

asf Tutorials

jens.schwarzbach@ukr.de

asf can be easily set up as a picture playback machine. For installation please refer to <https://github.com/jvschw/asf/wiki>. The stimulus-definition file (<any>.std) tells asf which stimuli are available, the trial-definition file <any>.trd tells asf when to show what and for how long, when to respond, and how results are logged.

Below we refer to <asf_install_dir> as the installation directory of asf. Usually this is ~/asf.

1 Show single pictures

This tutorial is located in <asf_install_dir>/documentation/tutorials/animals

You can run it with:

```
<asf_install_dir>/documentation/tutorials/animals  
ExpInfo = ASF('stimuli.std', 'trialdefs.trd', 'test')
```

1.1 Simple Image classification

This is an experiment to demonstrate basic functionality. Here you learn how to set up an experiment that consists of showing one image per trial and querying a manual response. Participants are shown pictures for 2 seconds each, which they have to classify into cats (left mouse button) and dogs (right mouse button).

1.1.1 Step by step

Start up Matlab or Octave and create a directory on your computer in which the experiment resides, which will be called the experiment-home from now on. For example choose ~/exp/animals as your experiment-home

```
mkdir ~/exp/animals  
cd ~/exp/animals
```

Create a subdirectory “stimuli” in which you store the images you want to present

```
mkdir stimuli
```

Find five images of cats and five images of dogs. Check out

<https://www.pexels.com/search/cats/> and <https://www.pexels.com/search/dogs/>. Store them in ./stimuli/stimuli (name them cats001.jpg – cats005.jpg, and dogs001.jpg – dogs005.jpg, respectively)

Create a text file stimuli.std in experiment-home that lists all ten pictures with their relative path (relative with respect to experiment-home), such as

```
./stimuli/cats001.jpg  
./stimuli/cats002.jpg  
./stimuli/cats003.jpg  
./stimuli/cats004.jpg  
./stimuli/cats005.jpg  
./stimuli/dogs001.jpg  
./stimuli/dogs002.jpg  
./stimuli/dogs003.jpg  
./stimuli/dogs004.jpg  
./stimuli/dogs005.jpg
```

Create a trd file `trialdefs.trd` that determines what to show when: We will start with showing only `cat001` (the first picture in the list) and `dog001` (the sixth picture in the list).

```
2 category cat dog
1 0 1 120 1 1 1
2 0 6 120 1 1 2
```

- The first line depicts design information: The experiment has one factor with two levels, the factor is called 'category', and the levels are called 'cat' and 'dog', respectively.
- The following lines depict trial information, with one line per trial. Here, the definition of a trial is show a stimulus and collect a response.
- The first column indicates the code of the experimental condition (1 for cat, 2 for dog).
- The second column allows to specify the absolute onset time of a trial (not used here, therefore always 0).
- Columns 3 and 4 depict a so called page. A page consists of a stimulus number (1 or 6 in this case) and its duration (in frames). With a frame rate of 60 Hz, the value 120 means 2 seconds.
- Columns 5 and 6 depict on which page to start and end response collection (here 1 and 1).
- Column 7 indicates the correct response key (1 for cat, 2 for dog).

Run this very first version, the syntax is `ExpInfo = ASF(stimFileName, trialFileName, expName, [Cfg])`. In our case you can start the experiment with:
`ExpInfo = ASF('stimuli.std', 'trialdefs.trd', 'test')`

To show the entire stimulus set, create a corresponding trd file `'trialdefs_10.trd'`:

```
2 category cat dog
1 0 1 120 1 1 1
1 0 2 120 1 1 1
1 0 3 120 1 1 1
1 0 4 120 1 1 1
1 0 5 120 1 1 1
2 0 6 120 1 1 2
2 0 7 120 1 1 2
2 0 8 120 1 1 2
2 0 9 120 1 1 2
2 0 10 120 1 1 2
```

Run the experiment

```
ExpInfo = ASF('stimuli.std', 'trialdefs_10.trd', 'test' )
```

You will notice that the trials are played in the order in which they have been defined. In general, `asf` leaves it up to the experimenter to provide properly shuffled trial-definition files, but you can also configure `asf` to shuffle the presentations for you. To this aim, you create a variable `Cfg`, which can have many different fields (check `help asf` on the command line).

```
Cfg.randomizeTrials = 1
ExpInfo = ASF('stimuli.std', 'trialdefs_10.trd', 'test', Cfg)
```

Now you have your first animal-categorization experiment in asf, which presents pictures of cats and dogs in randomized order and records your responses (which key and which reaction time).

Additional tasks

- Change response mapping (keyboard instead of mouse keys)

2 Show sequence of pictures

This tutorial is located in <asf_install_dir>/documentation/tutorials/animals

You can run it with:

```
<asf_install_dir>/documentation/tutorials/animals_dm  
ExpInfo = ASF('stimuli.std', 'trialdefs.trd', 'test')
```

Usually, trials in an experiment contain more than a single page. In this tutorial we will develop a little experiment in which we ask participants to judge whether two subsequent pictures show animals of the same category.

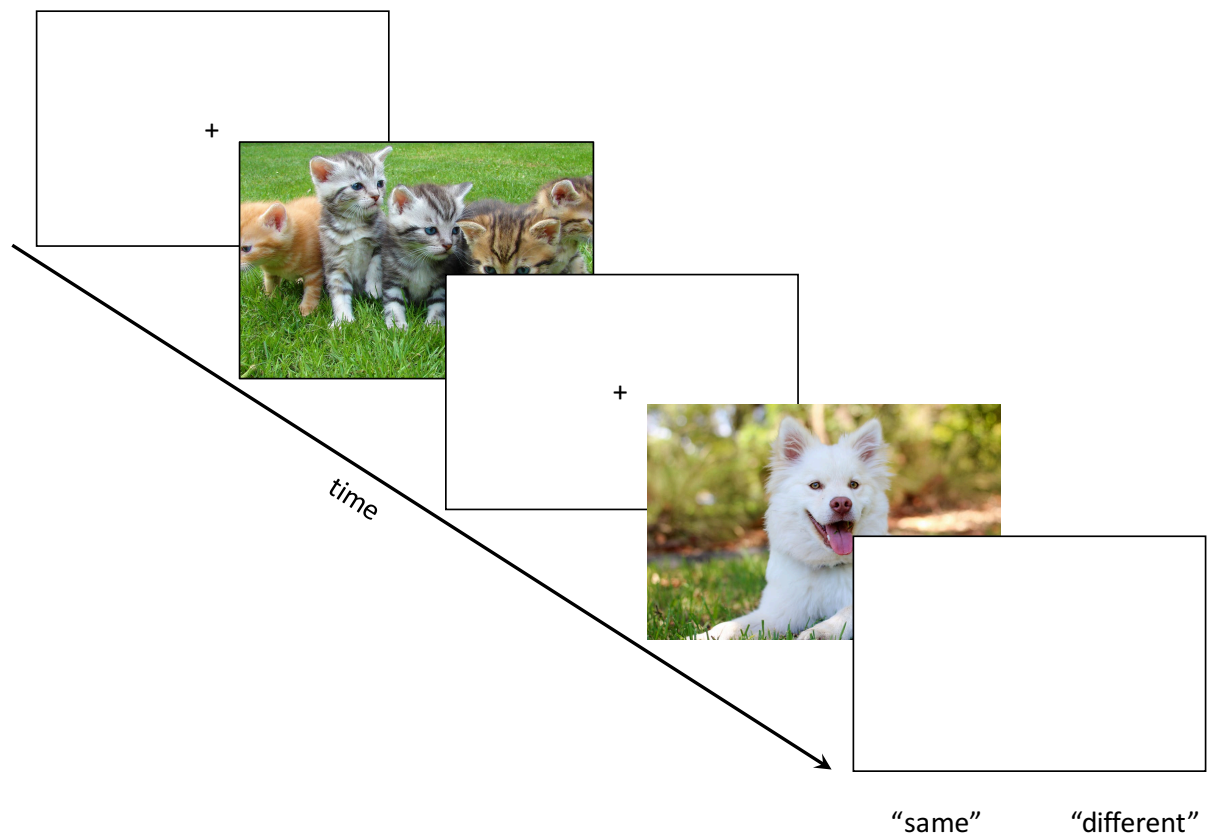


Figure 1. Example trial for our categorical decision making task. Each trial contains 5 pages. A trial starts with presenting a fixation cross for 500ms, followed by picture 1 (cats or dogs) for 500ms, a fixation cross (500 ms), picture 2 (cats or dogs) for 500 ms. The last page shows an empty screen for 1500ms during which participants are asked to indicate whether the two pictures showed animals of the same species(left mouse button) or different species (right mouse button).

3 Online rendering by means of a plugin: Show one picture or a sequence of pictures at different locations

This tutorial is located in <asf_install_dir>/documentation/tutorials/dots_example

You can run it with:

```
<asf_install_dir>/documentation/tutorials/dots_example  
run_dots
```

Imagine you want to program a perimetry experiment in which a small stimulus occurs at different locations on the screen and the participant must press a button in order to report whether she or he has detected a stimulus. in the end, the experimenter can analyze the speed and accuracy of visual detection as a function of stimulus location.

Let's say we want to test 25 locations on a 5x5 grid around the center of the screen.

-250, 250	-125, 250	0, 250	125, 250	250, 250
-250, 125	-125, 125	0, 125	125, 125	250, 125
-250, 0	-125, 0	0,0	125, 0	250, 0
-250, -125	-125, -125	0, -125	125, -125	250, -125
-250, -250	-125, -250	0, -250	125, -250	250, -250

Table 1. Positions of stimuli relative to center.

One way to achieve this would be to create 25 bitmaps each of which containing one stimulus at a defined location. Here we explore a more efficient alternative. We create one bitmap that contains the stimulus in the center of the image, and we present this bitmap at different positions on the screen.

So far you have learned that the default behavior of asf, which is to present one picture per frame, and that this picture is presented centrally. This is achieved by the built-in function ASF_showTrial.m (e.g. line numbers 89, 147, and 217 appendix).

```
Screen('DrawTexture', windowPtr,...  
    Stimuli.tex(atrial.pageNumber(iPage)) );
```

We will write our own ShowTrial function (ASF_showTrialDotXY.m), in which we add a little piece of code that draws a dot at a certain location relative to the screen-center.

We will also have to tell our function what stimPosX and stimPosY are early on in the function, which we can define early on in the code (near original line 48)

```
stimPosX = atrial.userDefined(1);  
stimPosY = atrial.userDefined(2);
```

Add right after the original line 147, which is

```
Screen('DrawTexture', windowPtr,...  
    Stimuli.tex(atrial.pageNumber(iPage)) );
```

the following code

```
drawFix(windowPtr, Cfg, fixColor)  
if iPage == atrial.startRTonPage  
    Screen('DrawDots', windowPtr,...  
        [ Cfg.Screen.centerX + stimPosX,...  
          Cfg.Screen.centerY + stimPosY], 10, [160 160 160], [], 2);  
end
```

This piece of code will draw a grey dot at the indicated stimulus position.

This also requires a certain type of trd-file, such as

```
5 5 px -250 -125 0 125 250 py -250 -125 0 125 250
```

1	0	-250	-250	1	1	1	120	1	2	1
1	0	-125	-250	1	1	1	120	1	2	1
1	0	0	-250	1	1	1	120	1	2	1
1	0	125	-250	1	1	1	120	1	2	1
1	0	250	-250	1	1	1	120	1	2	1
1	0	-250	-125	1	1	1	120	1	2	1
1	0	-125	-125	1	1	1	120	1	2	1
1	0	0	-125	1	1	1	120	1	2	1
1	0	125	-125	1	1	1	120	1	2	1
1	0	250	-125	1	1	1	120	1	2	1
1	0	-250	0	1	1	1	120	1	2	1
1	0	-125	0	1	1	1	120	1	2	1
1	0	0	0	1	1	1	120	1	2	1
1	0	125	0	1	1	1	120	1	2	1
1	0	250	0	1	1	1	120	1	2	1
1	0	-250	125	1	1	1	120	1	2	1
1	0	-125	125	1	1	1	120	1	2	1
1	0	0	125	1	1	1	120	1	2	1
1	0	125	125	1	1	1	120	1	2	1
1	0	250	125	1	1	1	120	1	2	1
1	0	-250	250	1	1	1	120	1	2	1
1	0	-125	250	1	1	1	120	1	2	1
1	0	0	250	1	1	1	120	1	2	1
1	0	125	250	1	1	1	120	1	2	1
1	0	250	250	1	1	1	120	1	2	1

As usual, the first two columns encode the trial-code and the onset time, both of which can be ignored here.

Columns 3 and 4 are the two user-supplied columns (relative x-position, relative y-position).

Columns 5 and 6 refer to page 1 (page, duration) on which we render our stimulus).

Columns 7 and 8 refer to page 2 (page, duration), which are part of the response period.

Columns 9 and 10 indicate the start- and end-pages for response collection.

Column 11 indicates the correct response (button 1).

4 Show several pictures at once

Here you learn how to set up an experiment that consists of showing two images side by side per trial and querying a manual response. Participants are shown two pictures of different animal categories (cats and dogs) for two seconds. One category will be on the left side of the screen, the other on the right. Which category is where will be randomized. Participants have to manually indicate the location (left or right) of the cat(s) using the left and right mouse button, respectively.

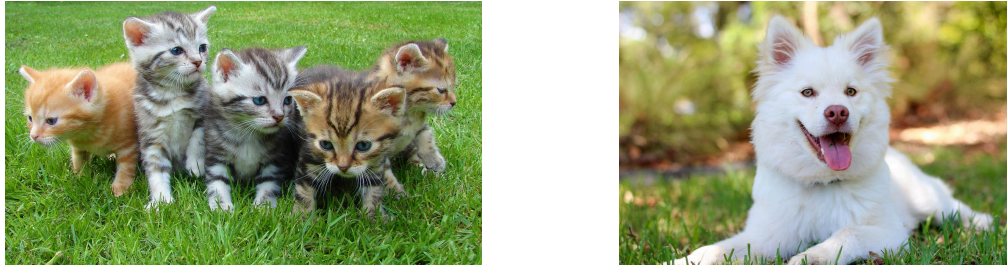


Figure 2. Sample trial from a “Where is the cat” experiment. Two pictures are presented simultaneously to the left and right of the center of the screen. One depicts a cat or cats, the other a dog or dogs. Participants indicate manually on which side they see a cat or cats.

4.1 Preparatory work

We create the experiment’s base directory and switch into it

```
mkdir ~/exp/animals_sim
cd ~/animals_sim/
```

Copy the stimuli from the animals tutorial.

```
copyfile('~exp/animals/stimuli/*', './stimuli')
```

Use some picture editor and create an empty (i.e. white) picture and save it under ./stimuli/stim_empty.bmp

Copy the stimuli and the stimulus definition file from the animals tutorial.

```
copyfile('~exp/animals/stimuli.std', 'stimuli.std')
```

Edit this file and add ./stimuli/stim_empty.bmp at the beginning such that it now contains:

```
./stimuli/stim_empty.bmp
./stimuli/cats001.jpg
./stimuli/cats002.jpg
./stimuli/cats003.jpg
./stimuli/cats004.jpg
./stimuli/cats005.jpg
./stimuli/dogs001.jpg
./stimuli/dogs002.jpg
./stimuli/dogs003.jpg
./stimuli/dogs004.jpg
./stimuli/dogs005.jpg
```

4.2 Strategy

So far you have learned how to show one picture per page. This is achieved by the built-in function ASF_showTrial.m. You may want to spend some time in trying to understand this

very central function (see appendix). If you want to show several pictures on the same page you have to determine their respective location. A very tedious approach would be to create new combined images by placing two images on the same canvas in an image processing program and store them as combined stimuli. This takes a lot of time, and you may end up having to create a large number of compound stimuli.

A much more efficient approach is to have asf combine these pictures online. To achieve that, you need to tell ASF to use a different function to present stimuli. Rather than using the builtin function ASF_showTrial.m use ASF_showTrial2Pics.m, which puts two pictures on the screen according to parameters you provide in the trd-file.

To this aim save a copy of ASF_showTrial2Pics.m to your local directory (because this allows you editing and adapting this function to your own needs):

```
copyfile(fullfile(fileparts(which('asf'))), ...  
'ASF_showTrial_plugins', 'ASF_showTrial2Pics.m'), '.')
```

Remember, the empty picture is the first picture in our std-file, cats have the indices 2-6, and dogs have the indices 7-11.

The function ASF_showTrial2Pics expects two so-called userDefinedSTMcolumns, which are extra entries in a trd-file.

A trd-file (trialdefs_short.trd) with only two trials, might look like this with userDefinedSTMcolumns printed in boldface:

```
2 whereiscat left right  
1 0 2 7      1 120 1 30    1 1    1  
2 0 7 2      1 120 1 30    1 1    2
```

The first line depicts design information: The experiment has one factor with two levels, the factor is called 'whereiscat, and the levels are called 'left' and 'right', respectively.

The following lines depict trial information, with one line per trial. Here, the definition of a trial is show a stimulus and collect a response.

The first column indicates the code of the experimental condition (1 for left, 2 for right).

The second column allows to specify the absolute onset time of a trial (not used here, therefore always 0).

Columns 3 and 4 depict the userDefinedSTMcolumns. userDefinedSTMcolumn1 denotes the index of the picture presented on the left. userDefinedSTMcolumn2 denotes the index of the picture presented on the right.

Columns 5 and 6 refer to the first page of the trial (show ./stimuli/stim_empty.bmp for 120 frames).

Columns 7 and 8 refer to the second page of the trial (show ./stimuli/stim_empty.bmp for 30 frames).

Columns 9 and 10 depict on which page to start and end response collection (here 1 and 1).

Column 11 indicates the correct response key (1 for left, 2 for right).

The plugin ASF_showTrial2Pics.m is written such that it interprets trial-definitions accordingly:

Lines 60-77 define the variables destinationRect1 and destinationRect1, which depict rectangular selections to the left and right of the center of the screen.

Lines 80-81

```
pic1 = atrial.userDefined(1);
pic2 = atrial.userDefined(2);
```

Lines 133-142

draw the image (an empty picture),
 put another image (pic1) on the left (destinationRect1), and
 put another image (pic2) on the right (destinationRect2)

```
%PUT THE APPROPRIATE TEXTURE(S) ON THE BACK BUFFER
Screen('DrawTexture', windowPtr, Stimuli.tex(atrial.pageNumber(i)));
if pic1 > 0
    %if valid picture index provided, put it on the left
    Screen('DrawTexture', windowPtr, Stimuli.tex(pic1), [], destinationRect1);
end
if pic2 > 0
    %if valid picture index provided, put it on the right
    Screen('DrawTexture', windowPtr, Stimuli.tex(pic2), [], destinationRect2)
end
```

In order for asf to use ASF_showTrial2Pics-plugin instead of the builtin default ShowTrial-function one needs to run asf with a configuration variable.

```
Cfg = [];
Cfg.userSuppliedTrialFunction = @ASF_showTrial2Pics;
Cfg.userDefinedSTMcolumns = 2;
ExpInfo = ASF('stimuli.std', 'trialdefs_short.trd', 'test', Cfg)
```

To get all possible combinations of all cats and all dogs on either side you need to define a trd file with 51 lines:

```
2 category catleft catright
1 0 2 7      1 120  1 30    1 1    1
1 0 2 8      1 120  1 30    1 1    1
1 0 2 9      1 120  1 30    1 1    1
1 0 2 10     1 120  1 30    1 1    1
1 0 2 11     1 120  1 30    1 1    1
1 0 3 7      1 120  1 30    1 1    1
1 0 3 8      1 120  1 30    1 1    1
1 0 3 9      1 120  1 30    1 1    1
1 0 3 10     1 120  1 30    1 1    1
1 0 3 11     1 120  1 30    1 1    1
1 0 4 7      1 120  1 30    1 1    1
1 0 4 8      1 120  1 30    1 1    1
1 0 4 9      1 120  1 30    1 1    1
1 0 4 10     1 120  1 30    1 1    1
1 0 4 11     1 120  1 30    1 1    1
1 0 5 7      1 120  1 30    1 1    1
1 0 5 8      1 120  1 30    1 1    1
1 0 5 9      1 120  1 30    1 1    1
1 0 5 10     1 120  1 30    1 1    1
1 0 5 11     1 120  1 30    1 1    1
1 0 6 7      1 120  1 30    1 1    1
1 0 6 8      1 120  1 30    1 1    1
1 0 6 9      1 120  1 30    1 1    1
```


1	0	6	10	1	120	1	30	1	1	1
1	0	6	11	1	120	1	30	1	1	1
2	0	7	2	1	120	1	30	1	1	2
2	0	7	3	1	120	1	30	1	1	2
2	0	7	4	1	120	1	30	1	1	2
2	0	7	5	1	120	1	30	1	1	2
2	0	7	6	1	120	1	30	1	1	2
2	0	8	2	1	120	1	30	1	1	2
2	0	8	3	1	120	1	30	1	1	2
2	0	8	4	1	120	1	30	1	1	2
2	0	8	5	1	120	1	30	1	1	2
2	0	8	6	1	120	1	30	1	1	2
2	0	9	2	1	120	1	30	1	1	2
2	0	9	3	1	120	1	30	1	1	2
2	0	9	4	1	120	1	30	1	1	2
2	0	9	5	1	120	1	30	1	1	2
2	0	9	6	1	120	1	30	1	1	2
2	0	10	2	1	120	1	30	1	1	2
2	0	10	3	1	120	1	30	1	1	2
2	0	10	4	1	120	1	30	1	1	2
2	0	10	5	1	120	1	30	1	1	2
2	0	10	6	1	120	1	30	1	1	2
2	0	11	2	1	120	1	30	1	1	2
2	0	11	3	1	120	1	30	1	1	2
2	0	11	4	1	120	1	30	1	1	2
2	0	11	5	1	120	1	30	1	1	2
2	0	11	6	1	120	1	30	1	1	2

5 Show text

6 Play sounds

7 Synchronize pictures and sounds

8 Issue a trigger when a stimulus is presented (EEG/MEG)

9 Synchronize your experiment to an external event (fMRI)

10 Appendix

11

11.1 Understanding the builtin function `ASF_showTrial.m`

```

1  function TrialInfo = ShowTrial(atrial, windowPtr, Stimuli, Cfg)
2  if ~isfield(Cfg, 'feedbackResponseNumber'), Cfg.feedbackResponseNumber = 1; else end; %IF YOU WANT TO GIVE
3  FEEDBACK REFER BY DEFAULT TO THE FIRST RESPONSE GIVEN IN THIS TRIAL
4  %SAVE TIME BY ALLOCATING ALL VARIABLES UPFRONT
5
6  % VBLTimestamp system time (in seconds) when the actual flip has happened
7  % StimulusOnsetTime An estimate of Stimulus-onset time
8  % FlipTimestamp is a timestamp taken at the end of Flip's execution
9  VBLTimestamp = 0; StimulusOnsetTime = 0; FlipTimestamp = 0; Missed = 0;
10 Beampos = 0;
11
12 StartRTMeasurement = 0; EndRTMeasurement = 0;
13 timing = [0, VBLTimestamp, StimulusOnsetTime, FlipTimestamp, Missed, Beampos];
14 nPages = length(atrial.pageNumber);
15 timing(nPages, end) = 0;
16 this_response = [];
17
18 %ON PAGES WITH WITH RESPONSE COLLECTION MAKE SURE THE CODE RETURNS IN TIME
19 %BEFORE THE NEXT VERTICAL BLANK. FOR EXAMPLE IF THE RESPONSE WINDOW IS 1000
20 %ms TOLERANCE MAKES THE RESPONSE COLLECTION CODE RETURN AFTER 1000ms-0.3
21 %FRAMES, I.E. AFTER 995 ms AT 60Hz
22 toleranceSec = Cfg.Screen.monitorFlipInterval*0.3;
23
24 %HOWEVER, THIS MUST NOT BE LONGER THAN ONE FRAME
25 %DURATION. EXPERIMENTING WITH ONE QUARTER OF A FRAME
26 responseGiven = 0;
27 this_response.key = [];
28 this_response.RT = [];
29
30
31 %-----
32 %TRIAL PRESENTATION HAS SEVERAL PHASES
33 % 1) WAIT FOR THE RIGHT TIME TO START TRIAL PRESENTATION. THIS MAY BE
34 %    IMMEDIATELY OR USER DEFINED (E.G. IN fMRI EXPERIMENTS)
35 %
36 % 2) LOOP THROUGH PAGE PRESENTATIONS WITHOUT RESPONSE COLLECTION

```

```

37 %
38 % 3) LOOP THROUGH PAGE PRESENTATIONS WHILE CHECKING FOR USER INPUT/RESPONSES
39 %
40 % 4) LOOP THROUGH PAGE PRESENTATIONS WITHOUT RESPONSE COLLECTION
41 % (AFTER RESPONSE HAS BEEN GIVEN)
42 %
43 % 5) FEEDBACK
44 %-----
45
46 %IF YOU WANT TO DO ANY OFFLINE STIMULUS RENDERING (I.E. BEFORE THE TRIAL
47 %STARTS), PUT THAT CODE HERE
48 responseCounter = 0; %COUNTS THE NUMBER OF RESPONSES GIVEN WITHIN THE ALLOWED PERIOD
49 %LOG DATE AND TIME OF TRIAL
50 strDate = datestr(now); %store when trial was presented
51
52 %-----
53 % PHASE 1) WAIT FOR THE RIGHT TIME TO START TRIAL PRESENTATION. THIS MAY BE
54 % IMMEDIATELY OR USER DEFINED (E.G. IN fMRI EXPERIMENTS)
55 %-----
56
57 %IF EXTERNAL TIMING REQUESTED (e.g. fMRI JITTERING)
58 if Cfg.useTrialOnsetTimes
59     wakeupTime = WaitSecs('UntilTime', Cfg.experimentStart + atrial.tOnset);
60 else
61     wakeupTime = GetSecs;
62 end
63 %LOG TIME OF TRIAL ONSET WITH RESPECT TO START OF THE EXPERIMENT
64 %USEFUL FOR DATA ANALYSIS IN fMRI
65 tStart = wakeupTime - Cfg.experimentStart;
66
67
68 if Cfg.Eyetracking.doDriftCorrection
69     EyelinkDoDriftCorrect(Cfg.Eyetracking.el);
70 end
71
72 %-----

```

```

73 %END OF PHASE 1
74 %-----
75
76 %MESSAGE TO EYELINK
77 Cfg = ASF_sendMessageToEyelink(Cfg, 'TRIALSTART');
78
79 %-----
80 % PHASE 2) LOOP THROUGH PAGE PRESENTATIONS WITHOUT RESPONSE COLLECTION
81 %-----
82 %CYCLE THROUGH PAGES FOR THIS TRIAL
83 atrial.nPages = length(atrial.pageNumber);
84 for iPage = 1:atrial.startRTonPage-1
85     if (iPage > atrial.nPages)
86         break;
87     else
88         %PUT THE APPROPRIATE TEXTURE ON THE BACK BUFFER
89         Screen('DrawTexture', windowPtr, Stimuli.tex(atrial.pageNumber(iPage)));
90
91         %PRESERVE BACK BUFFER IF THIS TEXTURE IS TO BE SHOWN
92         %AGAIN AT THE NEXT FLIP
93         bPreserveBackBuffer = atrial.pageDuration(iPage) > 1;
94
95         %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT IN THE
96         %BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED AGAIN TO THE SCREEN
97         [VBLTimestamp StimulusOnsetTime FlipTimestamp Missed Beampos] =...
98             ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
99             Cfg, bPreserveBackBuffer);
100
101         %SET TRIGGER (PARALLEL PORT AND EYELINK)
102         ASF_setTrigger(Cfg, atrial.pageNumber(iPage));
103
104
105         %LOG WHEN THIS PAGE APPEARED
106         timing(iPage, 1:6) = [atrial.pageDuration(iPage), VBLTimestamp,...
107             StimulusOnsetTime FlipTimestamp Missed Beampos];
108

```

```

109
110     %WAIT OUT STIMULUS DURATION IN FRAMES. WE USE PAGE FLIPPING RATHER
111     %THAN A TIMER WHENEVER POSSIBLE BECAUSE GRAPHICS BOARDS PROVIDE
112     %EXCELLENT TIMING; THIS IS THE REASON WHY WE MAY WANT TO KEEP A
113     %STIMULUS IN THE BACKBUFFER (NONDESTRUCTIVE PAGE FLIPPING)
114     %NOT ALL GRAPHICS CARDS CAN DO THIS. FOR CARDS WITHOUT AUXILIARY
115     %BACKBUFFERS WE COPY THE TEXTURE EXPLICITLY ON THE BACKBUFFER AFTER
116     %IT HAS BEEN DESTROYED BY FLIPPING
117     nFlips = atrial.pageDuration(iPage) - 1; %WE ALREADY FLIPPED ONCE
118     for FlipNumber = 1:nFlips
119         %PRESERVE BACK BUFFER IF THIS TEXTURE IS TO BE SHOWN
120         %AGAIN AT THE NEXT FLIP
121         bPreserveBackBuffer = FlipNumber < nFlips;
122
123         %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT
124         %IN THE BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED
125         %AGAIN TO THE SCREEN
126         ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
127             Cfg, bPreserveBackBuffer);
128     end
129 end
130 end
131 %-----
132 %END OF PHASE 2
133 %-----
134
135 %-----
136 % PHASE 3) LOOP THROUGH PAGE PRESENTATIONS WHILE CHECKING FOR USER
137 %     INPUT/RESPONSES
138 %-----
139 %SPECIAL TREATMENT FOR THE DISPLAY PAGES ON WHICH WE ALLOW REACTIONS
140
141 for iPage = atrial.startRTonPage:atrial.endRTonPage
142     if (iPage > atrial.nPages)
143         break;
144     else

```

```

145
146 %PUT THE APPROPRIATE TEXTURE ON THE BACK BUFFER
147 Screen('DrawTexture', windowPtr, Stimuli.tex(atrial.pageNumber(iPage)));
148
149 %DO NOT PUT THIS PAGE AGAIN ON THE BACKBUFFER, WE WILL WAIT IT OUT
150 %USING THE TIMER NOT FLIPPING
151 bPreserveBackBuffer = 0;
152
153 %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT
154 %IN THE BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED
155 %AGAIN TO THE SCREEN
156 [VBLTimestamp StimulusOnsetTime FlipTimestamp Missed Beampos] =...
157     ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
158     Cfg, bPreserveBackBuffer);
159
160 %SET TRIGGER
161 ASF_setTrigger(Cfg, atrial.pageNumber(iPage));
162
163 if iPage == atrial.startRTonPage
164     StartRTMeasurement = VBLTimestamp;
165 end
166
167 %STORE TIME OF PAGE FLIPPING FOR DIAGNOSTIC PURPOSES
168 timing(iPage, 1:6) = [atrial.pageDuration(iPage), VBLTimestamp,...
169     StimulusOnsetTime, FlipTimestamp, Missed, Beampos];
170
171 pageDuration_in_sec =...
172     atrial.pageDuration(iPage)*Cfg.Screen.monitorFlipInterval;
173
174 [x, y, buttons, t0, t1] =...
175     ASF_waitForResponse(Cfg, pageDuration_in_sec - toleranceSec);
176
177 if any(buttons)
178     % ShowCursor
179     %A BUTTON HAS BEEN PRESSED BEFORE TIMEOUT
180     if Cfg.responseTerminatesTrial

```

```

181         %ANY CODE THAT YOU FEEL APPROPRIATE FOR SIGNALING THAT
182         %PARTICIPANT HAS PRESSED A BUTTON BEFORE THE TRIAL ENDED
183         %Snd('Play','Quack')
184     else
185         %WAIT OUT THE REMAINDER OF THE STIMULUS DURATION WITH
186         %MARGIN OF toleranceSec
187         %MAKE THIS CONFIGURABLE AND GO HERE IF I ALLOW ONLY ONE
188         %RESPONSE? JVS20130329
189         wakeupTime = WaitSecs('UntilTime',...
190             StimulusOnsetTime + pageDuration_in_sec - toleranceSec);
191     end
192     if responseGiven == 0
193         responseGiven = 1;
194         responseCounter = responseCounter + 1;
195         %FIND WHICH BUTTON IT WAS
196         this_response.key = find(buttons);
197         %COMPUTE RESPONSE TIME
198         this_response.RT = (t1 - StartRTMeasurement)*1000;
199     end
200 end
201 end
202 end
203 %-----
204 %END OF PHASE 3
205 %-----
206
207 %-----
208 % PHASE 4) LOOP THROUGH PAGE PRESENTATIONS WITHOUT RESPONSE COLLECTION
209 % (AFTER RESPONSE HAS BEEN GIVEN) SAME AS PHASE 2
210 %-----
211 %OTHER PICS
212 for iPage = atrial.endRTonPage+1:nPages
213     if (iPage > atrial.nPages)
214         break;
215     else
216         %PUT THE APPROPRIATE TEXTURE ON THE BACK BUFFER
217         Screen('DrawTexture', windowPtr, Stimuli.tex(atrial.pageNumber(iPage)));

```



```

218
219 %PRESERVE BACK BUFFER IF THIS TEXTURE IS TO BE SHOWN
220 %AGAIN AT THE NEXT FLIP
221 bPreserveBackBuffer = atrial.pageDuration(iPage) > 1;
222
223 %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT
224 %IN THE BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED
225 %AGAIN TO THE SCREEN
226 [VBLTimestamp StimulusOnsetTime FlipTimestamp Missed Beampos] =...
227     ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
228     Cfg, bPreserveBackBuffer);
229
230 %SET TRIGGER (PARALLEL PORT AND EYELINK)
231 ASF_setTrigger(Cfg, atrial.pageNumber(iPage));
232
233
234 %LOG WHEN THIS PAGE APPEARED
235 timing(iPage, 1:6) = [atrial.pageDuration(iPage), VBLTimestamp,...
236     StimulusOnsetTime FlipTimestamp Missed Beampos];
237
238 %WAIT OUT STIMULUS DURATION IN FRAMES.
239 nFlips = atrial.pageDuration(iPage) - 1; %WE ALREADY FLIPPED ONCE
240 for FlipNumber = 1:nFlips
241     %PRESERVE BACK BUFFER IF THIS TEXTURE IS TO BE SHOWN
242     %AGAIN AT THE NEXT FLIP
243     bPreserveBackBuffer = FlipNumber < nFlips;
244
245     %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT
246     %IN THE BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED
247     %AGAIN TO THE SCREEN
248     ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
249     Cfg, bPreserveBackBuffer);
250 end
251 end
252 end
253

```

```

254 %-----
255 %END OF PHASE 4
256 %-----
257
258 %-----
259 % PHASE 5) FEEDBACK
260 %-----
261 %IF YOU WANT TO FORCE A RESPONSE
262 if Cfg.waitUntilResponseAfterTrial && ~responseGiven
263     [x, y, buttons, t0, t1] = ASF_waitForResponse(Cfg, 10);
264
265     if any(buttons)
266         %A BUTTON HAS BEEN PRESSED BEFORE TIMEOUT
267         responseGiven = 1;    %#ok<NASGU>
268         %FIND OUT WHICH BUTTON IT WAS
269         this_response.key = find(buttons);
270         %COMPUTE RESPONSE TIME
271         this_response.RT = (t1 - StartRTMeasurement)*1000;
272     end
273 end
274
275 %TRIAL BY TRIAL FEEDBACK
276 if Cfg.feedbackTrialCorrect || Cfg.feedbackTrialError
277     ASF_trialFeedback(...
278         this_response.key(Cfg.feedbackResponseNumber) == atrial.CorrectResponse, Cfg, windowPtr);
279 end
280
281 %-----
282 %END OF PHASE 5
283 %-----
284
285
286 %PACK INFORMATION ABOUT THIS TRIAL INTO STRUCTURE TrialInfo (THE RETURN
287 %ARGUMENT). PLEASE MAKE SURE THAT TrialInfo CONTAINS THE FIELDS:
288 %   trial
289 %   datestr

```

```
290 % tStart
291 % Response
292 % timing
293 % StartRTMeasurement
294 % EndRTMeasurement
295 %OTHERWISE DIAGNOSTIC PROCEDURES OR ROUTINES FOR DATA ANALYSIS MIGHT FAIL
296 TrialInfo.trial = atrial; %REQUESTED PAGE NUMBERS AND DURATIONS
297 TrialInfo.datestr = strDate; %STORE WHEN THIS HAPPENED
298 TrialInfo.tStart = tStart; %TIME OF TRIAL-START
299 TrialInfo.Response = this_response; %KEY AND RT
300 TrialInfo.timing = timing; %TIMING OF PAGES
301 TrialInfo.StartRTMeasurement = StartRTMeasurement; %TIMESTAMP START RT
302 TrialInfo.EndRTMeasurement = EndRTMeasurement; %TIMESTAMP END RT
```

