Using Pylink from PsychoPy Builder

EyeLink®
Fast, Accurate, Reliable Eye Tracking

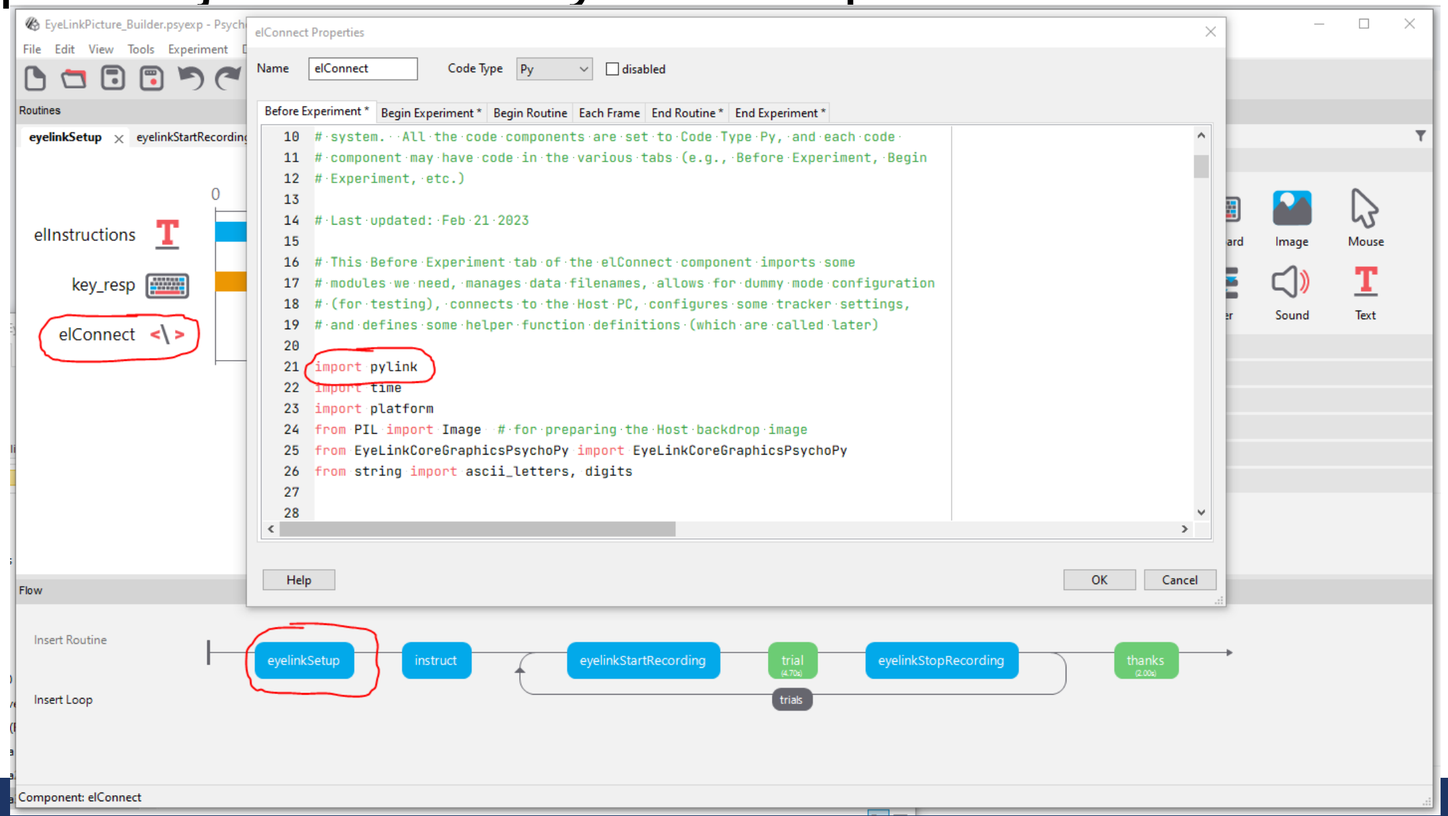
SR Research
EyeLink®

Setup and Overview

- PsychoPy has two main modes – Builder and Coder. Builder allows you to construct experiments in a GUI-based environment and then run from there or “compile” to Python code that can be run from Coder (or other Python environment)
- PsychoPy Builder has new Eyetracker interface, but it uses ioHub (their input/output/external device module). ioHub wraps Pylink, but uses its own core graphics for camera setup/calibration.
  - ioHub has lots of limitations with respect to EyeLink systems (no EDF logged by default, drift check doesn’t work, gaze data grabbing limited, no Data Viewer messages in examples, camera transfer/calibration not as smooth as with official EyeLink/Psychopy core graphics, etc.)
- These new Builder examples illustrate how to interface with EyeLink system from Builder using Pylink directly – so, none of those limitations apply

Setup and Overview

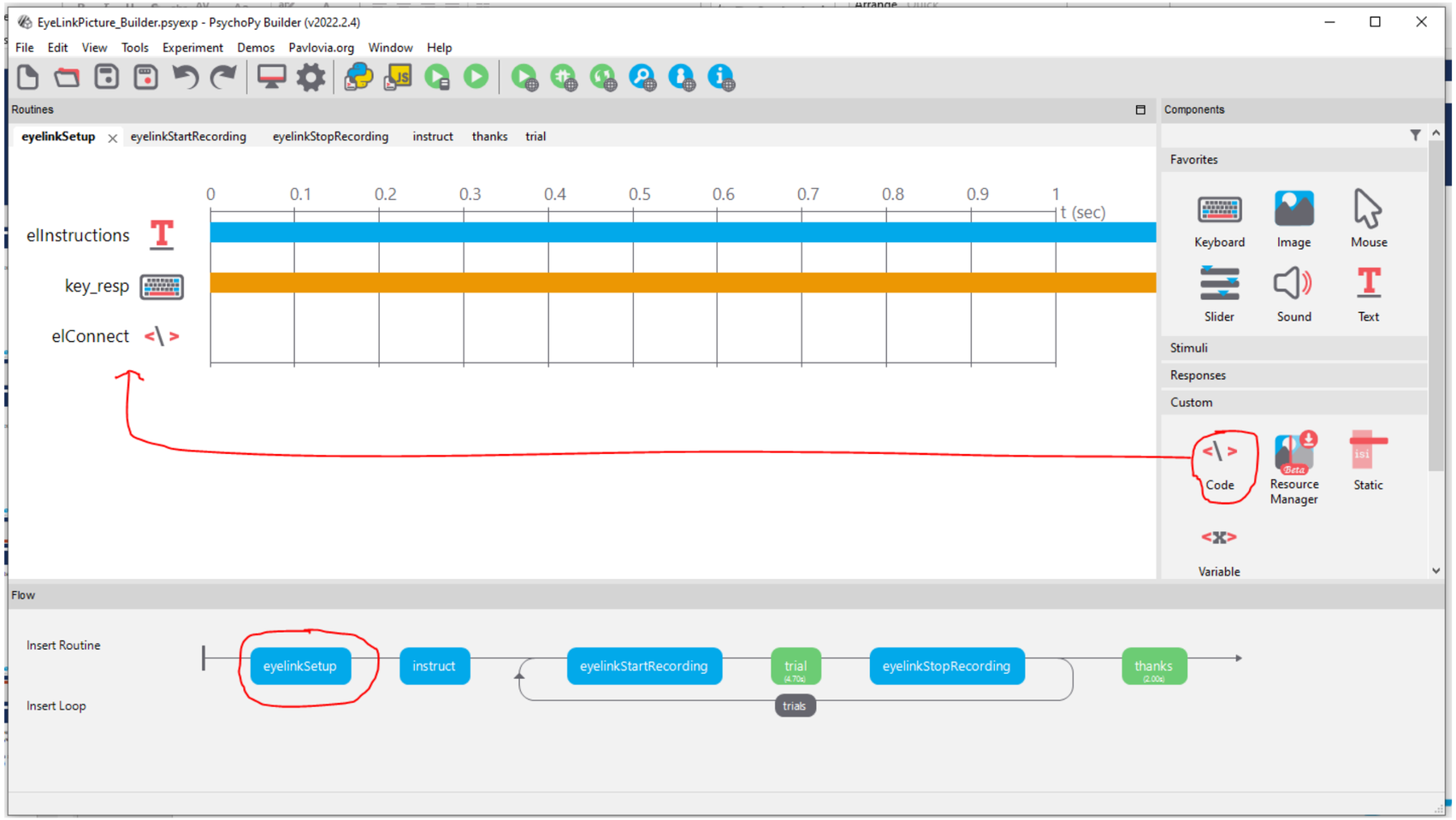
- General EyeLink comminication for Builder is the same as for the Coder (see Getting Started with PsychoPy / EyeLink Integration)
- Can think of Builder as mechanism for automatically generating Python code
- Pylink is imported just as in a Python script



EyeLink®
Fast, Accurate, Reliable Eye Tracking

Setup and Overview

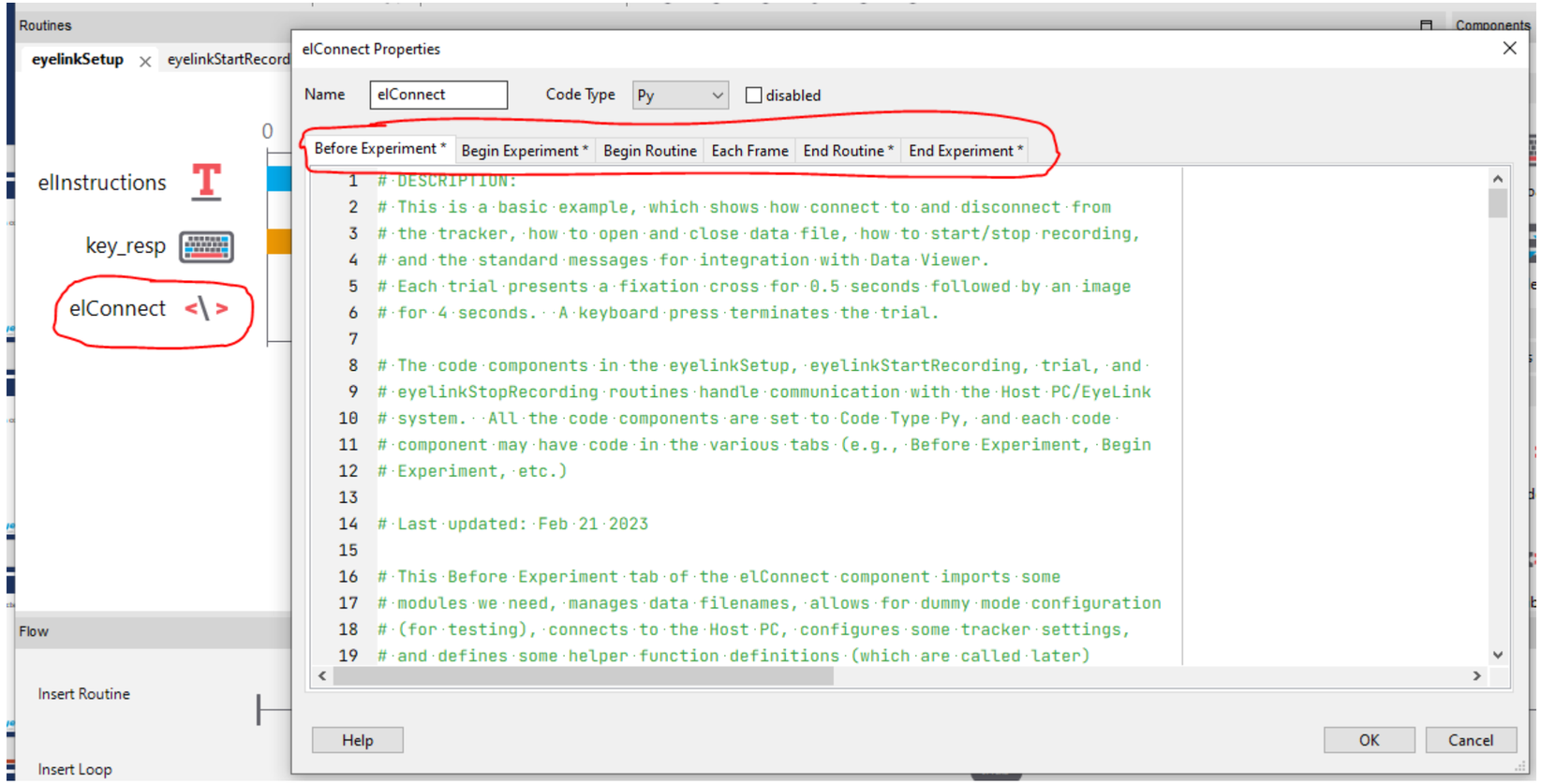
- In Builder, Routines can include Code objects



EyeLink®
Fast, Accurate, Reliable Eye Tracking

Setup and Overview

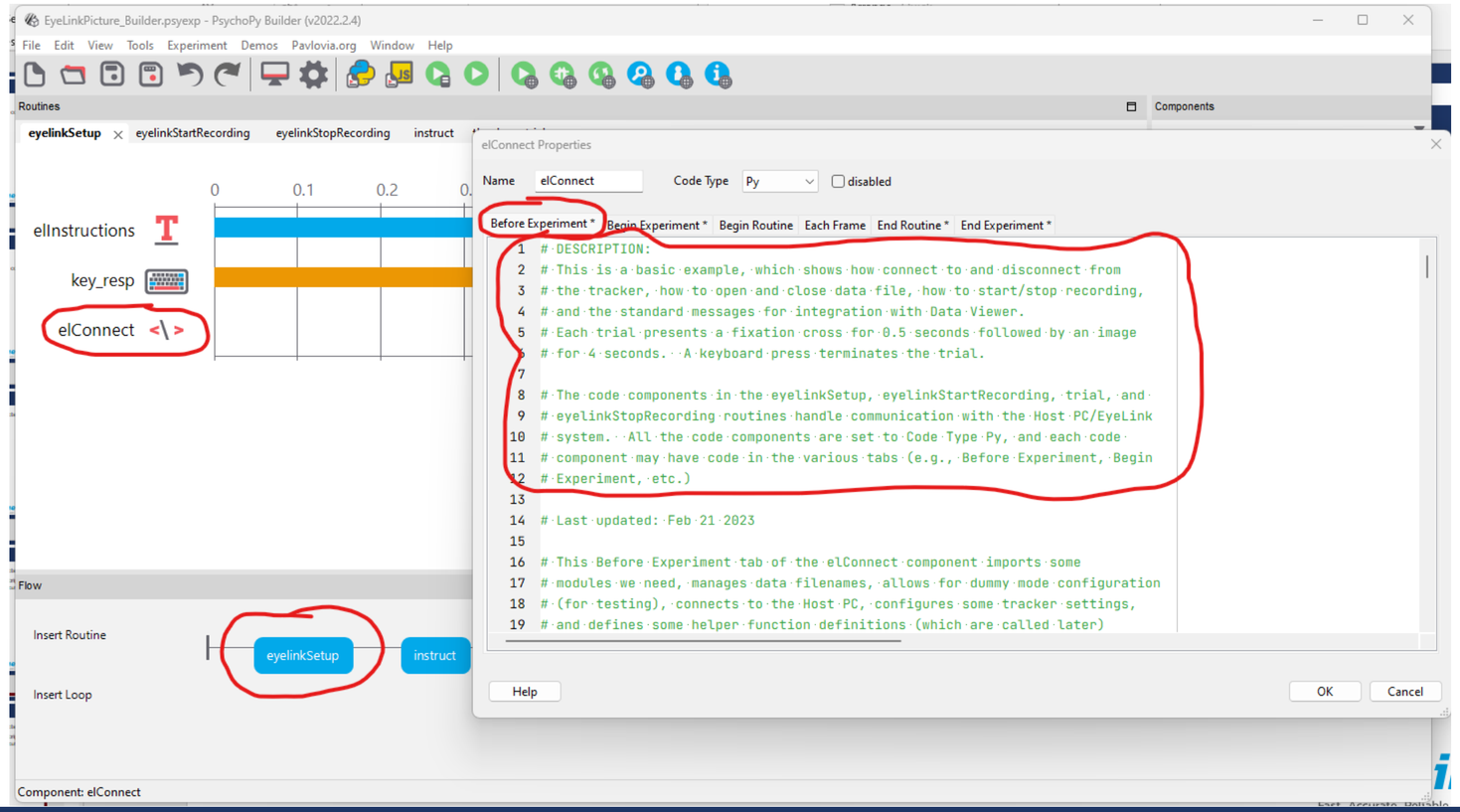
- In Builder, Routines can include Code objects
- Clicking on a Code object will bring up an interface to add the Python script
  - For a given code object, script can be added at several points in the routine's execution



EyeLink®
Fast, Accurate, Reliable Eye Tracking

Setup and Overview

- Comments in the Before Experiment tab of the elConnect object in the eyelinkSetup Routine describe each example and mention code components and tabs that contain custom script
  - Examples are similar to the Coder examples

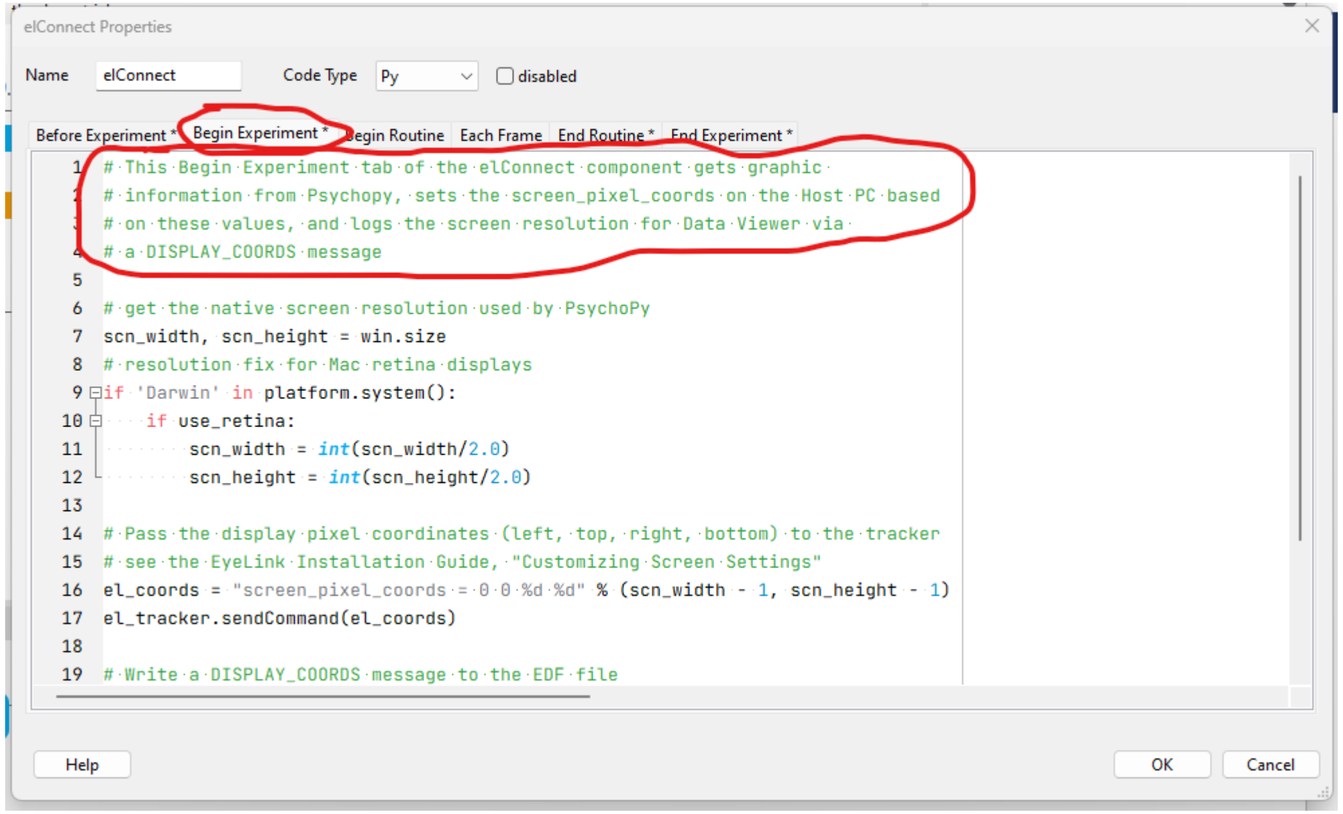


- EyeLinkFixationWindowFastSamples_Builder
- EyeLinkGCWindow_Builder
- EyeLinkMRIdemo_Builder
- EyeLinkPicture_Builder
- EyeLinkPursuit_Builder
- EyeLinkSaccade_Builder
- EyeLinkVideo_Builder

EyeLink®
Fast, Accurate, Reliable Eye Tracking

Setup and Overview

- Comments at the top of each tab provide overview of functionality of that tab
  - Comments before commands describe them in more detail
  - Code was largely taken from coder examples, put in right place, and tweaked



EyeLink®
Fast, Accurate, Reliable Eye Tracking

elConnect

elConnect component in eyelinkSetup Routine at beginning of experiment:

- Begin Experiment tab – Gets EDF name, connects to tracker, sets tracker setting, defines helper functions (e.g., abort_trial, terminate_task, clear_screen)
- Before Experiment tab – gets graphics info from PsychoPy, sets screen_pixel_cords, sends DISPLAY_COORDS message
- End Routine tab – Configures calibration options (colors, target type/file) and calls Camera Setup
- End Experiment tab – Calls terminate_task to retrieve EDF and close connection to Host PC

elStartRecord

elStartRecord component in eyelinkStartRecoring Routine at beginning of trial:

- Begin Routine tab is only one used – calls offline mode, does Host PC backdrop drawing, sends trial start messages, does drift check, offline mode again, then start recording

elTrial

elTrial component in trial Routine has code in several tabs

- Begin Experiment – initializes trial counter variable and defines a method that will be called upon each screen retrace – this method serves to mark experimental events and log useful information for data analysis. Also defines some helper functions to convert from PsychoPy position/size units to EyeLink position/size units
- Begin Routine – resets some variables that log whether we have marked certain events for the trial yet and sends condition TRIAL_VAR messages
- Each Frame – calls the method in the Begin Experiment tab that handles event marking/info logging. Also checks for special key presses (Ctrl-C) to allow proper early termination
- End Routine – checks whether responses were detected and, if so, mark them, and send TRIAL_VAR messages for them. Also send end of trial messages

elStartRecord

elStopRecord component in eyelinkStopRecoring Routine at end of trial:

- End Routine tab is only one used – clears screen (and marks event), stops recording, and sends TRIAL_RESULT message

Grabbing Gaze Data

- Can use Each Frame tab, which gets executed immediately after each screen flip, to handle some of the looping – it is built-in loop
- For buffered data, add while loop to Each Frame tab and then set temporal cutoff so that it doesn’t loop too long and block project

Marking Video Frames

- Depending on type of movie playback backend, PsychoPy may or may not provide current frame numbers.
- If frame numbers are provided, script will use those to determine VFRAME messages
- If frame numbers not provided, script will use time of current frame to determine VFRAME

Considerations

- PsychoPy (Coder and Builder) can miss sending messages if too many are sent in a row. Make sure to put 1 msec pause after every 3-5 messages
- TRIAL_VAR messages are sent at a few points in the trial. Condition type variables are sent near the beginning of the trial and response variables are sent at the end. This is to avoid having to send too many messages at once and potentially interrupting PsychoPy timing
- The examples show how to include temporal offsets in the event marking messages, which allow the time of the event to be precisely determined (these are taken into account automatically by Data Viewer) if the message cannot be sent at the time of the event. In PsychoPy, it seems like that may not usually be necessary. The Each Frame tab of code components appears to be executed immediately after screen drawing – i.e., in our tests, the automatically calculated offset values tended to work out to 0. We left this in in case there are settings/unknowns that may lead to the offsets being non zero in some cases.

EyeLink functions that examples illustrate

- EyeLinkPicture_Builder – simple example showing tracker connection/settings editing, image transfer to Host, drift check, basic event marking, file transfer, DV integration
- EyeLinkPursuit_Builder – smooth pursuit example illustrating dynamic IAs (to .ias file), target position messages
- EyeLinkSaccade_Builder – shows how to grab event/buffered data and use it in experiment
- EyeLinkFixationWindowFastSamples_Builder – shows how to grab latest sample and use it in fix window checking
- EyeLinkGCWindow_Builder – shows how to grab latest sample and use it to update window
- EyeLinkMRIdemo_Builder – shows how to sync with MRI, log sync times, and do continuous eye tracker recording through block of trials
- EyeLinkVideo_Builder – shows how to present videos and send VFRAME messages so videos can be visualized in DV