pylinkwrapper Documentation

Release 1.0

Nick DiQuattro

March 16, 2016

Contents

Contents:

CHAPTER 1

Installation

1. Download and place the pylinkwrapper folder somewhere you can easily reach in the file system.
2. Add the folder to Psychopy's python path as detailed here: http://www.psychopy.org/recipes/addCustomModules.html
3. Update pylink to the latest version provided by the SR Research development kit. It should be installed on the display computer.

That's it! Here's a quick demo of getting connected to the Eyelink:

```
import pylinkwrapper

win = visual.window(monitor='nickMon', fullScr=True, allowGUI=False, color=-1)
tracker = pylinkwrapper.connect(win, '1_nd')
```

Check out the documentation for the Connector Class class to see what functions are available.

CHAPTER 2

Connector Class

This class provides methods for interacting with the Eyelink from psychopy.

class connector.**Connect**(*window*, *edfname*)
Provides functions for interacting with the EyeLink via Pylink.

Parameters

- **window** – Psychopy window object.
- **edfname** (*str*) – Desired name of the EDF file.

calibrate(*cnum=13*, *paval=1000*)
Calibrates eye-tracker using psychopy stimuli.

Parameters

- **cnum** (*int*) – Number of points to use for calibration. Options are 3, 5, 9, 13.
- **paval** (*int*) – Pacing of calibration, i.e. how long you have to fixate each target in milliseconds.

convert_coords(*x*, *y*, *to='eyelink'*)
Converts from degrees visual angle units to EyeLink Pixel units.

Parameters

- **x** (*float or int*) – X coordinate in visual angle.
- **y** (*float or int*) – Y coordinate in viusal angle.
- **to** – Direction of conversion. Options: 'eyelink' or 'psychopy'.

Returns Two values in order x, y

draw_ia(*x*, *y*, *size*, *index*, *color*, *name*)
Draws square interest area in EDF and a corresponding filled box on eye-tracker display. Must be called after *set_trialid()* for interest areas to appear in the EDF.

Parameters

- **x** (*float or int*) – X coordinate in degrees visual angle for center of check area.
- **y** (*float or int*) – Y coordinate in degrees visual angle for center of check area.
- **size** (*float or int*) – length of one edge of square in degrees visual angle.
- **index** (*int*) – number to assign interest area in EDF
- **color** (*int*) – color of box drawn on eye-tracker display (0 - 15)

- **name** (*str*) – Name of interest area in EDF

draw_text(*msg*)
Draws text on eye-tracker screen.

Parameters **msg** (*str*) – Text to draw.

end_experiment(*spath*)
Closes and transfers the EDF file.

Parameters **spath** (*str*) – Absolute file path of where to save the EDF file.

fix_check(*size*, *ftime*, *button*)
Checks that fixation is maintained for certain time.

Parameters

- **size** (*float or int*) – Length of one side of box in degrees visual angle.
- **ftime** (*float*) – Length of time to check for fixation in seconds.
- **button** (*char*) – Key to press to recalibrate eye-tracker.

get_gaze()
Gets current gaze position of eye. Must be called between *record_on()* and *record_off()*. Sendlink must be set to True as well.

Returns list of coordinates in the form of [x, y].

record_off()
Stops recording.

record_on(*sendlink=False*)
Starts recording. Waits 50ms to allow eyelink to prepare.

Parameters **sendlink** (*bool*) – Toggle for sending eye data over the link to the display computer during recording.

sac_detect(*x*, *y*, *radius*)
Checks if current gaze position is outside a circular interest area.

Parameters

- **x** (*float or int*) – X coordinate in degrees visual angle for center of circle IA.
- **y** (*float or int*) – Y coordinate in degrees visual angle for center of circle IA.
- **radius** (*float or int*) – Radius of detection circle in degrees visual angle.

Returns List of whether saccade was detected and the gaze coordinates

at time of detection. In the form of [bool, [x, y]]. :rtype: bool, list

send_command(*cmd*)
Sends a command to the Eyelink.

Parameters **cmd** (*str*) – Command to send.

send_message(*txt*)
Sends a message to the tracker that is recorded in the EDF.

Parameters **txt** (*str*) – Message to send.

send_var(*name*, *value*)
Sends a trial variable to the EDF file.

Parameters

- **name** (*str*) – Name of variable.
- **value** (*float, str, or int*) – Value of variable.

set_status(*message*)
Sets status message to appear while recording.

Parameters **message** (*str*) – Text object to send, must be < 80 char

set_trialid(*idval=1*)
Sends message that indicates start of trial in EDF.

Parameters **idval** – Values to set TRIALID.

set_trialresult(*rval=0*, *scrcol=0*)
Sends trial result to indiciate trial end in EDF and clears screen on EyeLink Display.

Parameters

- **rval** (*float, str, or int*) – Value to set for TRIAL_RESULT.
- **scrcol** (*int*) – Color to clear screen to. Defaults to black.

CHAPTER 3

Sample Experiment

This is a very simple experiment that demonstrates use of pylinkwrapper. It can be found in the `sample` folder.

```
"""
Pylink Wrapper test experiment
N. DiQuattro - January 2015

This is a simple experiment where a circle appears randomly on the screen. It's
purpose is to provide examples of how to use the pylink wrapper with a psychopy
experiment.

There's help documentaiton available for each of the functions that show the
available parameters.
"""

# Import modules
from psychopy import visual
from psychopy import core, event
import numpy as np

import pylinkwrapper  # Here's the special one

# Window set-up
win = visual.Window(monitor='nickMon', units='deg', fullscr=True,
                    allowGUI=False, color=0)

# Initiate eye-tracker link and open EDF
tracker = pylinkwrapper.Connect(win, '1_test')

# Calibrate eye-tracker
tracker.calibrate()

# Stimulus
fix = visual.Circle(win, radius=1, pos=(0, 0), fillColor=[1, 0, 0],
                    lineColor=[1, 0, 0])

cfix = visual.Circle(win, radius=.15, fillColor=-1, lineColor=-1)

# Display stimulus 5 times
for t in range(5):
    # Find random coordinates and set them
    fx = np.random.randint(-10, 10)
    fy = np.random.randint(-10, 10)
    fix.setPos((fx, fy))
```

```
    # Eye tracker trial set-up
    stxt = 'Trial %d' % t
    tracker.set_status(stxt)  # Define status message that appears on eye-link
    #  display

    tracker.set_trialid()  # Sends trial start message for EDF
    tracker.send_message('Circle Trial')

    # Draw IA
    tracker.draw_ia(fx, fy, 1, 1, 5, 'circle')  # Draw interest area and box

    # Start recording
    tracker.record_on()

    # Draw and display circle
    cfix.draw()
    fix.draw()
    win.flip()

    # Wait for response
    keyp = event.waitKeys()

    # Stop Recording
    tracker.record_off()

    # Send response key to EDF file
    tracker.send_var('response', keyp[0][0])

    # End trial for EDF
    tracker.set_trialresult()

    # ISI with fixation check
    cfix.draw()
    win.flip()
    tracker.fixcheck(2, 1, 'z')

# Retrieve EDF
tracker.end_experiment('C:\\edfs\\')  # Closes and retrieves EDF file to
# specified path
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

C

C

D

E

F

G

R

S