

# The EyeLink Plugin for OpenSesame

## User Manual

Version 0.2

Last updated on July 4, 2018

## Licensing Info

The EyeLink plugin for OpenSesame was developed to help users to control and to communicate with the EyeLink trackers. The current release does not contain any eye-event triggers (e.g. wait for a fixation of 200 ms in a pre-defined interest area to initiate the presentation of stimulus), but it has most of the frequently used functions and configuration options of the tracker. We encourage users to download the version released on the SR Support website, which implements the tracking practice and options recommended by SR Research, but the user is allowed to modify the source code to meet their experimental needs. Please see the licensing info below.

*Eyelink Plugin for OpenSesame*  
*Copyright (C) 2018 Zhiguo Wang*

*This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.*

*This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.*

*You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.*

# 1 Installation

This plugin should also work on Linux and Mac, but we have only tested it on the Windows platform. The installation instructions presented below are for the Windows platform.

- 1) Download the latest version of the Plugin from the SR Support website, under the Getting Started with Experimental Programming section.
- 2) Unzip the downloaded .zip file and copy all the folders contained in the .zip file to C:\Program Files (x86)\OpenSesame\share\opensesame\_plugins\
- 3) Download and install the latest version of the EyeLink Developer's Kit from the SR Support website, from the Downloads->Eyelink Display Software section.  
<https://www.sr-support.com/forum/downloads/eyelink-display-software/39-eyelink-developers-kit-for-windows-windows-display-software>
- 4) OpenSesame needs the Pylink library to communicate with the tracker. Pylink is a Python wrapper of the EyeLink API (part of the Developer's Kit) and one can find this library from C:\Program Files (x86)\SR Research\EyeLink\SampleExperiments\Python. There are multiple versions, with the last two digits in the folder name corresponding to the Python version it was built against (i.e., pylink27 is for the Python 2.7 series). Please rename the correct version of pylink folder to "pylink" and copy it to C:\Program Files (x86)\OpenSesame\Lib\site-packages.
- 5) After the above steps, please be sure to set the IP address of the experimental PC to "100.1.1.2" (without quotes) and Subnet mask to "255.255.255.0" (without quotes) so the experimental PC is on the same network as the EyeLink Host PC. Open the example experiment that comes with the EyeLink plugin to test the link between the two machines.

OpenSesame supports Pygame (legacy), Psychopy, and Expyriment as its backend. We have found the Expyriment backend is crash-prone and would encourage users to stick to the Psychopy backend, which is robust and supports frequently used visual stimuli for visual psychophysics (e.g., gratings).

## 2 Usage of the Plugin

After the Plugin has been installed, one should see eight items come up in the item toolbar of the OpenSesame interface. To use the plugin, simply drag one of these items

to the required location in the experiment sequence. For general cognitive tasks, we recommend that users follow these integration steps:

### **1) Experiment level**

- a) Connect to the tracker when the script initializes. Please use the `el_Connect` item for this task. The configuration options available for this item will be elaborated in the next section.
- b) Calibrate the tracker at the beginning of each block. This will help the user to maintain optimal tracking accuracy. For fMRI or tasks in which interruption of the task should be avoided, users can calibrate the tracker once at the beginning of each run/session. The item for this function is `el_CamSetup`. This item will help users to transfer the camera image to the experimental PC, to adjust the pupil/CR threshold by using hot keys on the experimental PC keyboard, to calibrate the tracker and validate the calibration results.
- c) Run the experimental trials one-by-one and record eye movement data.
- d) Disconnect from the Eyelink Host PC and transfer the data file to the experimental PC.

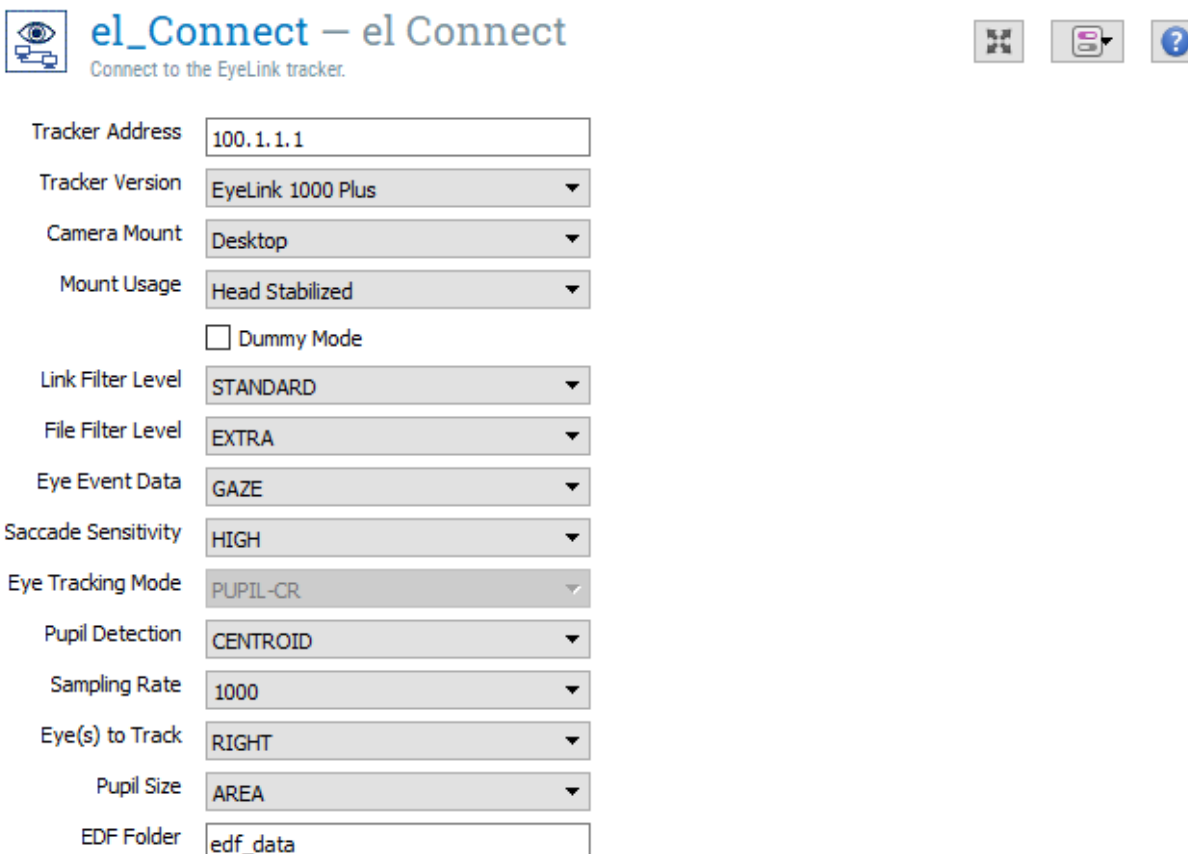
### **2) Trial level**

- a) Drift-correction or drift-check. This procedure will check the tracking accuracy and give the user a chance to re-calibrate the tracker, if needed.
- b) Send commands/messages to the tracker to draw reference landmarks on the Host PC screen (optional, use the `el_Command` or `el_Message` items).
- c) Start recording. We start and stop recording at the beginning and end of each trial, so the inter-trial intervals won't be recorded; this will reduce the size of the EDF data file. For EEG and tasks where continuous recording is preferred, please start recording at the beginning of each run/session. The user also has the option to send a "recording status" message to the tracker; this message will be shown in the bottom-right corner of the Host PC screen.
- d) Draw experimental graphics and send messages to the tracker to mark the onset of these stimuli, and maybe also the interest areas that will be used in data analysis. This is IMPORTANT, otherwise, there is no way to tell when and what stimuli was presented from the eye movement data file.
- e) Collect subject responses and send messages to the tracker.
- f) Stop recording and send all experimental variables to the tracker. These variables will be accessible from the Data Viewer software, a nifty data analysis and visualization tool developed by SR Research.

We have provided an example script with all the recommended usage of the tracker. The functions of each of the items in the plugin are briefly explained below.

## 2.1 el\_Connect

Establish a link to the EyeLink Host PC, configure the tracker, and automatically open a data file on the Host to record the eye movement data.



**Tracker Address** 100.1.1.1

**Tracker Version** EyeLink 1000 Plus

**Camera Mount** Desktop

**Mount Usage** Head Stabilized

☐ Dummy Mode

**Link Filter Level** STANDARD

**File Filter Level** EXTRA

**Eye Event Data** GAZE

**Saccade Sensitivity** HIGH

**Eye Tracking Mode** PUPIL-CR

**Pupil Detection** CENTROID

**Sampling Rate** 1000

**Eye(s) to Track** RIGHT

**Pupil Size** AREA

**EDF Folder** edf\_data

### Important Notes

The EyeLink data file will be saved on the Host PC and retrieved to the current '**EDF Folder**' at the end of a session. By default, the subject number the user specified at the beginning of a session will be used to name the EDF data file. Please bear in mind that the length of the EDF data file name and hence the subject number you specified CANNOT exceed 8 characters.

The options can be set with this item are explained in the table below.

Tracker Address	The IP address of the Eyelink Host PC. The default IP address of the Host PC is 100.1.1.1; the IP address of the experimental PC should be in the same network of the Host PC, i.e., in the range of 100.1.1.2-255.
Tracker Version	The model of EyeLink tracker being used for data collection,

	could be EyeLink I, EyeLink II, EyeLink 1000, EyeLink 1000 Plus, or EyeLink Portable DUO. Some configuration options may not be available for certain models, e.g., sampling rate cannot be set from the EyeLink Plugin for EyeLink I and II, and the Pupil-only tracking method is not available in EyeLink 1000, 1000 Plus and Portable DUO.
Camera Mount	The mounting solution of the EyeLink camera, i.e., Desktop, Tower, Arm, Long-range.
Mount Usage	Tracking in either Head Stabilized or Remote (head free to move) mode. The remote mode is unavailable for Tower and Long-range mounts. Tracking in remote mode requires the use of a target sticker on the subject's forehead.
Dummy Mode	Run the tracker in "simulation" mode, i.e., no physical connection to the tracker is established. In Dummy Mode, the user should press F1 to skip Camera setup/calibration, and the drift-correction/check target will briefly flashes and then disappear (as not tracker is physically connected to the experimental PC).
Link Filter Level	The EyeLink trackers utilizes a heuristic filtering algorithm for denoise purposes (see Stampe, 1993). Each increase in filter level reduces noise by a factor of 2 to 3 but introduces a 1-sample delay to the data available over the link. The default option is set to STANDARD, but user can turn off Link Filter if real time online access of gaze data is critical.
File Filter Level	Same as above, but applies to the data recorded in file.
Eye Event Data	Set how velocity information for saccade detection is to be computed. This option is almost always set to GAZE. Please see the EyeLink user manual (section 4) for details of various eye events (e.g., fixation, saccade).
Saccade Sensitivity	Sensitivity of the EyeLink online parse, see Section 4.3.9 of the user manual. For EyeLink II and newer models, HIGH-velocity=22 deg/sec, acceleration=3200 deg/sec <sup>2</sup> ; NORMAL-velocity=30 deg/sec, acceleration=8000 deg/sec <sup>2</sup> ."
Eye Tracking Mode	Select the tracking algorithm. EyeLink I operates in Pupil-only mode, while EyeLink II operates in either Pupil-only or Pupil-CR mode. EyeLink 1000 and newer models all operate in Pupil-CR mode. The Pupil-CR tracking algorithm is resilient to small head movements (i.e., drift-free to certain extent). This is why force drift-correcting the tracker is not recommended for EyeLink

	1000 and newer models.
Pupil Detection	Set the algorithm used to detect the pupil center. This option only applies to EyeLink 1000 and newer models.
Sampling rate	The sampling rate of the tracker.
Eye(s) to Track	Set the eyes to track; can be changed on the Host PC manually.
Pupil Size	Record the pupil size in arbitrary units, AREA and DIAMETER measures are equivalent: $DIAMETER = 256 * \sqrt{AREA/PI}$ . The pupil size recorded in the data files is in arbitrary units; calibration during testing is required if one needs to report pupil size in real world units (i.e., mm). Please see this SR Support forum post for details, <a href="https://www.sr-support.com/showthread.php?5021-Answers-to-EyeLink-Host-PC-and-Hardware-FAQ&amp;p=20283#post20283">https://www.sr-support.com/showthread.php?5021-Answers-to-EyeLink-Host-PC-and-Hardware-FAQ&amp;p=20283#post20283</a> .
EDF Folder	The EyeLink data file will be saved on the Host PC and retrieved to the current <b>'EDF Folder'</b> at the end of a session. By default, the subject number the user specified at the beginning of a session will be used to name the EDF data file. Please bear in mind that the length of the EDF data file name and hence the subject number you specified CANNOT exceed 8 characters.

## 2.2 el\_Disconnect

This item will disconnect from the EyeLink Host PC and retrieve the EDF data file over the link. No option to configure for this item.

## 2.3 el\_CamSetup

This item wraps all the functions a user may need to calibrate the tracker. Using animation calibration target (video) has not yet been implemented. One should put this item at the beginning of each block of trials. The configuration options are explained in the table below.



# new\_el\_CamSetup – el CamSetup

Connect to the EyeLink tracker.



Calibration Type

Pacing Interval

☒ Randomize Order

☒ Repeat First Point

☐ Force Manual Accept

Horizontal Screen Proportion to Calibrate

Vertical Screen Proportion to Calibrate

Calibration Target

Custom Target Image/Video

Browse


Calibration Type	Select the calibration type, i.e, HV9 for a 9-point calibration. When tracking in remote mode, it is recommended to use HV13, whereas in head-stabilized mode, HV9 gives the best calibration results.
Pacing Interval	Set the pacing interval for the calibration/validation targets, i.e., after how much time will the next calibration target be presented after the current calibration target has been accepted.
Randomize Order	Randomize the order of the calibration/validation targets
Repeat First Point	Repeat the first point. This option is enabled by default, helps to improve calibration results.
Force Manual Accept	Manually accept fixation duration calibration/validation by pressing SPACEBAR or ENTER. One can switch to automatic mode at any time during calibration/validation by pressing “A” on the Host or experimental PC keyboard.
Horizontal Screen Proportion to Calibrate	The horizontal proportion of screen to calibrate. This option is useful when the subject display is large and the top corners may be outside the tracking range of the tracker. One can manually specify the calibration/validation target positions, but this is the recommended way of controlling the size of the calibrated screen region.
Vertical Screen Proportion to Calibrate	The horizontal proportion of screen to calibrate.






Calibration Target	Select which type of calibration target to use. The default is a bull's eye shaped dot, but one can also use an image or a video as the calibration target.
Custom Target Image/Video	Select an image or a video file from the File Pool to use as the calibration target.

## 2.4 el\_DriftCheck

This item helps to drift-correct the tracker. The EyeLink II and EyeLink I trackers mount the eye camera on the headband and are susceptible to gaze-drifts as the headband may slip during recording. Drift-correction helps to maintain the tracking accuracy. For EyeLink 1000 and newer models, the tracker uses the Pupil-CR tracking method by default. This method is relatively drift-free and there is no need to drift-correct the tracker; by default the drift-correction routine only checks the gaze error without linearly correct the gaze data. The various configuration options of this item are listed in the table below.


**el\_DriftCheck – el DriftCheck**
Drift-correction/check

Target X

Target Y

☒ Allow Re-calibrate if Drift-correction/check Fails

☐ Apply Drift-correction [EyeLink II/I ONLY]

☐ Use Custom Target

Custom Target Image

☐ Use Animation Target

Animation Target Video

### Important Notice

Force drift-correct the tracker **ONLY** for EyeLink II/I systems! EyeLink 1000 and newer models by default use the Pupil-CR tracking algorithm, which is largely drift-free. Forcing the tracker to linearly correct the calibration matrix with the gaze error will reduce tracking accuracy.

Target X/Y	The X, Y coordinates of the drift-correction/check target in OpenSesame's default screen coordinates (i.e., 0,0 correspond to the center of the screen). The drift-correction/check target does not necessarily need to be presented at the center of the screen.
Allow Re-calibrate if Drift-	Allow the user to press ESCAPE to setup the tracker and to re-calibrate.

correction/check Fails	
Apply Drift-correction	Force the tracker to drift-correct only when using EyeLink II/I.
Use Custom Target	Use an image from the File Pool as the drift-correction/check target.
Custom Target Image	Select an image file to use as the drift-correction target.
Use Animation Target	Use a video as the drift-correction target.
Animation Target Video	Select a video to use as the drift-correction target.

## 2.5 el\_StartRecording

Start the track in recording mode. One may also send a status message to the tracker to show the current condition/trial, or the progress of the task on the Host PC screen. One can also specify what types of data is available over the link during recording, and what types of data are saved in EDF data files.



File with Eye Events	Store event data in the EDF data file.
File with Samples	Store event data in the EDF data file.
Eye Events Available Over Link	Allows accessing event data over the link during recording.
Samples Available	Allows accessing sample data over the link during recording.

Over Link	
Recording Status Message	Send a message to the Host PC screen to show the current trial number, condition, etc.

## 2.6 el\_StopRecording

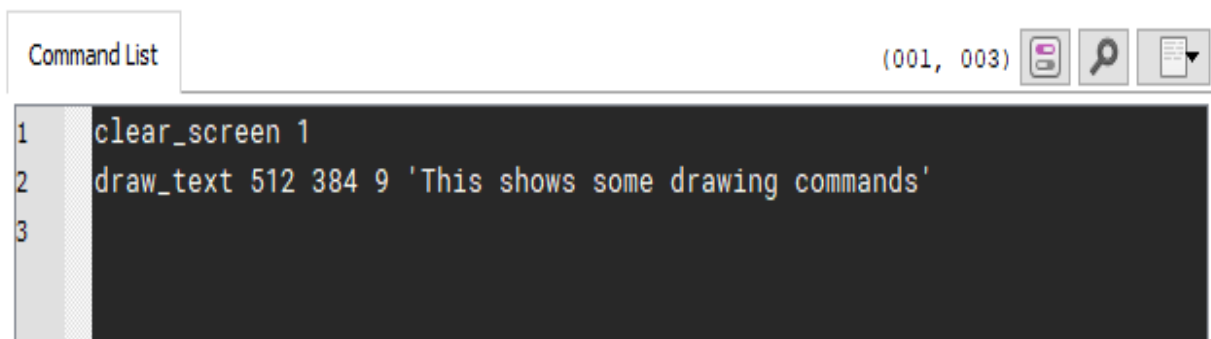
This item is usually placed at the end of a trial to stop the recording of eye movement data, and to do some housekeeping work. It should be noted that all experimental and trial VARIABLES available in the OpenSesame namespace will be automatically sent to the tracker by this node. No configuration option is available for this item.

## 2.7 el\_SendCommand

Send commands to the tracker. If you need to send multiple commands, put each command in a line, for instance,

```
sampling_rate 500
draw_cross 512 384
```

The various 'draw' commands can be very useful and one can use them to draw simple landmarks on the Host display during recording. These commands (e.g., clear\_screen, draw\_line, draw\_box, draw\_text) can be found in the COMMANDS.INI file on the Host PC, under /elcl/exe. One can also send various commands to configure the tracker options, for instance, setting the sampling rate to 500 Hz (see the example above).



## 2.8 el\_SendMessage

Messages are very IMPORTANT and we need messages in the DATA FILE to tell what

events happened during a trial and at what time. Messages should be sent to the tracker everytime a stimulus screen is on or a response has been issued. A message should not exceed 113 characters. One can send multiple messages, please put one message in a line.

One can also send “Data Viewer Integration Messages” to the tracker. These messages will be used by the Data Viewer software, a data analysis and visualization tool provided by SR Research to load the interest areas, background images, etc. Please see the Data Viewer user manual for a full list of Data Viewer integration messages. A few frequently used ones are listed below.

### Image Loading Messages

!V IMGLOAD FILL relative\_image\_path

!V IMGLOAD TOP\_LEFT relative\_image\_path x\_position y\_position width height

!V IMGLOAD CENTER relative\_image\_path x\_position y\_position width height

### Interest Area Messages

!V IAREA RECTANGLE id left top right bottom label

!V IAREA ELLIPSE id left top right bottom label

!V IAREA FREEHAND id x1, y1 x2, y2 ... xn, yn label

!V IAREA FILE relative\_file\_path (load interest area template file)



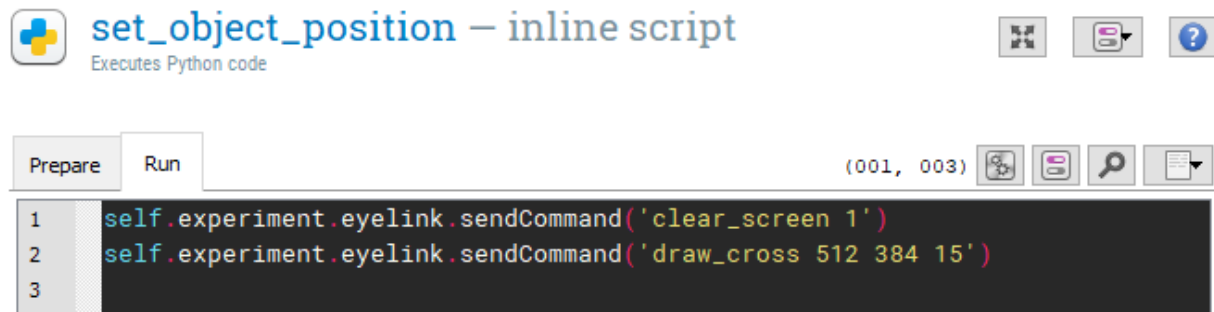
## 3 Hacks and Caveats

### 3.1 Accessing the tracker from inline scripts

The tracker instance initialized by the plugin can be accessed from inline scripts by referencing to “self.experiment.eyelink”. For instance, put the following commands in an inline script will clear the screen of the Host PC and draw a cross at the center.

```
self.experiment.eyelink.sendCommand('clear_screen 1')
self.experiment.eyelink.sendCommand('draw_cross 512 384 15')
```

For all the attributes and methods of the tracker instance, please refer to the Pylink user manual for details. The Pylink user manual is packaged within the EyeLink Developer's Kit and can be accessed from the Start menu on Windows machines (Start->All Programs -> SR Research->Manuals) after the Developer's Kit has been installed.



The ability to access the tracker from inline scripts allows user to access of all functions wrapped in the Pylink library. Be sure to put your custom code in the “Run” rather than the “Prepare” tab.

## 3.2 Interest area definitions

It is IMPORTANT to keep in mind that the Interest Area definitions recognizable by the Data Viewer software requires users to specify the Interest Area in a screen coordinates where 0,0 is the top-left corner of the screen. In OpenSesame, the origin of the default screen coordinates is the center of the screen instead. So, one need to do a bit math when creating interest area messages. In the example inline Python scripts, we first grab the stimulus named “target” from the canvas of the “stimulus\_display”, then, we extract the left, top, right, and bottom of this stimulus; then, we send the interest area message in the “Run” tab.

```
sti_display = items['stimulus_display'].canvas

tar_ia = (int(sti_display['target'].x - sti_display['target'].width/2+var.width/2),
          int(sti_display['target'].y - sti_display['target'].height/2+var.height/2),
          int(sti_display['target'].x + sti_display['target'].width/2+var.width/2),
          int(sti_display['target'].y + sti_display['target'].height/2+var.height/2))

self.experiment.eyelink.sendMessage('!V IAREA RECTANGLE 1 %d %d %d %d target'%tar_ia)
```

## 4 The Visual World example

This example script shows how to program a Visual World type task in OpenSesame.

One may believe that a tool with a nice graphical user interface will require minimum scripting, this is almost always NOT the case. OpenSesame and E-Prime both require quite a bit “inline” scripting for tasks of medium to high complexity. I will explain the scripting part in more detail in a later section.

## 4.1 Overview of the task

The task is straightforward. A drift-correction/check target (cross) first appears on the screen, then 4 objects appears. After a preview period of 1000 msec, an audio file starts to playback. The subject’s task is to quickly move the mouse cursor to click the target object (grapes in the present task) when they hear the target word (“grapes”). By performing a time-binning analysis over the sample gaze data, one can plot a nice figure to show the “decision-making” process after the target word has been played. For an overview of the Visual World paradigm, please see the review paper by Huettig, Rommers, & Meyer (2011).

*Huettig F, Rommers, J, & Meyer, A. S. (2011). Using the visual world paradigm to study language processing: A review and critical evaluation. psychologica 137(2):151-171. DOI: 10.1016/j.actpsy.2010.11.003*



This task obviously requires precise timing for audio playback. Please bear in mind that SR Research has not done any sort of timing verification for OpenSesame and we encourage users to perform their own tests if stimulus timing is critical for their tasks.

The overall organization of the script is not complicated, for introductory tutorials on

programming in OpenSesame, please check the Tutorial section on the OpenSesame website, <http://osdoc.cogsci.nl>.

## 4.2 Inline scripts

The present task uses a few inline scripts and the most complicated one is the “sti\_preparation” inline. This script is responsible for quite some tasks we would recommend if one plans to use (or may use) the EyeLink Data Viewer software to analysis and visualize the eye movement data later on.

### A) Controls the position of the objects

Here we first create a list of four screen locations with the `xy_circle()` function. Then, we change the position of the objects based on the location specified in the “block” loop. The “location” of each object is specified in the block loop as integers, i.e., 1, 2, 3, 4. We use this variable as index to set the position of each object by using the list of screen coordinates (`sti_pos`) we created with `xy_circle()`.

```
# the positions of the objects
sti_pos = xy_circle(n=4, rho = 250)

# get the canvas on which the objects are shown
sti_display = items['stimulus_display'].canvas

# position the objects
sti_display['target'].x, sti_display['target'].y = sti_pos[var.target_loc-1]
sti_display['distractor_1'].x, sti_display['distractor_1'].y = sti_pos[var.distractor_1_loc-1]
sti_display['distractor_2'].x, sti_display['distractor_2'].y = sti_pos[var.distractor_2_loc-1]
sti_display['distractor_3'].x, sti_display['distractor_3'].y = sti_pos[var.distractor_3_loc-1]
```

### B) Define the interest areas

The Interest area definitions can be sent to the tracker by a simple message. For this message, we need to know the left, top, right and bottom position of the interest areas. This is done by the following lines of code in the Prepare tab.

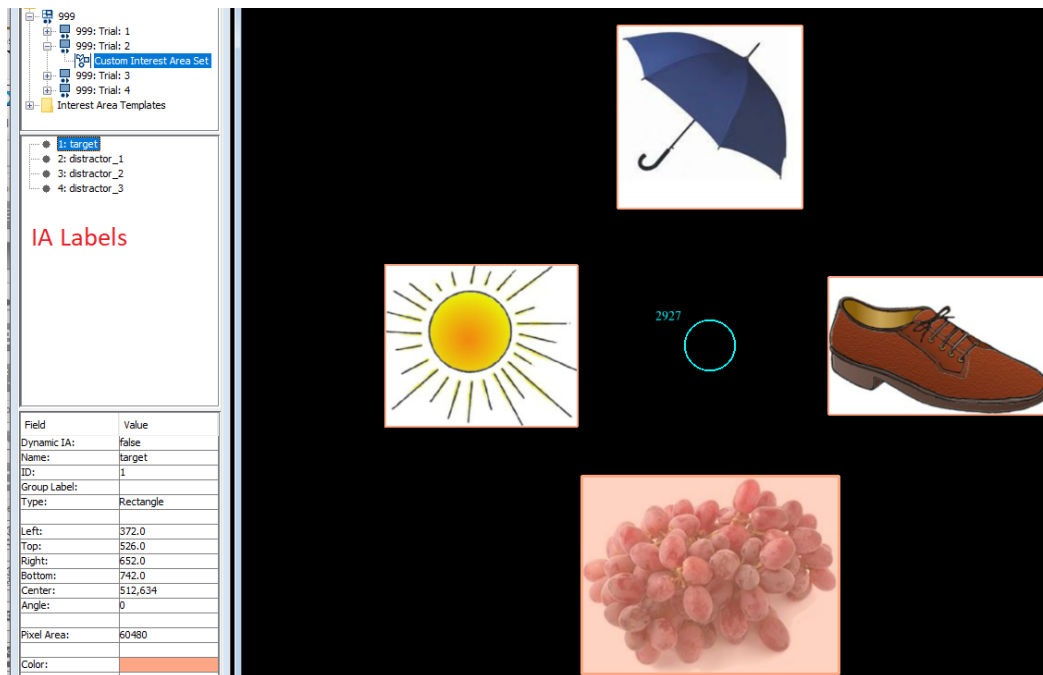
```
tar_ia = (int(sti_display['target'].x - sti_display['target'].width/2.+var.width/2),
          int(sti_display['target'].y - sti_display['target'].height/2.+var.height/2),
          int(sti_display['target'].x + sti_display['target'].width/2.+var.width/2),
          int(sti_display['target'].y + sti_display['target'].height/2.+var.height/2))
```

It is important to bear in mind that the screen coordinates used by the tracker has the origin (0,0) at the top-left corner of the screen, whereas that in OpenSesame by default is the center of the screen. Transformation of the coordinates is needed here.

In the Run tab, we send the following message to the tracker.

```
self.experiment.eyelink.sendMessage('!V IAREA RECTANGLE 1 %d %d %d %d target"%tar_ia)
```

For more information about the format of Interest Area messages, please see the Data Viewer User Manual, section 7. With these messages, the Interest area definitions will be automatically loaded into Data Viewer when an EDF data file is opened.



### C) Load background images

With image loading messages, Data Viewer will be able to load the correct background images when an EDF data file is imported into a Data Viewer viewing session. For this to happen, we need to know the path to the background images, relative to where the EDF data files are stored. In the example script, the EDF data files, by default, will be saved in the “edf\_data” folder, while the image files are stored in the “images” folder. Relative to the EDF data files, the path to the images is “../images”. Consequently, the target object corresponds to the following file path. Here we use “var.target\_img” to refer to the “target\_img” column in the “block” loop.

```
target_img_path = '../images/' + var.target_img
```

Of course, we will also need to know where the images should be presented.

```
target_img_x = int(sti_display["target"].x+var.width/2)
target_img_y = int(sti_display["target"].y+var.height/2)
```



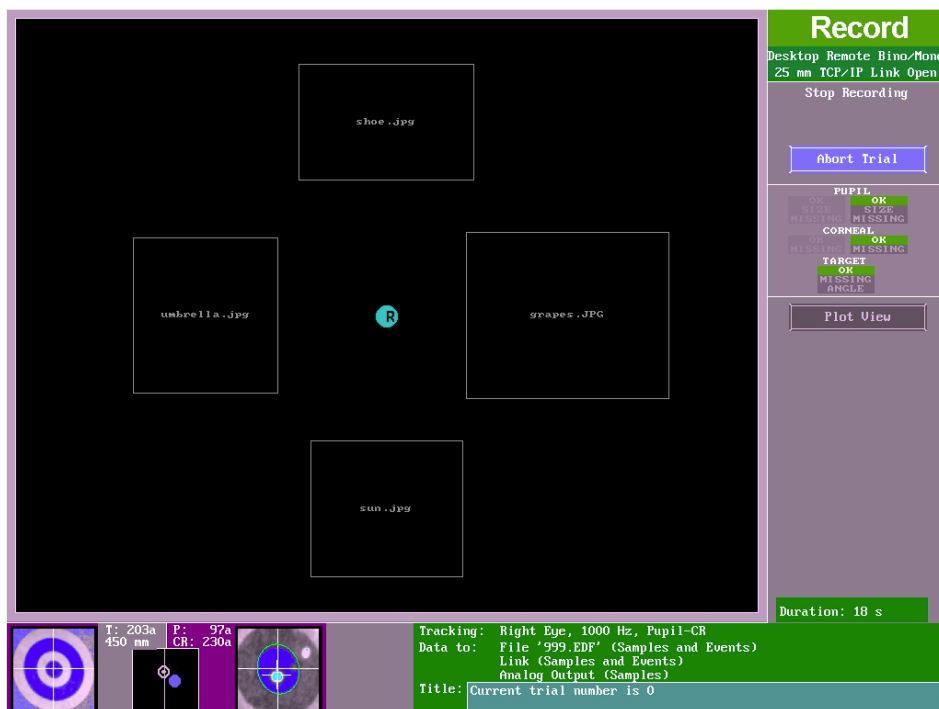
With the above information, we can put the following message in the Run tab to send a valid image loading message to the tracker.

```
self.experiment.eyelink.sendMessage('!V IMGLOAD CENTER %s %d %d'%(target_img_path,
target_img_x, target_img_y))
```

## D) Draw landmark boxes on the Host PC

One can transfer the images being presented to the subject to the Host PC screen, so the experimenter can examine which object is being fixated during recording. However, transferring images to the Host can be time consuming and it is not advisable for timing critical tasks. Instead, one can always use the drawing commands to draw some landmarks on the Host PC. This can be done with the following line of code in the Run tab. Here we also used the “draw\_text” command to show which image is presented in each landmark box on the Host PC.

```
self.experiment.eyelink.sendCommand('clear_screen 0')
self.experiment.eyelink.sendCommand('draw_box %d %d %d %d 15'%tar_ia)
self.experiment.eyelink.sendCommand('draw_text %d %d 15 %s'%(target_img_x, target_img_y,
var.target_img))
```

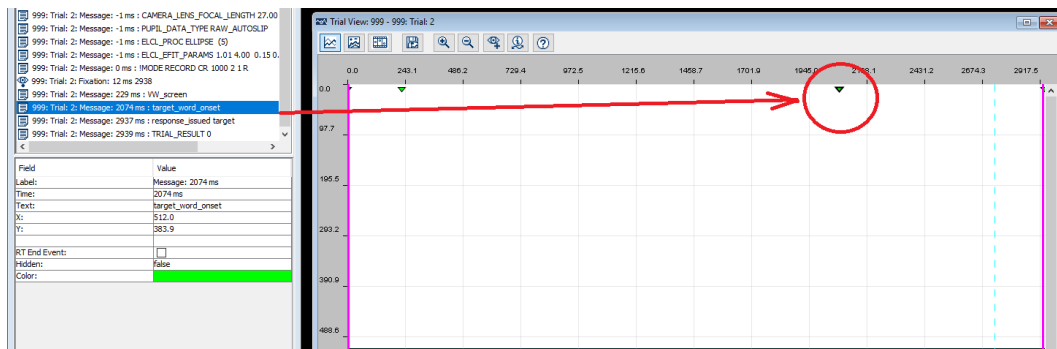


The other inline script in this task is “target\_word”. It simply wait for a while until the target word is played in the audio file, then it sends a message to the tracker to mark

the onset of the target word (“grapes”). This message is critical to our data analysis, without this message, one cannot align the eye movement data to the onset of the target word.

*# wait for the onset of the target word, then should the mouse cursor  
clock.sleep(var.target\_word\_onset\_time)*

*# send a message to let the tracker know the onset of the target word  
self.experiment.eyelink.sendMessage('target\_word\_onset')*



## 4.3 Trial variables

At the end of each trial, when the `el_StopRecording` node is called, all variables in the OpenSesame namespace will be recorded in the EDF data file. These variables are critical for group-level analysis in Data Viewer and can be accessed from the Trial Variable Value Editor within Data Viewer.

Trial Variable Value Editor								
_garbage_col...	distractor_1_img	distractor_1_loc	distractor_2_img	distractor_2_loc	distractor_3_img	distractor_3_loc	experiment_file	expe
	sun.jpg	4	umbrella.jpg	1	shoe.jpg	2	VW_click.osexp	C:\Us
	sun.jpg	3	umbrella.jpg	4	shoe.jpg	1	VW_click.osexp	C:\Us
	sun.jpg	2	umbrella.jpg	3	shoe.jpg	4	VW_click.osexp	C:\Us
	sun.jpg	1	umbrella.jpg	2	shoe.jpg	3	VW_click.osexp	C:\Us

Reset Cancel OK

## 5 Known issues and limitations

Please bare in mind that the EyeLink plugin is still beta software and we have noticed the following issues during our testing.

- a) During camera setup, the image is transferred to the subject PC from the EyeLink Host PC, so one can conveniently adjust the focus of the lens and use the keyboard hot keys to adjust pupil/CR threshold. The keyboard may sometimes appear laggy or unresponsive. We are still investigating the source of this problem.
- b) The Expyriment backend tend to be buggy and crash-prone. The drawing functions are not properly wrapped in OpenSesame. The search limit appears as a figure 8 instead of an oval in the camera setup screen. Keyboard responses are extremely laggy and may freeze from time to time. One need to be patient.
- c) The `.set_pos()` function only works when the backend is Psychopy. So, in the Visual World task, the mouse cursor won't necessarily be placed at the center of the screen after the target word has been played.
- d) Animated calibration target can be useful for infant studies. The current implementation of animated calibration targets (videos) is more of a "hack", e.g., we present the frames of the video as images one-by-one. There is no sound track. For simplicity, no fail-safe code is included in this initial release. One should always check if the selected media file matches the "Calibration Target" type being selected. OpenSesame will terminate and give a warning if the "Calibration Target" option is set to "AnimationVideo" whereas the "Calibration Target Image/Video" field is actually an image.

If you run into any issues when using the EyeLink plugin, please feel free to contact [zhiguo@sr-research.com](mailto:zhiguo@sr-research.com).