peyeCoder

an open-source program for coding eye movements

User Manual

Software created by Rob H. Olson
User manual created by Ron Pomper & Christine E. Potter

https://rholson1.github.io/peyecoder/

Olson, R. H., Pomper, R., Potter, C. E., Hay, J. F., Saffran, J. R., Ellis Weismer, S., & Lew-Williams, C. (2020). Peyecoder: An open-source program for coding eye movements (Version v1.1.5). Zenodo. http://doi.org/10.5281/zenodo.3939234

Introduction

Many experiments use eye-tracking methods to study language acquisition. For instance, in the looking-while-listening (LWL) paradigm (Fernald et al., 2008), children are shown two images on a screen and hear a sentence identifying one image (e.g., “Where’s the ball?”). In order to measure language comprehension, we must identify where children are looking (e.g., left vs. right image) as they hear the sentence unfold over time. With this information, researchers can quantify language comprehension by measuring how quickly and how accurately participants fixate a target image after it is labeled. Participants’ fixation locations can be tracked automatically using an eye-tracker or manually by trained human coders.

peyeCoder is custom software built for coding gaze location on a frame-by-frame basis. It can be used on its own or in conjunction with an automatic eye-tracker (e.g., allowing researchers to hand-code trials where there was a loss in track status or entire participants who could not be calibrated). peyeCoder is nearly identical to its predecessor iCoder, which was developed at the Center for Infant Studies at Stanford University (PI: Anne Fernald) in 2003 by Jonathan Berger (with updates in 2009 by Wayne Packer, in consultation with Virginia Marchman and Christine Potter). Unlike iCoder, however, peyeCoder can be run on both 32- and 64-bit operating systems and can be run on Macs, Windows, or Linux computers.

This manual will teach you how to install and use peyeCoder software. Moreover, it will provide an introduction on how to code participants’ eye movements in experiments using the LWL paradigm. Included with the manual is an example video that has both automatic eye-tracking (Tobii) and hand-coded (peyeCoder) data.

peyeCoder was created in 2020 by Rob Olson with input from Ron Pomper and Christine Potter. Funding for the project was provided by Casey Lew-Williams and Susan Ellis Weismer. The project is open-source and the code is publicly available at: https://github.com/rholson1/peyecoder.

When publishing data that were coded using peyeCoder, please include the following citation in your publication:

Olson, R. H., Pomper, R., Potter, C. E., Hay, J. F., Saffran, J. R., Ellis Weismer, S., & Lew-Williams, C. (2020). Peyecoder: An open-source program for coding eye movements (Version v1.1.5). Zenodo. http://doi.org/10.5281/zenodo.3939234

If you have any questions, please send an email to ron.pomper@gmail.com

Installation

To install peyeCoder visit: https://rholson1.github.io/peyecoder/. There will be periodic updates to fix any bugs. Each update will be made available as a new release online. Until there is a stable version (which will be indicated in the release), please periodically check the website to make sure that you have the most recent release. You can do this by comparing the version of peyeCoder installed on your computer (in the menu bar select Help > About peyecoder) with the version included in the most recent release on the website.

peyeCoder can be installed by:

- downloading the source code and all of the dependent python packages
- using the Python Package Index: pip install peyecoder
- or by downloading an executable version

The most common way to install peyeCoder is to download the appropriate executable file (for Mac, Windows, or Linux) from: https://github.com/rholson1/peyecoder/releases/

The executable file can be placed anywhere on your computer. Double-clicking the file will open the software.

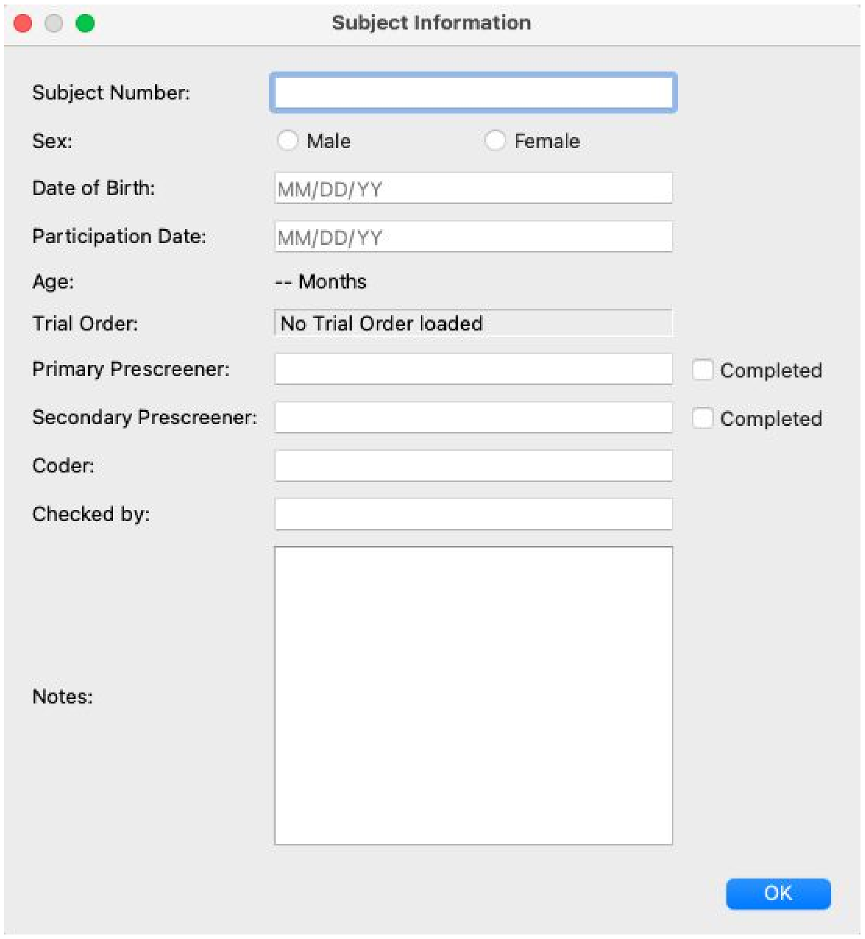
> Note: it can take approximately one minute for the software to load. macOS will initially block peyeCoder from opening, because it was *not* installed via the App Store. If you are using peyeCoder for the first time on a Mac, double-click on the file, then open System Preferences, select Security & Privacy, go to the General tab, and near the bottom of the window click “Allow anyway.”

Opening Screens

Upon startup, peyeCoder will open with two windows.

Subject Information

The first window, titled **Subject Information** contains both required and optional fields. The following fields must be filled to avoid generating errors: Subject Number, Sex, Date of Birth, Participation Date, and Trial Order (which must be loaded by dragging in a .txt file, discussed in detail below). The remaining fields (Primary Prescreener, Secondary Prescreener, Coder, Checked by, Notes) are optional. The Notes section can be used to record any useful information, such as any problems with the video or trials that were particularly difficult to code, but it should be noted that information from the Notes field is stored with file, but not exported in the data.



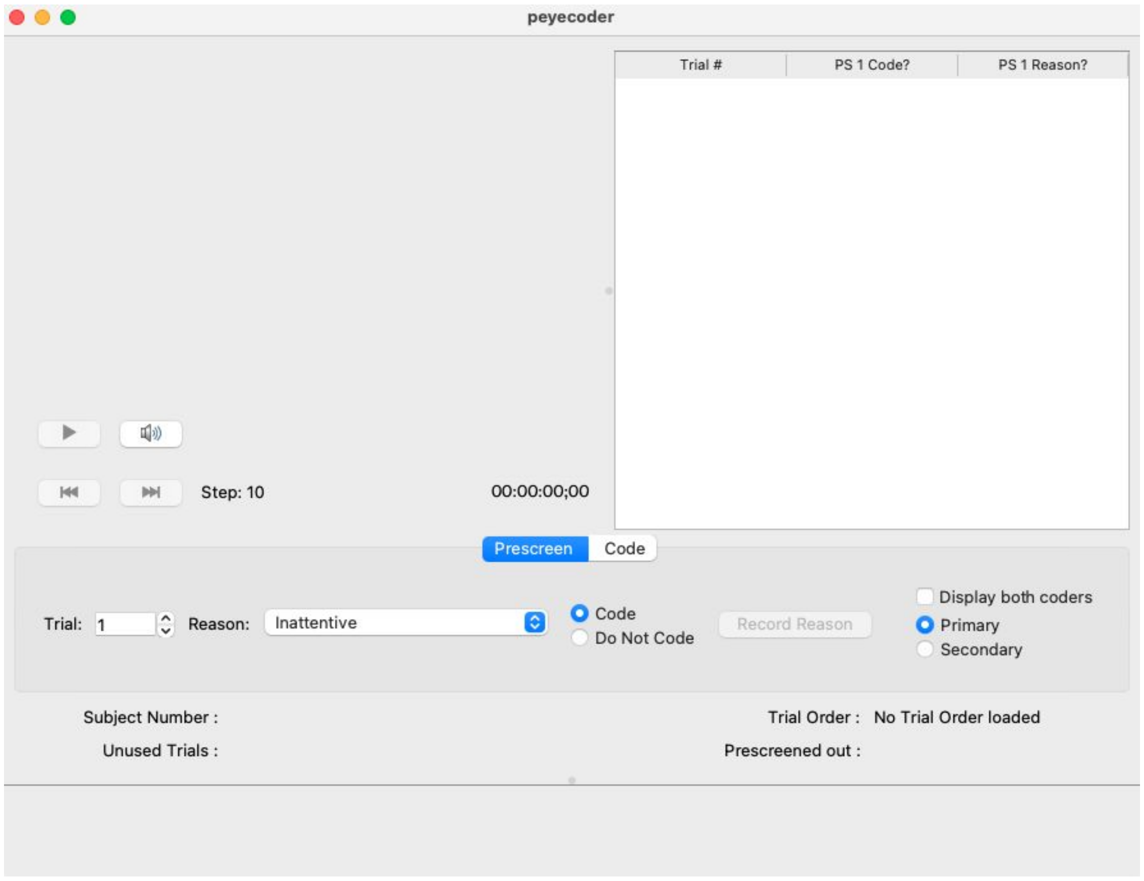
Subject Number:		
Sex:	◯ Male ◯ Female	
Date of Birth:	MM/DD/YY	
Participation Date:	MM/DD/YY	
Age:	-- Months	
Trial Order:	No Trial Order loaded	
Primary Prescreener:		☐ Completed
Secondary Prescreener:		☐ Completed
Coder:		
Checked by:		
Notes:		

Figure 1. Screenshot of Subject Information window without any information entered.

After you have entered all of the Subject Information, this window can be left open or closed (either by clicking the red x in the top left corner or by clicking the *OK* button in the bottom right corner). Once closed, you can access this window by clicking Edit > Subject Info in the menu bar or by using the keyboard shortcut command+I.

Peyecoder

The second window, titled **peyecoder**, is where all of the coding occurs.



Step: 10

00:00:00;00

Trial: 1 Reason: Inattentive

[✓ Code] Do Not Code

☐ Display both coders

[✓ Primary] Secondary

Subject Number :

Trial Order : No Trial Order loaded

Figure 2. Screenshot of peyecoder window when the Coding tab is selected.

The *left panel* is where the video will be displayed, once it is loaded. Beneath the video are buttons to Play/Pause the movie, mute/unmute the audio, advance the video one frame backward or one frame forward, display the number of frames that will be skipped when using the “jump” feature, and the Time Code associated with the current frame of the video.

> Note: the Time Code is in the format hours : minutes : seconds ; frames. peyeCoder will automatically detect the frame rate and adjust when to increment the seconds counter. For instance, most cameras record at 30 (actually 29.97, see below) frames per second (so 33.333 ms elapse every frame) in which case the frames will count up to 29, before resetting back to 0 and incrementing the second by 1. Some video recording methods (like Zoom) record at a slower 25 frames per second (so 40 ms elapse every frame) in which case the frames will count up to 24, before resetting back to 0 and incrementing the second by 1.

The *right panel* will display either Prescreen responses or Code responses, depending on which tab is selected. peyeCoder defaults to open to the Code tab.

The *middle panel* will display coding options underneath the left and right panels. These options change depending on whether Prescreen or Code tab is selected.

- When Prescreen is selected (see Figure 2, Prescreening section for more details), coders can manually enter or set the Trial Number (using the up and down arrows), select a Reason for marking a trial (using the drop down menu), indicate whether the trial should or should not be coded (using the drop down menu), and indicate whether they are the primary or secondary prescreener. Once all of the appropriate information has been entered, the data can be recorded by clicking the Record Reason button or by pressing the Enter/Return key.
- When Code is selected (see Figure 3, How to code section for more details), coders can manually enter the Trial Number or increment it (using the up and down arrows), set the trial status to be on/off (using the drop down menu or keyboard shortcut), and set the Response (using the drop down menu or keyboard shortcut), which is the participant's gaze location. Once all of the appropriate information has been entered, the data can be recorded by clicking the Record Event button or by pressing the Enter/Return. This data will be associated with the current frame (Time Code) of the video.

Figure 3. Screenshot of peyecoder window when the Code tab is selected.

The *bottom panel* will display information that was entered in the **Subject Information** window, including the Subject Number, the Trial Order, and trials that should not be coded because they were marked as Unused (in the Trial Order) or Prescreened out. This panel will display any error messages when coding. For more information on errors see the Automatic Error Checking section.

Menu Bar

Finally, several additional features can be found in the **menu bar**.

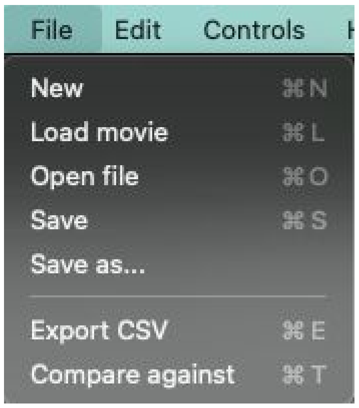


Figure 4. Screenshot of File menu options.

Selecting File in the Menu Bar will allow you to open a new coding window (note: this will close any currently open file, see Opening Multiple Instances of peyeCoder section if you need to have multiple files open at once), load the movie, open a previously coded and saved file, save your current coding (as a .vcx file), export your data as a .csv file, or compare your coding against another previously coded and saved file.

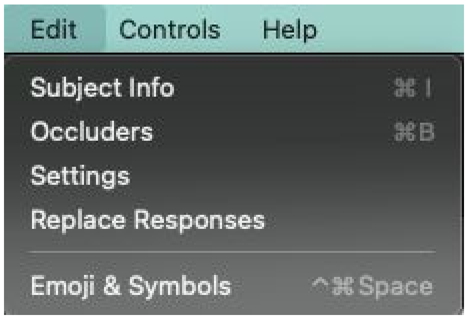


Figure 5. Screenshot of Edit menu options.

Selecting Edit in the Menu Bar will allow you to re-open the Subject Info menu, add/remove/adjust Occluders, open the Settings menu, and Replace Responses.

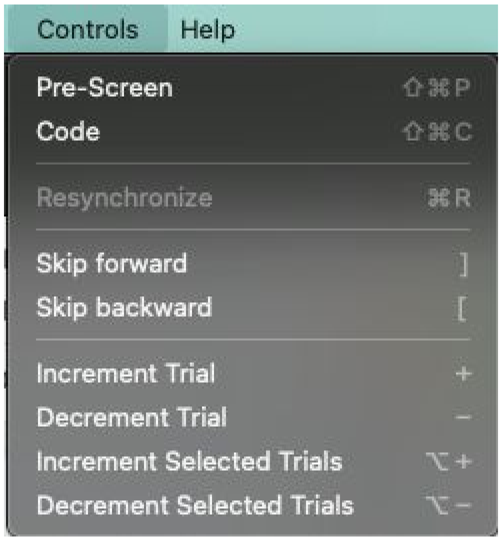


Figure 6. Screenshot of Controls menu options.

Selecting Edit in the Menu Bar will allow you to switch between the Pre-Screen and Code tabs in the peyeCoder window, Rescynchronize your Time Code (note: this is only necessary if the Time Code skips or duplicates a frame number at some point in your video), Skip a pre-specified number of frames (the default is set to 10) forward or backward in the video, increment or decrement the trial number in the Coding tab of the peyeCoder window, and increment or decrement the trial number for a coded response or responses that are selected (highlighted) in the Coding tab of the peyeCoder window.

Other functions and fixes

Editing responses, shortcut keys, and step frames

Figure 7. Screenshot of Settings window.

Selecting Edit > Settings in the menu bar will open the Settings window. Here you can manually adjust how many frames will be skipped when you jump through the video. The default is set to 10 frames. Pressing the left and right arrow keys on your keyboard will advance you through the video one frame at a time. Pressing the left brack [or right bracket] keys will advance the video by the specified number of frames (this is useful when you do not need to code an entire trial and need to skip a prespecified amount of time at the beginning of each trial). You can also specify gaze locations on the screen (Response) and their associated shortcut key. The default responses are based on the original iCoder. Finally, you can specify which shortcut key will toggle the Trial Status between on and off.

Editing prior responses

Individual responses can be changed by highlighting the appropriate line (which will jump to that point in the video and then display all values in the drop down menus), making the necessary changes, and then recording the new event and deleting the prior one. However, this can be inconvenient if multiple events need to be changed. The trial number can be adjusted for multiple responses at once by highlighting a series of events and choosing "Increment/Decrement Selected Trials" from the Control menu (or using the shortcut Option +/-).

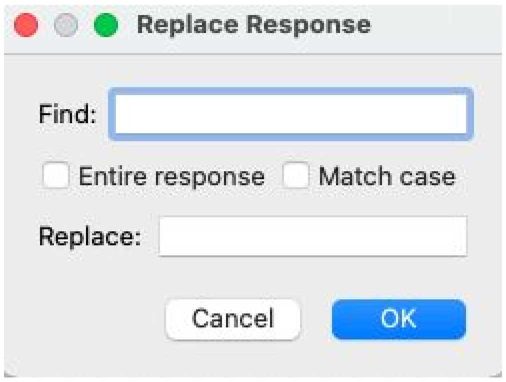
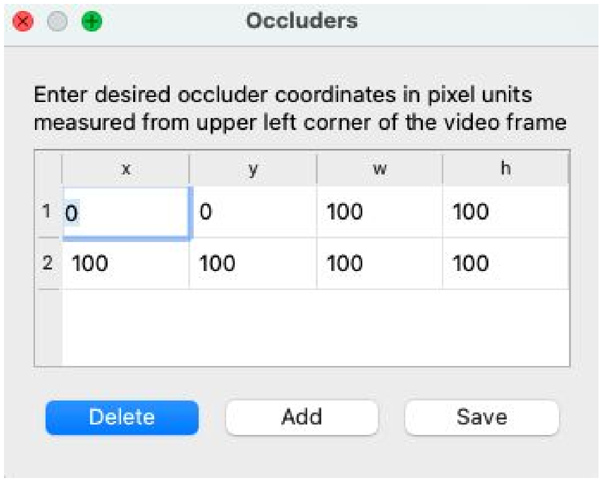


Figure 8. Screenshot of Replace Responses window.

It is also possible to change all responses of a particular type, e.g., changing 'right' to 'topRight.' To do this, select Edit > Replace responses in the menu. This will allow you to enter in the original response label that was used and what it should be replaced with. Additionally, you have options to constrain the Find process. Checking *Entire response* will only identify responses to be changed where the entire text of the response matches the entry in *Find*, while leaving the box unchecked will identify responses where only a subset of the text matches the entry in *Find* (e.g., Right in 'topRight' and 'bottomRight'). Checking *Match case* will only identify responses to be changed where the entire text matches the entry in *Find* in both upper and lower case marking (e.g., 'Right' will not match 'right' if this option is checked).

Occluders


Occluders

Enter desired occluder coordinates in pixel units measured from upper left corner of the video frame

	x	y	w	h
1	0	0	100	100
2	100	100	100	100

Figure 9. Screenshot of Occluders window.

Selecting Edit > Occluders in the menu bar will open the Occluders window. This will allow you to superimpose a gray box on the video that is displayed over the video on the left panel of the peyecoder window. The location of this occluder can be adjusted by its horizontal (x) and vertical (y) distance from the top left corner of the screen in pixels. Larger x values move the occluder further to the right. Larger y values move the occluder lower. The size of the occluder can be adjusted by its width (w) and height (h) in pixels. Additional occluders (rows) can be added by clicking the Add Occluder button. And existing occluders can be removed by clicking the Delete Occluder button.

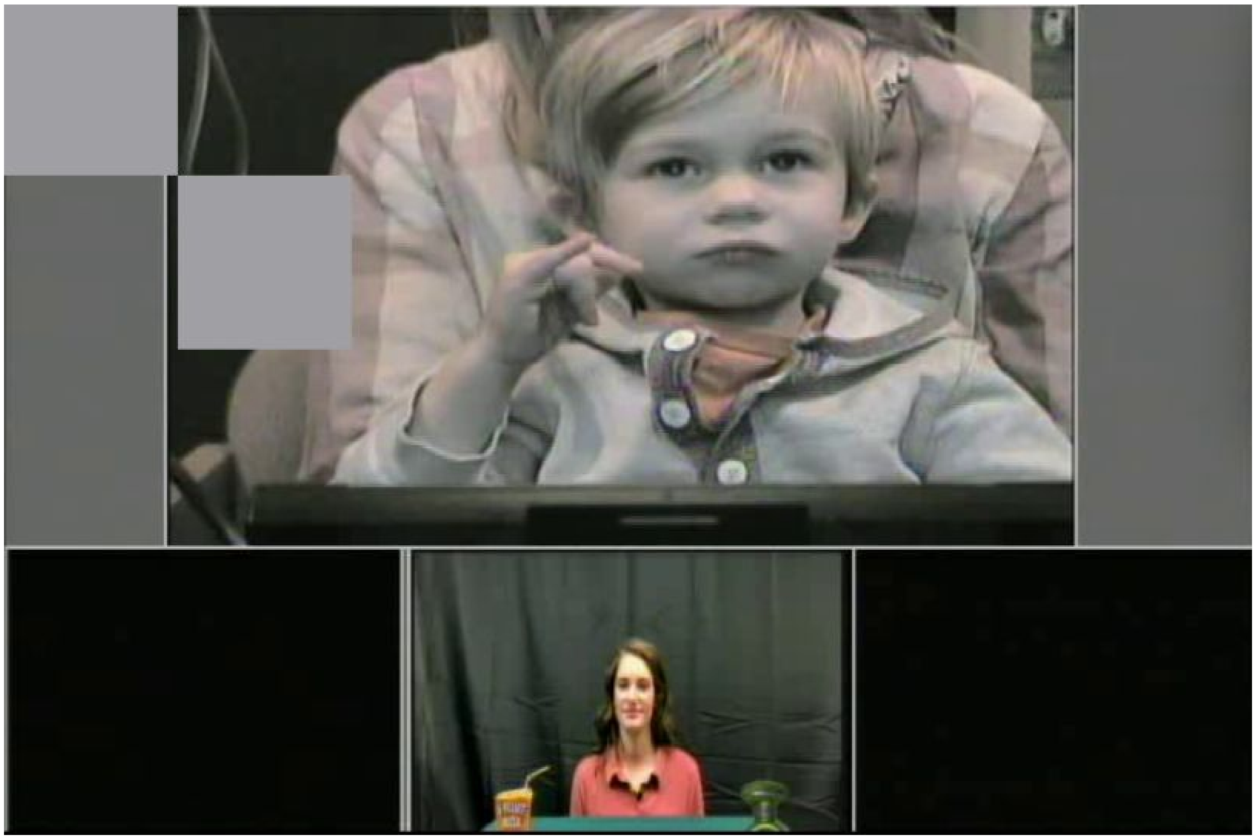


Figure 10. Screenshot of Occluders displayed over the video. The top left square is from line 1 in *Figure 9*. The square below it and to the right is from line 2 in *Figure 9*.

Resynchronizing (this should be rare)

Depending on how videos are recorded and compressed, some labs have had issues with losing frames over the course of a 5-minute session, and the time code that appears on the video may not match what the coder records in the time code column. It is important to remember that peyeCoder does not read anything from the video directly, and that every trial is independent, so this issue does not necessarily create any problems in the data. The goal is to generate output that indicates when events happen *relative to the onset of that trial*, and the purpose of the timecode is to help the human coder; peyeCoder is simply tracking how many frames elapse for each event that occurs *relative* to the first event of the trial.

Note: because peyeCoder tracks everything relative to the first frame of each trial, the loss of a frame when recording the video will only affect data for the trial where the loss occurred (e.g., the loss of the fourth frame in the video will cause peyeCoder to think that events occurred one frame/33 ms earlier than they actually occurred in real time).

As an example, it could happen that a frame was dropped in the recording (e.g., the timestamp in the video skips frame 00:00:00.01 to 00:00:00.03). So, then what the video shows as frame 00:00:00.03, peyeCoder would interpret as 00:00:00.02. Resynchronizing would allow you to correct this error and make the recorded timestamp match what is shown in the video.

To update the time code, select Controls > Resynchronize in the menu bar or use the shortcut command-R. This will bring up a pop-up box (Figure 11) that allows the coder to enter the current time code, and all subsequent responses will be linked to this time.

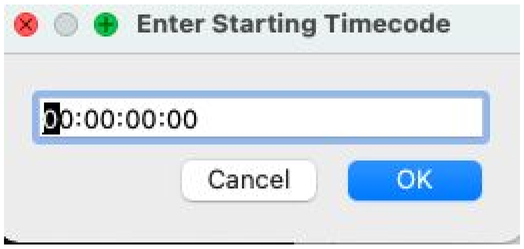


Figure 11. Timecode box that appears when a file is first opened or at any time that the coder needs to resynchronize the movie.

Note: peyeCoder keeps track of the exact frame in the video where you resynchronized. This means that if you need to go back to earlier in the experiment (i.e., before you had to resynchronize) then peyeCoder will automatically adjust the timestamp back to the original synchronization.

Complete list of keyboard shortcuts

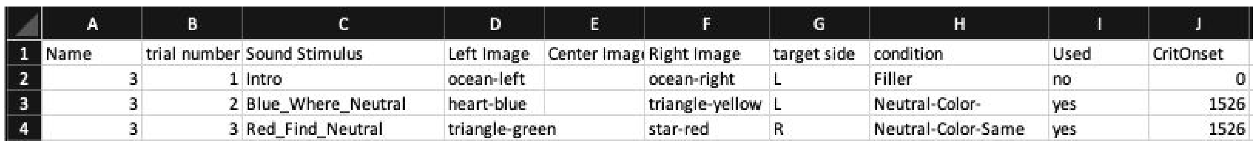
Command + N	New file
Command + O	Open file
Command + L	Load Movie
Command + S	Save
Command + Q	Quit
Command + I	Subject Info
Command + T	Compare Against
Command + B	Add occluders
Command + ,	Settings menu

Command + R	Resynchronize
Command + E	Export .csv file
Shift-Command + P	Prescreen tab
Shift-Command + C	Coding Tab
1	*Left*
2	*Off*
3	*Right*
4	*Away*
5	*Center*
6	*Change Trial Status (on/off)*
Option + (=/+)	Increase trial # for selected trials
Option + (-/_)	Decrease trial # for selected trials
+	Increase trial #
-	Decrease trial #
]	Skip *n* frames forward
[	Skip *n* frames backward

Table 1. List of shortcuts. Note that the italicized numerical keys that correspond with gaze location responses are the defaults, but can be edited

Trial Order File

When entering Subject Information, one of the required entries is a .txt (or .csv) file providing information about all of the trials in the experiment. An example file has been included. This information is important both for coding (by identifying which trials should and should *not* be coded) and for data exporting (by identifying target image location, onset of the target word, etc.).



	A	B	C	D	E	F	G	H	I	J
1	Name	trial number	Sound Stimulus	Left Image	Center Imag	Right Image	target side	condition	Used	CritOnset
2	3	1	Intro	ocean-left		ocean-right	L	Filler	no	0
3	3	2	Blue_Where_Neutral	heart-blue		triangle-yellow	L	Neutral-Color-	yes	1526
4	3	3	Red_Find_Neutral	triangle-green		star-red	R	Neutral-Color-Same	yes	1526

Figure 12. Screenshot of information entered for the first 3 trials of an experiment.

Below are descriptions for each column in the order they appear in the .txt file. Each column header must be labeled according to this convention

Name

- name of the trial order
- this is helpful for coding (e.g., peyeCoder will display the name of the loaded order, which should match any order information displayed in the video) and for data analysis (e.g., for testing whether there are differences between different experimental orders)

Trial Number

- 1, 2, 3, etc.
- this is necessary for both coding (i.e., peyeCoder will generate an error if coders code a trial number beyond the range of the order) and for data analysis purposes (e.g., plotting results on a by-trial basis)

Sound Stimulus

- description of the audio that the child heard
- this is not used for coding and is only used for data analysis purposes (e.g., subsetting to only include trials with a specific word)

Left Image

- name of the image on the left side of the screen (from the participant's perspective)
- this is only used for data analysis purposes

Center Image (optional)

- name of the image displayed in the center of the screen
- this can be left blank and is only used for data analysis purposes

Right Image

- name of the image on the right side of the screen (from the participant's perspective)
- this is only used for data analysis purposes

Target Side

- location of the target image (from the participant's perspective)
- when using the default response options can be L, R, or N (N is used for filler or attention-getting trials for which there is no target image)
- when using custom response options this can be anything (e.g., bottomLeft, bottomRight, topLeft, topRight)
- this information is only used for data analysis purposes; peyeCoder uses this to convert coded responses into accuracies upon exporting (see next section on Data Export)

Condition

- name for the trial type
- this is only used for data analysis purposes (e.g., comparing children's accuracy between conditions)

Used
- yes, no
- this is for coding purposes (i.e., peyeCoder will generate an error if the coder trials to code a trial that is not included)
- peyeCoder will NOT export any data from a trial that is coded but marked as "no" in the Used column
- this feature is useful when filler or attention-getting trials are included in an experiment

CritOnset
- number quantifying the time in milliseconds from when the coder begins to code the trial until the onset of the critical part of the trial (often a target word, e.g., the onset of *baby* in a sentence such as *Where is the baby?*)
- this is used for data analysis purposes - peyeCoder will use this number to create a centered time variable when exporting the data (i.e., time = 0 not at the beginning of the trial, but rather the onset of the critical word)
- for instance, if there is 2,000ms of silence and a 1,000ms carrier phrase (e.g., "Find the") before the onset of the target word ("ball") and coding begins at the onset of the trial then CritOnset=3000
    - alternatively, you may choose to have coders skip an initial portion of the trial and have coding begin 500ms into the trial (i.e., coders skip ahead 15 frames from the onset of the trial before coding) then CritOnset=2500

You can include additional columns beyond those specified here. These can be used for your own purposes, they will not, however, be read by peyeCoder and will not be included when exporting the data.

Note: both iCoder and peyeCoder orders can have multiple rows for the same trial; this allows you to export data for a trial using two different critical onsets. Additional columns can be added to peyeCoder, they will not be processed by peyeCoder (and therefore will not be included when exporting).

Prescreening

Prescreening is the process of determining whether there are trials that do not need to be coded. Some labs choose to do this, others include all trials where data could be collected (much like studies using an automatic eyetracker, where sessions may not have been videotaped to have this option). peyeCoder includes the option for the video to be prescreened by zero, one, or two individuals and for trials to simply be marked or to be excluded altogether.

Most often, prescreening involves watching the video in real-time and flagging trials that the prescreener thinks should not be included in the data to be analyzed. For every trial, the prescreener has the option of flagging that trial and choosing 'Code' or 'Do Not Code' and marking a reason (Reasons: *Inattentive, Child Talking, Parent Talking, Parent Interference, Child Not Looking Before Sound, Equipment Malfunction, Experiment Ended Early, Other*, Figure 13).

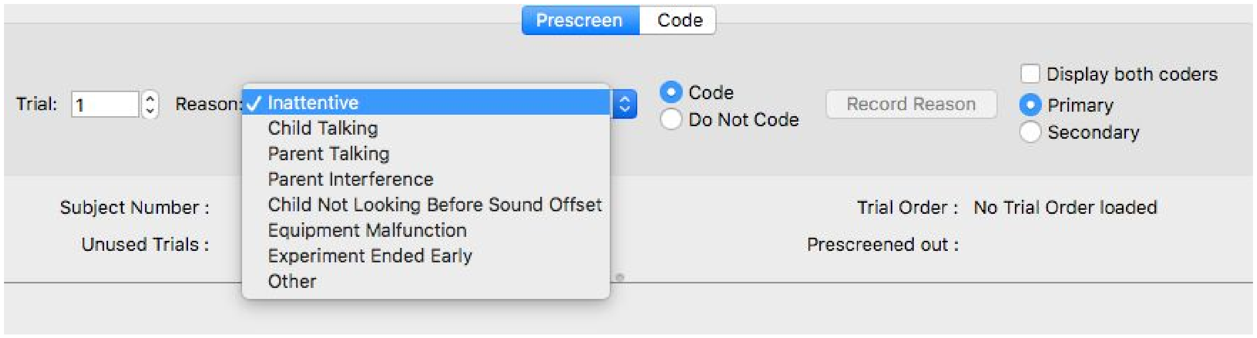
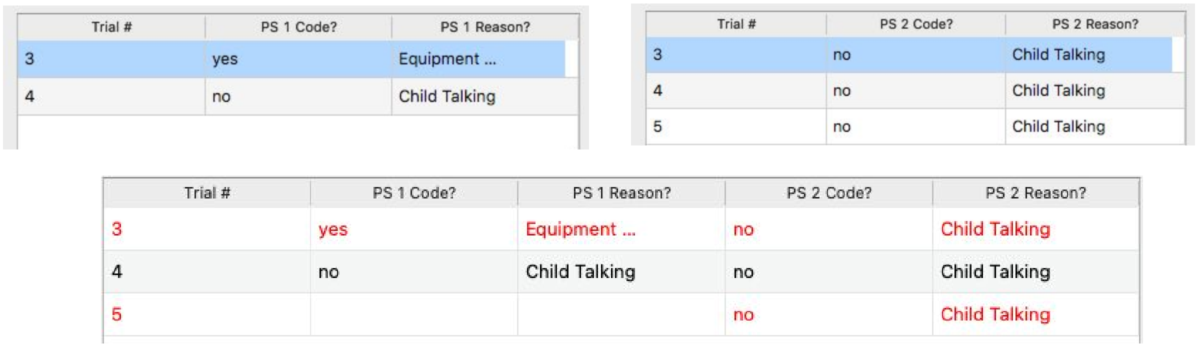


Figure 13. Menu options available during prescreening

A primary and secondary prescreener can make decisions independently, using the toggle menu to indicate which person marked the trials. When the 'Display both coders' box is checked, responses from both prescreeners appear so that any differences can be noted (see Figure 14 for example). Lab standards will vary for how much they choose, for example, to exclude trials where the child is looking away for long stretches, as that information can also be gleaned from coding the eye movements, but may waste the coder's time. A more conservative approach may be to only exclude trials with outside interference (e.g., times when the child was prevented from hearing the stimuli, when the parent directed their behavior, or when there was a problem with the presentation of the stimuli), as children's eye movements in those cases may reflect factors other than what the researchers intend.



Trial #	PS 1 Code?	PS 1 Reason?
3	yes	Equipment ...
4	no	Child Talking

Trial #	PS 2 Code?	PS 2 Reason?
3	no	Child Talking
4	no	Child Talking
5	no	Child Talking

Trial #	PS 1 Code?	PS 1 Reason?	PS 2 Code?	PS 2 Reason?
3	yes	Equipment ...	no	Child Talking
4	no	Child Talking	no	Child Talking
5			no	Child Talking

Figure 14. Sample prescreening from Prescreener 1 (left) and Prescreener 2 (right), and then the combined output (center). Note that on trial 3, there is disagreement about whether or not the trials should be coded, agreement on trial 4, and only Prescreener 2 found anything to mark on trial 5. peyeCoder will highlight trials where there are disagreements in red (this will only occur when 'Display both coders' is checked).

Eliminating trials. Trials marked 'Do Not Code' by the primary prescreener will appear in the list of unused trials along with those that are indicated as not used by the order file and will be exported as a blank line in the output. Note that while the secondary prescreener has identical menu options, peyeCoder reads only from the primary prescreener in determining which trials should be coded and what information should be exported.

Exporting data. Any trial marked as "Do Not Code" is not exported. Any trial marked as "Code" will be exported and will include the Reason in a column titled Prescreen Notes (e.g., Child Talking).

How to code

The main goal of peyeCoder is to allow the coder to classify the location of the child's gaze at each time point in the video. It is often difficult to tell from a single frame where a child is looking, so it is generally advisable to also look at the surrounding frames and even to watch part of the session in real time when necessary to get a sense of that particular child's looking patterns. Individual labs are likely to have their own criteria and systems for coding; below, we describe one common approach, with references to the example coding key.

Recording events

It is unnecessary (and impractical) to record a response for every frame in a 5-minute session. Instead, the coder is tasked with indicating every time there is a meaningful change. This change is either related to the appearance or disappearance of stimuli (e.g., changing the trial number) or in the child's behavior (e.g., no longer looking at the left picture). The guiding principle that we use to maximize consistency between coders is that we mark events based on the *first indication of change*. For every event, the coder must indicate four things: **Trial** number, **Trial Status**, **Response** (where the child is looking), and the corresponding **Time Code**. Recording a series of events will create a list like the one in Figure 15 below, each timestamped to show the time code from that frame in the video.

Figure 15. Lists of events from Trial 1 of the coding key.

Trial

peyeCoder is organized by trial, so it is important to make sure that the looking behavior can be matched with trial information from the order file. Any trials that are marked as unused in the order or have been flagged in prescreening do not need to be coded and should be skipped. Because this may happen, it is important to make sure that the trial number that is recorded matches the events of the video.

Trial Status

To ensure that information about the child's gaze is time-locked to the presentation of the experimental stimuli, the coder must indicate the beginning and end of a trial, using the Trial Status menu. The first frame for which the Trial Status is coded as *on* will be used as the reference point for the timing of the Critical Onset. Often, this may be signaled by the appearance of the images in a PIP box. It is also possible to use a different time point as the starting point of the coding window, such as the onset of the auditory stimulus. In the example provided, we chose to mark the onset of the trial as 18 frames *after* the first indication of the target images. This window which was chosen to offset a 3-frame lag between the appearance in the recording vs. in the child's view and to avoid coding 500ms where the images are displayed in silence that was not going to be included in the baseline window in later analyses. Similarly, the end of a trial may be indicated by the disappearance of stimuli, a time-based criterion, or any other convention on which coders can reliably agree, and the last frame of trial should be marked *off*. Note that in our key, the first indication that the trial has ended is a subtle change in brightness and pixelation that might not immediately be obvious.

Response

The primary challenge for the coder is to determine where the child is looking. To eliminate the need for mental rotation, we code right vs. left from the CODER's perspective, and the order file is organized accordingly (see Order File and Left-Right Inversion sections for more information). For a typical LWL study, there are likely to be three responses of interest: looking to the image on the **right**, looking to the image on the **left,** or looking at neither image (**off**). By default, peyeCoder also allows for two additional responses: **center** (if there is a fixation stimulus in the center of the screen, for example), and **away** (typically used to differentiate between when the child is merely shifting between pictures, i.e., *off* the images, vs. not paying attention at all, i.e., looking *away* from the screen). Note that while older versions of iCoder required the separate use of offs and aways, we do not recommend using that distinction, and it is unnecessary in peyeCoder. The coder also has the option of manually specifying other possible responses (e.g., topRight) in the settings if these are appropriate for a particular study (see editing responses for more details).

Possible response options can be categorized as either fixations (*right, left, center*), transitions (*off*), or inattentiveness (*away*). A fixation is defined as the point at which the child is clearly looking at a defined location on the screen, usually indicated by eyes that are clearly visible, open, and not in motion. Transitions are marked as soon as the child begins to shift gaze away from the location, often indicated by blurring of the eyes or movement of the eyelids. The optional *away* response is used when the child is not looking at either image for an extended period of time (e.g., fixating on the camera, looking at a parent) or when the eyes are not visible (often because the child moves out of frame or blocks their own eyes by pointing). In the files provided, one version of the key includes *away* events; the primary key marks the child as *off* for all those same frames.

What to do when the child's eyes are not visible

It is relatively common for children's eyes to be obstructed at some point during the session (e.g., the child is blinking, moving out of frame, places an arm in front of their eyes while pointing, etc.). The most conservative approach is to mark any frame where the coder cannot see at least one of the eyes as *off* (or *away*). However, because of physiological constraints, we assume that children cannot move their eyes to a new point and back in <300ms. Therefore, our current standard is to use an 8-frame guideline. That is, if the obstruction lasts 7 frames or fewer (and the child is looking in the same location before and after the obstruction), we code as if there was no obstruction. The reasoning is that they could not realistically have transitioned off the image and back in that amount of time. However, if there are ≥8 consecutive frames where the eyes cannot be seen, we mark all frames where the eyes are hidden as *off* and code the first frame where the eyes are visible as a fixation (see Trial 23 in our example coding file).

Automatic error checking

peyeCoder is designed to check for mistakes and to display errors in red. The primary errors are: missing participant information (e.g., not including Order information, or the participants' DOB), failing to mark the end of a trial by recording an event with Trial Status *off*, and illegal sequences of responses. Illegal coding sequences include two events with the same timecode, repeated responses (e.g., two consecutive *right* responses, unless the second one is when the trial status is turned *off*), out-of-order trial numbers, or changes between fixations without an intervening *off* to signal the transition (i.e., *right-left* or *left-right;* direct transitions involving *away* or *center* are permitted). See differences from iCoder for more information about permissible vs. impermissible sequences below.

Reliability

To evaluate consistency across coders, peyeCoder allows you to compare two files that were separately coded for the same video. The reliability report generates both numerical measures of reliability and a list of errors that highlights differences between the files. In general, reliability comparisons serve two primary purposes: (1) providing feedback to new coders who are learning coding conventions and (2) assessing (and reporting in publications) the interrater reliability that coders are able to achieve.

Numerical measures

The reliability report includes three separate values: Frame agreement, Comparable trials, and Shift agreement.

Frame agreement

Frame agreement is designed to capture the global agreement between coders about where the child is looking. It is defined as the ***percentage of coded frames for which the two files have the same response*** for all coded trials. It is important to note that for the purposes of Frame agreement, *off* and *away* are considered equivalent, as the measure is intended to describe whether coders are consistent in deciding when the child is looking at an image (*right*, *left*, or *center*).

Comparable trials

The measure of Comparable trials is computed as the ***percentage of trials for which the two files have recorded the same number of events****.* This measure is useful because it provides information about how many trials were included in the calculation of Shift Agreement (see below). It can also provide insight into how consistent the two coders were in what they considered ‘away’. Most commonly, trials are *not* comparable because one coder included an away that the other coder did not (e.g., Coder A assumed the child was scanning an image and looking at the edge, while Coder B thought the child looked away and then looked back). Trials can also be non-comparable because there is disagreement around the beginning or end of a trial.

Shift agreement

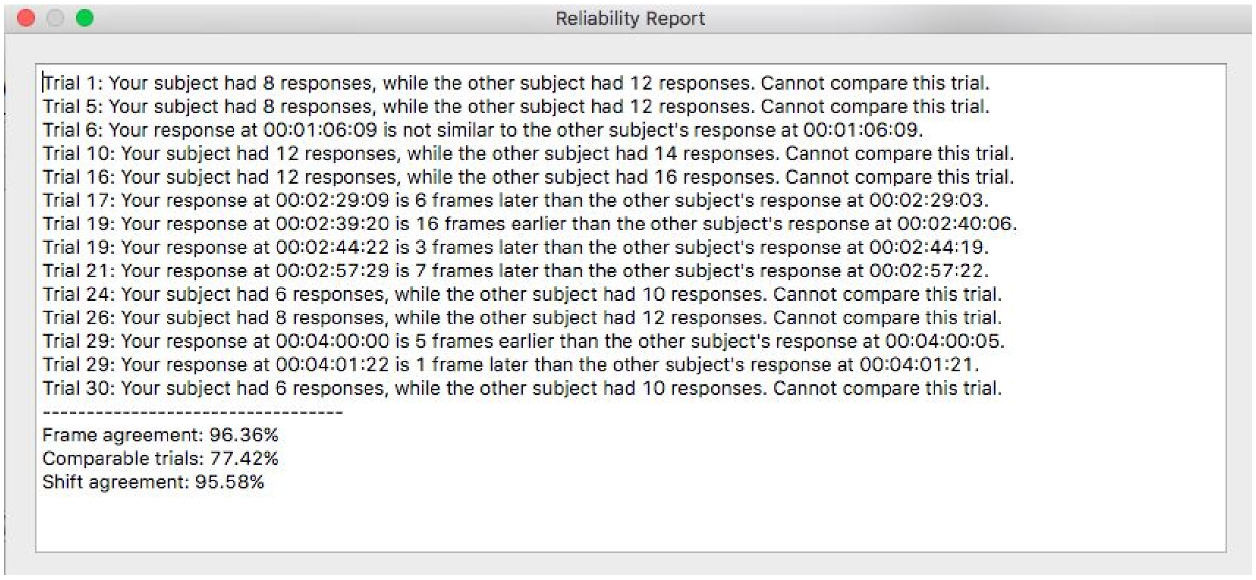
Shift agreement is intended to evaluate how well coders agree on the timing of coded events. It is defined as the ***percentage of coded events that occur within the agreed threshold (+/- 1 frame)***. Shift events (any response recorded other than the start/end of a trial) are considered to agree if they are no more than one frame apart (e.g., if Coder A marks an event at 00:00:01.05 and Coder B marks the event at 00:00:01.04, 00:00:01.05, or 00:00:01.06, any of those responses is considered to agree). Any other time of response is considered *not* to agree. Shift events are also considered *not* to agree if the coders agree on the timing, but not the event (e.g., Coder A marks the event at 00:00:01.05 as *right* and Coder marks 00:00:01.05 as *away*). As with Frame agreement, no distinction is made between *off* and *away* in the calculation of Shift Agreement.

How to use the reliability function

Only files with the same Subject and Order information (Subject number, birthdate, date of test, and order) and no errors (i.e., illegal sequences, missing information) can be compared. Prescreening information does not have match for comparison, and because the comparison is done on a trial-by-trial bases, the files do not needed to have the same number of trials coded (thereby allowing for coders-in-training to compare as they go, or for only selected trials to be included in evaluating reliability).

To compare to files, open one of them and from the File menu, choose ‘Compare against’ or use the shortcut command-T. Using the navigation find and select the second file you wish to compare. This function will automatically open two text boxes. The first box contains the Reliability Report, which includes a detailed list of errors and the three numeric reliability values. The second box is a list of responses from the comparison file, which allows the person checking reliability to easily compare events across the two files.

Interpreting the Reliability Report



```
Trial 1: Your subject had 8 responses, while the other subject had 12 responses. Cannot compare this trial.
Trial 5: Your subject had 8 responses, while the other subject had 12 responses. Cannot compare this trial.
Trial 6: Your response at 00:01:06:09 is not similar to the other subject's response at 00:01:06:09.
Trial 10: Your subject had 12 responses, while the other subject had 14 responses. Cannot compare this trial.
Trial 16: Your subject had 12 responses, while the other subject had 16 responses. Cannot compare this trial.
Trial 17: Your response at 00:02:29:09 is 6 frames later than the other subject's response at 00:02:29:03.
Trial 19: Your response at 00:02:39:20 is 16 frames earlier than the other subject's response at 00:02:40:06.
Trial 19: Your response at 00:02:44:22 is 3 frames later than the other subject's response at 00:02:44:19.
Trial 21: Your response at 00:02:57:29 is 7 frames later than the other subject's response at 00:02:57:22.
Trial 24: Your subject had 6 responses, while the other subject had 10 responses. Cannot compare this trial.
Trial 26: Your subject had 8 responses, while the other subject had 12 responses. Cannot compare this trial.
Trial 29: Your response at 00:04:00:00 is 5 frames earlier than the other subject's response at 00:04:00:05.
Trial 29: Your response at 00:04:01:22 is 1 frame later than the other subject's response at 00:04:01:21.
Trial 30: Your subject had 6 responses, while the other subject had 10 responses. Cannot compare this trial.
----------------------------------
Frame agreement: 96.36%
Comparable trials: 77.42%
Shift agreement: 95.58%
```

Figure 16. Sample reliability report

The reliability report (Figure 16) is intended to highlight coding discrepancies by pointing to the specific disagreements between coders. The error reports are related to, but do not map perfectly onto, the numerical measures of reliability. Some errors are highlighted but not penalized (e.g., differences in fixed events, see below), and not every disagreement appears in the error report (e.g., disagreements that are just one frame apart are factored into Frame agreement, but not included as errors). Three categories of errors are highlighted in the Reliability report: Comparability errors, Gaze Location errors, and Timing Errors.

Comparability errors

Any trials that do not have the same number of responses are listed, with information about why (e.g., *Trial 1: Your subject had 8 responses, while the other subject had 12 responses. Cannot* *compare this trial.*). Most often, these errors reflect disagreement about the presence of off-task responses in the trial, and to find and resolve disagreement, coders can look to the response box that lists events from the other file.

Gaze Location errors

Gaze location errors are recorded when coders disagree about where the child is looking. Errors are noted if the two responses are not the same (e.g., Coder A records *left* when Coder B records *right*), and the reliability report highlights this event (e.g., *Trial 6: Your response at 00:01:06:09 is not similar to the other subject's response at 00:01:06:09.*). Because responses are compared sequentially, it may be that two responses with different timecodes are being compared (e.g., *Trial 6: Your response at 00:01:06:08 is not similar to the other subject's response at 00:01:06:09*).

Timing errors

Timing errors are recorded when events occur outside of the agreed upon threshold. Errors are described, including the duration of the disagreement (e.g., *Trial 19: Your response at 00:02:44:22 is 3 frames later than the other subject's response at 00:02:44:19.*
Trial 29: Your response at 00:04:00:00 is 5 frames earlier than the other subject's response at 00:04:00:05). As with calculating Shift agreement, shift events that are within one frame are considered equivalent and not marked as errors. Fixed events (first/last event of a trial when the trial status changes) must be coded on the exact same frame.

Limitations, potential workarounds, and other notes

Dealing with Non-comparable trials

The most notable limitation of the automatic reliability comparison is that not all trials are included in the calculation of Shift Agreement, which is often the most useful metric of reliability to report. Because events are compared sequentially, it is not possible to get a meaningful comparison, and if a large number of the trials are not comparable, then the Shift Agreement calculated based on a small subset of trials may be either inflated or reduced. There are three common solutions to this issue: (1) An independent third coder can make changes to the coding of the trials to make them comparable (e.g., adding in an ‘away’); (2) the third coder can manually calculate Shift Agreement for those trials; (3) the automatic values can be reported, including the Comparable Trials value.

Fixed Events

Fixed events are not included in Shift Agreement, even though we are stricter in our expectations about coders agreeing exactly. This apparent contradiction allows us to ensure high precision in the timing of trials (since all analyses rely on us knowing when events occurred in relation to the beginning of a trial) but not to conflate these decisions with our ability to say that coders agree on shifting behavior.

Reporting data to match analyses

All coded frames count equally in our reliability measures, yet most often analyses are done based on a subset of data that might include less than 50% of the trial (e.g., an 1800ms accuracy window, out of a trial of 6s), and moreover, children tend to switch more, and there are often more disagreements at other parts of the trial, such as the beginning when the child is inspecting images in silence. We take this approach to be conservative, and also because for training purposes, it is useful to provide coders as much feedback as possible. However, if these standards are significantly stricter than those used by other labs, it can occasionally be a problem in peer review. Another reasonable approach is to simply compare the data that is used in analysis. For example, after data is exported, it is possible to calculate trial-by-trial accuracy and RT and to compare the outputs to see whether differences between coders yield differences in the resulting data.

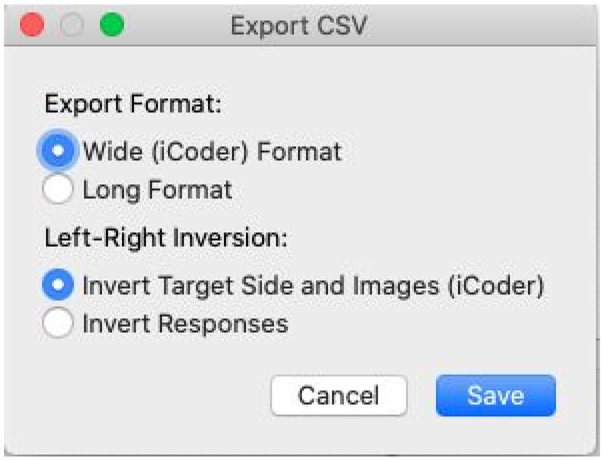
Current conventions

All labs can and should develop their own standards for training new coders and for periodic reliability checks. Currently, our labs expect coders to achieve ~97-98% reliability (compared to either a standardized key or between any two coders on average) for both Frame and Shift Agreement. It should be noted that peyeCoder has a slightly stricter definition of Frame Agreement than iCoder, so labs that in the past might have been able to achieve 99% Frame Agreement may need to reconsider. There is less standardization around what is reasonable to expect for Comparable Trials, particularly as new conventions for coding ‘away’ emerge, but 75-80% has been a reasonable target in the past.

Individual videos will vary vastly in how difficult they are to code, so a single file may not be a good reflection of an individual coder’s skill. Typically, it will take new coders 4-6 files (particularly if chosen appropriately to increase with difficulty and to contain different types of challenges) to become reliable. The difficulty increases considerably if there are more than two locations to consider, or if the images are in unusual locations (either because of a difference in the camera or screen arrangement), so it may be useful to have experienced coders work together and discuss discrepancies before coding large quantities of data.

Data Export

After selecting File > Export CSV, the following menu will open


Export CSV

Export Format:

[✓ Wide (iCoder) Format]

Long Format

Left-Right Inversion:

[✓ Invert Target Side and Images (iCoder)]

Invert Responses

Figure 17. Screenshot of data exportation window

When exporting the data you have two decisions to make:

- whether to export the data in wide vs. long format
- whether to invert the target side or the coded responses

Each choice is explained in greater detail in the next two sections.

Wide vs. Long Format

Both formats will export the following information for each row:

Sub Num

Months

Sex

Order

Tr Num

Prescreen Notes

L-image

C-image

R-image

Target Side

Target Image

Condition

CritOnset

For each trial, there are multiple data points (i.e., one every frame/33 ms).

In wide format, there is one row per trial and one column for each data point (e.g., a column for when Time = 0, another column for when Time = 33).

In long format, there are multiple rows per trial (with the number equal to the number of data points) and two columns (one specifying Time and the other Accuracy).

peyeCoder will default to export data in the same wide format as iCoder.

Exporting in long format enables extra information to be included in the .csv file.

- with wide format, the header row will be labeled using time that is centered (i.e., time = 0 at critical onset) and children's accuracy (1 = target, 0 = distractor, 0.5 = center, - = away, . = off) will be provided for each frame
- with long format, there will be columns with values for both time that is centered and time that is uncentered (time = 0 on the first frame that is coded) and columns with values for both children's accuracy and the coded response (e.g., left, right, center)

The wide format export option, however, should ONLY be used when the file has been coded using the default responses (left, off, right, away, center).

- this is because the accuracy value for each column unambiguously identifies whether the child was looking at the target (Accuracy=1), distractor (Accuracy=0), center (Accuracy=0.5), off (Accuracy=.) or away (Accuracy=-).

With more images, however, this information is lost. For instance, if the target is the topRight image then Accuracy would = 0 if the child was looking at the bottomLeft, bottomRight, or topLeft images.

Left-Right Inversion

When coding, the participant’s left is *not* the same as the coder’s left (e.g., when the participant is coded as looking to the left, they are actually looking at the image on the right side of the screen). Therefore, an inversion is necessary.

In the Trial Order file, Left Image, Right Image, and target side are all from the *participant’s* perspective.

Invert Target Side and Images: By default (and consistent with iCoder), peyeCoder will invert the Left Image, Right Image, and target side entries when exporting the data. Therefore, the entry for Left Image in the Trial Order will match the entry for R-image in the exported .csv file.

- for instance, a trial could have Left Image = cat; Right Image = dog; target side = R from the participant’s perspective (in the Trial Order)
- using this option will export L-image = dog; R-image = cat; Target Side = L, target = dog in the exported .csv file
- so when the coder marked the child as looking to the coder’s left (image of the dog), that will be exported as a look to the target (Accuracy = 1)

Invert Responses: as an alternative, peyeCoder can be set to export the L-image, R-image, and Target Side as they occurred from the participant’s perspective, but *invert* the coded response.

- for instance, a trial could have Left Image = cat; Right Image = dog; target side = R from the participant’s perspective (in the Trial Order)
- using this option will export L-image = cat; R-image = dog; Target Side = R, target = dog in the exported .csv file
- so when the coder marked the child as looking to the coder’s left (image of the dog) that will be inverted and exported as looking to the right and a look to the target (Accuracy = 1)

If you will be combining hand-coded data with data from an automatic eye-tracker, then you should select the Long format and Invert Responses options (the two non-default options).

Other Helpful Information

Modifying Default Settings

By default, peyeCoder will open with Step set to 10 frames and the following responses & associated keys: left (1), off (2), right (3), away (4), center (5) and toggling trial status between on/off is set to the key 6.

All of these options can be changed when you are coding a file. Any changes that you make for an individual file, however, will not carry over to other files you open or when you create a new file (i.e., the step frames and responses will revert back to the defaults).

You can, however, change the default values for peyeCoder. To do this open a new file, make your desired changes to Step and responses. Then save this file as default.vcx.

If you are using a Linux or Windows computer, simply save this computer in the same folder where the peyeCoder executable file is located (e.g., having both the default.vcx file and peyeCoder executable on your Desktop). The next time you open peyeCoder it will detect the default.vcx file and use the modified default settings.

If you are using a Mac computer, you will need to complete one additional step. After saving the default.vcx file (to any location). Right click on the peyeCoder executable file and select “Show Package Contents”.

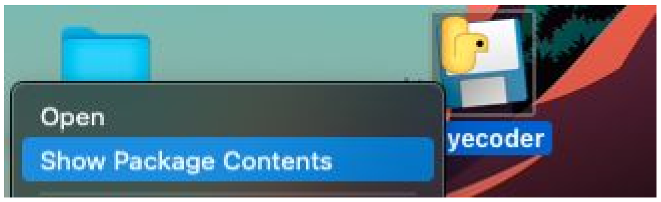


Figure 18. Screenshot with the portion of the menu displayed when right-clicking on the peyeCoder executable file.

This will open up a window in Finder with a “Contents” folder. Within the Contents folder navigate to the MacOS folder. Drag and drop the default.vxc file into this folder. The next time you open peyeCoder it will detect the default.vcx file and use the modified default settings.

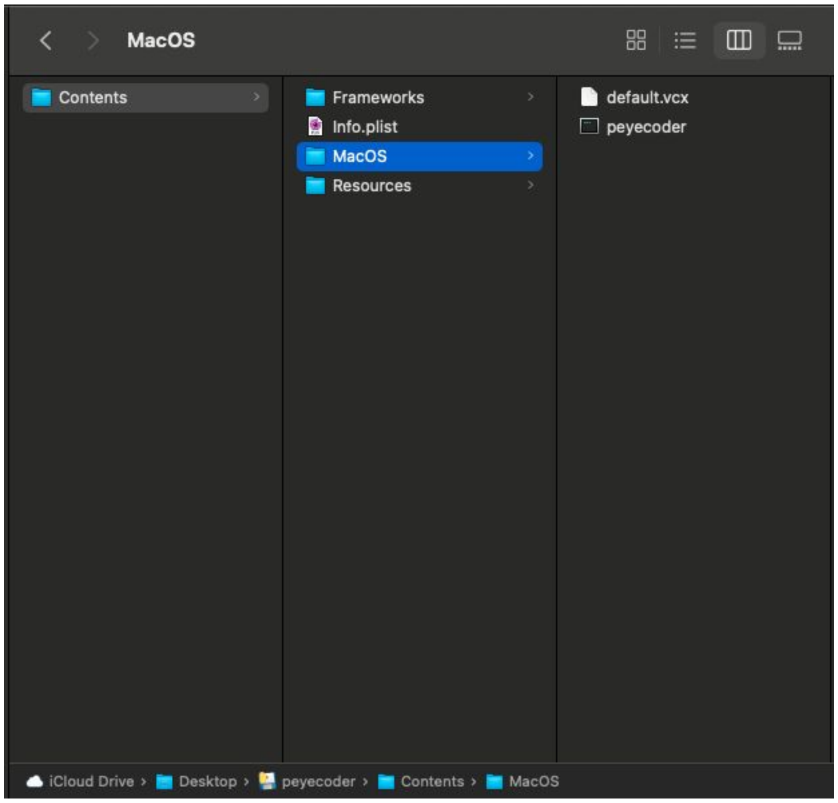


Figure 19. Screenshot of the Finder window that is opened when right-clicking on the peyeCoder and selecting Show Package Contents.

Timecode

If you do not digitally impose a Time Code (either when recording or exporting) on your videos, this can be done afterwards using ffmpeg. This software is free and available at: http://ffmpeg.org.

After installing ffmpeg, you can use Terminal to navigate (*cd…*) to the folder containing the recorded video without a time stamp and enter the following code:

```
ffmpeg -i zoom_4.mp4 -vf
"drawtext=timecode='00\:00\:00\:00':rate=25:fontsize=48:fontcolor=white:box=1:boxborderw=6:b
oxcolor=black@0.75:x=(w-text_w)/2:y=h-text_h-20" -c:a copy zoom_4_stamped.mp4
```

This code, opens a specific video from the using the file name

zoom_4.mp4

adds a digital Time Code in the hours:minutes:seconds:frames format

drawtext=timecode='00\:00\:00\:00':

for a video with 25 frames per second rate

rate=25:

the Time Code is added as white, size 48 font on a black background

:fontsize=48:fontcolor=white:box=1:boxborderw=6:boxcolor=black

That is centered at the bottom of the video,

@0.75:x=(w-text_w)/2:y=h-text_h-20

the edited video is then saved as a new file

-c:a copy zoom_4_stamped.mp4

Opening Multiple Instances of peyeCoder

On Windows and Linux, you can double click on the peyeCoder executable file to open a second instance of the program. This will allow you to have two or more coding files open simultaneously. This is mostly helpful if you are doing a reliability comparison when two different coders have coded the same participant and want to simultaneously edit the coding files.

On Mac, you cannot double click on the peyeCoder executable file to open a second instance. If you have installed peyeCoder via Terminal, you can open two (or more) Terminal windows and have multiple coding files open simultaneously. If you are only using the executable file, there is a workaround (see https://www.cultofmac.com/200062/run-multiple-copies-of-an-app-at-once-on-your-mac-os-x-tips/). This workaround will have you create an AppleScript that you can double-click to launch peyeCoder (rather than the executable file itself) and will allow you to have multiple instances of peyeCoder open simultaneously.

Frame rates vs. ms per frame

Most cameras record at 29.97 frames per second, which means that 33.3667 ms elapse between frames.

Zoom, however, records at 25 frames per second, which means that 40 ms elapse between frames.

Unlike iCoder (which can only work with videos at 29.97 frames per second), peyeCoder will determine the frame rate of a video when it is loaded and take this into account when incrementing the time stamp. So when coding a Zoom video, the time stamp in peyeCoder will count up to 24, the next frame will increment the second by 1 and flip the count back to 0. When coding a regular video at 29.97 frames per second, peyeCoder will count up to 29, the next frame will increment the second by 1 and flip the count back to 0.

When coding videos with a 29.97 frames per second rate, both iCoder and peyeCoder assume that 30 frames occur per second, which means 33.333 ms elapse between frames. So when outputting the data, the ms for successive frames go from 0 to 33 to 67 to 100. This means, however, that the time output from iCoder/peyeCoder successively falls behind how much time actually elapsed by ~.03 ms for each frame in the video. This turns out to be a trivial amount, because we are concerned about the amount of time that elapses from the first frame that is coded in each trial. So if you're coding a trial that lasts 4004 ms, that is 120 frames. And if each frame falls .03 ms behind the actual time elapsed, iCoder/peyeCoder will say that 4000 ms elapsed during the trial (120 frames * 33.333 ms per frame). So, by the end of a trial, iCoder/peyeCoder has fallen 4 ms behind. That's not large enough of a difference to worry about. For instance, it's not until we get up to trials that last 34 seconds that the lag would reach 33 ms (i.e., the time reported in iCoder/peyeCoder would be 1 frame behind the amount of time that actually elapsed), where we would realistically start considering making an adjustment (i.e., by having iCoder/peyeCoder increment the time stamp twice when incrementing one frame forward).

Accompanying Materials

The example video is from Ron Pomper’s dissertation (2020), which is in preparation for publication. In this experiment, three-year-olds watched videos of an adult labeling familiar and novel objects placed before her on a table (teaching trials). Children were then shown pairs of images and heard a sentence labeling one of the images (test trials). By tracking children’s eye-movements we can measure their accuracy in identifying the referents of novel words (during teaching) and their success in learning and remembering the word-object mappings (during testing). Children were taught the names of the novel objects in 3 different conditions and trials were blocked by condition. Within each block, children saw all of the teaching trials (n=5) and all of the test trials (n=7) for a specific condition.

Included with the video are several .vcx files in the Coding subfolder. These are saved **peyeCoder files**. Using peyeCoder, three different people coded test trials from the example video. All three coders were compared and a consensus file was created (CueCue_101_KEY.vcx). The initial coding includes three possible responses (right, left, and off), but the key was then edited to include a fourth response of ‘away’ for instances when the child is inattentive (CueCue_101_KEY_withAways.vcx). Finally, one person coded both teaching and test trials from the example video (CueCue_101_RP.vcx). This last file was used for plotting and comparisons with the Tobii data (see below).

Within the Coding folder is an Orders subfolder. This contains two .txt files with information about the **Trial Order.** These can be used as an example to be adapted for your own experiments and can also be used if you wish to code the example video yourself to compare against the key. One order should be used if you only code the test trials

(CueCue-1-test-only.txt) and a second order should be used if you code both teaching and test trials (CueCue-1.txt).

The Data subfolder contains the exported data from both the Tobii X2-60 eyetracker and the peyeCoder (CueCue_101_RP.vcx) file. Finally, there are several example R scripts included to **process and plot** the data.

The Tobii X2-60 eye tracker tracks children's fixation location every 16 ms (i.e., 60 Hz). These fixation locations are coded as coordinates in terms of pixel location with the axis starting in the top-left corner of the screen (0,0) and extending to the bottom-right corner of the screen (1920, 1080). At the end of the experiment all of the fixation locations are written to a raw data file (101_TOBII_output.tsv). An accompanying R script (1_Format_TobiiData.R) **processes the raw data.** This script is idiosyncratic to the experimental setup and would need to be adapted to other eye-trackers. There are, however, several custom functions in the libraries folder that are generally useful for dealing with automatic eye-tracking data. PyGaze_processGazeData.R reformats the data to be in a more user-friendly format. PyGaze_addAOIHits.R determines for each frame whether the x and y coordinates for the child's gaze location fall within different Areas of Interest (AOIs) on the screen. PyGaze_InterpolateAOIHits.R interpolates missing data, which occurs when automatic trackers briefly lose track of children's fixation location throughout the experiment. When this happens the AOI will be marked as away. For instance if the child fixates the right image from 300 to 600 ms, but the eye-tracker lost track at 400 ms, that frame will have its AOI set to away. The script will identify these instances (i.e., aways that occur for less than 300 ms where the child fixated the same AOI before and after) and switch them to match the AOI before/after the loss in tracking. The processed data is then saved as a new .csv file (CueCue_101_TobiiData.csv).

An optional R script (2_Plot_Gaze_Locations.R), will **load the processed data and plot** every fixation using its x- and y-coordinates. The different AOIs are drawn as squares on the plot and each data point is colored based on the AOI. These plots are particularly useful in visually identifying whether children were sufficiently calibrated. If calibration was poor there will be significant drift (e.g., rather than being centered within an AOI, clusters of fixations will be centered above and to the left of the AOIs). If there is poor calibration and significant drift, this can be an indication that the participant's data needs to be hand-coded.

The data exported from peyeCoder (CueCue_101_peyeData.csv) indicates whether children are fixating different AOIs on the screen every 33 ms. This is the standard frame rate for most experiments, because most video cameras record at 30 Hz. Therefore, there is an' R script (3_Combine_TobiiLWL.R) that will **align the formatting of data that is exported from Tobii and peyeCoder**. This includes downsampling the Tobii data to report children's fixation location every 33 ms. The downsampling function bins children's fixations into 33 ms bins and retains only the first fixation within each bin.

An additional R script (4_Plots.R), includes code to **filter individual trials** (removing trials where the child is not fixating any of the AOIs for more than 50% of the critical window), aggregating and plotting changes in children's accuracy in fixating the target image over time. Most of this code is accomplished using different tidyverse packages (e.g., dplyr and ggplot). There is excellent documentation for these packages available online (https://tidyverse.tidyverse.org).

Finally, there is an R script (5_Plots.R) that will f**ormat the data exported from peyeCoder in wide format** (CueCue_101_peyeData_wide.csv) to exactly match the formatting of the data that is exported from iCoder. Specifically, this script adds three columns for each trial. Response indicates whether the child was looking at the target (T), distractor (D), or away (A) at the onset of the target word (Time = 0). First Gap is the number of frames children took when they first shifted from one image to the other (i.e., target to distractor or vice versa) after the onset of the target word (Time = 0). If a child did not complete a shift from one image to the other, then First Gap will be set to NA for that trial. RT is children's reaction time, this is the time (in ms) where the First Gap began. For example, if the child is fixating the distractor from 0 to 567ms, then shifts from 600 to 667 ms, and fixates the target at 700 ms - Response will be D, First Gap will be 3 (shifting for the 600, 633, and 667ms frames), and RT will be 600.

Comparisons to iCoder

Who can use the software

- iCoder works only for Macs running macOS Mojave (10.14) or earlier
- peyeCoder works on Macs running macOS Mojave (10.14) or later, as well as Windows and Linux computers

What types of experiments can be coded

- iCoder only works for videos recorded at a rate of 29.97 frames per second
- peyeCoder works videos with different frame rates
  - Zoom recordings are a slower frame rate of 40ms/frame
  - peyeCoder detects this and will increment the second counter in the time stamp after 25 frames, rather than 30, and will adjust the ms per bin when exporting
- iCoder can only be used to code trials with a left, right, and center image
- peyeCoder can be used to code trials with any configuration of stimuli, since the responses can be customized
  - e.g., 4AFC tasks can be coded in peyeCoder where images are displayed bottomLeft, bottomRight, topLeft, and topRight

What types of response patterns are required

- iCoder required aways
    - sequences like left-off-left were not permissible, generating an error that prevented exporting the data
- peyeCoder does not require aways
    - sequences like left-off-left are now permissible
    - so coding schemes that do not include aways can be used
- peyeCoder eliminates other previously illegal sequences
    - only direct transitions between right-left and left-right are now flagged as errors (e.g., left-away, left-center transitions are allowed)

Ways to make coders unaware to visual stimuli

- peyeCoder has a new feature that allows you to add gray occluder boxes to cover parts of the video

What information is required in the Trial Order

- iCoder orders required many columns that are no longer used (i.e., users often filled them with 0’s). Moreover, only some of these columns affected how the data was exported from iCoder
- Therefore, in peyeCoder we have removed the following columns from order file, which simplifies the process
    - Trial Onset
        - used to indicate whether coders began coding a trial from Picture (i.e., when images appear on the screen) or Sound (i.e., when the sound file begins to play) onset
        - this did not affect how iCoder exported the data and is no longer necessary, because most labs code from picture onset (coding from sound onset would require a visual stimulus to be displayed in the PIP box indicating the frame on which the sound file begins to play)
    - TrStart
        - used to indicate from what point in the trial should data be exported
        - 0 export all
        - +xxx ms will not export the first xxx ms of coding
        - this is no longer necessary, because this data can be trimmed during the analysis process and it is better practice to retain this data upon export (to prevent the need to re-export data if it becomes necessary to analyze gaze data at the beginning of the trial)
    - PictOn
        - used to indicate from what point in the trial where coding began did the images appear

        - -xxx ms if coding from sound onset (i.e., pictures appeared before coding began for the trial)
        - +xxx ms if coding from picture onset (i.e., pictures appeared at the onset of coding or after coding began for the trial)
        - this did not affect how iCoder exported the data and is no longer necessary (again because few if any labs code from sound onset)
    - SoundOn
        - used to indicate from what point in the trial where coding began did the sound play
        - -xxx ms if skipping into trial to start coding (i.e., sound began playing before coding began for the trial)
        - +xxx ms if coding from picture onset (i.e., sound began playing after coding began for the trial)
        - this did not affect how iCoder exported the data and is no longer necessary (again because few if any labs code from sound onset)
    - TrEnd
        - number quantifying the time in milliseconds from when the coder begins to code the trial until the end of the trial
        - in principle, this could be used to limit whether coded frames beyond the trial are exported
        - this did not affect how iCoder exported the data, however, and is no longer included
    - CritOffset
        - number quantifying the time in milliseconds from when the coder begins to code the trial until the end of the critical window
        - in principle, this could be used to limit how RTs are calculated (i.e., not calculating RTs that occur outside of the critical window)
        - this did not affect how iCoder exported the data, however, and is no longer included and it is better practice to calculate all RTs and then eliminate RTs that are too longer during data analysis (to prevent the need to re-export data if the critical window of analysis changes)
        - moreover, peyeCoder does not calculate RTs for each trial when exporting the data
- iCoder orders allowed you to add extra information at the bottom of the file, this information was not used by iCoder, but could be helpful notes about counterbalancing, changes in experimental procedure, logging general issues, etc.; this information cannot be included peyeCoder orders, because it will interfere with the programs ability to read in the data

How the data are exported

- iCoder required the use of another program (dataWiz) to aggregate coded files and export the data into a single .csv file
- peyeCoder exports data for each coded file, rather than aggregating (so there's no additional software required)
- peyeCoder provides extra options when exporting to change how the data is formatted (wide vs. long format and inverting the target side vs. coded response when determining accuracy)
- when exporting the data, iCoder/dataWiz would determine for each trial…
    - Response: whether the subject was looking at the target (T), distractor (D), or away (A) at the onset of the critical word (Time=0)
    - FirstGap: the length (# of frames) it took the subject to shift from one image to the other image on their first change in fixation following the onset of the target word
    - RT: the time (in ms) it took the subject to initiate the first shift from one image to the other image following the onset of the target word
    - canonically, this information was used to calculate children's speed in word recognition by measuring the average RT on target-initial trials where the child shifted to the distractor within a specific time frame (e.g., no later than 2,100 ms after the onset of the critical word) and directly shifted between the two pictures (e.g., FirstGap no greater than 8 frames)
- peyeCoder does not calculate Response, firstGap, and RT when exporting data
    - but we have included in the Accompanying Materials R code that will do this

How reliability works

- peyeCoder opens another panel when comparing coding between files (responses you select in one coded file are also highlighted in the other so that you can follow along where you click in both coded files)

- in iCoder % frame agreement only was calculated for comparable trials
- in peyeCoder % frame agreement is calculated using all trials