

EyeLink Plugin for PsychoPy

User Manual

SR Research Ltd., Ottawa, Canada

Last updated: June 8, 2026

1 Introduction and Installation	1
1.1 Introduction	1
1.2 Install the EyeLink Plugin for PsychoPy	3
1.3 Configure the IP address of the Display PC	8
1.4 Deploy the example sets and documentation	8
2 Plugin Components	9
2.1 Initialize Connection	10
2.2 Camera Setup	12
2.3 Drift Check	15
2.4 Host Drawing	17
2.5 Start Recording	21
2.6 Mark Events	23
2.7 Stop Recording	28
2.8 Gaze Trigger	28
2.9 Latest Sample	30
2.10 Buffered Data	31
3 Example walkthrough	34
3.1 Experiment configuration	35
3.2 Initialize EyeLink connection	36
3.3 Perform camera setup / eye tracker calibration	38
3.4 trials loop organization	39
3.5 Draw trial stimuli on the Host PC backdrop	42
3.6 Perform a drift check	45
3.7 Start eye tracker recording	46
3.8 Mark and log experimental events	47
3.9 Stop eye tracker recording	50
3.10 Online eye data access options	51

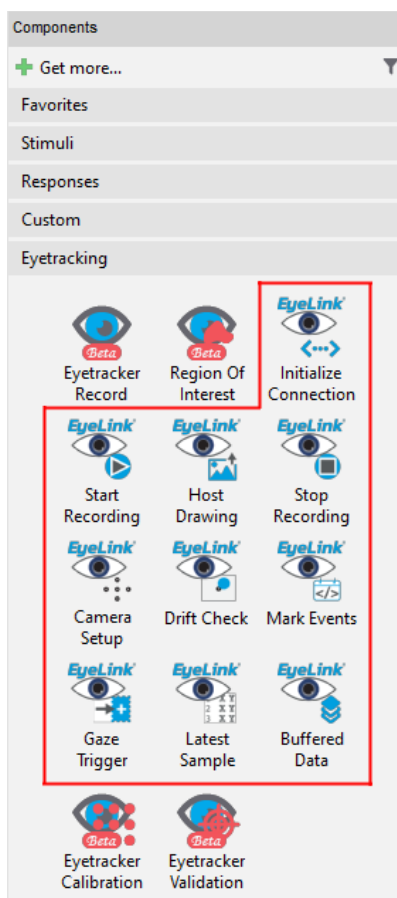
3.11 MRI Synchronization and continuous block recording	52
4 Known issues and trouble-shooting tips	54
4.1 Retina Displays on macOS	54
4.2 Experiment quitting after Camera Setup/Calibration/Validation/Drift Checks	55
4.3 EyeLink Plugin components showing up as puzzle pieces	56

1 Introduction and Installation

1.1 Introduction

The EyeLink Plugin for PsychoPy includes everything you will need for interacting with EyeLink systems from PsychoPy. First, it includes all the Python packages and libraries that are needed to communicate with the EyeLink system. It also includes a set of Builder components that can be added to PsychoPy Builder experiments to enable fully-featured interaction with EyeLink systems. Finally, it includes examples and documentation that can serve as starting point for programming EyeLink experiments in PsychoPy.

Once installed, the components will appear in the Components panel in the PsychoPy Builder interface in the Eyetracking section.



The components provide mechanisms for:

a) Initializing a connection with the EyeLink Host PC/eye tracking system

- b) Performing camera setup/participant calibration
- c) Performing drift check/drift correct between trials or blocks of trials
- d) Populating the Host PC backdrop with trial image stimuli and/or landmarks representing the location of stimuli for online feedback to the experimenter regarding the gaze location relative to stimuli
- e) Starting/stopping eye tracker recording
- f) Marking experimental event occurrence and sending special messages to Data Viewer to allow stimulus visualization, to log interest area and target position information, and to log trial variable information
- g) Providing an easy way to perform fixation window gaze triggering
- h) Accessing EyeLink sample and event data in real-time

Example experiments illustrating how to use the EyeLink Plugin for PsychoPy are available after installation (see section 1.4 below) and are also posted on the SR Research Support Forums at the following link.

[Getting Started with PsychoPy / EyeLink Integration](#)

These examples are grouped into three sets of similar examples:

- 1) The first set shows how to use EyeLink code directly in Coder-built experiments.**
- 2) The second set shows how to use EyeLink code in Code components in PsychoPy's Builder (this is a legacy set that was developed before plugins were a feature of PsychoPy).**
- 3) The third set shows how to use the EyeLink Plugin's components to add EyeLink integration to Builder experiments, which simplifies things compared to the code-based methods.**

This manual will focus on the techniques illustrated by the third, plugin-based set of examples.

A description of the examples that are included is below.

<i>EyeLinkPicture_Plugin</i>	A simple example, which shows how to initialize a connection with the Host PC, control eye tracker recording, populate the Host PC backdrop, and mark critical experimental information in the EDF, including stimulus event marking and sending Data Viewer drawing, interest area, and variable logging.
<i>EyeLinkPursuit_Plugin</i>	Shows how to record the target position in a smooth pursuit task. This script also shows how to record

	dynamic interest area and target position information to the EDF data file so Data Viewer can recreate the interest area and playback the target movement.
<i>EyeLinkSaccade_Plugin</i>	Shows how to retrieve eye events (saccades) during testing. A visual target appears on the left side, or right side of the screen and the participant is required to quickly shift gaze to look at the target (pro-saccade) or a mirror location on the opposite side of the central fixation (anti-saccade).
<i>EyeLinkFixWindow_Plugin</i>	Shows how to implement a gaze-based trigger. First a fixation cross is shown at the center of the screen. The trial moves on only when gaze has been directed to the fixation cross.
<i>EyeLinkGCWindow_Plugin</i>	Shows how to manipulate visual stimuli based on real-time gaze data. A mask is shown at the current gaze position in the "mask" condition; in the "window" condition, the image is masked, and a window at the current gaze position will reveal the image hidden behind the mask.
<i>EyeLinkMRIdemo_Plugin</i>	Shows how to do continuous eye tracker recording through a block of trials (e.g., in an MRI setup), and how to synchronize the presentation of trials with a sync signal from the MRI. With a long recording, we start and stop recording at the beginning and end of a testing session (run), rather than at the beginning and end of each experimental trial. The MarkEvents component is set to send trial onset/offset markers so that the long recording can be broken down into separate trials during analysis.
<i>EyeLinkVideo_Plugin</i>	A simple video demo. In each trial eye movements are recorded while a video stimulus is presented on the screen. Integration messages from the MarkEvents component allow the video to play back in Data Viewer

1.2 Install the EyeLink Plugin for PsychoPy

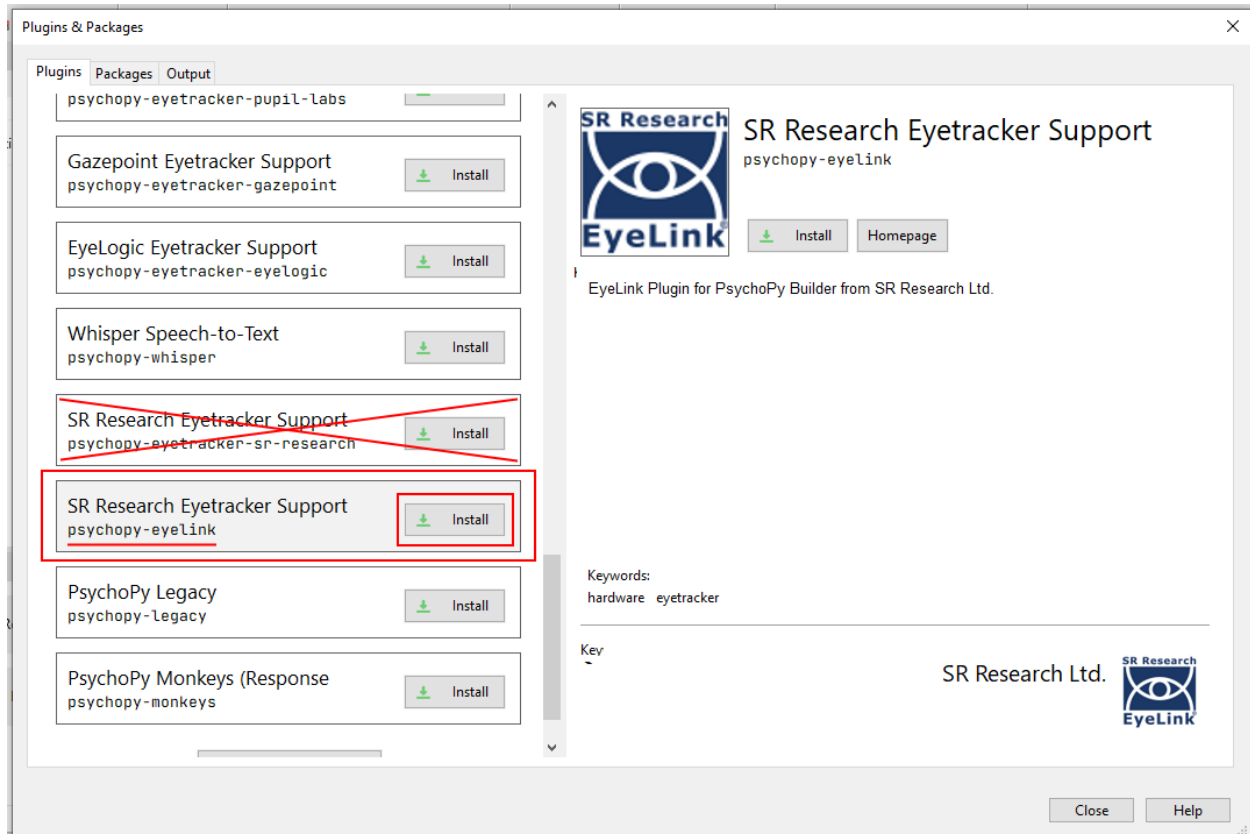
1) Install PsychoPy. The plugin is compatible with 2024.1.0 and up.

<https://www.psychopy.org/download.html>

2) Launch PsychoPy and go to “Tools -> Plugin/packages manager...”

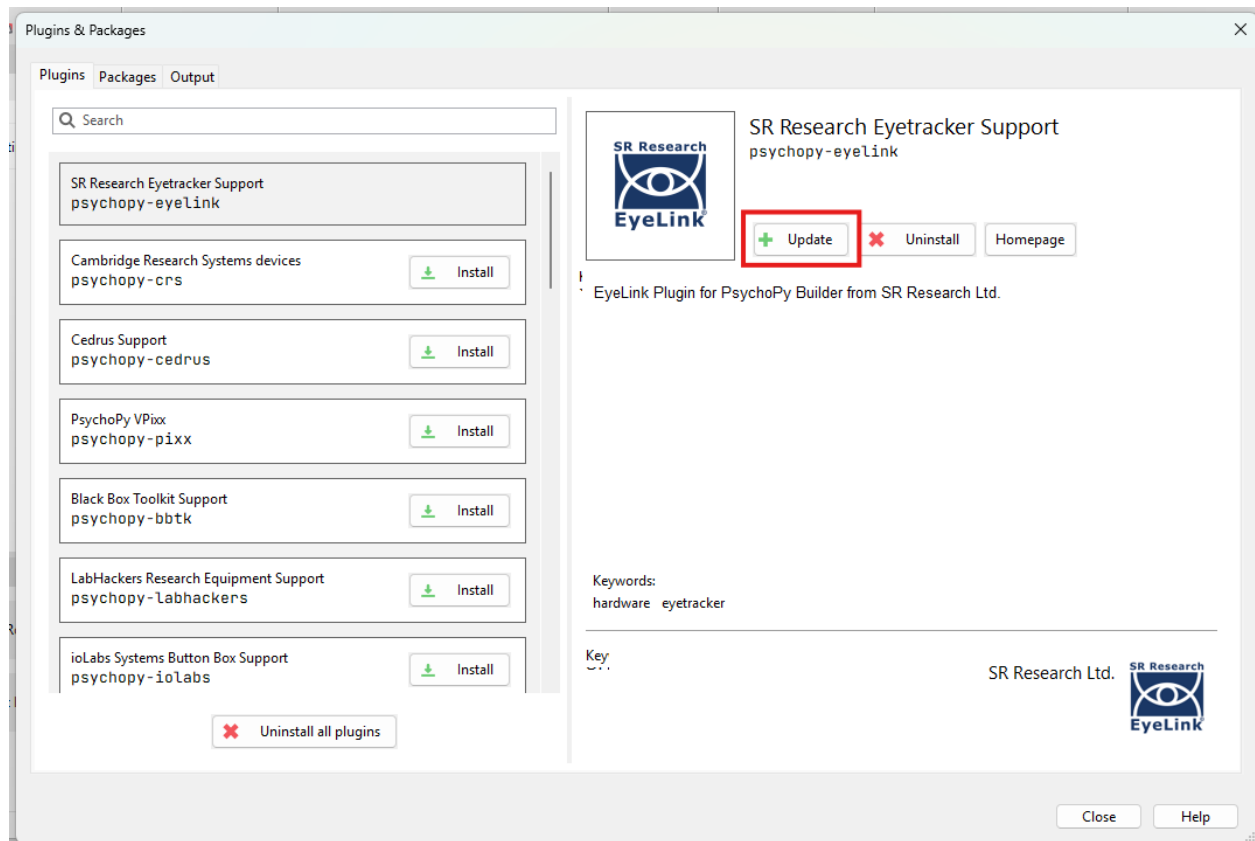
Scroll down until you find the SR Research Eyetracker Support plugin called

psychopy-eyelink



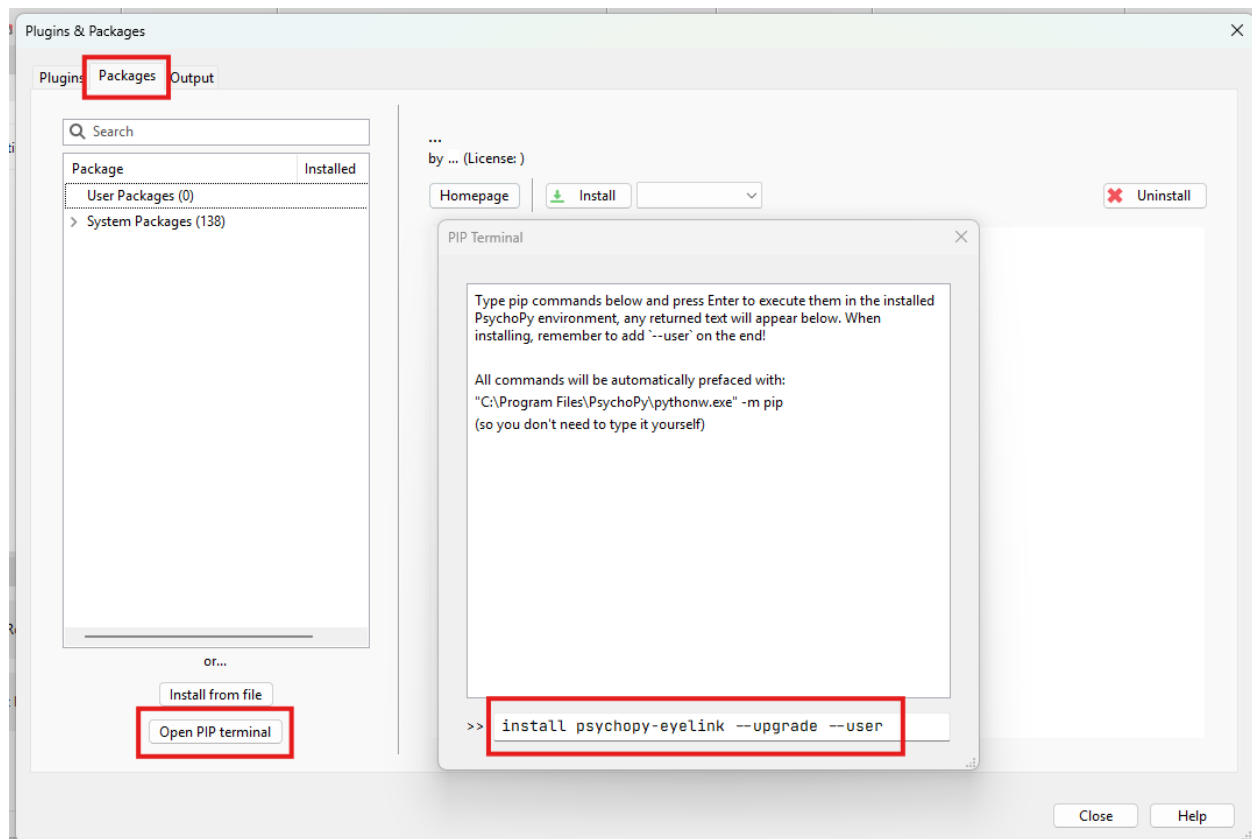
Note, do not install the plugin called psychopy-eyetracker-sr-research – that plugin uses the built in ioHub eye tracker extension for PsychoPy and does not provide full-featured EyeLink integration support. Installing it will not add any functionality to the EyeLink Plugin for PsychoPy that is covered in this manual and that is provided by SR Research. Instead, make sure to install the one called simply psychopy-eyelink. Installing psychopy-eyelink will install all Python modules and dependencies needed to run EyeLink experiments through PsychoPy’s Builder and Coder. These dependencies include Pylink and EyeLinkCoreGraphicsPsychoPy – these dependencies are used by the EyeLink Plugin to allow communication with the EyeLink Host PC and to provide mechanisms for eye tracker setup, calibration/validation, and drift check procedures.

If the EyeLink Plugin has already been installed on your computer, then you can use the Update button to update the EyeLink Plugin as well as its dependencies.



Alternatively, you may install the EyeLink Plugin for PsychoPy using a Python pip command. To do so, from the Plugins & Packages dialog, go to the Packages tab, click Open PIP terminal, and enter the following command in the to the PIP Terminal. Then press Enter. You will see some text indicating the plugin is being installed.

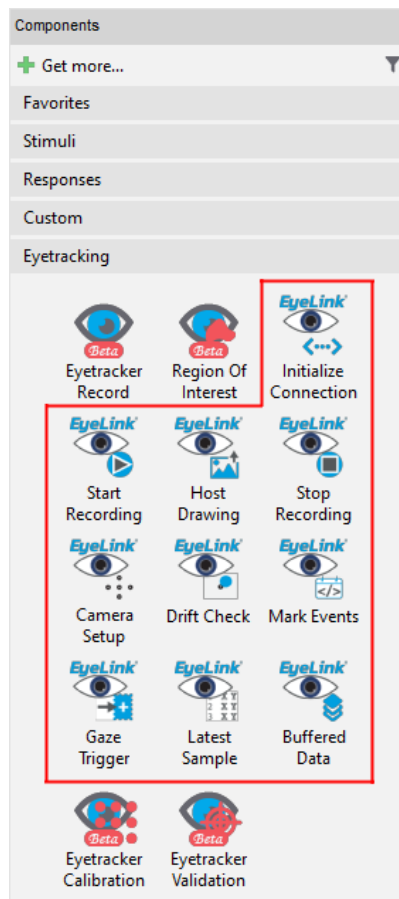
install psychopy-eyelink --upgrade --user



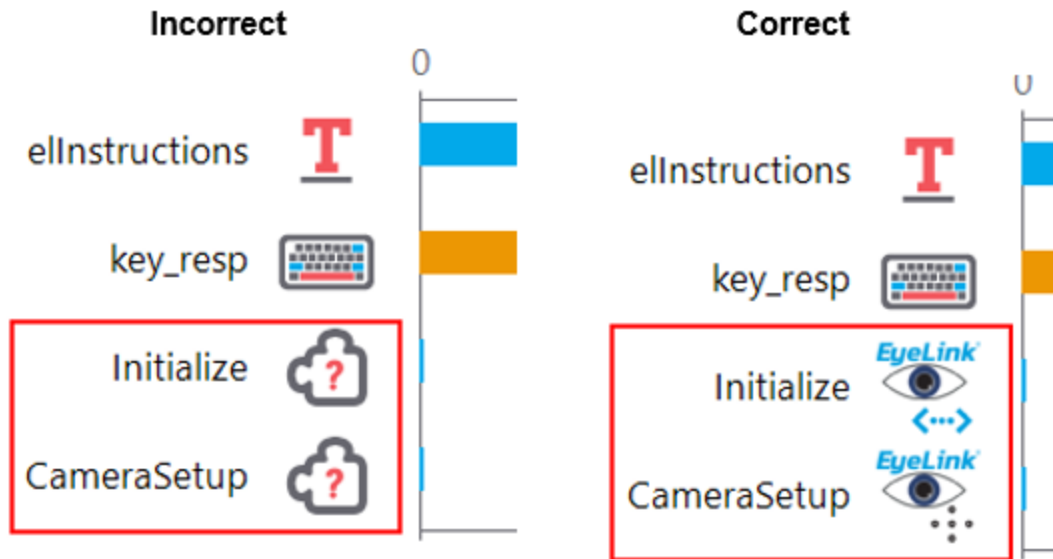
Note: For versions of PsychoPy earlier than 2025.1.1, the command must include pip at the beginning: `pip install sr-research-pylink --user --upgrade`.

Then press Enter. Running that PIP command will install the plugin.

You can close and then relaunch PsychoPy to verify that the plugin has been installed properly –if it is installed correctly then you will see the EyeLink Plugin components in the Eyetracking tab of the Components panel. They will look like this:



Important Note – always make sure to launch PsychoPy first and verify that you see the EyeLink Plugin components in the Eyetracking panel before opening an experiment. Then use the File -> Open command to open any experiments that have EyeLink Plugin components in them. If you instead double-click a .psyexp Builder project file (before PsychoPy has been launched) to launch PsychoPy / open the experiment, then the EyeLink Plugin will likely not have loaded properly, and the components will appear as puzzle pieces. See the image below for an illustration of correct vs incorrect plugin component loading.



If you see those puzzle pieces instead of the normal EyeLink Plugin components then you can either just use File -> Open to open the same project again or re-launch PsychoPy and use File -> Open to open the project to resolve the issue. See section 4.3 for more information.

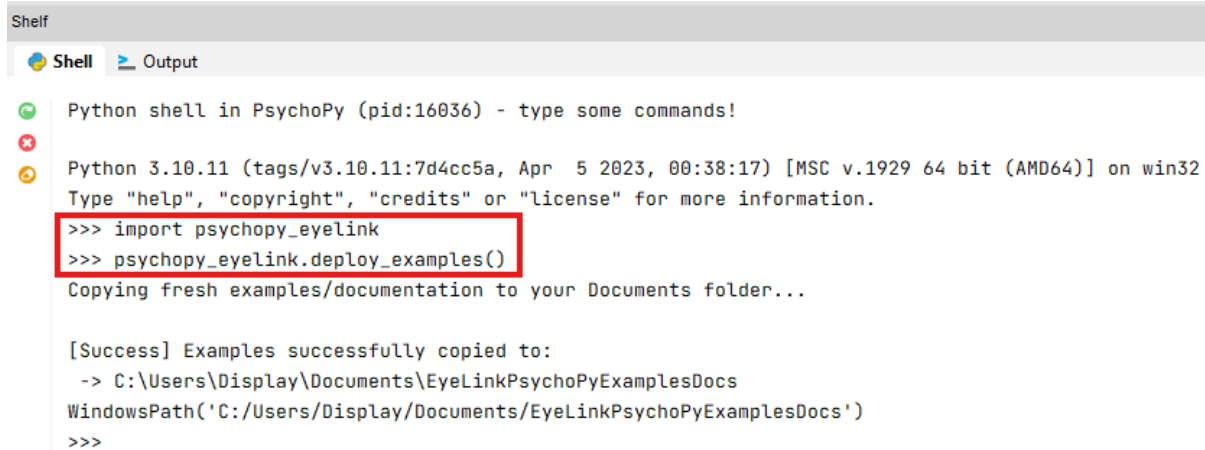
1.3 Configure the IP address of the Display PC

A typical setup of an EyeLink tracker involves two computers, the EyeLink Host PC for eye tracking data registration and a Display PC for stimulus presentation. The Display PC communicates with the Host PC through an Ethernet connection. For successful Ethernet communication between the two computers, the Display PC has to be on the same network as the Host PC. One frequently seen difficulty when connecting to the tracker is caused by failing to set the IP address configuration on the Display PC. The Display PC IP address is typically set to 100.1.1.2, and the subnet mask should be 255.255.255.0. You can see the FAQ: [How do I configure my network settings to connect to the EyeLink Host PC?](#) for instructions on how to do this. You can also find more information in the EyeLink Installation Guide for the system (<https://www.sr-research.com/support/forum-34.html>)

1.4 Deploy the example sets and documentation

The EyeLink Plugin provides examples that illustrate how to add EyeLink integration to both Coder-based and Builder-based experiments. To access the examples, first deploy them to your computer's Documents folder. You can do so by launching Coder (Window -> Show Coder), pressing Enter to start a Python shell, and then typing the following two commands to the Python shell (note, these commands use underscores between the text "psychopy" and "eyelink" and between "deploy" and "examples"):

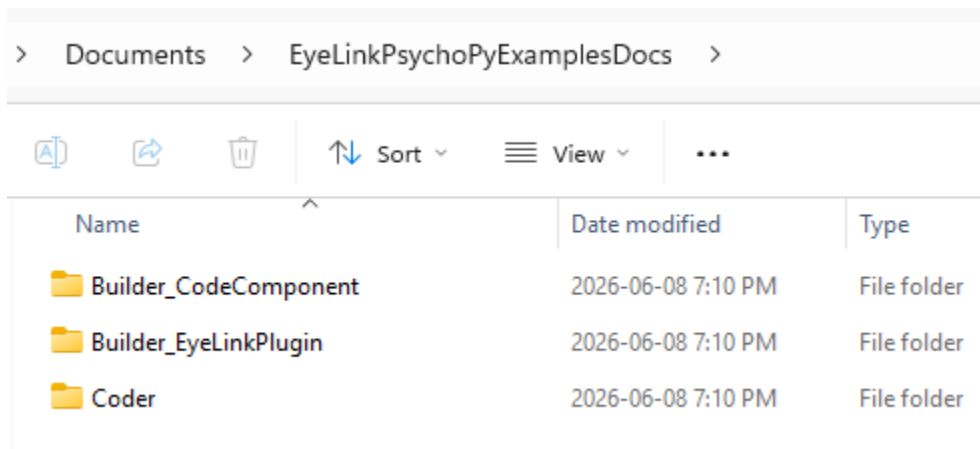
```
>>> import psychopy_eyelink
>>> psychopy_eyelink.deploy_examples()
```



```
Shelf
Shell Output
Python shell in PsychoPy (pid:16036) - type some commands!
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr 5 2023, 00:38:17) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import psychopy_eyelink
>>> psychopy_eyelink.deploy_examples()
Copying fresh examples/documentation to your Documents folder...

[Success] Examples successfully copied to:
-> C:\Users\Display\Documents\EyeLinkPsychoPyExamplesDocs
WindowsPath('C:/Users/Display/Documents/EyeLinkPsychoPyExamplesDocs')
>>>
```

These commands will place a copy of the example sets in your operating system's Documents folder, in a folder called EyeLinkPsychoPyExamplesDocs. If an existing folder with the same name is detected, the folder that is already in the Documents folder will be renamed with the date/time appended to the folder name before the fresh set of examples is placed there. The examples that are relevant to the EyeLink Plugin that is discussed in this manual will be in the Builder_EyeLinkPlugin folder.



2 Plugin Components

This section of the document describes the functionality of each of the components in the EyeLink Plugin for PsychoPy and provides usage instructions and all property definitions for each component.

2.1 Initialize Connection



The Initialize Connection component initializes a connection to the EyeLink Host PC from the Display PC (i.e., from the computer running the PsychoPy script). It also allows for configuration of some settings for the session.

The Initialize Connection component will also make it so that at the end of the experimental run, the script will send a command to the Host PC to have it transfer the EyeLink data file for the run to a participant-specific folder within the results folder in the experimental project directory. Note, this EDF file contains pointers to other files in the experiment (e.g., to interest area set files, accessory graphics files, and to images/videos used in the experiment to help Data Viewer with its functionality). So, when doing analysis, make sure you are working from a full copy of your PsychoPy experimental folder to ensure that when you import the EDF to Data Viewer it can find the other files that it needs, as well.

Its “Dummy Mode” property can be checked to skip connection to the Host PC and to make it so other EyeLink components like Drift Checks will be skipped automatically. This feature is useful when testing the project without an eye tracker/Host PC.

In rare cases where the Host PC’s IP address may have been changed, you can also check the “Use Custom Host IP Address” and then enter the updated IP address in the “Host IP Address” property that will become available when “Use Custom Host IP Address” is checked.

It is critical to set the “PsychoPy Screen Unit Type” property to match the screen unit type of the stimulus components in the project. This property is used in converting location/size data from PsychoPy units to EyeLink screen units, which are in pixels and where 0,0 corresponds to the top-left corner of the screen and values increase as position moves to the right and down – see section 3.2 for further discussion on this topic.

The “Disable EyeLink Audio” can be used to fully disable all EyeLink audio that may be played during calibration/validation/drift checks.

The execution of the code written by the Initialize Connection component occurs at the beginning/end of the experiment and at the end of the routine it is in, so the Start/Stop properties are not used.

The screenshot shows the 'Initialize Properties' dialog box with the following settings:

- Name:** InitializeConnection
- Start:** \$ time (s) dropdown set to 0.0; Expected start (s) field is empty.
- Stop:** \$ duration (s) dropdown set to 0.001; Expected duration (s) field is empty.
- Dummy Mode:** ☐ (unchecked)
- Use Custom Host IP Address:** ☐ (unchecked)
- PsychoPy Screen Unit Type:** pix (selected in dropdown)
- Disable EyeLink Audio:** ☐ (unchecked)

Properties of Initialize Connection:

Dummy Mode	When checked, the script will not connect to the Host PC and EyeLink components will be skipped. Checking this property allows for testing the script in the absence of an EyeLink system/Host PC. This property should not be checked for real data collection.
Use Custom Host IP Address	By default, the EyeLink Host PC has an IP address of 100.1.1.1. If this property is unchecked the script will connect to that IP address when initializing a connection to the Host PC. If the Host PC has been reconfigured to use a different IP address, then this property can be checked to reveal the "Host IP Address" property, which

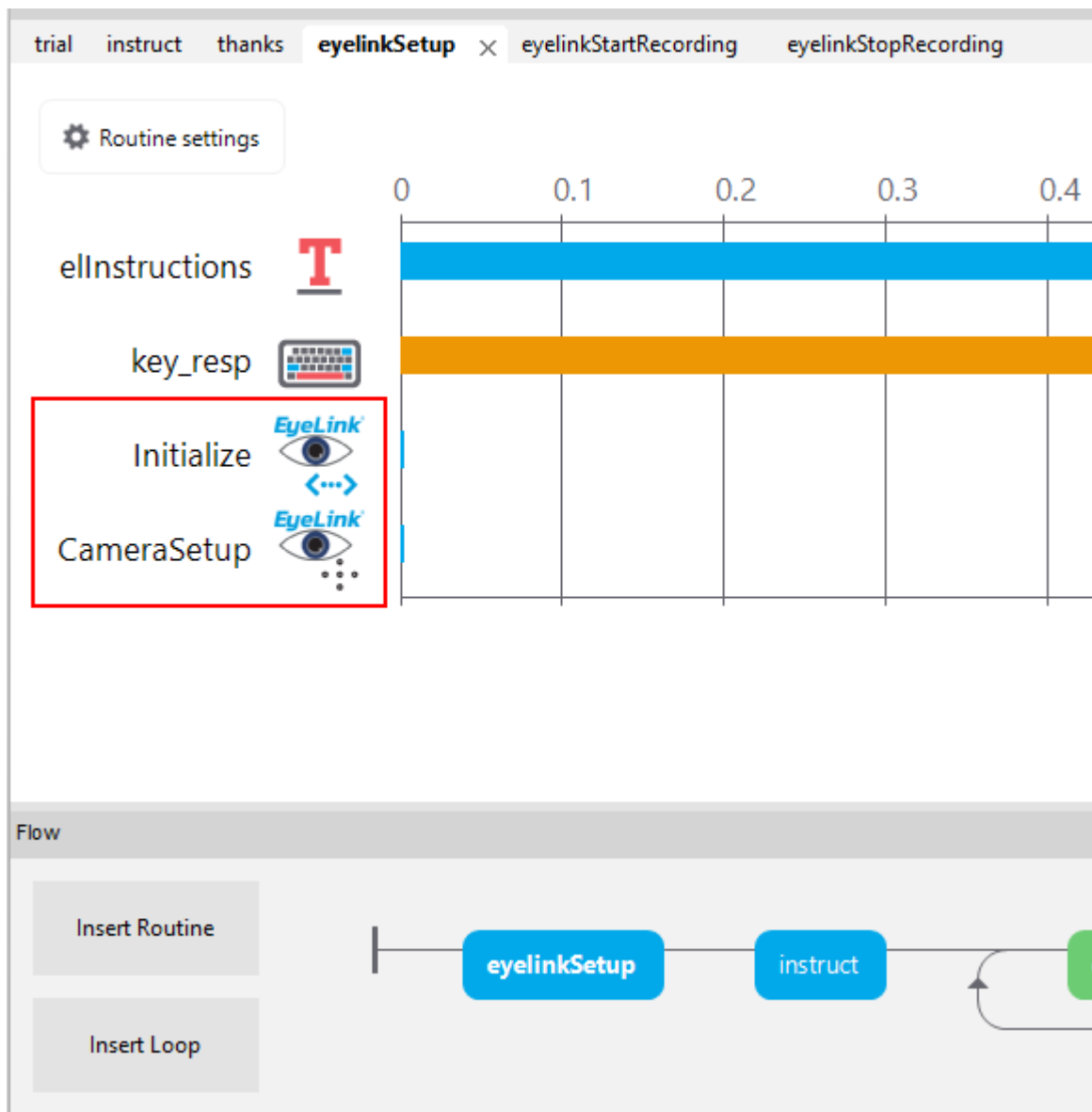
	can be edited to match the custom IP address the Host PC is using.
Host IP Address	When "Use Custom Host IP Address" is checked, this property to should be edited to match the custom IP address that the Host PC is using.
PsychoPy Screen Unit Type	This property should be set to match the screen coordinate system being used by the experiment. Some conversion is done by the plugin to convert position values to the EyeLink coordinate system, and that conversion depends on the proper PsychoPy coordinate system being selected here.
Disable EyeLink Audio	If checked, all calibration/validation/drift check audio will be disabled.

2.2 Camera Setup



The Camera Setup component will put the Host and Display PCs in Camera Setup mode. From there, the experimenter can set up the participant and perform eye tracker calibration/validation. For details on the system setup process, please see the EyeLink system User Manual (<https://www.sr-research.com/support/forum-34.html>) and the video tutorials for the system (<https://www.sr-research.com/support/forum-12.html>). The Camera Setup component includes properties that allow the appearance of the calibration/validation target and the sounds played during these processes to be customized. This includes the ability to use image files or videos as calibration/validation targets.

If the Camera Setup component is placed in a routine along with the Initialize Connection component, then these should be the bottom-most components in the routine, and the Camera Setup component should be below the Initialize Connection component (as illustrated in the examples provided with the plugin).



The execution of the Camera Setup component occurs at the end of the routine, so its Start/Stop properties are not used.

Properties of Camera Setup:

CameraSetup Properties

Basic Data Testing

Name: CameraSetup

Start: \$ time (s) 0.0
Expected start (s)

Stop: \$ duration (s) 0.001
Expected duration (s)

Calibration Type: HV9

Target Type: picture

Target Filename: images/fixTarget.bmp

Background Color: win.color

Calibration/Validation Sounds: defaults

Help OK Cancel

Properties of Camera Setup:

Calibration Type	Allows configuration of the number of calibration targets. Options include a horizontal-only 3 point routine (H3), and horizontal/vertical 3, 5, 9, and 13 point routines (HV3, HV5, HV9, and HV13).
Target Type	Allows configuration of the target appearance. Options include circle, spiral, picture, and movie. Picture and movie options require image or video files in the experimental folder.
Target Size	Specify the size of the target. This property is only available for circle and spiral targets.
Target Filename	The image or movie filename. This property is only available for picture and movie targets. Can include path relative to the psyexp file.
Foreground Color	Color of the target. This property is only available for circle and spiral targets.

Background Color	Background color for calibration. Make sure to match this with the background of the experimental stimuli to optimize accuracy of the eye tracker. Can be set to win.color to match the background color of the experiment.
Calibration/Validation Sounds	This can be set to defaults (which uses the files qbeep.wav, type.wav, and error.wav that are included in example experiments), off, or to use custom wav files. If custom wav files is selected then Calibration/Validation Filenames need to be additionally specified.
Calibration/Validation Filenames	If custom wav files is selected then Calibration/Validation Filenames need to be additionally specified. The filenames should be listed in the order target, good, error. Target -- sound to play when target moves; Good -- sound to play on successful operation; Error -- sound to play on failure or interruption.

2.3 Drift Check



The Drift Check component will put the Host and Display PCs in Drift Check mode. This will present a single fixation target to the participant. The experimenter will be able to see a representation of the target, along with a representation of the current gaze position, on the Host PC. When the gaze is steady on the target, either the experimenter or the participant can press the spacebar on the Display PC or Host PC to proceed. Alternatively, the experimenter can press Esc from a drift check to go back to the Camera Setup mode to potentially recalibrate/revalidate the eye tracker. When ready to proceed back to the drift check, the experimenter can then click the Output/Record button on the Host PC or press O to pick up at the drift check where Esc had previously been pressed. A drift check allows the experimenter to check/assess the accuracy of the calibration model, ensure that the participant is fixating a certain location before proceeding (usually to the next trial in the experiment), and to recalibrate the eye tracker if needed.

The execution of the Drift Check component occurs at the end of the routine, so its Start/Stop properties are not used.

DriftCheck Properties

Basic Data Testing

Name: DriftCheck

Start: \$ time (s) 0.0
Expected start (s)

Stop: \$ duration (s) 0.001
Expected duration (s)

Target Position: [0,0]

Target Type: picture

Target Filename: images/fixTarget.bmp

Background Color: win.color

Drift Check Sounds: defaults

Help OK Cancel

Properties of Drift Check:

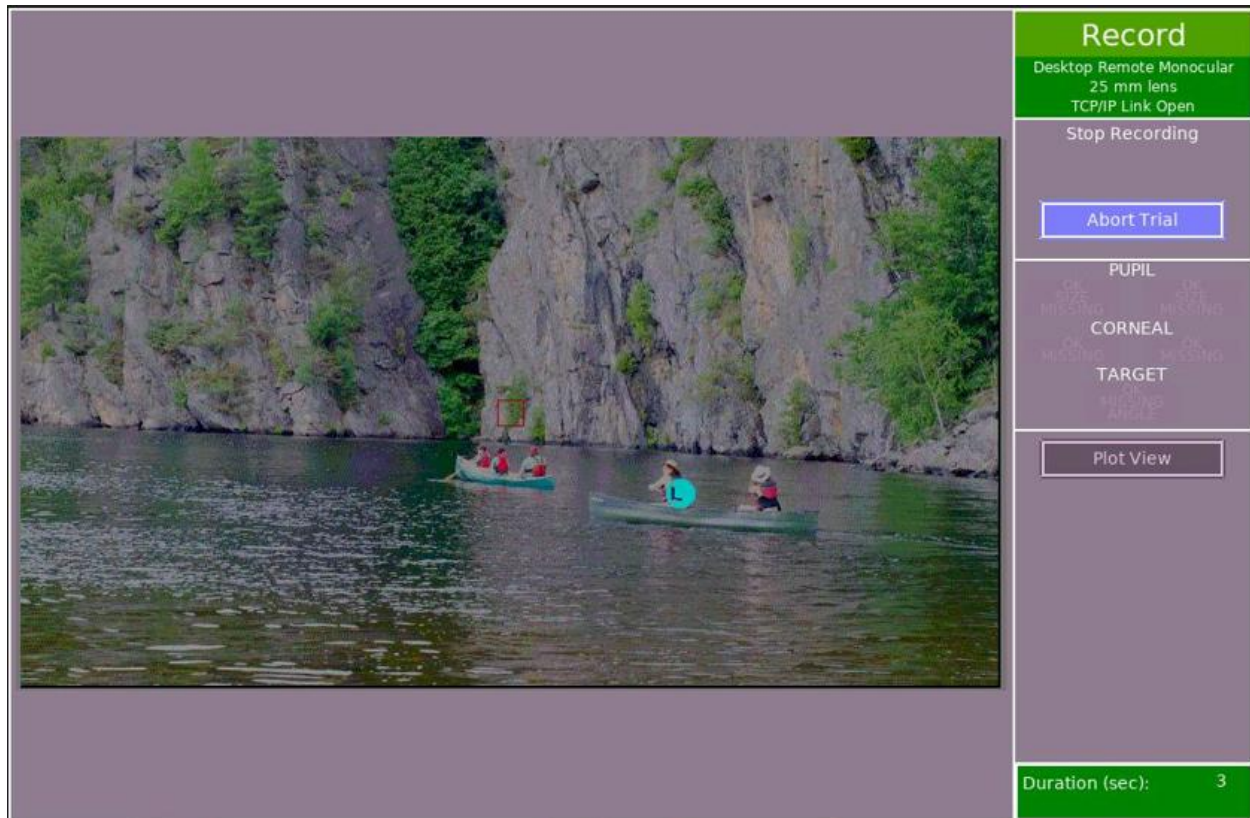
Target Position	The X,Y location of the Drift Check target, in the PsychoPy coordinate system (using the coordinate system specified in the PsychoPy Screen Unit Type property of the experiment's EyeLink InitializeConnection component.
Target Type	Allows configuration of the target appearance. Options include circle, spiral, picture, and movie. Picture and movie options require image or video files in the experimental folder.
Target Size	Specify the size of the target. This property is only available for circle and spiral targets.
Target Filename	The image or movie filename. This property is only available for picture and movie targets. Can include path relative to the psyexp file.

Foreground Color	Color of the target. This property is only available for circle and spiral targets.
Background Color	Background color for calibration. Make sure to match this with the background of the experimental stimuli to optimize accuracy of the eye tracker. Can be set to win.color to match the background color of the experiment.
Drift Check Sounds	This can be set to defaults (which uses the files qbeep.wav, type.wav, and error.wav that are included in example experiments), off, or to use custom wav files. If custom wav files is selected then Calibration/Validation Filenames need to be additionally specified
Drift Check WAV Filenames	If custom wav files is selected then Drift Check Filenames need to be additionally specified. The filenames should be listed in the order target, good, error. Target -- sound to play when target moves; Good -- sound to play on successful operation; Error -- sound to play on failure or interruption.

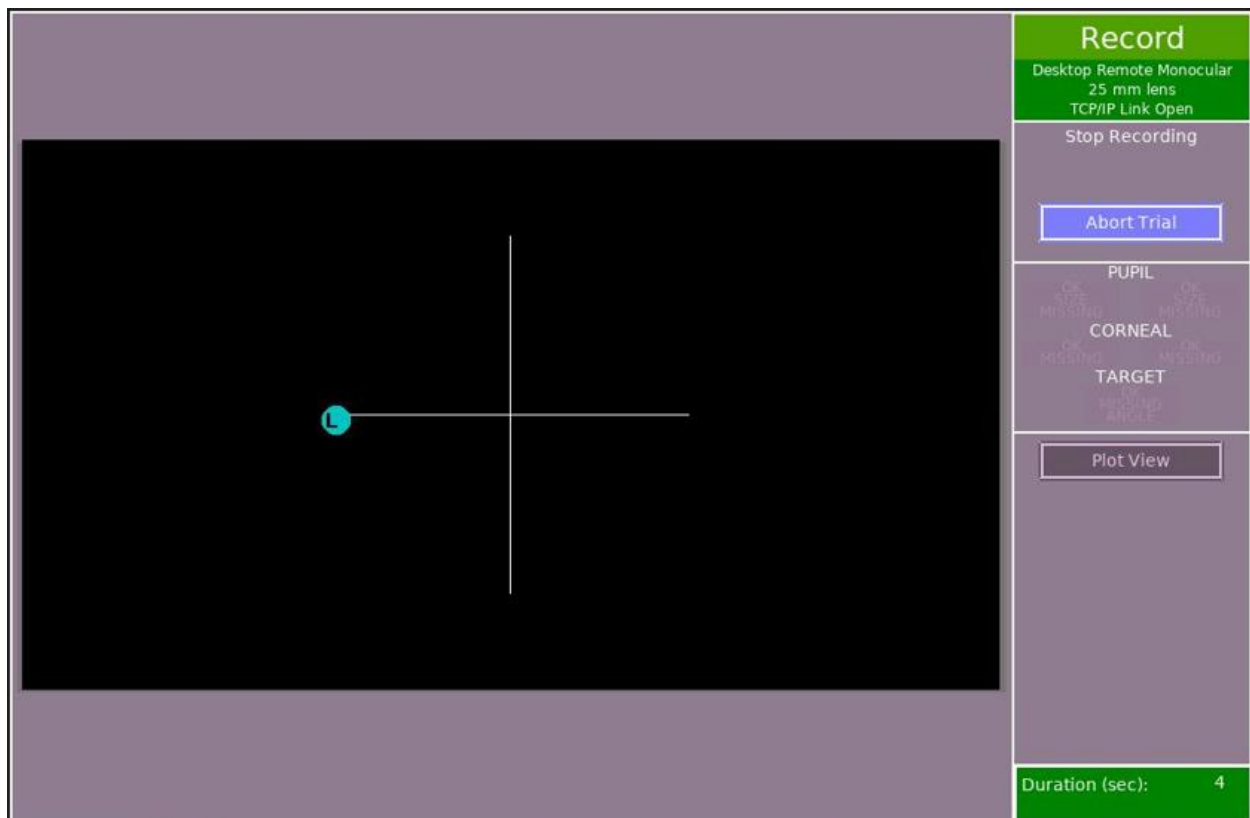
2.4 Host Drawing



The Host Drawing component sends feedback to the experimenter via the EyeLink Host PC that can be helpful during eye tracker data recording. First, in the gaze view during recording, the Host PC shows the experimenter a representation of the participant's Display PC monitor, giving the experimenter an indicator of the participant's current gaze position relative to stimuli on the Display PC monitor.



The “Images for Host PC Backdrop” property allows for sending copies of image components to the Host PC backdrop, and the “Components for Draw Commands to Host PC Backdrop” will draw a rectangle representing each stimulus component’s boundaries on the Host PC backdrop. The “Additional Draw Commands for Host PC Backdrop” property allows any of the other Host PC’s drawing options to be used in a flexible manner, which can be useful for things like smooth pursuit tasks, where the position of the stimulus may change (e.g., you can draw lines to represent the trajectory of the stimulus; see the EyeLinkPursuit_Plugin example for an illustration of how to do this).

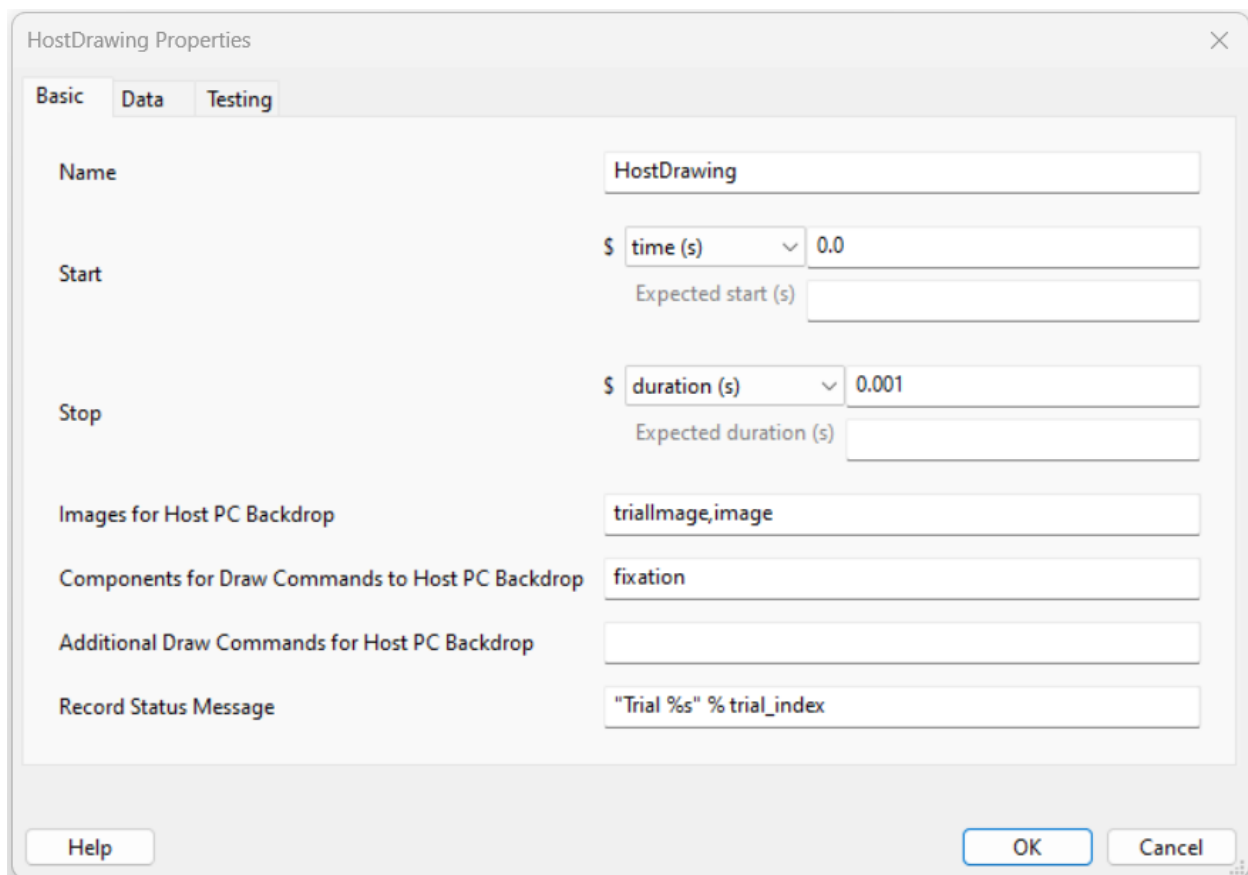


Note, once the Host PC has started recording, its backdrop cannot be updated. So, even though the participant's Display PC screen may change, the Host PC's backdrop must remain static. This limitation does not apply to visualizations in Data Viewer after data has been collected (e.g., you will be able to see dynamic representations of the stimuli the participant saw during analysis in Data Viewer's Trial View window, provided the Mark Events component is configured appropriately).

The Host Drawing component can also send a message to the experimenter via the "Record Status Message" property. For example, you can use this to send yourself a message indicating the current trial number in the experimental session/run. You can use common Python string concatenation, along with names of PsychoPy objects, when populating this field.



All the properties of the Host Drawing component are optional, and the Host Drawing component itself is optional. If it is used, then it should be placed at the top of the routine that includes the Start Record component – positioning it at the top if the routine will ensure it is executed before the subsequent Start Record component (and the Drift Check component, if the routine includes that, as well). The execution of the Host Drawing component occurs at the end of the routine, so its Start/Stop properties are not used.



The image shows a 'HostDrawing Properties' dialog box with three tabs: 'Basic', 'Data', and 'Testing'. The 'Basic' tab is selected. It contains several input fields for configuring the Host Drawing component. The 'Name' field is set to 'HostDrawing'. The 'Start' section has a '\$ time (s)' dropdown set to '0.0' and an empty 'Expected start (s)' field. The 'Stop' section has a '\$ duration (s)' dropdown set to '0.001' and an empty 'Expected duration (s)' field. The 'Images for Host PC Backdrop' field contains 'trialImage,image'. The 'Components for Draw Commands to Host PC Backdrop' field contains 'fixation'. The 'Additional Draw Commands for Host PC Backdrop' field is empty. The 'Record Status Message' field contains '"Trial %s" % trial_index'. At the bottom, there are 'Help', 'OK', and 'Cancel' buttons.

Property	Value
Name	HostDrawing
Start	\$ time (s) 0.0
Expected start (s)	
Stop	\$ duration (s) 0.001
Expected duration (s)	
Images for Host PC Backdrop	trialImage,image
Components for Draw Commands to Host PC Backdrop	fixation
Additional Draw Commands for Host PC Backdrop	
Record Status Message	"Trial %s" % trial_index

Properties of Host Drawing:

Images for Host PC Backdrop	This optional property can be used to transfer copies of one or more image components to the Host PC backdrop. Each image component should be provided as a pair, where the first item in the pair is the name of the image file (or name of the variable or column from the routine's Conditions file that specifies the image file to be used) and the second item in the pair is the name of the component. The items in the pair should be separated by a comma and the pairs should be separated by semicolons. For example, for two image components names image1Comp and image2Comp whose image
-----------------------------	--

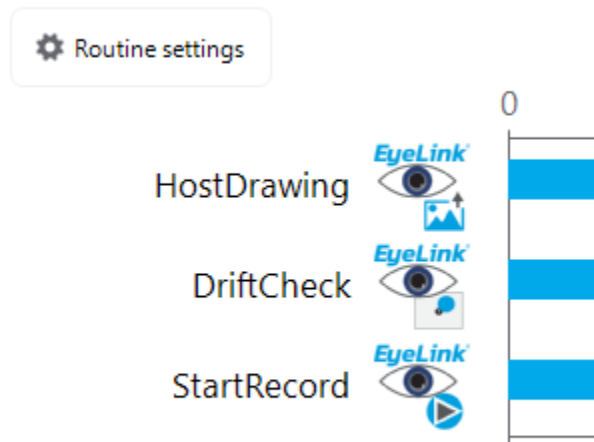
	files are specified as columns image1File and image2File in the Routine Condition file, you would use: image1File, image1Comp; image2File, image2Comp
Components for Draw Commands to Host PC Backdrop	This optional property can be used to draw a rectangle representing stimulus boundaries for one or more components. Any stimulus component type can be included in this list. When including more than one component, separate component names with commas.
Additional Draw Commands for Host PC Backdrop	This optional property can be used to send custom drawing commands to the Host PC backdrop. This can be useful for dynamic stimuli, like in smooth pursuit studies, to provide a trajectory of the stimulus motion. The command should follow expected EyeLink drawing command messages, which are documented in the EyeLink Programmers Guide in the section "Useful EyeLink Commands -> Drawing Commands"
Record Status Message	This optional property can be used to send a text string message to the experimenter that will be presented during eye tracker recording. The string can be build using Python concatenation and can include PsychoPy object names.

2.5 Start Recording



The Start Recording component will send a command to the EyeLink Host PC to start recording eye tracking data. It should be placed in its own routine before the looping routine that handles the critical experimental events during which eye tracking data will be recorded. If Host Drawing and/or Drift Check components are included in that same routine, then make sure that the Start Recording component is at the bottom of the routine. You can right click the Start Recording component and choose "Move to bottom" to ensure it is at the bottom of the routine. This will

ensure recording is only started after the Host PC drawing and Drift Check have been performed.



Starting/stopping recording is typically performed between experimental trials, but can be performed between experimental blocks (where a block consists of multiple trials) as well. Most of the examples provided for the EyeLink Plugin for PsychoPy illustrate how to start recording before each experimental trial and stop recording at the end of each trial, but the EyeLinkMRIdemo_Plugin example shows how to start recording eye tracking data at the beginning of each block of experimental trials and stop recording at the end of the block.

Stopping/starting recording between experimental trials allows you to update the Host PC backdrop (see the Host Drawing component) and perform a Drift Check (see the Drift Check component) between experimental trials – these tasks cannot be performed while the eye tracker is recording data. Sometimes, though, continuous recording through a block of trials is required by the experimental paradigm, e.g., to help control timing, which is often important in an MRI experiment. In these cases, you can position the Start Recording and Stop Recording components in routines before/after routines that handle blocks of trials. If this is the case, make sure to check “Send Trial Onset Offset Messages” under Mark Events to ensure the continuous recording has markers that divide into separate trials.

The Start Recording component does not have any properties to configure. The execution of the start recording component occurs at the end of the routine, so its Start/Stop properties are not used.

2.6 Mark Events



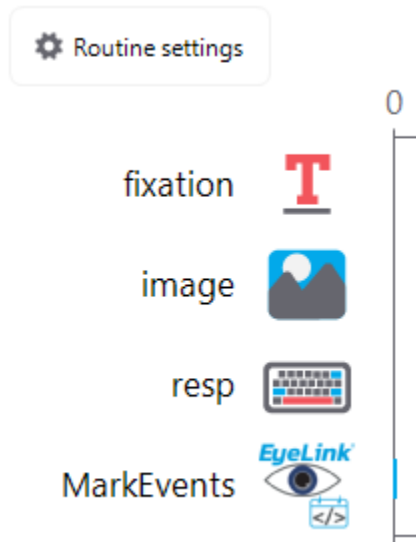
The Mark Events component sends messages to the EyeLink data file to mark the occurrence of experimental events and to log trial data that will be helpful for later data analysis (in Data Viewer or in other data analysis pipelines). It is critical to use a Mark Events component in any routine containing important experimental events, as its roles include the following:

- 1) Sending event-marking messages that will be used in data analysis to identify the times when the critical experimental events occurred. These event-marking can be used in Data Viewer (or other analysis software) to focus analysis on certain interest periods.**
- 2) Logging trial condition information as Trial Variables that can be included in Data Viewer's output reports and used to sort/group the data in Data Viewer's Inspector.**
- 3) Send Data Viewer image and drawing commands to make it so representations of the stimuli the participants viewed can be shown in Data Viewer's various visualization options. This includes showing images, drawing shapes to represent static stimuli that are not image-based, and to play back videos that were presented during the experiment.**
- 4) Automatically create/log interest areas for experimental stimuli that can be used in analysis.**
- 5) Send special target position messages which will allow the export of target position data, which can be visualized and included in data output reports using Data Viewer.**
- 6) Logging behavioral data (e.g., reaction time, accuracy, key pressed) for response components as Trial Variables for Data Viewer.**
- 7) Sending trial onset/offset messages -- this is not necessary unless your experiment involves continuous recording throughout a block of trials, as the marking of trial onset/offset is automatic when starting/stopping recording.**
- 8) Allowing for the Ctrl-C key combination to be used during the routine to allow early termination of the session/run that includes retrieval of the EyeLink data file (EDF) for that session from the EyeLink Host PC along with a proper shutdown of the connection to the Host PC.

** See [Getting Started with Data Viewer](#) for more information/learning resources involving Data Viewer usage. Even if you do not plan to use Data Viewer for analysis, the data

logged by the Mark Events component is useful, and having this data in the EyeLink data file will make it available in other data analysis pipelines, as well.

The Mark Events component should be positioned at the bottom of the components in the routine:



Execution of the Mark Events is ongoing throughout the routine, so its Start/Stop properties are not used.

MarkEvents Properties

Basic

Data

Testing

Name

MarkEvents

Start

\$ time (s) 0.0

Expected start (s)

Stop

\$ duration (s) 0.001

Expected duration (s)

Trial Condition Variables for Logging

trialImage,condition

Stimulus Components for Event Marking

fixation,image

Include All Event Marked Stimulus Components For DV Drawing Messaging

☒

Send Video Frame Onset Messages to Enable Playback in Data Viewer

☐

Include All Event Marked Stimulus Components For Interest Area Messaging

☐

Stimulus Components for Interest Area Messaging

Interest Area Margins

0

Interest Area Shape

Rectangle

Stimulus Components for Target Position Messaging

Response Components for Event Marking

resp

Send Trial Onset Offset Messages

☐

Allow Ctrl-C Quit Key to be Used

☒

Help

OK

Cancel

Properties of Mark Events:

Trial Condition Variables for Logging	Can be used to log the values of any PsychoPy variables (e.g., variables from the loop's Conditions CSV file) as Trial Variables to the EDF at the end of the trial. Trial Variables can be used to group the data in Data Viewer and can be included in any of the output reports that Data Viewer can create Variable names should be separated by commas. It is typically a good idea to include all columns from any Conditions files here.
Stimulus Components for Event Marking	Specify a list of the components whose onset/offset should be marked in the EDF. The Message format in the EDF will be the name of the component followed by _ONSET or _OFFSET. E.g., for a component called myImage, its messages will be

	myImages_ONSET and myImages_OFFSET. The components in the list should be separated by commas. Typically the list would include all the stimulus components in the routine.
Include All Event Marked Stimulus Components for DV Drawing Messaging	If checked, then the script, for each component in the "Stimulus Components for Event Marking" list, will send special Data Viewer integration messages which that allow the stimuli to be viewed in Data Viewer in its Trial View window (in Spatial Overlay and Animation Playback View, as well as in Fixation/Heat Maps). For image components, you will see the image as it was presented, and for non-image visual stimulus components, you will see rectangles marking the positions of the stimuli. If the experiment involves movies, then to ensure playback for movie components, make sure to additionally check "Send Video Frame Onset Messages to Enable Playback in Data Viewer".
Stimulus Components for Data Viewer Drawing Messaging	The script, for each component in the list, will send special Data Viewer integration messages which allow the stimuli to be viewed in Data Viewer in its Trial View window (in Spatial Overlay and Animation Playback View, as well as in Fixation/Heat Maps). For image components, you will see the image as it was presented, and for non-image visual stimulus components, you will see rectangles marking the positions of the stimuli. If the experiment involves movies, then to ensure playback for movie components, make sure to additionally check "Send Video Frame Onset Messages to Enable Playback in Data Viewer". Note, this property will only be available if "Include All Event Marked Stimulus Components for DV Drawing Messaging" is unchecked.
Send Video Frame Onset Messages to Enable Playback in Data Viewer	If checked, then the script, for each movie component in the "Stimulus Components for Event Marking" list, will send special Data Viewer integration messages which that allow the movie to be played back in Data Viewer in the Trial View's Animation Playback mode.
Include All Event Marked Stimulus Components for Interest Area Messaging	If checked, then the script, for each component in the "Stimulus Components for Event Marking" list, will send special Data Viewer integration messages that create interest areas that can be used for analysis. The label of the interest area will be based on the name of the component and the ID will be based on its vertical position within the routine.
Stimulus Components for Interest Area Messaging	The script, for each stimulus component in the list, will send special Data Viewer integration messages that create interest areas that can be used for analysis. The label of the interest area will be based on the name of the component and the ID will be based on its vertical position within the routine. Note, this property will only be available if "Include All Event Marked Stimulus Components for Interest Area Messaging" is unchecked.

Interest Area Margins	The number of pixels to extend each edge (left, top, right, and bottom) beyond the component edges for the interest areas. Note, interest area properties can be adjusted during analysis in Data Viewer as well.
Interest Area Shape	Rectangle or ellipse. Note, interest area properties can be adjusted during analysis in Data Viewer as well.
Stimulus Components for Target Position Messaging	The script, for each stimulus component in the list (up to two supported), will send target position messages that allow the component position/trajectory to be visualized in Data Viewer's Trial View window and to be included in the data output reports that Data Viewer can generate. See the EyeLinkPursuit_Plugin example for an illustration of how to use this in a smooth pursuit experiment.
Response Components for Event Marking	For each response component in the list, send an event-marking message to the EyeLink Data File (EDF) to mark its occurrence. The Message format in the EDF will be the name of the component followed by _EVENT or _OFFSET. E.g., for a component called myKeyboard, its message will be myKeyboard_EVENT. The components in the list should be separated by commas. Typically the list would include all the response components in the routine.
Send Trial Onset Offset Messages (usually should be false, should be true for continuous recording with multiple trials)	If checked, then at the beginning/end of the routine (i.e., for each iteration of the loop containing the routine), the script will send a message marking the trial onset (e.g., TRIALID 1) and trial offset (e.g., TRIAL_RESULT). This should only be used if the eye tracker will be continuously recording through all trials in a block (e.g., see the EyeLinkMRIDemo_Plugin example). In most cases, this should be unchecked, as trial start/stop markers are automatically sent if each trial is a separate recording (see EyeLinkPicture_Plugin or any of the other examples for this more typical recording scheme).
Allow Ctrl-C Quit Key to be Used	If checked, then Ctrl-C may be pressed at any point during the execution of the routine to end the experimental run early. Using Ctrl-C in this way will properly close and retrieve the EyeLink Data File (EDF) from the EyeLink Host PC and properly shutdown the connection to the Host PC.

2.7 Stop Recording



The Stop Recording component will send a command to the EyeLink Host PC to stop recording eye tracking data. It should be placed in its own routine after the routine that handles the critical experimental events during which eye tracking data will be recorded. See the discussion in section 2.5 Start Recording for more information about options/considerations in determining when to start/stop eye tracker recording.

The Stop Recording component does not have any properties to configure. The execution of the Stop Recording component occurs at the end of the routine, so its start/stop values are not used.

2.8 Gaze Trigger



The Gaze Trigger component allows for the specification of a gaze triggering region, requiring the participant to maintain gaze within that region for a minimum consecutive duration before the experiment can proceed. Once the gaze criteria are met, the routine containing this component will end.

The triggering region is specified using the same PsychoPy units that are chosen by the EyeLink Initialize Connection component. If Within is checked the the gaze trigger will fire once the gaze has been within the region for the specified minimum duration. If Within is unchecked then gaze must be outside the region for the specified minimum duration. If Send Draw Command to Host PC is checked then a box representing the triggering region will be drawn on the Host PC's backdrop (i.e., on the Host PC's representation of the Display PC screen).

This component will send a message to the EDF marking the time of the event with the text "COMPONENTNAME_FIX_WINDOW_COMPLETED", where COMPONENTNAME will be the name of the component used in the experiment. The component will also log a trial variable with the PsychoPy globalClock Display PC time of the event. The variable will be called COMPONENTNAME.gazeWindowGazeCompletedTime.

See the EyeLinkFixationWindowFastSamples_Plugin example for an illustration of how to use this trigger.

GazeTrigger Properties

Basic Data Testing

Name: GazeTrigger

Start: \$ time (s) .2
Expected start (s):

Stop: \$ duration (s)
Expected duration (s):

Position: [0,0]

Size: [100,100]

Within: ☒

Minimum Duration: 0.5

Send Draw Command to Host PC: ☒

Help OK Cancel

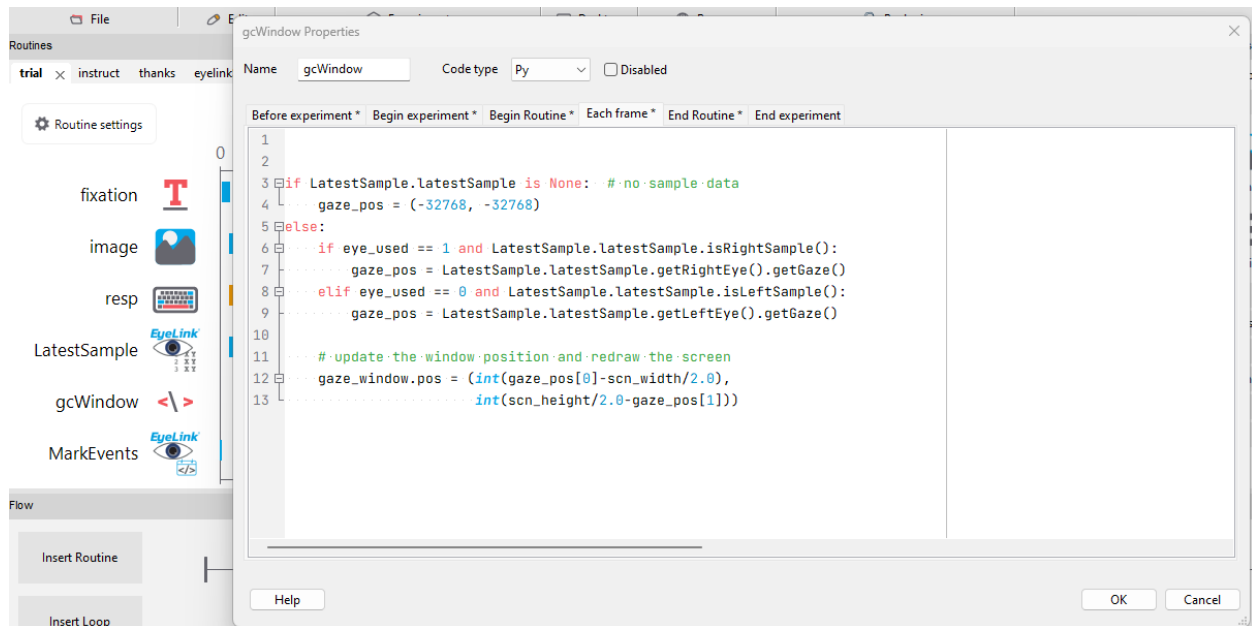
Properties of Gaze Trigger:

Start	Start time for the gaze trigger data checking
Stop	Stop time for the gaze trigger data checking
Position	X,Y position of the gaze-triggered region in PsychoPy project units, as specified in the EyeLink Initialize Connection Component
Size	Height and width of the gaze-triggered region in PsychoPy project units, as specified in the EyeLink Initialize Connection Component)
Within	Whether the requirement is for gaze to be inside the defined region (checked) or outside the defined region (unchecked)
Minimum Duration	Minimum duration that gaze must be in the triggering region consecutively

2.9 Latest Sample



The Latest Sample component will retrieve the most recent eye tracking sample from the EyeLink Host PC (if a new sample is available). It will be checked/retrieved in a loop during the time window defined by its Start and Stop properties. The latest sample data will be available as part of the data structure of the component, and can be accessed via that component's .latestSample property. For example, if the component is called LatestSample, then the sample data will be stored/accessible via LatestSample.latestSample. See the EyeLinkGCWindow_Plugin example for an illustration of how to access/use the data – that example illustrates how to implement a gaze-contingent moving window using the data from a Latest Sample component. The sample data are used in the Each Frame tab of the code component in the trial routine.



Documentation regarding the structure of each latestSample can be found in the Pylink User Manual in the section “Pylink Module -> SampleData”.

Properties of Latest Sample:

Start	Time in the routine to start checking for the eye tracking data
Stop	Time in the routine to stop checking for the eye tracking data

2.10 Buffered Data



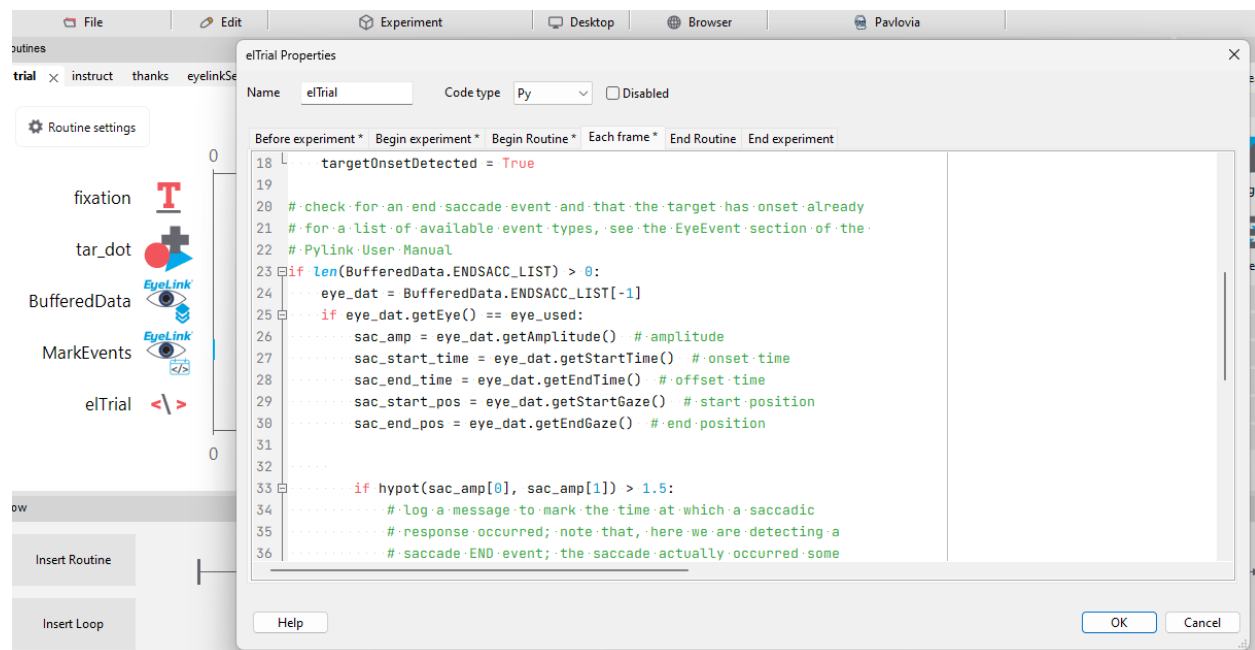
The Buffered Data component will retrieve from the EyeLink data buffer and store in a local PsychoPy buffer any of the event data types specified in its “Event Data Types to Include” property. These event types can include the following, with items in the list of types to include separated by commas.

STARTBLINK -- start of blink events

ENDBLINK -- end of blink events
STARTSACC -- start of saccade events
ENDSACC -- end of saccade events
STARTFIX -- start of fixation events
ENDFIX -- end of fixation events
FIXUPDATE -- fixation update events
MESSAGEEVENT -- message events
BUTTONEVENT -- button events
INPUTEVENT -- input events

Documentation regarding the structure of each event data type can be found in the Pylink User Manual in the section “Pylink Module”.

During the routine, between the start time and stop time of the Buffered Data component, the script will retrieve all events of the types included in “Event Data Types to Include”. Each type will be stored in a separate list as part of the data structure of the component. The name of the list in the component data structure will be [EVENTTYPE]_LIST. For example, if the component is called BufferedData, and “Event Data Types to Include” includes ENDSACC, then all end saccade events that occur between the start and stop time of the component will be logged to a variable called BufferedData.ENDSACC_LIST. Each new event will be appended to the end of the list, so the latest event of a given type can always be accessed using the index [-1] (which is the index for the last item in a Python list). See the EyeLinkSaccade_Plugin example for an illustration of how to access event data – it shows how to grab end saccade events and end a trial when a saccade lands in a specified region. The event data are accessed/checked in the Each Frame tab of the code component in the trial routine.



As data are collected, if the length of the event data type list exceeds the value of “Maximum Local Event List Length” then the oldest item in the list will be removed as each new item is added to ensure the list does not exceed a certain length.

If “Include Spanned End Events” is unchecked, then the component will not include 1) end saccade, end fixation, and end blink events that occur between the component’s start/stop times but whose start was before the component’s start time and 2) end saccade, end fixation, and end blink events that occur after the component’s stop time but whose start time was between the component’s start/stop times. If this property is checked then those events will be included. These events can be thought of as spanning the component’s start or stop time boundaries

Sample data can also be retrieved by the Buffered Data component (if “Include Sample Data” is checked), but unless it is critical that the experimental script retrieve all samples then often the Latest Sample component is a better choice for retrieving sample data – it provides a simple way to access the most recent eye tracking sample.

The image shows a software dialog box titled "BufferedData Properties". It has three tabs: "Basic", "Data", and "Testing", with "Basic" currently selected. The dialog contains several input fields and checkboxes. The "Name" field is set to "BufferedData". The "Start" section has a dropdown menu set to "time (s)" with a value of ".7", and an empty "Expected start (s)" field below it. The "Stop" section has a dropdown menu set to "duration (s)" with a value of "3.0", and an empty "Expected duration (s)" field below it. The "Event Data Types to Include" field contains the text "ENDSACC". The "Maximum Local Buffer Size" field is set to "100". There are two checkboxes at the bottom: "Include Spanned End Events" and "Include Sample Data", both of which are currently unchecked. At the bottom of the dialog are three buttons: "Help", "OK", and "Cancel".

Property	Value
Name	BufferedData
Start	time (s) .7
Expected start (s)	
Stop	duration (s) 3.0
Expected duration (s)	
Event Data Types to Include	ENDSACC
Maximum Local Buffer Size	100
Include Spanned End Events	☐
Include Sample Data	☐

Properties of Buffered Data:

Start	Time in the routine to start checking for the eye tracking data
Stop	Time in the routine to stop checking for the eye tracking data
Event Data Types to Include	Include any of the following, separated by commas: STARTBLINK -- start of blink events ENDBLINK -- end of blink events STARTSACC -- start of saccade events ENDSACC -- end of saccade events STARTFIX -- start of fixation events ENDFIX -- end of fixation events FIXUPDATE -- fixation update events MESSAGEEVENT -- message events BUTTONEVENT -- button events INPUTEVENT -- input events These events will be saved in a variable of the component which will be called EVENTNAME_LIST, e.g., BufferedData.ENDSACC_LIST will be the list of end saccade events for a component named BufferedData
Maximum Local Event List Length	Maximum number of events to be stored in local PsychoPy list for each event data type
Include Spanned Eye Events	If unchecked, then the component will not include 1) end saccade, end fixation, and end blink events that occur between the component's start/stop times but whose start was before the component's start time and 2) end saccade, end fixation, and end blink events that occur after the component's stop time but whose start time was between the component's start/stop times. If checked then those events will be included.
Include Sample Data	Whether to retrieve sample data from the EyeLink link data buffer
Maximum Local Sample List Length	Maximum number of samples to be stored in local PsychoPy list. Note, this property will only be available if "Include Sample Data" is checked

3 Example walkthrough

In this section, we will walk through how the EyeLink Plugin components work in the EyeLinkPicture_Plugin example, which illustrates how to perform an experiment in which each trial presents a simple picture to the participant and waits for a response. The goal here is to illustrate how the EyeLink Plugin components can be added to existing PsychoPy experiments to implement the EyeLink integration. At the end of the walkthrough we will discuss online gaze data access options, MRI synchronization, and an alternative eye tracker recording structure

that can allow for continuous eye tracker recording through a block of trials. Please see section 1.2 of this manual for a list and description of the examples provided with the plugin.

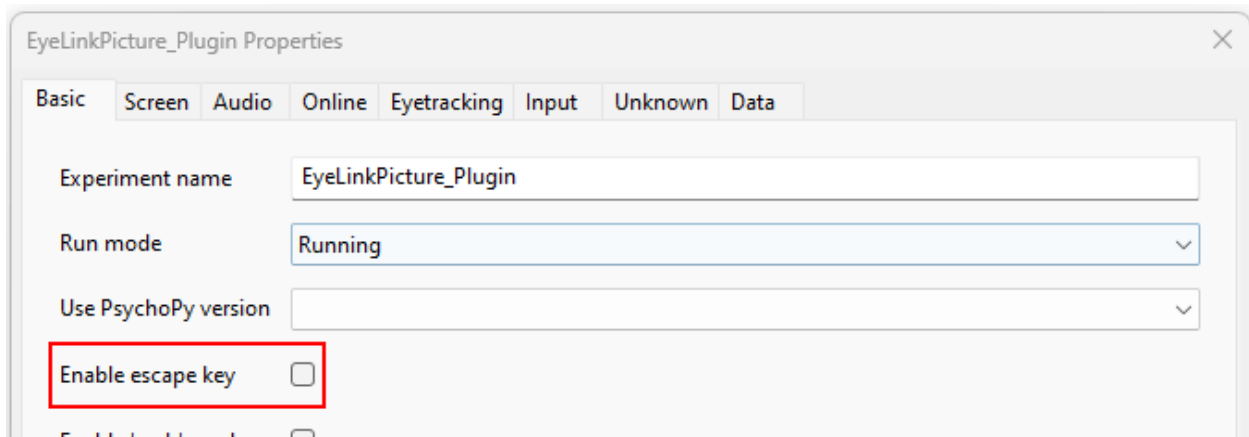
3.1 Experiment configuration

Before adding any of the EyeLink Plugin components, it is important to ensure the experiment has been configured optimally.

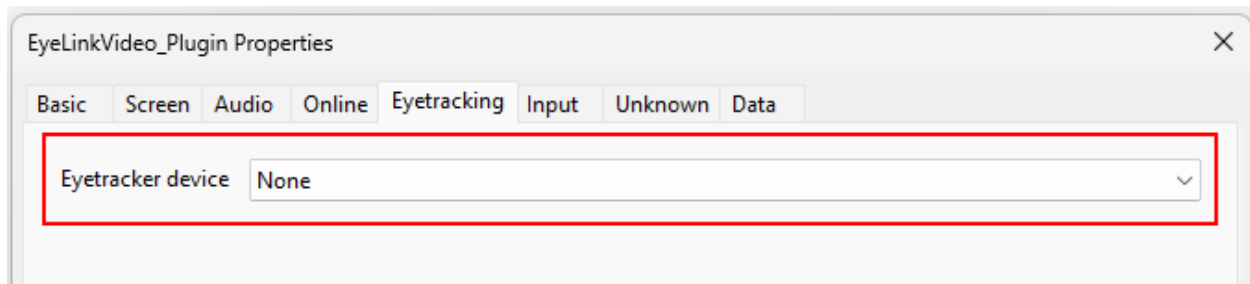
Experiment Settings



In the Basic tab, the “Enable escape key” property should be unchecked. The Esc key can be used during eye tracker calibration/validation/drift checks, so it is important to uncheck this property to ensure pressing Esc will not also end the experimental run. You can enable “Allow Ctrl-C Quit Key to be Used” property of Mark Events components to enable the Ctrl-C key combination to be used to end an experimental run early (and also properly retrieve the EDF from the Host PC and shut down connection to the Host PC). See sections 2.6 and 3.8 for further discussion of this property.



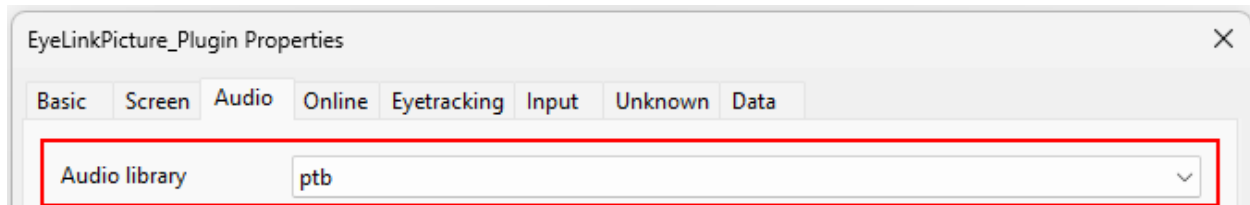
In the Eyetracking tab, “Eyetracker device” property should be set to None. The EyeLink Plugin will handle all eye tracker integration, so the built-in eye tracker options included with PsychoPy should not be used.



Note, the built-in PsychoPy eye tracker options provide only very limited EyeLink integration. Most critically, the built-in eye tracker options do not provide a mechanism for marking the occurrence of experimental events in the EyeLink data file (EDF). This means, when using the built-in eye tracker options of PsychoPy, there will be no way to use its eye tracker components to send event-markers to the data file to help you in analysis (so you can know what was being presented to the participant at different points in the experiment).

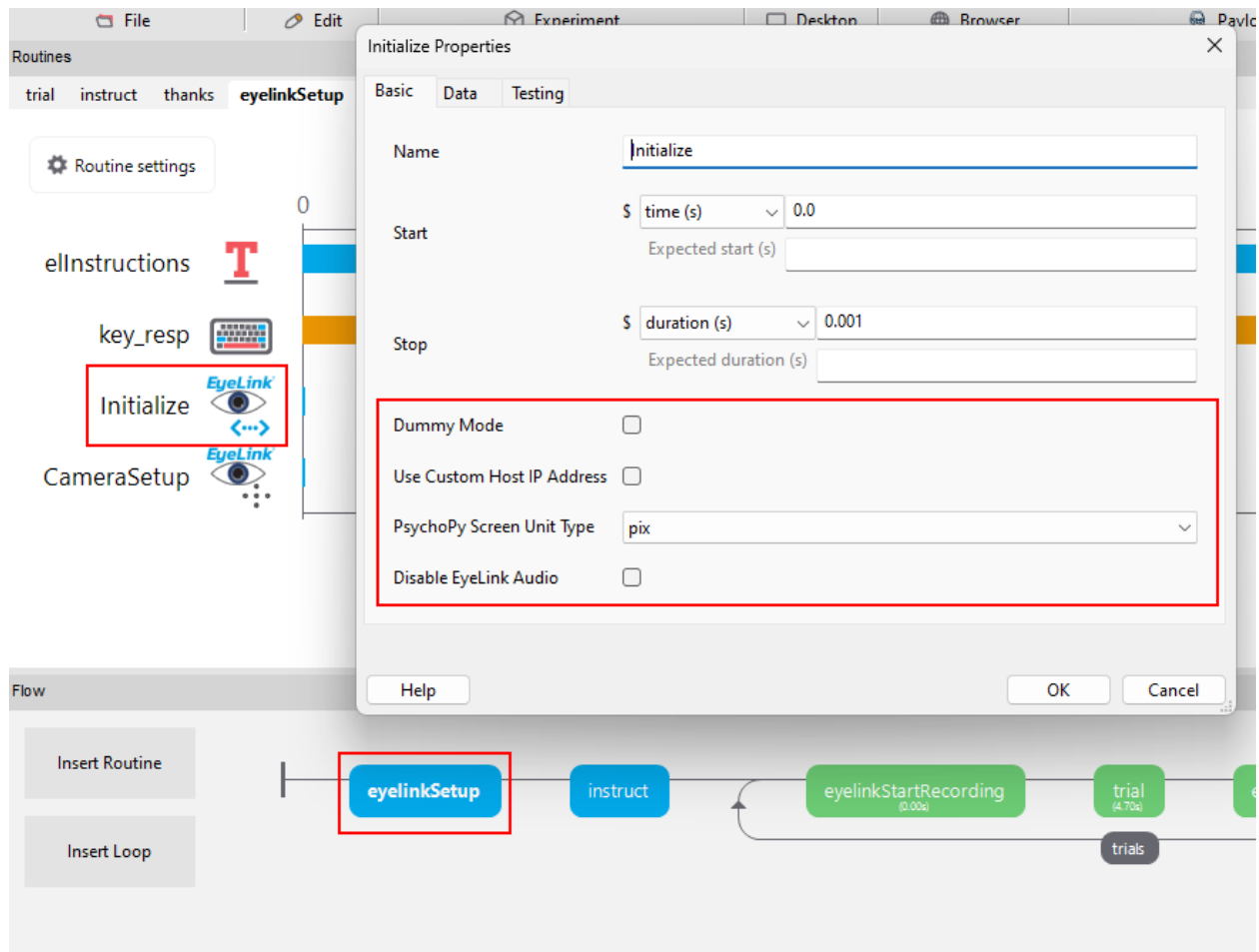
Fortunately, the EyeLink Plugin for PsychoPy discussed here not only allows for simple event marking, it also allows for stimulus, response, and interest area data logging, allowing for visualization of the data in Data Viewer (see sections 3.7 and 2.6 for information on how this is implemented). Additionally, the built-in PsychoPy eye tracker functions do not allow for drift checks and many other aspects of EyeLink API usage. The EyeLink Plugin for PsychoPy discussed here allows for fully-featured integration with EyeLink systems.

In the Audio tab, the “Audio library” property is set to ptb. This audio library is strongly recommended by the PsychoPy developers and is also best for use in EyeLink experiments.



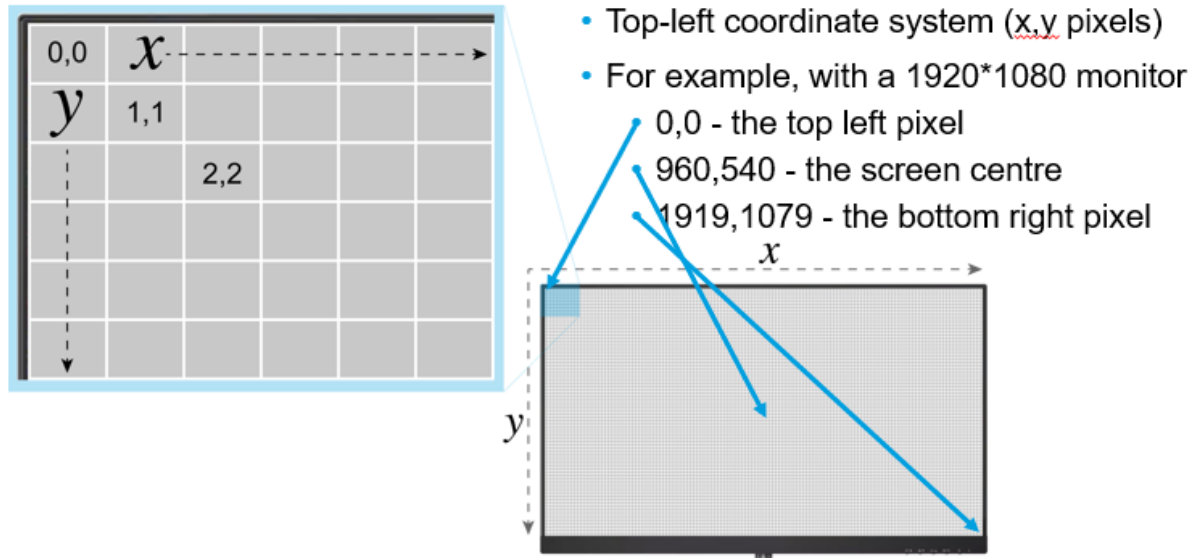
3.2 Initialize EyeLink connection

The first EyeLink component in the example is the Initialize Connection component, which is part of a routine called `eyelinkSetup` at the very beginning of the experiment. with a text component that presents some example eye tracker usage instructions at the beginning of the experiment and is placed below the routine’s stimulus and response components.



To test the experiment when you don't have a Host PC / EyeLink system with you, you can enable "Dummy Mode". This will make it so the experiment bypasses the EyeLink functionality. The "PsychoPy Screen Unit Type" property has been set to pix to match with the screen unit type of the stimulus components used in the experiment. This ensures proper conversion from the experiment's screen unit type to the EyeLink coordinate system, where the top-left corner of the screen is point 0,0, where the X coordinate increases as the eye moves to the right from the left edge of the screen and where the Y coordinate increases as the eye moves down from the top edge of the screen.

EyeLink coordinate system:



The eye data in the EyeLink data file (EDF) will use the EyeLink coordinate system, and setting the PsychoPy Screen Unit Type correctly will make it so interest area and screen stimuli data are correctly logged in the EyeLink coordinate system for analysis.

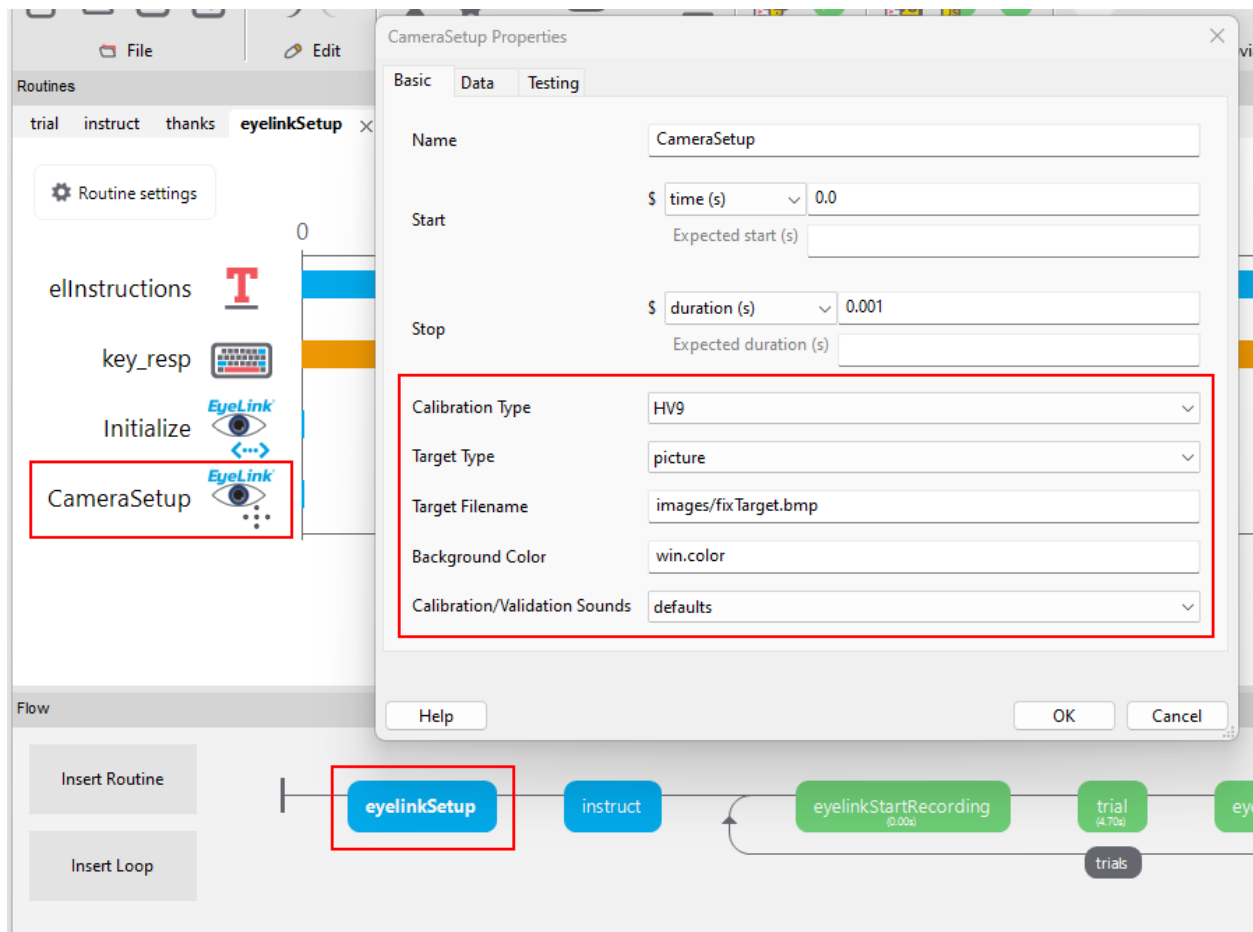
The examples connect to the default 100.1.1.1 Host PC IP address. If the Host PC IP Address has been changed in your setup (which, note, is typically not the case) you can check “Use Custom Host IP Address” and then enter your custom IP in the “Host IP Address” property that will appear.

3.3 Perform camera setup / eye tracker calibration

Before beginning the experiment, the experimenter will set up the participant and eye tracker and then will calibrate/validate the tracker. These procedures will all be performed while in the camera setup mode. For details on these procedures, please see the [Video Tutorials](#) on system setup and usage. Once eye tracker calibration/validation are complete, the experimenter can press O on either keyboard (or click Output/Record on the Host PC) to continue with the script. The Camera Setup component is below the Initialize Connection component, to ensure it is only executed after a connection with the EyeLink Host PC has been established.

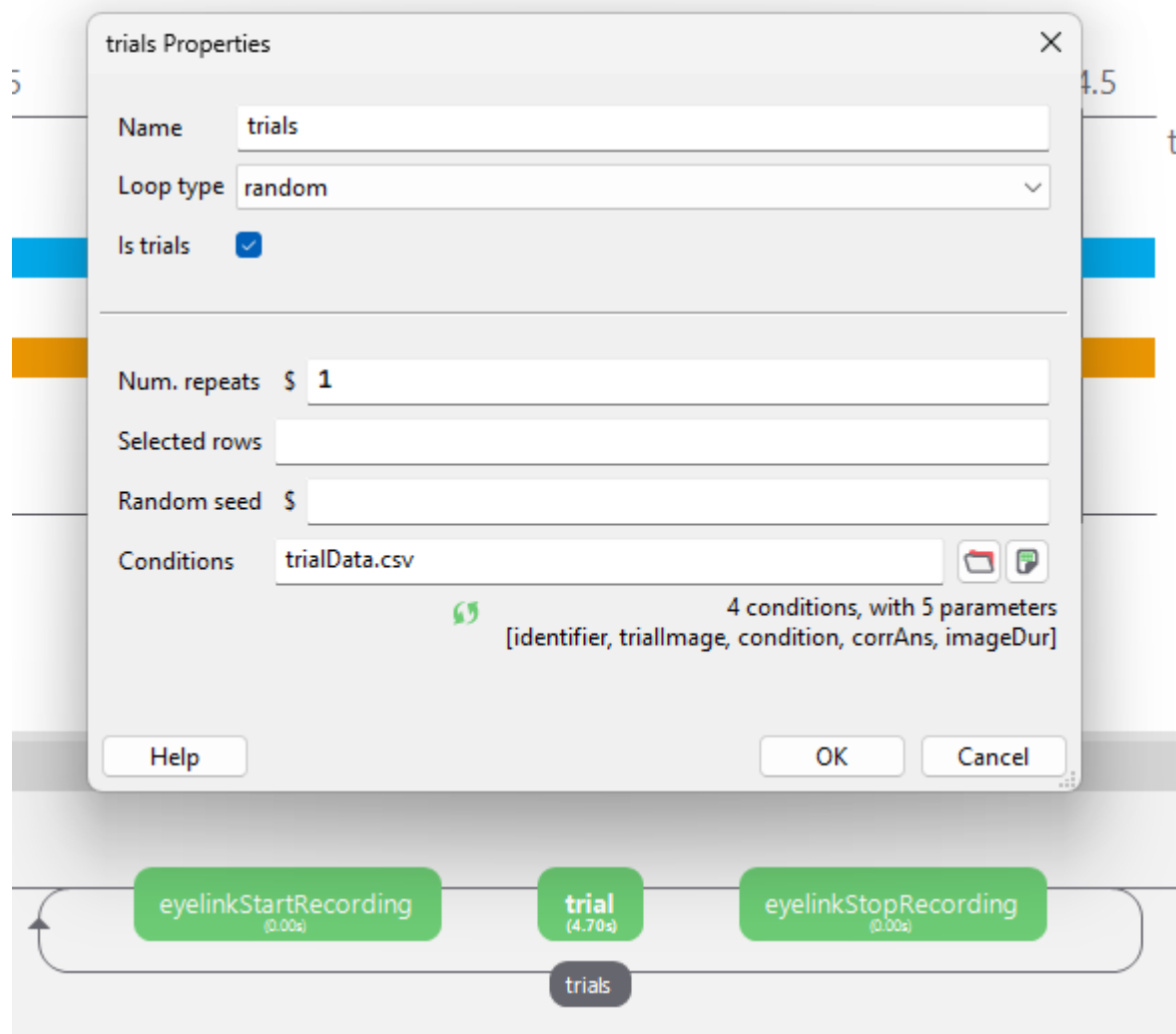
The properties of the Camera Setup allow you to configure the eye tracker calibration/validation. Here, the “Calibration Type” is set to 9 point horizontal/vertical (HV9), the calibration/validation “Target Type” is a picture, and the “Target Filename” points to an image file called fixTarget.bmp that is located in the project’s images subfolder. The “Background Color” property is set to win.color to match the background color of the stimuli in the project. This way, the pupil size doesn’t change too much from setup/calibration to trial stimulus presentation. Eye

tracker accuracy is optimized when the calibration is performed under similar luminance as the luminance that will be used during the experiment. Finally, the sounds are set to the defaults, which uses three wav files ('qbeep.wav', 'type.wav', and 'error.wav') for the target sound, good sound, and error sound that are presented during eye tracker calibration/validation.



3.4 trials loop organization

The loop that handles the presentation of the different trials is much like that of any other PsychoPy experiment. It points to a Conditions file called trialData.csv

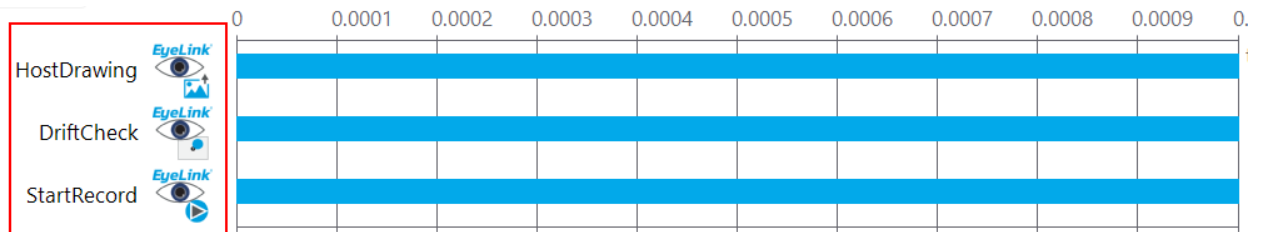


identifier	trialImage	condition	corrAns
1	images/img_1.jpg	color	m
2	images/img_2.jpg	color	z
3	images/img_3.jpg	blackAndWhite	m
4	images/img_4.jpg	blackAndWhite	z

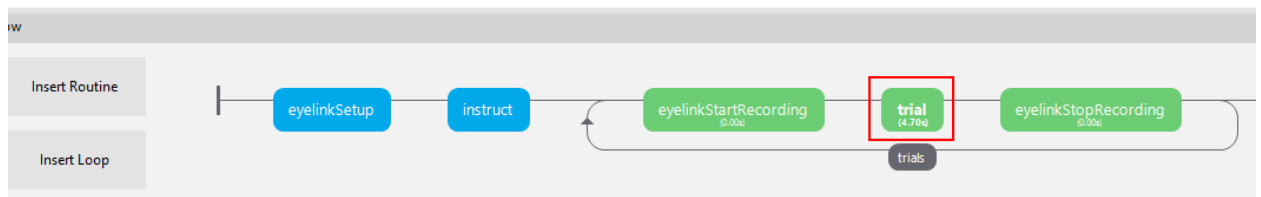
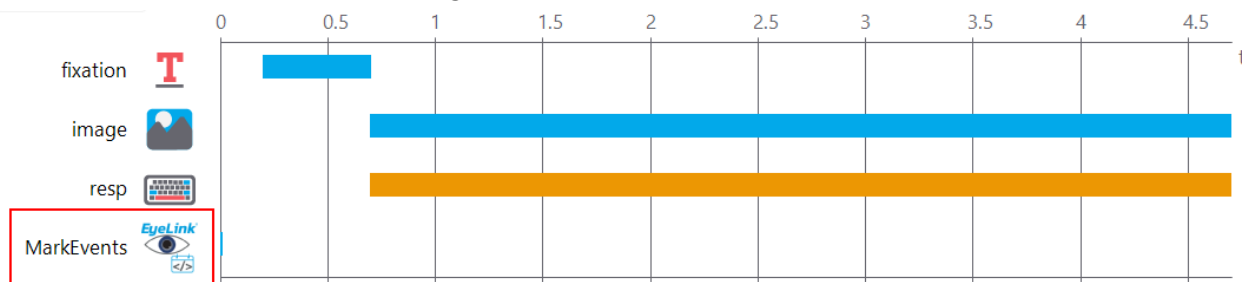
The identifier column provides a trial ID (just to potentially help in analysis), the triallImage column points to the image to be shown on the trial, the condition column codes the image type condition, and the corrAns column is checked in determining accuracy.

The notable differences in the trials loop, which make this a proper EyeLink experiment, are as follows. These differences will be discussed in more detail in sections 3.5 to 3.9.

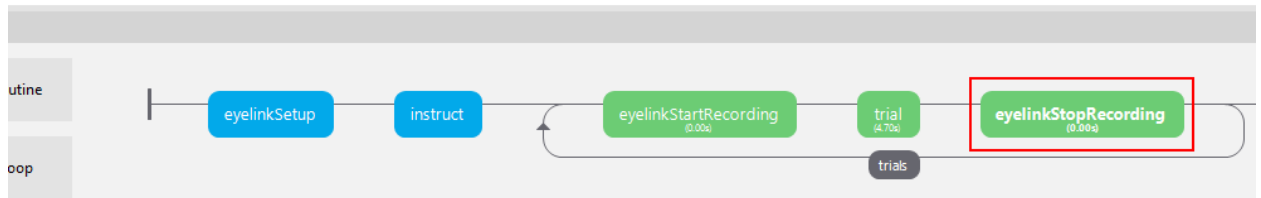
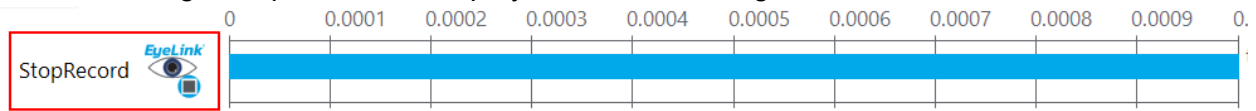
- a) It includes a routine at the beginning of the trials loop, called `eyelinkStartRecording`, which contains three EyeLink plugin components. These components perform pre-trial EyeLink functions, including Host PC backdrop drawing, performing a drift check, and starting eye tracker recording.



- b) The trial routine includes a Mark Events component at the bottom of the routine that handles all experimental stimulus and response event marking, sends special Data Viewer messages to allow stimulus visualization in Data Viewer, and logs all trial variable data that might be needed for analysis, including behavioral response data, and trial condition information from the Conditions `trialData.csv` file.



- c) It includes a routine at the end of the trials loop, called `eyelinkStopRecording` that contains a single `Stop Record` to stop eye tracker recording at the end of each trial.

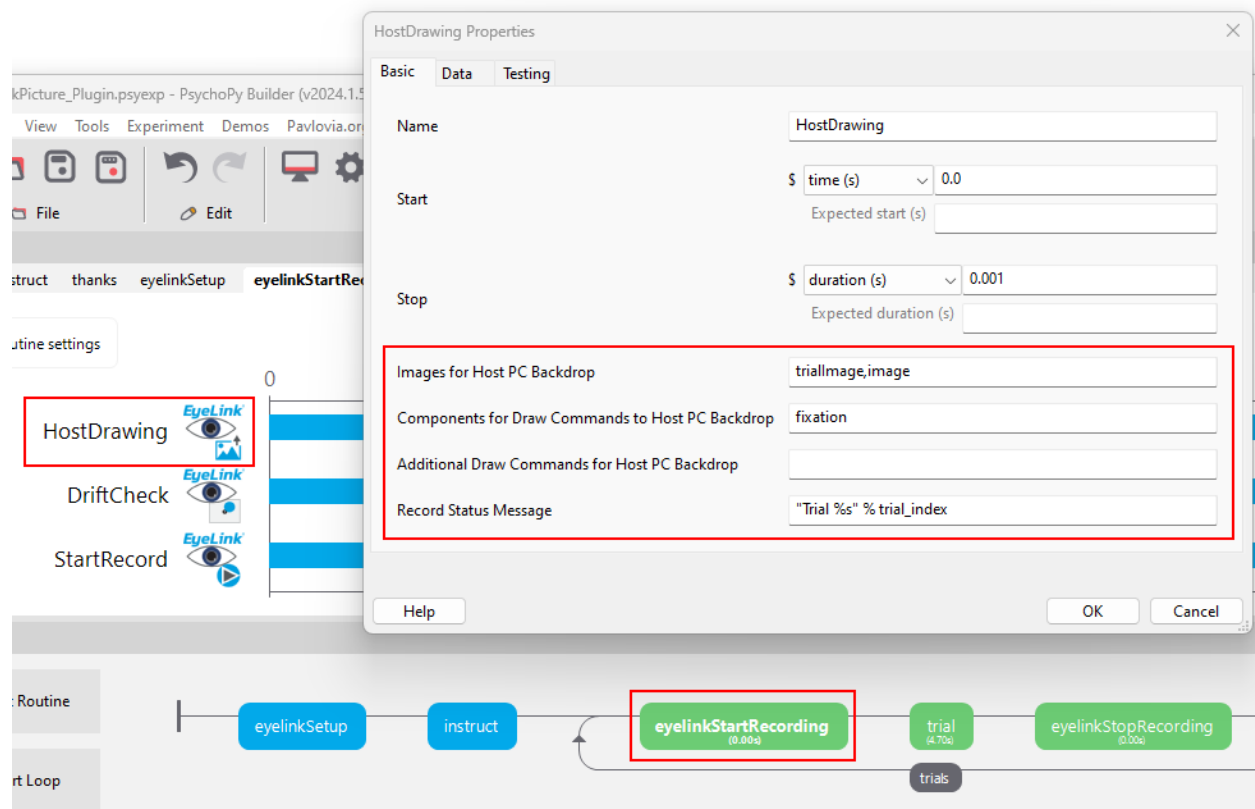


- d) The trial routine does not have any stimulus events before 200 msec into the sequence. In order to ensure optimal stimulus presentation timing, the routine immediately after the routine that starts eye tracker recording should not have any events before 200 msec.

3.5 Draw trial stimuli on the Host PC backdrop

The first/top-most component in the `eyelinkStartRecording` routine populates the Host PC backdrop with the upcoming trial's image stimulus and a fixation cross marker (a red box) so that the current gaze position can be visualized on top of what the participant is viewing during the trial by the experimenter. It also sends a message to the bottom-right corner of the Host PC to tell the experimenter the current trial number.

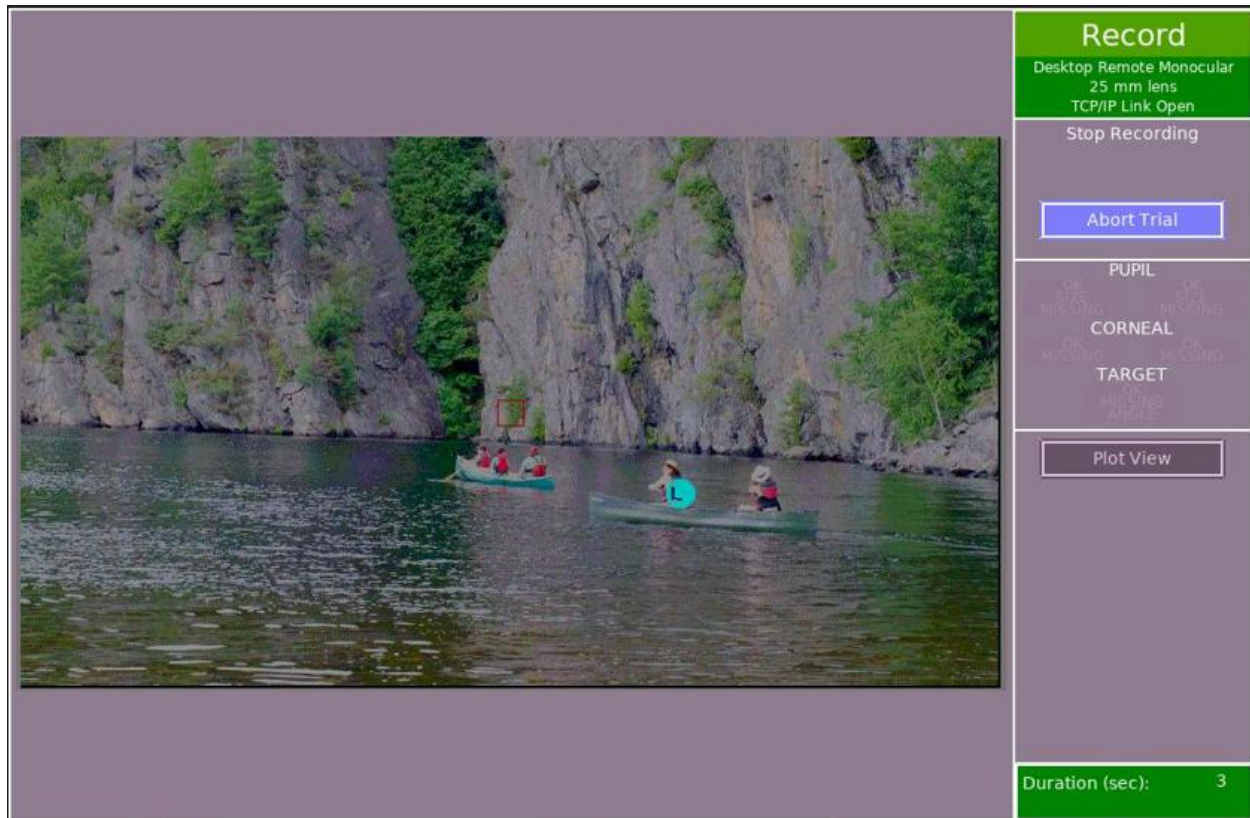
These functions cannot be performed during eye tracking (i.e., while we are recording eye data), so this component is at the top of the routine to ensure it is performed before eye tracker recording starts. Since the routine contains a Drift Check, too, it is above the Drift Check in the routine as well -- it is important that Host Drawing happen before the Drift Check (not the other way around).



The “Images for Host PC Backdrop” sends the trial’s image file to the backdrop by specifying the variable from the trialData.csv Conditions file followed by the component name that handles the actual image presentation with the variable name and the component name separated by a comma.

trialImage,image

The image will be shown on the Host PC at the same location/size as it is presented to the participant.



If multiple images need to be sent to the Host PC backdrop, the variable name/component name pairs should be separated by semicolons. E.g.,

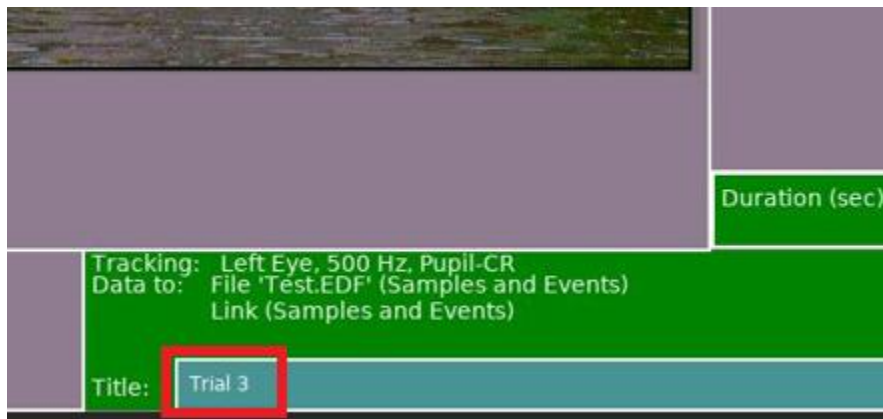
trialImage1,image1; trialImage2,image2

The “Components for Draw Commands to Host PC Backdrop” can be used for non-image components to tell the Host PC to draw a red rectangle on its backdrop representing the location/size of the stimulus. Because the fixation component has been included for this property you can see a small, red rectangle representing the fixation cross on the backdrop of the Host PC during recording (it is just above the left-most canoe in the screen capture above). If multiple components are included for this property then they should be separated by commas.

The Record Status Message can be used to send a text string to the experimenter, often to provide an update about experiment progress. In this case, the property is set up to use a little Python code to build a string including the text Trial followed by the trial number, where trial number comes from the trial_index variable.

“Trial %s” % trial_index

This text-string will be shown to the experimenter during recording at the bottom-right corner of the Host PC screen:



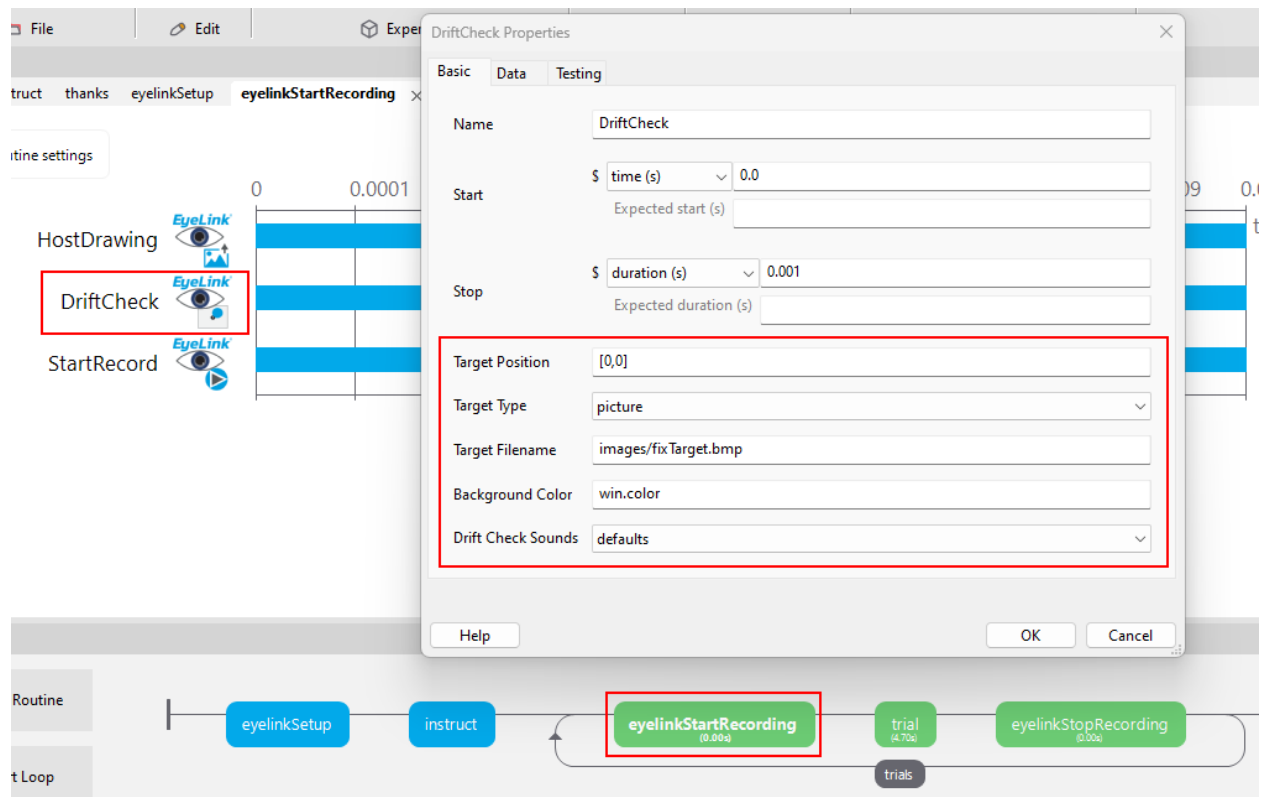
3.6 Perform a drift check

After the Host Drawing is performed, the Drift Check component calls for a Drift Check to be performed. At this point, the stimulus Display PC will present a single target on a blank background. The participant needs to look at the target and then, while gaze position is steady on the target the experimenter (or the participant) can confirm the gaze position by pressing the spacebar or Enter on either keyboard or by clicking the Accept Fixation button on the Host PC.

Drift checks are optional, but they serve a few purposes:

- a) They provide a mechanism for controlling where the participant is looking when the trial starts.
- 2) They allow you to check the accuracy of the calibration model between experimental trials.
- 3) They allow you to recalibrate as needed. Rather than pressing spacebar or Enter to accept the drift check, if the accuracy looks unacceptable then you can press Esc to go back to Camera Setup mode and recalibrate/revalidate. When you are finished with recalibration/revalidation you can press O or click Output/Record on the Host PC to pick up at the drift check where you left off.

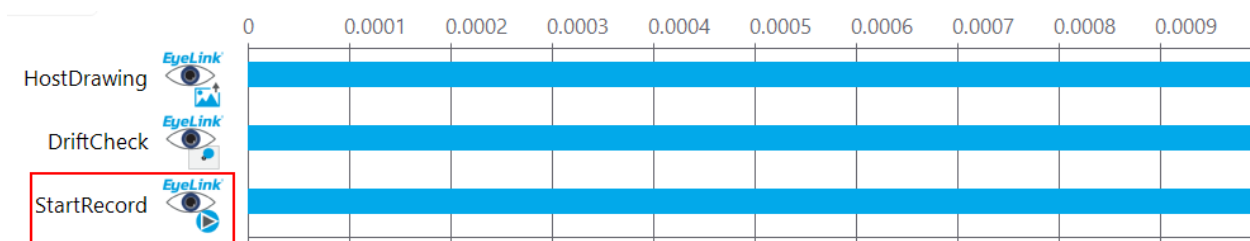
How often to perform drift checks depends on the goals of the study. In some experiments, researchers perform drift checks on each trial. In others, a drift check may be performed every N trials. In others, not at all. Do you need to control the gaze position when the trial starts? If so, you can perform a drift check to make sure gaze position is on the target region before starting. If not, then you could skip them or perform them at the beginning of each block.



The properties of the Drift Check allow configure the eye tracker drift check's appearance and the sounds that may be played during the procedure. Here, the "Target Position" is set to the center of the screen [0,0], the drift check "Target Type" is a picture, and the "Target Filename" points to an image file called fixTarget.bmp that is located in the project's images subfolder. The "Background Color" property is set to win.color to match the background color of the stimuli in the project. Finally, the sounds are set to the defaults, which uses three wav files ('qbeep.wav', 'type.wav', and 'error.wav') for the target sound, good sound, and error sound that are presented during eye tracker drift check. These audio files can be copied from any of the example experiments that come with the plugin over to the main folder of your experiment if you would like to use them, as well.

3.7 Start eye tracker recording

Following the drift check, the Start Record component starts eye tracker recording.

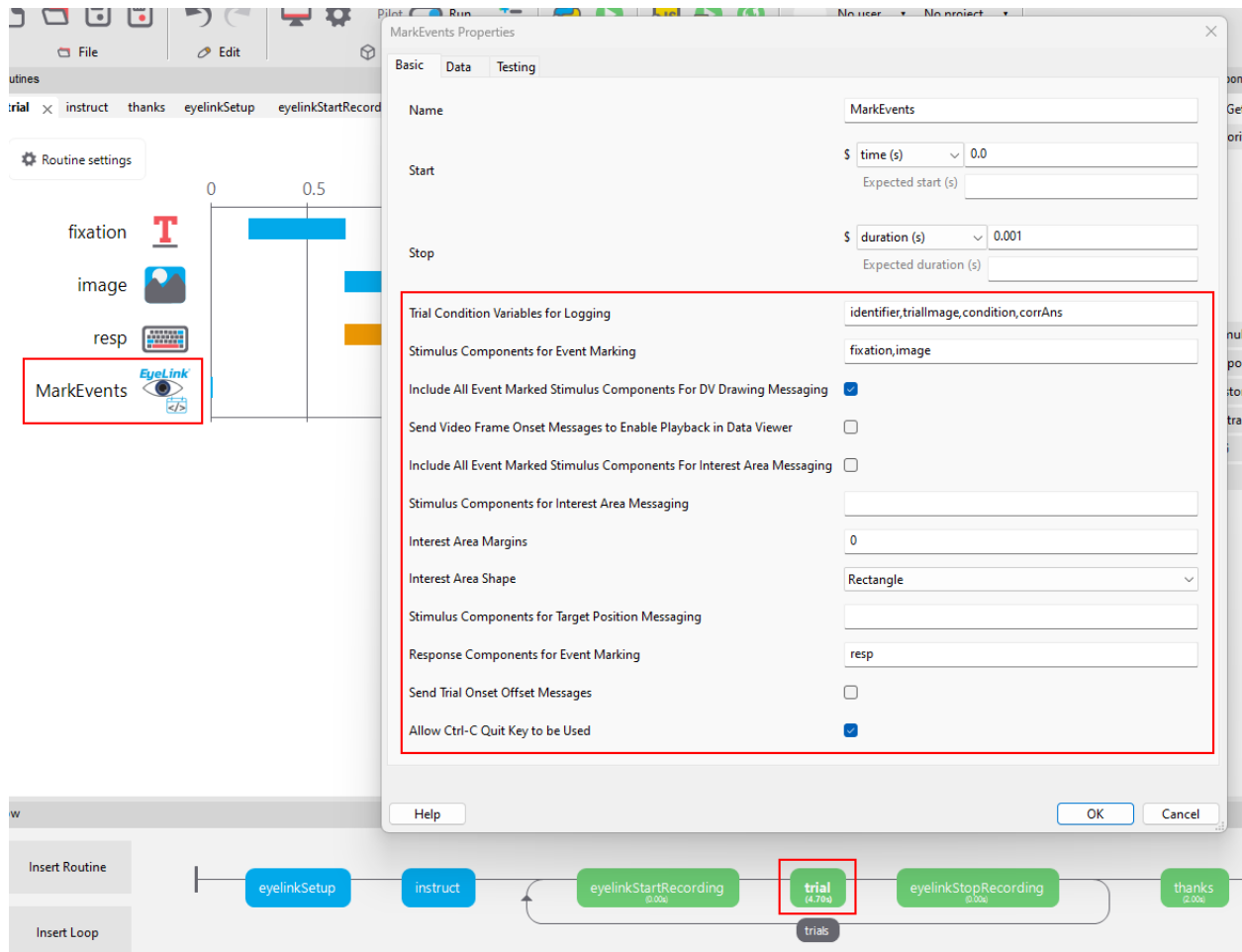


The EyeLinkPicture_Plugin example is set up so as to start recording at the beginning of each trial and stop recording at the the end of each trial. Doing so allows for the Host backdrop to be updated and for a drift check to be performed between trials. If the experimental paradigm requires continuous eye tracker recording through a block of trials then please see the EyeLinkMRIdemo_Plugin example as well as the discussions in sections 2.5, 2.6, and 3.11 of this manual.

The Start Record component does not have any properties to configure.

3.8 Mark and log experimental events

The Mark Events component that is in the trial sequence serves the critical role of marking experimental events in the EyeLink data file and logging lots of useful information about the trial's events that will help during data analysis.



The “Trial Condition Variables for Logging” property will handle sending special Trial Variable messages to make the values of any variables that are included be logged on each trial. Here, all columns from the trialData.csv Conditions file that is used by the trials loop are included. The messages that are sent to log this information will be formatted so that these variables become Trial Variables in Data Viewer, which makes it so they can be included in any output data reports you create in Data Viewer and so that you can sort the data by these values within the viewing session. This information also will be available in any other data analysis pipeline, as well.

The “Stimulus Components for Event Marking” property includes all the stimulus components in the routine, and the components are separated by commas. Including a stimulus component for this property makes it so that the script will automatically send stimulus onset and offset messages to the EyeLink data file (EDF) for that component. The Message format in the EDF will be the name of the component followed by _ONSET or _OFFSET. E.g., here, we will see the messages image_ONSET, image_OFFSET, fixation_ONSET, and fixation_OFFSET.

The property “Include All Event Marked Stimulus Components for DV Drawing Messaging” is checked. This makes it the script will send special Data Viewer draw/image load commands for all stimulus components in the “Stimulus Components for Event Marking” list so that we can see the stimuli in Data Viewer’s various visualization options. Images and videos will be shown as they were presented to the participant in Data Viewer. Other stimulus components will be represented by simple shapes in Data Viewer’s visualization options.

The “Response Components for Event Marking” property includes the only response component in the routine – resp. Any response components included in this property will have their occurrence marked with a message to the EDF. The Message format in the EDF will be the name of the component followed by _EVENT or _OFFSET. E.g., for the resp component called resp, its message will be resp_EVENT. The components in the list should be separated by commas.

Since this experiment starts recording at the beginning of each trial and stops recording at the end of each trial it does not (and should not) have its “Send Trial Onset Offset Messages” property checked. That property should only be checked in experiments that start eye tracker recording before a block of trials and stop recording at the end of the block of trials so that the script will send messages to mark the start and end of trials, which allows Data Viewer to divide that continuous eye tracker recording into multiple trials. See the EyeLinkMRIdemo_Plugin example for an illustration of continuous recording through a block of trials and see sections 2.5 and 3.11 of this document for further discussion.

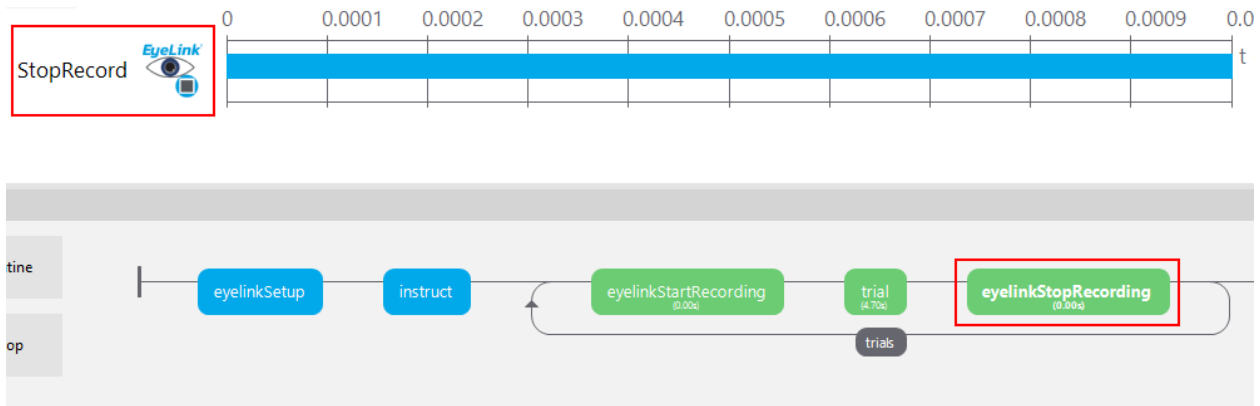
The property “Allow Ctrl-C Quit Key to be Used” makes it so that during the execution of this routine, the Ctrl-C key combination can be pressed on the Display PC to end the experimental session early. Doing so will send commands to the Host PC to close the current EDF file, transfer a copy of the EDF to the results folder of the experimental project directory (each participant’s EDF file will be saved in a subfolder within the experiment’s results folder) and to shut down the connection to the EyeLink Host PC.

The EyeLinkPicture_Plugin example does not utilize all the properties of the Mark Events component. Mark Events can also be used to auto-log interest area data that can help in analysis -- this example does not include interest area messaging, because for image studies like this it is often easiest/best to create the image-specific interest areas in Data Viewer. See the Interest Area videos in the [Data Viewer Tutorial Series](#) for more information on how interest areas can be create/applied to similar trials across participants in Data Viewer. See also the EyeLinkSaccade_Plugin, EyeLinkFixWindowFastSamples_Plugin, and EyeLinkPursuit_Plugin examples for illustrations of Mark Events components that include interest area messaging.

Additionally, it does not use the “Stimulus Components for Target Position Messaging” property, which will make it so that the movement of any stimulus components will be logged via custom Target Position messages that allow you to visualize the stimulus trajectory and output the

target position data alongside the eye data in Data Viewer's output reports (this is especially useful for smooth pursuit studies – see the EyeLinkPursuit_Plugin example for an illustration).

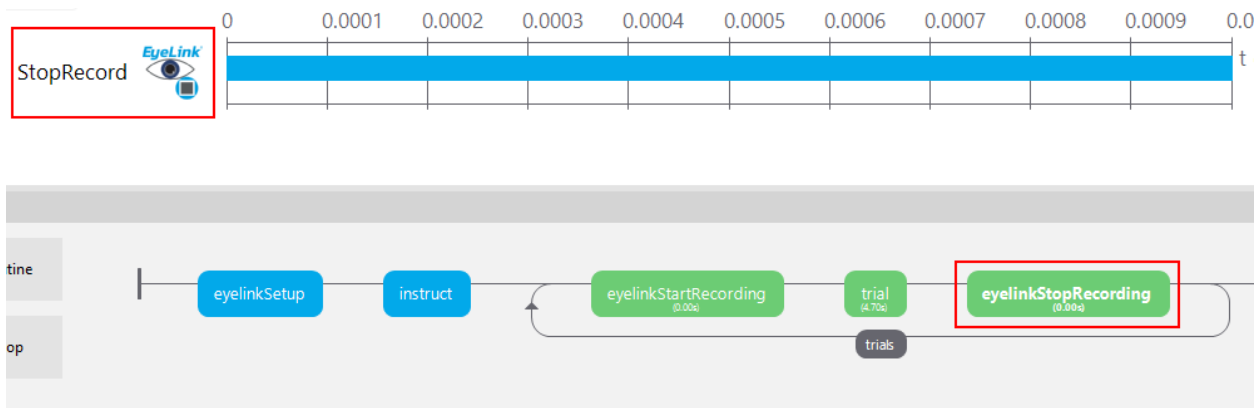
Finally, it does not show how to use the “Send Video Frame Onset Messages to Enable Playback in Data Viewer” property to send messages for each video frame onset, which allows Data Viewer to play the video back with gaze position overlaid in its Animation Playback view (see the EyeLinkVideo_Plugin example for an illustration).



3.9 Stop eye tracker recording

At the end of the trials loop the Stop Record component in the eyelinkStopRecording routine stops eye tracker data recording. There are no properties to configure for the Stop Record component.

Note, this is the last EyeLink Plugin component in the EyeLinkPicture_Plugin example. The Initialize Connection component (in the eyelinkSetup routine) handles retrieval of the EDF from the Host PC and shutting down



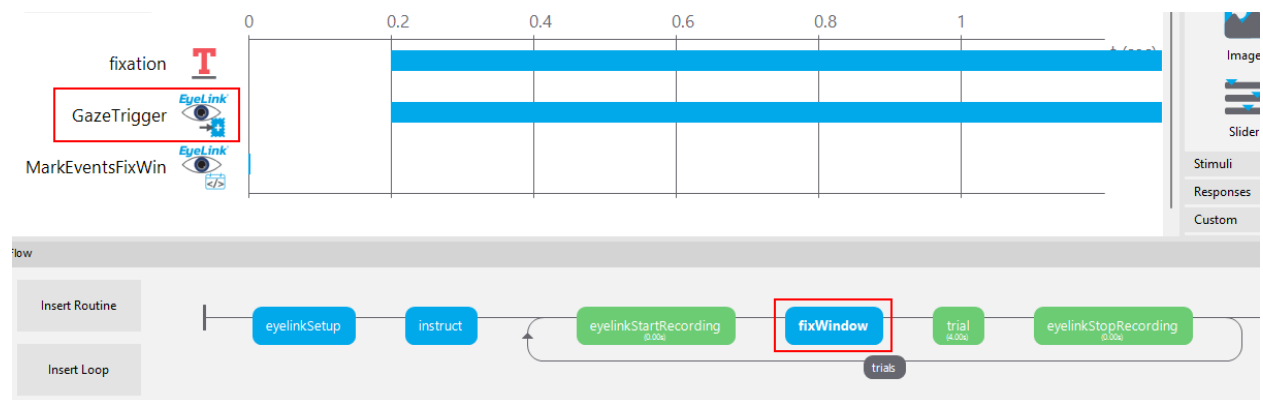
3.10 Online eye data access options

The EyeLinkPicture_Plugin example does not illustrate how to access eye data from the eye tracker in real-time, but the EyeLink Plugin includes several components that allow online data access.

Note, though, that unless the experiment requires eye data online, like for gaze-contingent paradigms or other cases where the experimental script must have access to the eye data during the run, there is typically no need to access that data online. EyeLink systems use a dedicated Host PC to handle all eye data recording so the Display PC doesn't have to be responsible for executing the eye tracking processes (leaving its resources free for stimulus presentation).

If the experimental paradigm does require eye data access online, then there are a few examples illustrating how to do so.

The EyeLinkFixationWindowFastSamples_Plugin example uses a Gaze Trigger component to wait for the participant to look at a target before advancing to trial stimulus presentation.



The EyeLinkGCWindow_Plugin example uses a Latest Sample component to grab the most recent sample from the Host PC and then uses a simple code component to update a gaze-contingent moving window (i.e., to make the window follow the eye position).



Finally, the EyeLinkSaccade_Plugin example uses a Buffered Data component to grab all end saccade events and then uses a simple code component to check whether a saccade end events is within a certain distance from the target in order to determine whether to end the routine/trial.

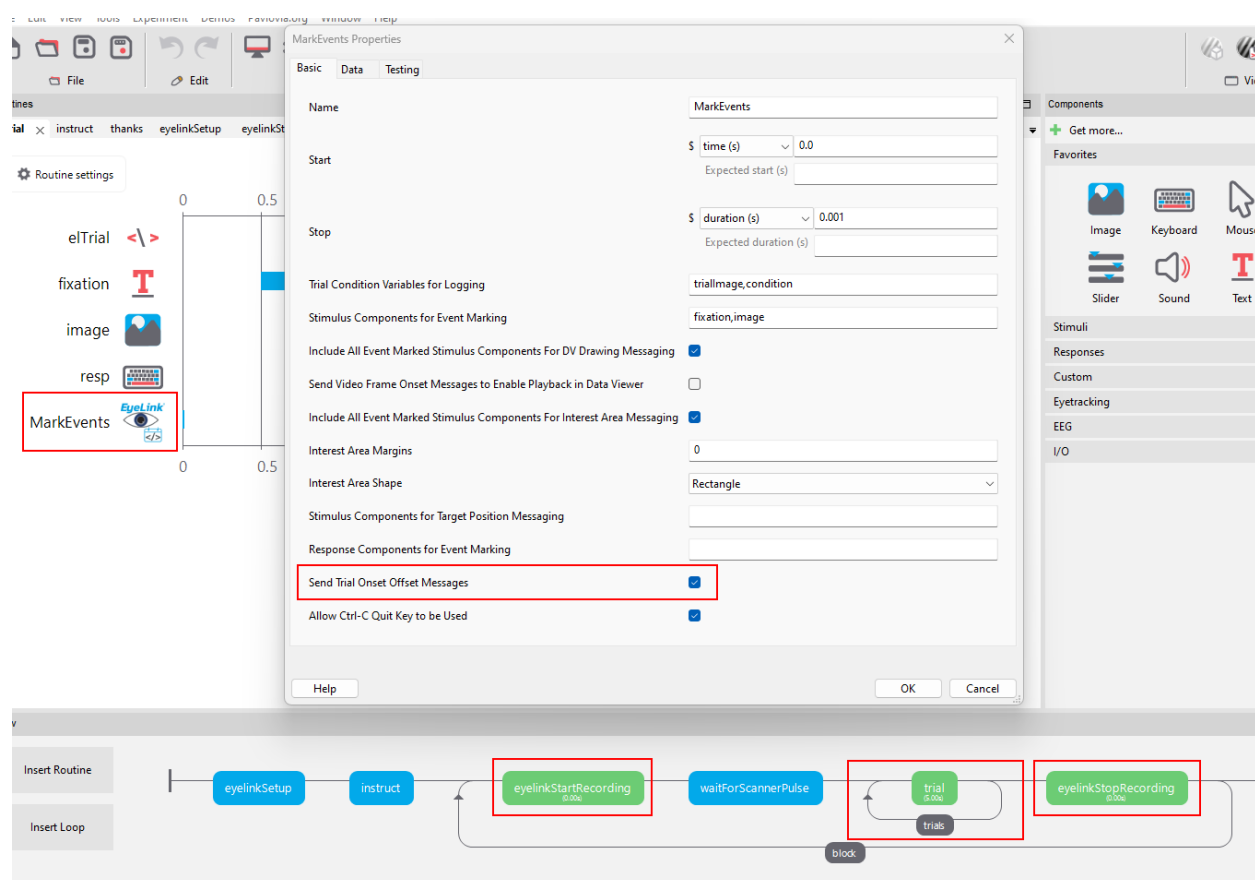


3.11 MRI Synchronization and continuous block recording

As mentioned in other sections of this document (see 2.5, 2.6, and 3.7), a typical EyeLink experiment would start eye tracker recording before each trial and then stop eye tracker

recording at the end of each trial. Doing so allows for a chance to update the Host PC backdrop via the Host Drawing component and to perform a drift check with the Drift Check component (these things cannot be performed while recording eye data). In some situations, though, it may be more important to record data continuously throughout a block of trials. In an MRI environment, for example, the trials of a block/scan cycle may all need to be of a certain duration to ensure alignment with regular scanner signals. In these types of cases, the timing overhead of starting/stopping eye tracker recording on each trial may be disruptive to the experimental design. Fortunately, it is easy to perform continuous eye tracker recording while also ensuring that the recording can still be properly divided into trials.

The EyeLinkMRIdemo_Plugin example shows how to implement this continuous recording structure, and there are really only two things that accomplish this. First, the Start Record/Stop Record components (in the eyelinkStartRecording/eyelinkStopRecording routines) are positioned before and after the trials loop (so that multiple iterations of the trials loop are executed within one continuous eye tracker recording). And next, the Mark Events component in the trial routine has its “Sent Trial Onset Offset Messages” property checked.



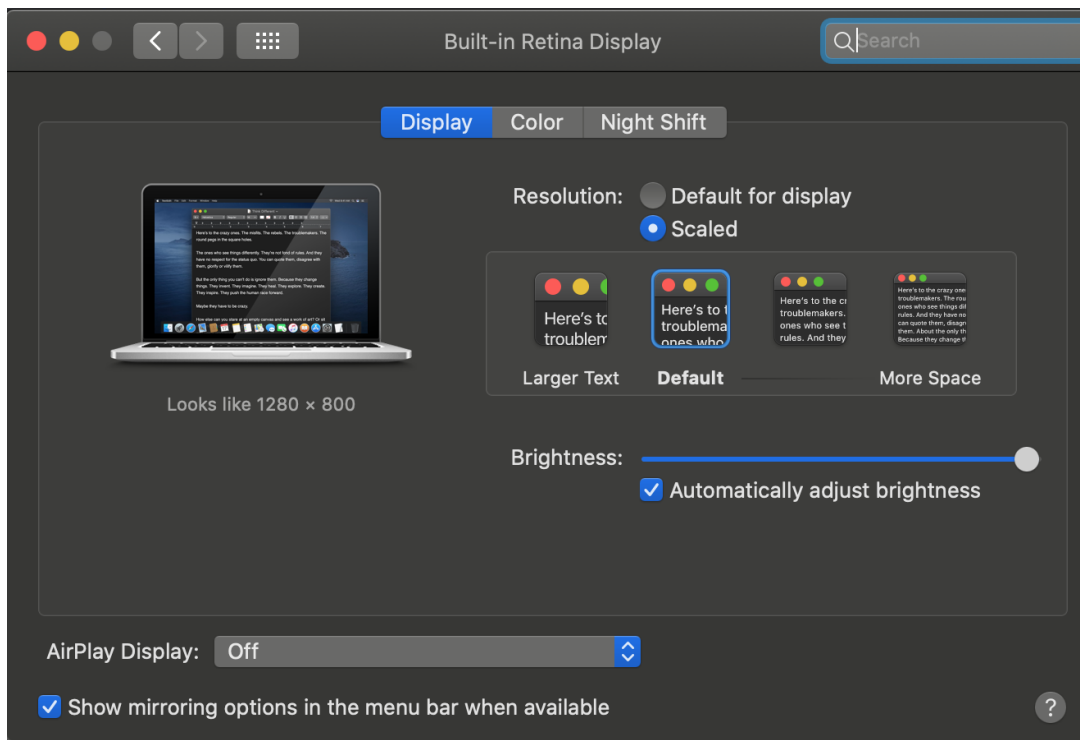
The EyeLinkMRIdemo_Plugin also illustrates how to wait for the first sync signal from the scanner on each block and use that as a trigger to begin the trials of the block (and event markers and data associated with the sync signal to the EDF). See the components in the

waitForScannerPulse and the eITrial component in the trial sequence for illustrations of how this works.

4 Known issues and trouble-shooting tips

4.1 Retina Displays on macOS

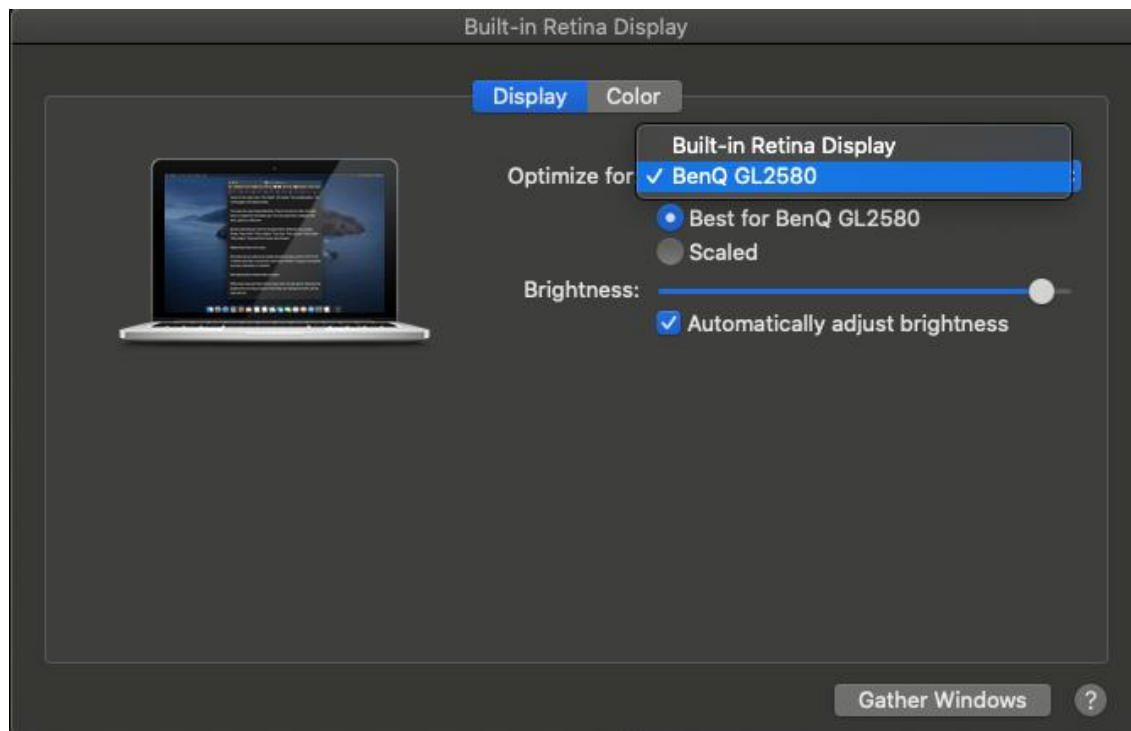
The high-res retina displays on macOS may give people issues when using PsychoPy. One easy solution is to use the “scaled down” resolution instead of the native pixel resolution of the retina display. If you go to the Retina Display settings, tick the “Scaled” option, and hover the mouse on the option of your choice (e.g., Default). You will see the scaled-down resolution on the left side (e.g., 1280 x 800 in the screenshot below). This is the resolution PsychoPy will use to draw your stimuli. However, if you read the “size” of the full screen window you opened PsychoPy will report the actual pixel resolution (e.g., 2560 x 1600). Users should be aware of this discrepancy.



If you are using an external monitor connected to a MacBook (equipped with retina display), macOS will allow you to either optimize screen resolution for the built-in retina display or the external monitor.

When running a PsychoPy Builder project with the EyeLink Plugin on macOS a dialog will ask whether the task will run on a built-in retina display or on an external monitor. If you will be presenting the experiment on an external monitor, then make sure choose to optimize the screen resolution for the external monitor and be sure to choose the external monitor in this dialog. If the experiment will be presented on the built-in retina monitor, then in the Apple settings choose to optimize the screen resolution for the built-in retina display then choose the retina option in the dialog.

[DIALOG SCREEN CAPTURE HERE]

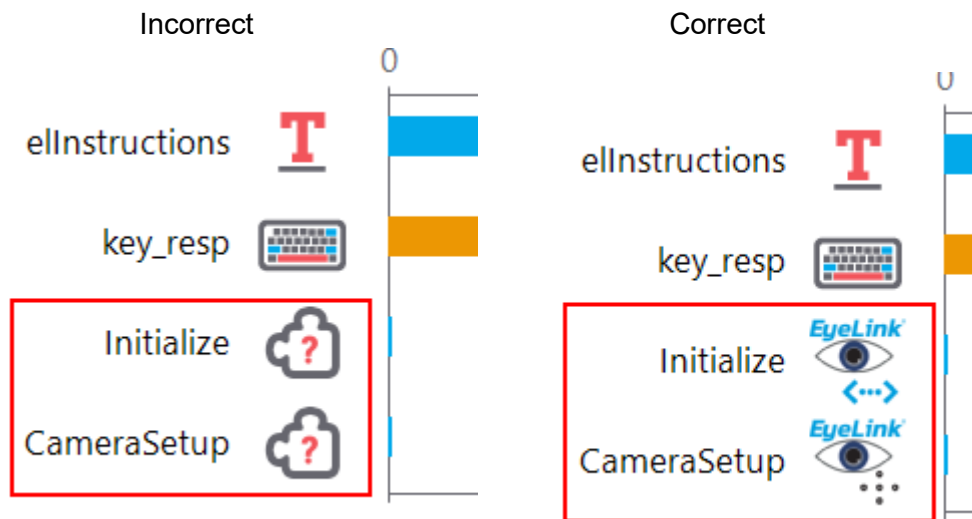


4.2 Experiment quitting after Camera Setup/Calibration/Validation/Drift Checks

If the experiment sometimes ends without notice immediately after moving on from a Camera Setup action (e.g., after pressing O or clicking Output/Record to move on to the real experiment), then it is likely that the “Enable escape key” property in the Experiment settings -> Basic tab is checked. This should be disabled, as the Esc key is sometimes used during eye tracker camera setup/calibration/validation/drift check (and we don’t want it also to end the experimental run). The “Allow Ctrl-C Quit Key to be Used” property of Mark Events components can be enabled to allow the Ctrl-C key combination to be used to end an experimental run early (and also properly retrieve the EDF from the Host PC and shut down connection to the Host PC). See sections 2.6, 3.1, and 3.8 for further discussion of this property.

4.3 EyeLink Plugin components showing up as puzzle pieces

If you double-click an experiment's .psyexp file to launch PsychoPy and open your experiment, then you may see the EyeLink Plugin components appear as puzzle piece icons rather than their typical EyeLink Plugin icons. This will cause the EyeLink components not to work.



If you see this puzzle piecing of the components' icons, simply use the File -> Open dialog in the PsychoPy Builder to re-open the experiment's .psyexp file. Then double-check that the components look correct again.