# Pyhab User Manual

v1.2

Summer 2017

Dr. Jonathan F. Kominsky

## 1. What is PyHab?

Infant and toddler looking-time studies primarily rely on manual looking-time coding software like jHab and xHab, which are old, opaque, and not open-source. PyHab is a script written for PsychoPy, a free, open-source python-based stimulus presentation software (with which the author is not affiliated), that fully replicates the capabilities of xHab and jHab while adding several features.

The single biggest advantage of PyHab over its predecessors is that it can be used to directly control stimulus presentation. Previously, you typically needed one person coding looking times and a second person controlling the stimuli, and communicating when to end one trial and advance to the next, and when to advance from habituation trials to test trials. This introduces logistical hassle and imprecise timing. PyHab can present movie files or animated stimuli and advance trials and trial types appropriately based on live looking time coding. One person can run an entire experiment.

You might worry that this means you can't run these studies with a blind coder. Not at all! Provided that the coder can't determine what the participant is seeing or hearing (which will just depend on your setup), PyHab can be configured to tell a coder only when a trial starts and ends, and use a masked condition file so that the coder only needs to input a code and not know what condition is being presented. In habituation designs when the test trial can come after a variable number of hab trials, PyHab doesn't even need to tell the coder when the test trial happens, in some ways making it *more* blind than other techniques!

## 2. What do you need to be able to use PyHab?

- You need PsychoPy, first of all (available free from psychopy.org). PyHab is just a script that runs in a PsychoPy environment.

- You will need the script itself in a folder. If you are doing stimulus presentation, you will need the attention-getting sound file (upchime1.wav) plus any movie files in the same folder.

- Once you have everything set up, simply open the script in PsychoPy's 'coder' interface and run it from there.

- To use PyHab for stimulus presentation, you will also need a computer with a two-screen setup. One screen will be your stimulus presentation screen, which will display stimuli to your participant. The other will be the coder's screen, which will display a small graphical interface telling you when a trial begins and ends.

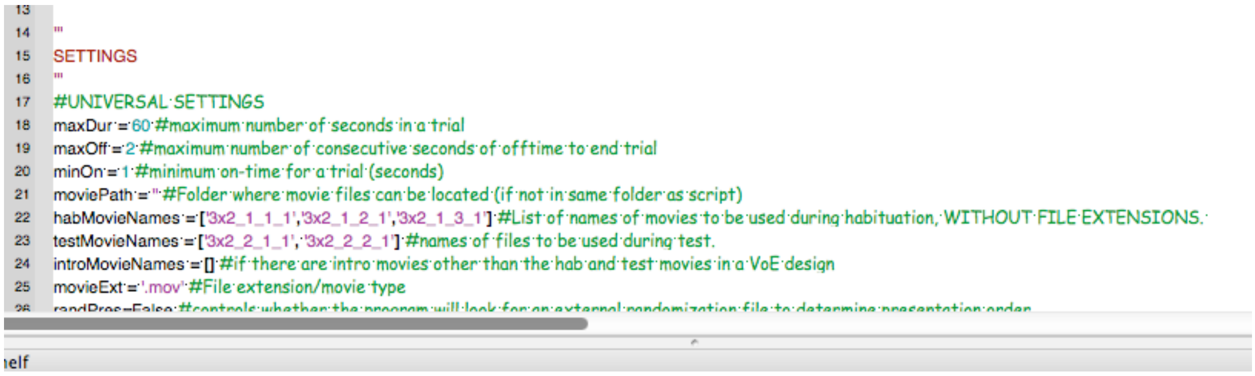
- You DON'T need to know python! I've done my best to make it easy for you to build your study in PyHab by modifying a few simple lines of code. Python experience will help you with troubleshooting and allow you to do more ambitious things with PyHab, but it shouldn't be necessary just to use PyHab as designed.

- If you want to present movie files, you will want your stimuli in .mov format for the most reliable performance. PsychoPy's MovieStim works with other formats, but I have not tested them with PyHab.

## 3. How to build your study in PyHab

### *a. Important parameters and settings*

When you open PyHab in PsychoPy's coder view, it should open a window with a bunch of text. If you don't know python, a lot of it won't make sense to you. Don't worry about it! The only stuff you need to mess with is the section labeled "SETTINGS".



```
'''
SETTINGS
'''
#UNIVERSAL SETTINGS
maxDur = 60 #maximum number of seconds in a trial
maxOff = 2 #maximum number of consecutive seconds of offtime to end trial
minOn = 1 #minimum on-time for a trial (seconds)
moviePath = '' #Folder where movie files can be located (if not in same folder as script)
habMovieNames = ['3x2_1_1_1','3x2_1_2_1','3x2_1_3_1'] #List of names of movies to be used during habituation, WITHOUT FILE EXTENSIONS.
testMovieNames = ['3x2_2_1_1', '3x2_2_2_1'] #names of files to be used during test.
introMovieNames = [] #if there are intro movies other than the hab and test movies in a VoE design
movieExt = '.mov' #File extension/movie type
randPres=False #controls whether the program will look for an external randomization file to determine presentation order
```

Which settings you need to care about will depend on what version of PyHab you are using (offline coding, movie presentation, other stimulus presentation), but five settings are universal to every version:

- maxDur: This is the maximum length of a trial, in seconds. Note that this is NOT maximum gaze-on-time, this is maximum amount of time between the start of trial and end of trial, period.
- maxOff: Maximum continuous gaze-off-time in a trial, in seconds. For example, if set to 2, trial will end if the participant looks away for two continuous seconds.
- minOn: Minimum gaze-on-time (not continuous). The maximum off-time only applies after the minimum on-time has been reached, but the maximum total duration applies regardless!
- prefix: Something, in single-quotes, to appear at the start of the name of the data files. This is basically just a way to identify which project the data file goes with.
- habituationDesign: PyHab currently supports habituation/dishabituation designs and Violation of Expectation (VoE) designs. Set this to "True" for habituation/dishabituation studies, and "False" for VoE designs.

### *b. Offline coding vs. PyHab-controlled stimuli*

Do you just want a replacement for jHab and xHab? Then you want the "NoStim" version of PyHab, which works almost exactly like xHab. The main difference in the settings is that an offline coding experiment doesn't require any information about stimuli, so the settings mentioned in the next section are absent. The habituation-specific and VoE-specific settings are still there, however.

### *c. Using movie files vs. programming your own stimuli*

If you are using PyHab to present stimuli, there are two ways to do it. The first way is relatively low-maintenance. Make your stimuli into movie files, one per trial. Quicktime (.mov) files are the only ones I have tested with PyHab, but any format supported by AVbin should work in principle (AVI, DivX, MPEG, WMV, etc.).

The movies can be any length. If the movies are short, the program will automatically loop them until the trial ends. At the start of trial, the movie will be paused on the first frame until you start the first gaze-on (see "Running a study in PyHab")

There are a few settings for movie-based stimuli:

- moviePath: If the movies are in the same folder as the script file, leave this as ''. If they are in another folder, this needs to be the path to that folder (e.g. '/Movies/')
- habMovieNames: A list of all the file-names of the movies you want to use for habituation or familiarization trials, enclosed in square brackets, each one in single quotes, separated by commas. (e.g., ['MovieName1', 'MovieName2', 'MovieName3'])
    - In a habituation design, the program assumes that you will only use one type of habituation trial per participant, so this is a list of your different habituation conditions (between-subjects).
    - In a VoE design, the program assumes that you want each participant to see every movie listed, so it's a within-subjects list. Note that you don't need to list the same movie multiple times if you want it displayed multiple times, that's a different setting (see the section on VoE designs).
    - For now, if you want different participants to see completely different sets of familiarization and/or test trials in a VoE design, you will need to create multiple

copies of the script, each with its own list. Future versions may be able to support multiple sets of hab or test stimuli within one script without needing to change the code (experienced programmers should be able to pull this off without too much difficulty).

- testMovieNames: Just like habMovieNames, but for test trials.
    - In a habituation design, PyHab assumes you will have on type of test trial per participant, but you can repeat that test trial multiple times (see below)
    - In a VoE design, it is again assumed that this will be within-subjects.
- introMovieNames: In a VoE design, you sometimes want a set of 'introductory' movies before the actual familiarization movies. If you have those, list them here. If not, leave blank.
- movieExt: The movie format, enclosed in single quotes (e.g., '.mov').

The alternative is to code your own stimuli using PsychoPy's built-in stimulus display capabilities. This is something you should only be considering if you already know how to code in Python, and ideally you have some experience with PsychoPy as well. Given that assumption, you want to look at the dispTrial function, and whatever you build make sure you can work on a global frame-count (contact jkominsky [at] g [dot] Harvard [dot] edu for an example).

## *d. Conditions and condition files*

Now things get a little complicated. There are two settings relating to conditions and randomization, and they work a little differently for habituation and VoE designs. Note that these settings do not apply to offline coding.

- randPres: True or False. If True, PyHab will expect a list of conditions and an external conditions file (see below), and will set the order of presentation according to that. If False, PyHab will ask you to enter a label for the condition, but all stimuli will be presented in the order they are listed.
    - For hab designs, setting this to False requires you to type in the index of the hab movie as the condition, and it will cycle through all of the movies listed in testMovieNames in order.

    - For VoE designs, setting this to False means PyHab will go through each set of movies in the order they appear in the list, and cycle through the list multiple times as needed (see below).
- condFile: If randPres is set to true, then you need a .csv file that has a list of the conditions (see below) and corresponding randomization information. That CSV file must be formatted in a very specific way, and it is different for hab and VoE designs.
    - For hab designs, the file consists of two columns. The first column is just a list of conditions corresponding to the conditions list in the next setting. The second column tells PyHab which hab movie and which test movie to use for a participant in that condition, with the following format:
      [1,1]
      The first number is which hab movie should be used for that participant (1 = first listed in habMovieNames, 2 = second listed, etc.).[1] The second number is which test movie should be used.
    - You can also set the "test" side to -1 if you have multiple different test videos you wish to present. If you do, it will go through the list of test videos in the order that they are listed in testMovieNames. (As of 6/23/17, there is currently no way to manipulate this order in hab designs except by changing the testMovieNames list.)
    - For VoE designs, the file is very similar, but the second column has a different format:
      [[1,2],[1,2,3],[1,2]]
      Where the first set of numbers is the order of the intro movies (if no intro movies, just leave as []), the second is the order of familiarization movies, and the third is the order of test movies, as they appear in the relevant list of names. Since it is possible to cycle through all of the movies multiple times, the same order will be used every time (see below).

[1] Programmers: You will probably be thinking that I should be counting from 0 because that's how Python does array indexes. For ease of use by non-programmers, I decided it was simpler to start from 1, and the code corrects appropriately, so '1' is indeed index 0 of the relevant list.

    - NB: You must have as many different numbers as there are entries in the corresponding list. So, if you have a list of three things in habMovieNames, you must have the numbers 1, 2, and 3 in the middle brackets.

- condList: A list of conditions or ‘keys’ corresponding to the first column of the condition file. Note that you need to enclose each one in single quotes here, but NOT in the condition file. When you run the program, these will appear in a dropdown list. Make sure each one matches what appears in the condition file EXACTLY (except for the single quotes).
    - NB: If you want to make the coder blind to condition, rather than creating a list of conditions that you use multiple times (for example, ‘A’, ‘B’, ‘C’, ‘D’), you can have a list of every single subject you plan to run. That way, all the person running the program sees is the subject code, but it doesn’t tell them what condition that actually connects to as long as they never open the condition file.

## *e. Habituation vs. VoE designs*

Finally, we come to the different settings for habituation and VoE designs. Whichever one you are not using is basically ignored, so you only have to worry about the settings for the design you are using.

For hab designs, there are two settings:

- maxHabTrials: The maximum number of habituation trials before the test trial.
    - Right now, PyHab only allows for one habituation rule. After the third trial, PyHab sums up the looking times over the first three trials and divides that by 2, and the resulting value is your habituation criterion. Starting on trial 6, it sums up the looking time over the most recent three trials, and if that is less than the habituation criterion, it proceeds on to the test trial. This setting is the number of trials it will go if that criterion is never met.
    - Future versions will let you set your own way of computing the habituation criterion.
- numTestTrials: Quite simply, the number of test trials. PyHab currently requires that all test trials be the same in a habituation design, but you can present that test trial multiple

times. Future version may allow you to present multiple different test trials in hab designs.

For VoE designs, there is only one setting, but it’s a big one:

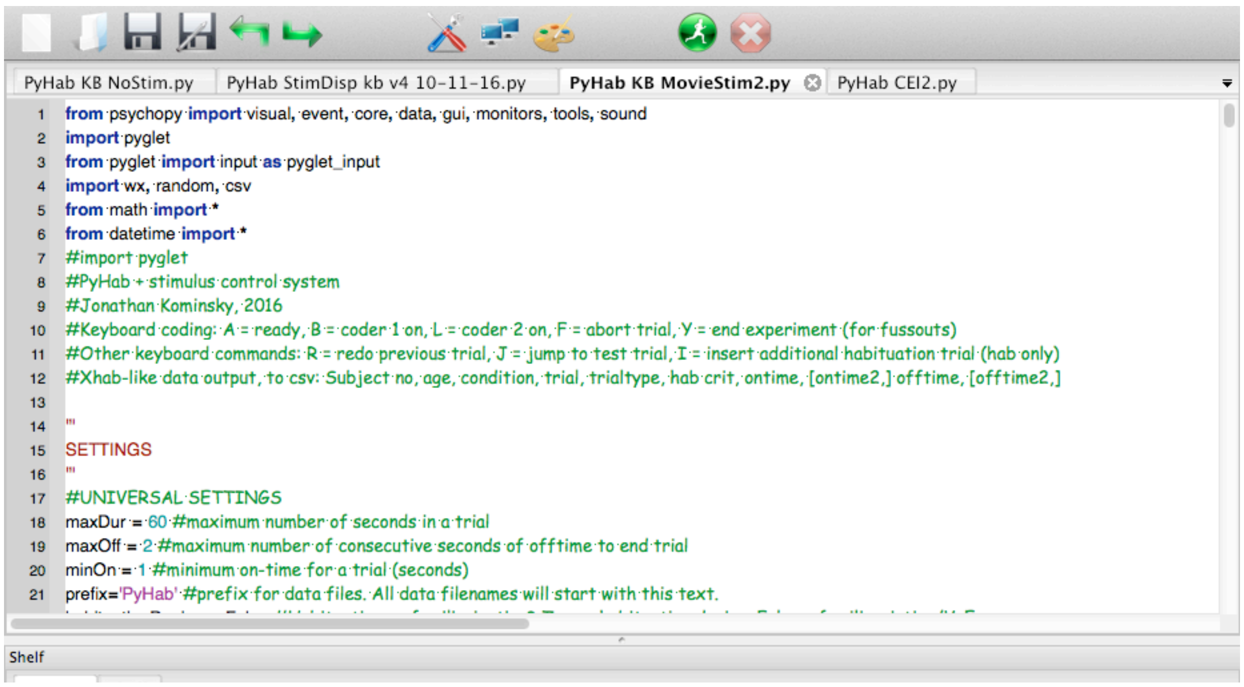
- trialOrder: This is basically the master list of how many of each kind of trial you want, and in what order. It will look something like this: [1,1,2,2,2,2,2,2,3,3,3,3]
    - 1 = an intro trial. 2 = a familiarization trial. 3 = a test trial
    - If you don’t have any of a given kind of trial, just don’t list them.
    - PyHab will cycle through the movies for each kind of trial in order (either the one pulled from the condition file, or by default the order they appear in the corresponding MovieNames list). So, if you have three familiarization moves (call them, A, B, and C), the six 2s in the trial order above would be ABCABC. This is why you only need to list each movie once.
    - If you want even presentation, make sure that each trial type appears a multiple of the number of items in that list.
    - **Advanced:** If you aren’t a programmer and really want to MacGyver a VoE design where you have non-overlapping sets of movies for a given type of trial between participants, you can have fewer instances of that trial type in the trialOrder than you have items in the corresponding MovieNames list. For example, if you list 8 movies in habMovieNames, but your trialOrder is [2,2,2,2,3,3], PyHab will only show the first four movies (either the first four in the list you write, or in the list order in the condition file). This will probably require you to list a given movie multiple times in the MovieNames list, but it is doable without extensive modification to rest of the code.

## 4. Running a study in PyHab

You (or your PI) built a study in PyHab and now you're ready to run some participants! These instructions are meant to be accessible to anyone, whether you're the one who designed the study or an RA helping to run it. Once the study is built, this is everything you need to know.

When you open PyHab, it will be in a PsychoPy window, and look like a bunch of code. To start the program running, click the big green button in the top bar with the silhouette of a running person. If, for whatever reason, something crashes, use the big red button next to it to kill the program.

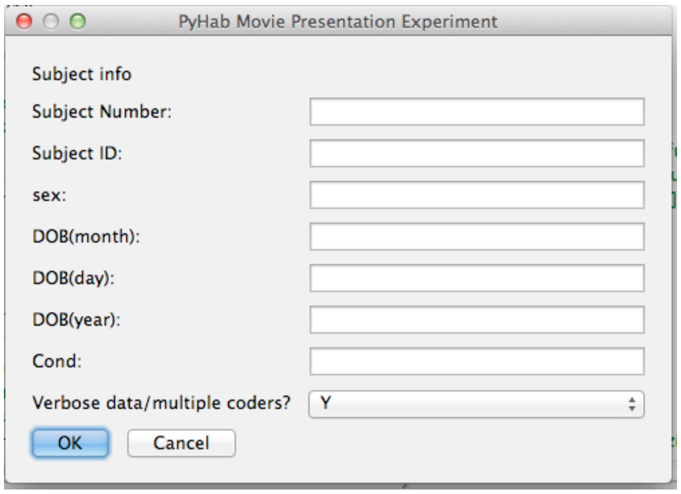
Here!



```
from psychopy import visual, event, core, data, gui, monitors, tools, sound
import pyglet
from pyglet import input as pyglet_input
import wx, random, csv
from math import *
from datetime import *
#import pyglet
#PyHab + stimulus control system
#Jonathan Kominsky, 2016
#Keyboard coding: A = ready, B = coder 1 on, L = coder 2 on, F = abort trial, Y = end experiment (for fussouts)
#Other keyboard commands: R = redo previous trial, J = jump to test trial, I = insert additional habituation trial (hab only)
#Xhab-like data output, to csv: Subject no, age, condition, trial, trialtype, hab crit, ontime, [ontime2,] offtime, [offtime2,]

'''
SETTINGS
'''
#UNIVERSAL SETTINGS
maxDur = 60 #maximum number of seconds in a trial
maxOff = 2 #maximum number of consecutive seconds of offtime to end trial
minOn = 1 #minimum on-time for a trial (seconds)
prefix='PyHab' #prefix for data files. All data filenames will start with this text.
```

When you hit the button, after a few seconds, a dialog box will pop up.


PyHab Movie Presentation Experiment

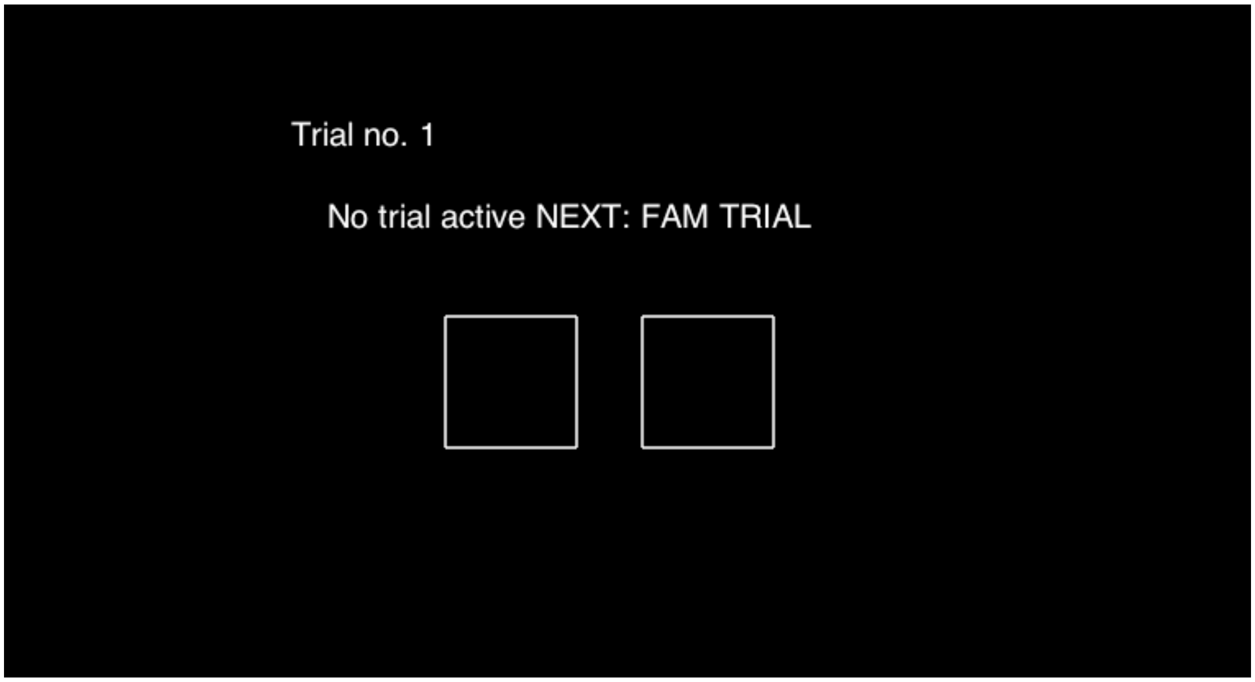
Subject info

| | |
|---|---|
| Subject Number: | |
| Subject ID: | |
| sex: | |
| DOB(month): | |
| DOB(day): | |
| DOB(year): | |
| Cond: | |
| Verbose data/multiple coders? | Y |

- This is what you will see if you are using the stimulus-presentation version. In the offline-coding version, there will be three more lines at the bottom, to record the date of the run (this is so you can code data days or weeks later but still accurately record the participant's age).
- The subject number and subject ID just go into the name of the data file. Unfortunately you can't leave them blank or the data won't save properly (working on that), but otherwise they don't affect anything about the program.
- Sex is also something you can't leave blank, but it's just recorded in the data file.
- The DOB fields ask for month, date, and year. **Year must be in two-digit format**. The program assumes anyone you are running in a study was born between 2000 and today's date, and it assumes that today's date is no later than the year 2099.[2]
- Cond will either be a text box, in which case you can put in whatever you want and it won't affect the program, or a drop-down menu with a list of the different conditions or subjects, which corresponds to the conditions file.
- "Verbose" mode will create a separate data file with the data gaze-by-gaze. The normal data file just summarizes each trial's total on-time, off-time, and number of gazes. The verbose file has the timestamps for each individual event. Also, if you are using multiple coders to get inter-rater reliability, you need to turn this on.

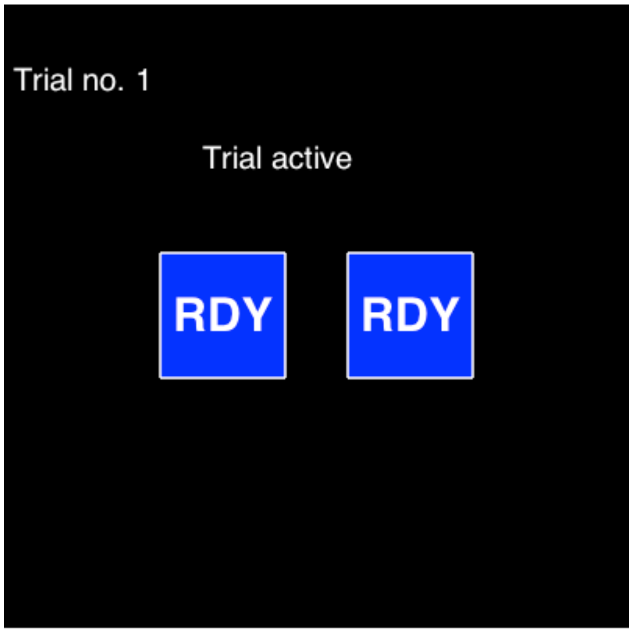
[2] If you are running this program in the year 2100 or later (over 80 years after its initial creation), the age-at-run computation code may need to be modified for people born in the 2090s. Also holy sh*t.

When you’ve filled in all the information, hit “OK” and the windows will open. You probably won’t be able to see the stimulus presentation window (if you’re using PyHab for stimulus presentation) but you will see a window that looks something like this.



This is the status window. The boxes show the current status for each coder. The left box is coder 1, the right box is coder 2.

Then, press the “A” key to play the attention-getter or start a trial. The boxes will turn blue and say “RDY”, like so:



To record the first gaze, hold down the “B” key. As long as a coder holds down their gaze key, the box will be green and say “ON”. During a trial, if the key is not held down, the box will be red and say “OFF”.
When the trial ends, both boxes will turn black until you play the attention-getter for the next trial. The boxes turning black is how you know that the trial is over.

## a. Keyboard commands

| KEY | FUNCTION |
|---|---|
| A | Play attention getter/put trial into “ready” |
| B | Coder 1 gaze on. Hold down as long as participant is looking at screen. First press starts trial from “Ready”. |
| L | Coder 2 gaze on. Coder 2 cannot affect trial start and end, those calculations are based entirely on Coder 1 |
| Y | End experiment. Quits and saves data at any point. The program should quit automatically at the end, this is designed for fuss-outs. |
| R | Abort/redo trial. If pressed during a trial, aborts it, codes it as “bad”, and the next trial will be a repeat of the aborted trial. If pressed between trials, will redo the last GOOD trial and replace it for computing hab criterion, etc. |
| J | Manually jump to first test trial (from hab or familiarization). Normally the program will advance to the test trial on its own, and this key should not be needed, but for offline coding it can be useful. |
| I | (Habituation experiments only): Insert additional habituation trial after test trial criterion has been reached. |

## b. Data

At the end of the experiment, or when you hit the “end experiment” button, the windows will automatically close (or attempt to, they do not always succeed), and all of the looking times will be saved to a .csv file in the same folder as the script. **Note that PyHab will over-write any existing csv file with the same name as the data file it is trying to create.** The name is determined by the subject number and ID, so make sure you have a different subject number and/or ID for every participant or move data files out of the script folder.

The default data format has the following columns:

- Snum: The subject number and ID combined into one string
- Months: Subject age in months
- Days: Subject age in days
- Sex: As entered in the start-of-experiment dialog
- Condition: As entered in the start-of-experiment dialog if typed in. If using an randomized presentation file, whatever is in the corresponding row of that file for the selected condition.
- Trial #
- GNG: Stands for Good/NoGood. If the trial was aborted or re-done, this will be set to "0" for that trial, and the following row will have the same trial again. Good trials have "1" in this column.
- TrialType: Depends on design. Hab design: 1 = hab trial, 2 = test. VoE designs: 1 = intro, 2 = familiarization, 3 = test. Either: 4 = "Y was pressed during this trial, so who cares".
- habCrit: For Hab designs, after trial 3 shows the calculated habituation criterion. For VoE designs, always 0.
- onTime: Total looking time for that trial (summed over any number of independent gazes).
- numGazes: Number of separate instances of the participant looking at the stimuli
- offTime: Total time participant was NOT looking at the stimuli during the trial.
- numOff: Number of looks away.
- The same four columns again with "B" after every one of them: All of the above, but for the second coder.

If you set "Verbose" to "Y" when you ran the program, you will also get up to two "verbose" .csv files that contain not only summary data of each trial, but the onset and offset of each individual gaze throughout the experiment. Note that there will be separate verbose files for each coder, if there were multiple coders. These files are very similar to the regular data files, except that each row represents a single look at the screen or look away, and there are multiple rows per trial. Most of the columns are as above, except for three:

- GazeOnOff: Indicates whether this is a look at the screen, or a look away. 1 = look at screen, 0 = look away.
- StartTime: When that look at the screen or look away initiated, relative to the start of the trial.
- EndTime: When that look at or look away ended, relative to the start of the trial.
- Duration: Duration of that look at or look away.

### *c. Preferential looking designs*

In addition to habituation and VoE designs that only examine whether or not a infant is looking at the stimulus, some studies want to examine, on certain trials, whether an infant is looking at one of two stimuli, on the left or right side.

There are separate scripts for preferential looking designs, with “PrefLook” in their names. They are very similar to the other PyHab scripts with a few important differences:

- They only support one coder at a time. No simultaneous dual-coding.
- B now means “gaze-on LEFT”, and M is “gaze-on RIGHT”
- For trials on which you are not doing preferential looking, you can simply use the B key as you would normally. On preferential looking trials, just use the B and M keys for left and right. Looks away are recorded when neither key is being pressed.
- The data format is re-ordered for the last several columns:
    - OnTime L: On-time for left (i.e., pressing B)
    - NumGazes L: Number of looks to the left
    - OnTime R: On-time for right (i.e., pressing M)
    - NumGazes R: Number of looks to the right
    - OffTime: As normal
    - NumOff: As normal.
- In the verbose data file, there are now three codes in the GazeOnOff column: 0, 1, and 2. 0 is still “off”, 1 is now “left”, and 2 is “right”.

### *d. Stand-alone reliability script*

Sometimes you want to calculate reliability, but might not be able to get two coders to sit down and simultaneously code the same video. (In the case of the preferential-looking version of PyHab, this is currently impossible anyways.)

It was always possible to manually calculate reliability from verbose data files, but I've also created a script that is just the reliability function that you would get by having two simultaneous coders, modified so that it takes any two verbose data files and spits out reliability statistics. It works on both regular looking time and preferential looking data.

When you run the script, the window will ask for the subject number and ID (as any other version of PyHab), and then three important things: the path to the verbose data files (must end with a / on OSX or \ on Windows), and then fields for the names of the two verbose data files (must include their file extensions, which should be .csv). The system is very precise and case-sensitive, and it will either work immediately or crash immediately. When it works it will create a "stats" file in the same folder as the script, and print out the four reliability stats to the PsychoPy output window.

## 5. Troubleshooting

If you don't know Python, troubleshooting PyHab could be difficult. It's not a particularly refined program, so it's easy for things to go weird. If you know Python, you can probably figure out the issue.

Here are a few issues you might encounter:

**Program crashes without ever opening dialog box.**
Most likely due to something being formatted incorrectly in the settings. PsychoPy will give you an error message (you may have to scroll up in the 'output' window to find it) with some blue text identifying what line the issue is on. Click on that to go straight to it, and see if you can figure out what's missing. A stray comma, bracket, or quotation mark can throw the whole thing off. Also, don't use the # symbol in the names of anything, because Python treats everything after that as a comment, not executable code.

**Program crashes right after hitting OK.**
Most likely due to the condition file being mis-formatted, because the first thing the program does after hitting OK is read those off and try to organize the movielists. Make sure you follow the formatting instructions in section 3d very carefully.

**Program crashes while trying to play a movie.**
Either due to the program trying to play a movie that doesn't exist, or an issue with the PsychoPy MovieStim function. Make sure the movie file is in the right directory for moviePath, the name is accurate in the MovieNames list, and the movie extension is accurate. If all of those are correct, make sure that your condition file is properly formatted. If that is also correct, it's a PsychoPy problem. Try using a different movie file format.
Note: Recently I have had issues on my own computer that seem to be related to the behavior of the ffmpeg codec in MacOS Yosemite. I haven't seen this issue on any computer other than mine, but I also can't fix it, and it crashes semi-randomly. If this starts happening to you, let me know.

**Program reaches the end, saves main data file, doesn't save verbose or reliability data or fails to close window.**
This is an ongoing issue I've been trying to solve. If it saves all the data but the windows don't close, that's some unknown issue with Python (particularly on Mac systems) that I have no ability to deal with. The reliability calculator should now never fall into a loop, as of June 2017, so it should always save the data successfully. Even if it does fail when calculating reliability, the good news is that it will still save all the data up to that point, so you should be able to manually calculate reliability in a pinch. Either way, alt-tab over to the script window and hit the big red X to kill the program, and if that doesn't work, command-option-escape (on a Mac) or ctrl-alt-delete and force quit.