x * self._size[0]/self._w/2.0
        y = y * self._size[1]/self._h/2.0
        return ((x, y), state[0])
    def get_input_key(self):
        """  handle key input and send it over to the tracker"""
        ky = []
        for ev in pygame.event.get():
            # check keyboard events
            if ev.type == KEYDOWN:
                keycode = ev.key
                if keycode == K_F1:
                    keycode = pylink.F1_KEY
                elif keycode == K_F2:
                    keycode = pylink.F2_KEY
                elif keycode == K_F3:
                    keycode = pylink.F3_KEY
                elif keycode == K_F4:
                    keycode = pylink.F4_KEY
                elif keycode == K_F5:
                    keycode = pylink.F5_KEY
                elif keycode == K_F6:
                    keycode = pylink.F6_KEY
                elif keycode == K_F7:
                    keycode = pylink.F7_KEY
                elif keycode == K_F8:
                    keycode = pylink.F8_KEY
                elif keycode == K_F9:
                    keycode = pylink.F9_KEY
                elif keycode == K_F10:
                    keycode = pylink.F10_KEY
                elif keycode == K_PAGEUP:
                    keycode = pylink.PAGE_UP
                elif keycode == K_PAGEDOWN:
                    keycode = pylink.PAGE_DOWN
                elif keycode == K_UP:
                    keycode = pylink.CURS_UP
                elif keycode == K_DOWN:
                    keycode = pylink.CURS_DOWN
                elif keycode == K_LEFT:
                    keycode = pylink.CURS_LEFT
                elif keycode == K_RIGHT:
                    keycode = pylink.CURS_RIGHT
                elif keycode == K_BACKSPACE:
                    keycode = ord('\b')
                elif keycode == K_RETURN:
                    keycode = pylink.ENTER_KEY
                    # probe the tracker to see if it's "simulating gaze
                    # with mouse". if so, show a warning instead of a blank
                    # screen to experimenter do so, only when the tracker
                    # is in Camera Setup screen
```

```
                if self._tracker.getCurrentMode() == pylink.IN_SETUP_MODE:
                    self._tracker.readRequest('aux_mouse_simulation')
                    pylink.pumpDelay(50)
                    if self._tracker.readReply() == '1':
                        # draw a rectangle to mark the camera image
                        rec_x = int((self._w - 192*2) / 2.0)
                        rec_y = int((self._h - 160*2) / 2.0)
                        rct = pygame.Rect((rec_x, rec_y, 192*2, 160*2))
                        pygame.draw.rect(self._disp, self._fgColor, rct, 2)
                        # show some message
                        msg = 'Simulating gaze with the mouse'
                        msg_w, msg_h = self._fnt.size(msg)
                        t_surf = self._fnt.render(msg, True, self._fgColor)
                        txt_x = int((self._w - msg_w)/2.0)
                        txt_y = int((self._h - msg_h)/2.0)
                        self._disp.blit(t_surf, (txt_x, txt_y))
                        pygame.display.flip()
            elif keycode == K_SPACE:
                keycode = ord(' ')
            elif keycode == K_ESCAPE:
                keycode = pylink.ESC_KEY
            elif keycode == K_TAB:
                keycode = ord('\t')
            elif(keycode == pylink.JUNK_KEY):
                keycode = 0
            ky.append(pylink.KeyInput(keycode, ev.mod))
    return ky
def exit_image_display(self):
    """  exit the camera image display"""
    self.clear_cal_display()
def alert_printf(self, msg):
    print(msg)
def setup_image_display(self, width, height):
    """  set up the camera image display
    return 1 to request high-resolution camera image"""
    self._size = (width, height)
    self.clear_cal_display()
    self.last_mouse_state = -1
    return 1
def image_title(self, text):
    """  show the camera image title
    target distance, and pupil/CR thresholds below the image. To prevent
    drawing glitches, we cache the image title and draw it with the camera
    image in the draw_image_line function instead"""
    self._title = text
def draw_image_line(self, width, line, totlines, buff):
    """  draw the camera image"""
    for i in range(width):
        try:
            self._imagebuffer.append(self._pal[buff[i]])
        except:
            pass
    if line == totlines:
        try:
            # construct the camera image from the buffer
            try:
                tmp_buffer = self._imagebuffer.tobytes()
            except:
                tmp_buffer = self._imagebuffer.tostring()
            cam = pygame.image.frombuffer(tmp_buffer,
                                          (width, totlines), 'RGBX')
            self._img = cam
            self.draw_cross_hair()
            # prepare the camera image
            img_w, img_h = (width*2, totlines*2)
            self._resizedImg = pygame.transform.scale(cam, (img_w, img_h))
            cam_img_pos = ((self._w/2-img_w/2),
                           (self._h/2-img_h/2))
            # prepare the camera image caption
            txt_w, txt_h = self._fnt.size(self._title)
            txt_surf = self._fnt.render(self._title, True, self._fgColor)
            txt_pos = (int(self._w/2 - txt_w/2),
                       int(self._h/2 + img_h/2 + txt_h/2))
            # draw the camera image and the caption
            surf = pygame.display.get_surface()
            surf.fill(self._bgColor)
            surf.blit(self._resizedImg, cam_img_pos)
            surf.blit(txt_surf, txt_pos)
            pygame.display.flip()
        except:
            pass
        self._imagebuffer = array.array('I')
def set_image_palette(self, r, g, b):
    """  get the color palette for the camera image"""
    self._imagebuffer = array.array('I')
    sz = len(r)
    i = 0
```

```
        self._pal = []
        while i < sz:
            rf = int(b[i])
            gf = int(g[i])
            bf = int(r[i])
            self._pal.append((rf « 16) | (gf « 8) | (bf))
            i = i + 1
# A short testing script showing the basic usage of this library
# We first instantiate a connection to the tracker (el_tracker), then we open
# a Pygame window (win). We then pass the tracker connection and the Pygame
# window to the graphics environment constructor (CalibrationGraphics).
# The graphics environment, once instantiated, can be configured to customize
# the calibration foreground and background color, the calibration target
# type, the calibration target size, and the beeps we would like to
# play during calibration and validation.
#
# IMPORTANT: Once the graphics environment is properly configured, call the
# pylink.openGraphicsEx() function to request PyLink to use the custom graphics
# environment for calibration instead.
def main():
    """ A short script showing how to use this library.
    We connect to the tracker, open a Pygame window, and then configure the
    graphics environment for calibration. Then, perform a calibration and
    disconnect from the tracker.
    The doTrackerSetup() command will bring up a gray calibration screen.
    When the gray screen comes up, press Enter to show the camera image,
    press C to calibrate, V to validate, and O to quit calibration"""
    # initialize Pygame
    pygame.init()
    # get the screen resolution natively supported by the monitor
    disp = pylink.getDisplayInformation()
    scn_w = disp.width
    scn_h = disp.height
    # connect to the tracker
    el_tracker = pylink.EyeLink("100.1.1.1")
    # open an EDF data file on the Host PC
    el_tracker.openDataFile('test.edf')
    # open a Pygame window
    win = pygame.display.set_mode((scn_w, scn_h), FULLSCREEN | DOUBLEBUF)
    # send over a command to let the tracker know the correct screen resolution
    scn_coords = "screen_pixel_coords = 0 0 %d %d" % (scn_w - 1, scn_h - 1)
    el_tracker.sendCommand(scn_coords)
    # Instantiate a graphics environment (genv) for calibration
    genv = CalibrationGraphics(el_tracker, win)
    # Set background and foreground colors for calibration
    foreground_color = (0, 0, 0)
    background_color = (128, 128, 128)
    genv.setCalibrationColors(foreground_color, background_color)
    # The calibration target could be a "circle" (default) or a "picture",
    genv.setTargetType('circle')
    # Configure the size of the calibration target (in pixels)
    genv.setTargetSize(24)
    # Beeps to play during calibration, validation, and drift correction
    # parameters: target, good, error
    # Each parameter could be ''--default sound, 'off'--no sound, or a wav file
    genv.setCalibrationSounds('', '', '')
    # Request PyLink to use the graphics environment (genv) we customized above
    pylink.openGraphicsEx(genv)
    # calibrate the tracker
    el_tracker.doTrackerSetup()
    # close the data file
    el_tracker.closeDataFile()
    # disconnect from the tracker
    el_tracker.close()
    # quit pygame
    pygame.quit()
    sys.exit()
if __name__ == '__main__':
    main()
```

Index