Gaze Contingent Code Documentation

This document only highlights the parts of code to be tweaked for future changes.

Introduction

General SR Research Template Code Structure.

GC WINDOW

```
from __future__ import division
from __future__ import print_function

import pylink
import os
import platform
import random
import time
import sys
from EyeLinkCoreGraphicsPsychoPy import EyeLinkCoreGraphicsPsychoPy
from psychopy import visual, core, event, monitors, gui
from PIL import Image  # for preparing the Host backdrop image
from string import ascii_letters, digits

# Switch to the script folder
script_path = os.path.dirname(sys.argv[0])
if len(script_path) != 0:
    os.chdir(script_path)

# Show only critical log message in the PsychoPy console
from psychopy import logging
logging.console.setLevel(logging.CRITICAL)

# Set this variable to True if you use the built-in retina screen as your
# primary display device on macOS. If have an external monitor, set this
# variable True if you choose to "Optimize for Built-in Retina Display"
# in the Displays preference settings.
use_retina = False

# Set this variable to True to run the script in "Dummy Mode"
dummy_mode = False

# Set this variable to True to run the task in full screen mode
```

```
# It is easier to debug the script in non-fullscreen mode
full_screen = True

# Store the parameters of all trials in a list, [condition, image]
trials = [
    ['mask', 'img_1.jpg'],
    ['mask', 'img_2.jpg'],
    ['window', 'img_1.jpg'],
    ['window', 'img_2.jpg'],
    ]

# Set up EDF data file name and local data folder
#
# The EDF data filename should not exceed 8 alphanumeric characters
# use ONLY number 0-9, letters, & _ (underscore) in the filename

--- CODE TO SET FILE NAMES AND RESULTS FOLDER

    # THIS SECTION OF CODE is to establish and set results folder and edf file name

--- SETTING EDF FILE NAME


# Step 1: Connect to the EyeLink Host PC
#
# The Host IP address, by default, is "100.1.1.1".
# the "el_tracker" objected created here can be accessed through the Pylink
# Set the Host PC address to "None" (without quotes) to run the script
# in "Dummy Mode"
if dummy_mode:
    el_tracker = pylink.EyeLink(None)
else:
    try:
        el_tracker = pylink.EyeLink("100.1.1.1")
    except RuntimeError as error:
        print('ERROR:', error)
        core.quit()
        sys.exit()

# Step 2: Open an EDF data file on the Host PC
edf_file = edf_fname + ".EDF"
try:
    el_tracker.openDataFile(edf_file)
except RuntimeError as err:
    print('ERROR:', err)
    # close the link if we have one open
```

```
    if el_tracker.isConnected():
        el_tracker.close()
    core.quit()
    sys.exit()


--- CONFIGURATION CODE STARTS HERE

    # This section includes sophistcated code to connect and configure Eyelink
programatically, it doesn't require any tweaking so leave it untouched

--- CONFIGURATION CODE ENDS HERE


def clear_screen(win):
   pass


def show_msg(win, text, wait_for_keypress=True):
    pass


def terminate_task():
    pass


def abort_trial():
    pass


def run_trial(trial_pars, trial_index):
    pass

# Step 5: Set up the camera and calibrate the tracker

# Show the task instructions
if dummy_mode:
    task_msg = 'Cannot run the script in Dummy mode,\n' + \
        'Press ENTER to quit the script'
else:
    task_msg = 'In the task, you may press the SPACEBAR to end a trial\n' + \
        '\nPress Ctrl-C if you need to quit the task early\n' + \
        '\nNow, press ENTER twice to calibrate tracker'
show_msg(win, task_msg)

# Terminate the task if running in Dummy Mode
```

```
if dummy_mode:
    print('ERROR: This task requires real-time gaze data.\n' +
          'It cannot run in Dummy mode (with no tracker connection).')
    terminate_task()
else:
    try:
        el_tracker.doTrackerSetup()
    except RuntimeError as err:
        print('ERROR:', err)
        el_tracker.exitCalibration()

# Step 6: Run the experimental trials, index all the trials

# construct a list of 4 trials
test_list = trials[:]*1

# randomize the trial list
random.shuffle(test_list)

trial_index = 1
for trial_pars in test_list:
    run_trial(trial_pars, trial_index)
    trial_index += 1

# Step 7: disconnect, download the EDF file, then terminate the task
terminate_task()
```

From the above template Structure the only code that should be altered will be inside the `run_trial()` function. Rest of the code and helper functions **Should not be modified** . These essentially allow us to follow a conventional execution process that Eyelink mandates. i.e Connection → Caliberation → Validation → Trial

---

run_trial( ) Function

```
    trials_data = []

    # unpacking the trial parameters
    #cond, pic = trial_pars

    # disable the GC window at the beginning of each trial
    gaze_window.enabled = False

    # load the image to display, here we stretch the image to fill full screen
    img = visual.ImageStim(win,image='example.png',)
    #text = visual.TextStim(win, text='Gaze on image!', pos=(0, -200), color='white')
```

- This section of the code includes, the trial_data empty list instantiation which holds response time and key_press for each iteration/trial
- The next line of code indicates, the gaze_window, which in this case is invisible gaze marker.
- Image stimulus instantiation is in the following line.

```
# show the image for 5-secs or until the SPACEBAR is pressed
    # move the window to follow the gaze
    event.clearEvents()  # clear cached PsychoPy events
    RT = -1  # keep track of the response time
    gaze_pos = (-32768, -32768)  # initial gaze position (outside the window)
    get_keypress = False
    while not get_keypress:
        # present the picture for a maximum of 5 seconds ~ changed to 100
        if core.getTime() - img_onset_time >= 100:
            el_tracker.sendMessage('time_out')
            break

        # abort the current trial if the tracker is no longer recording
        error = el_tracker.isRecording()
        if error is not pylink.TRIAL_OK:
            el_tracker.sendMessage('tracker_disconnected')
            abort_trial()
            return error

          for trial in trial_range:

            ## MAIN EXPERIMENT CODE
```

This part of the code is a conditional check through the while loop. The main functionality lies nested inside this while (as explained in the next code chunk). This segment of the code checks for keypresses to abort or perform other events such as caliberation or validation

```
for trial in trial_range:

            number = visual.TextStim(win, text=cond.loc[trial, 'Target'], color='black',
height=100)
            left_number = visual.TextStim(win, text=cond.loc[trial, 'Target'],
color='black', height=100, pos=(-300, 0))
            right_number = visual.TextStim(win, text=cond.loc[trial, 'Foil'],
color='black', height=100, pos=(300, 0))


            # grab the newest sample
            dt = el_tracker.getNewestSample()
            if dt is None:  # no sample data
                gaze_pos = (-32768, -32768)
            else:
                if eye_used == 1 and dt.isRightSample():
                    gaze_pos = dt.getRightEye().getGaze()
                elif eye_used == 0 and dt.isLeftSample():
                    gaze_pos = dt.getLeftEye().getGaze()

                # check if the center of the gaze_window is within the bounds of the img
stimulus
                if (img.pos[0] - img.size[0]/2) <= gaze_window.pos[0] <= (img.pos[0] +
img.size[0]/2) and \
                (img.pos[1] - img.size[1]/2) <= gaze_window.pos[1] <= (img.pos[1] +
img.size[1]/2):

                    # EXPERIMENT SEQUENCE
```

The Gaze Contigent functionality is triggered in this segment of code. final `if` statement in this code chunk checks if the gaze marker position is in the vicinity of the Image position. If so, the experiment is triggered.

The rest of the code is similar to `experiment.py` under psychopy folder.

Function parameters

```
# Read in the conditions from the CSV file
conditions = pd.read_csv("./conditions/Sample_Conditions.csv").drop('Unnamed: 0',axis=1)
CONDITION_NUM = 2 ## CHANGE THE CONDITION NUMBER TO TOGGLE BETWEEN THE TRIAL TYPE (
Current options 1 or 2 )
cond = conditions[conditions['Conditions']==CONDITION_NUM]
cond = cond.reset_index(drop=True)
cond = cond.sample(frac=1) # sample rows and with replacement ( Shuffles all the samples )
trial_range = cond['Trials'] # Trial range - 0 to 9



run_trial(trial_range, cond)
```

This code is outside of any function, the code reads the condition file, shuffles the trials and parameterizes the run_trial function.
