Workshop OpenSesame

Cats, Dogs & Capybara's

Before you start:

This workshop contributes 1/3 of your workshop participation grade. The workshop participation grade is 10% of your final grade for the course – hence, today's workshop contributes 1/3 points. Hand in your experiment before September 17th, 23:59, via Canvas.

When you start OpenSesame, you will be given a choice of template experiments, and (if any) a list of recently opened experiments (see Figure 1).

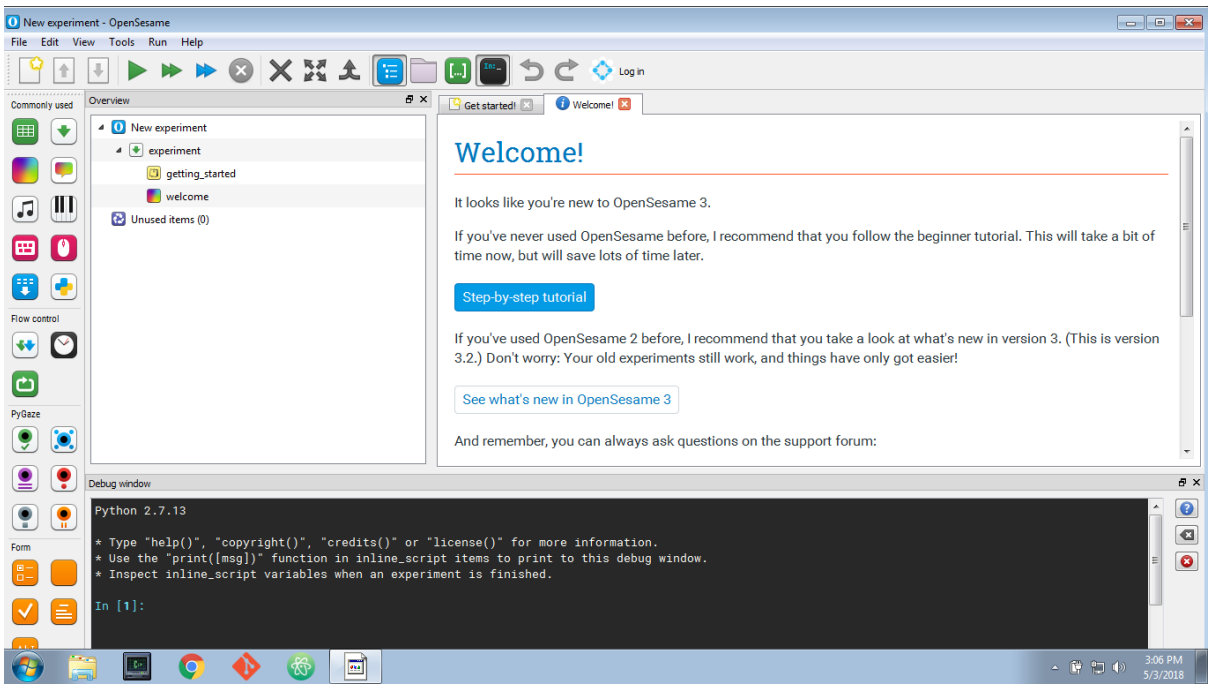

Figure 1. The OpenSesame window on start-up.

The *Extended template* provides a good starting point for creating trial-based experiments. However, in this tutorial we will create the entire experiment from scratch. Therefore, we will continue with the 'default template', which is already loaded when OpenSesame is launched (Figure 2). Therefore, simply close the 'Get started!' and (if shown) 'Welcome!' tabs.

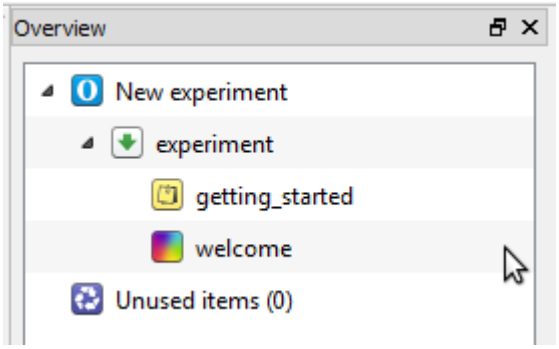

Figure 2. The structure of the 'Default template' as seen in the overview area.

Background box 1: Basics

OpenSesame experiments are collections of *items*. An item is a small chunk of functionality that, for example, can be used to present visual stimuli (the sketchpad item) or to record key presses (the keyboard_response item). Items have a type and a name. For example, you might have two items of the type keyboard_response with the names *t1_response* and *t2_response*. To make the distinction between item types and item names clear, we will use this_style for types, and *this style* for names.

To give structure to your experiment, two types of items are especially important: the loop and the sequence. Understanding how you can combine loops and sequences to build experiments is perhaps the trickiest part of working with OpenSesame, so let's get that out of the way first.

A loop is where, in most cases, you define your independent variables. In a loop you can create a table in which each column corresponds to a variable, and each row corresponds to a single run of the 'item to run'. To make this more concrete, let's consider the following *block_loop* (unrelated to this tutorial):

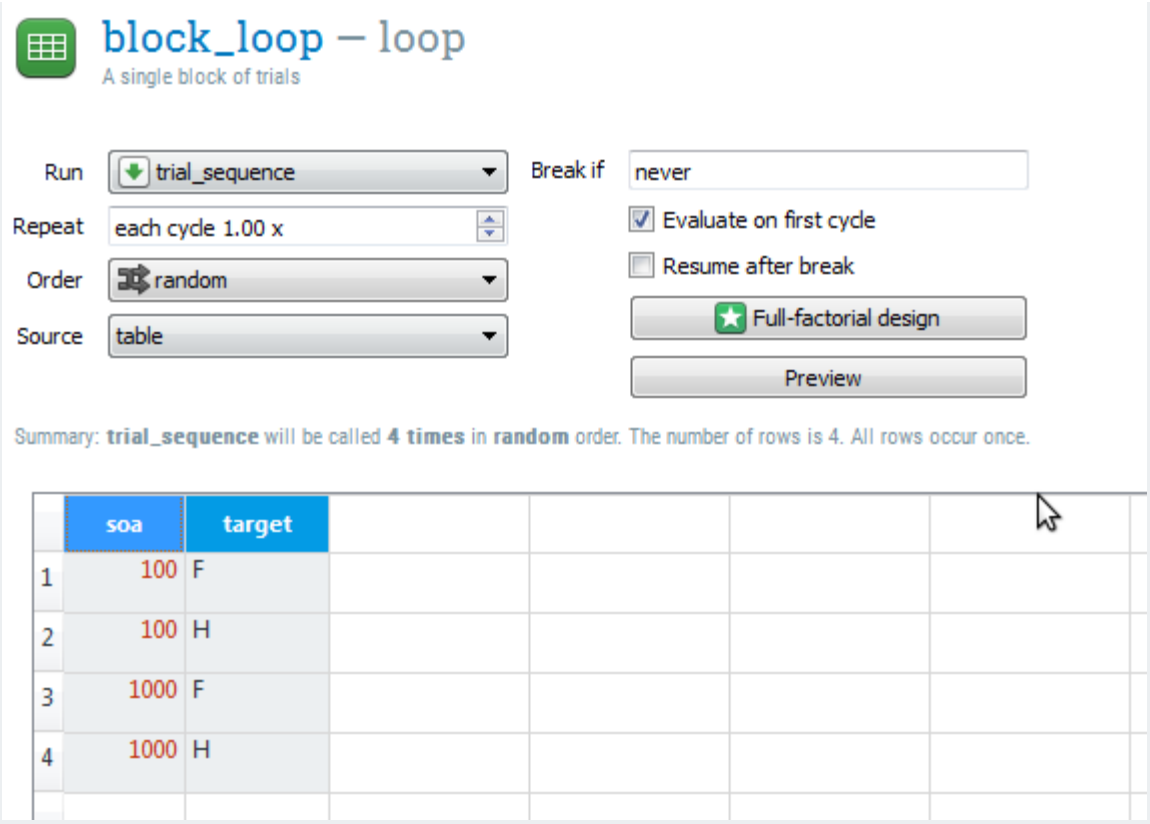

Figure 3. An example of variables defined in a loop table. (This example is not related to the experiment created in this tutorial.)

This *block_loop* will execute *trial_sequence* four times. Once while `soa` is 100 and `target` is 'F', once while `soa` is 100 and `target` is 'H', etc. The order in which the rows are walked through is random by default, but can also be set to sequential in the top-right of the tab.

A sequence consists of a series of items that are executed one after another. A prototypical sequence is the *trial_sequence*, which corresponds to a single trial. For example, a basic *trial_sequence* might consist of a sketchpad, to present a stimulus, a keyboard_response, to collect a response, and a logger, to write the trial information to the log file.

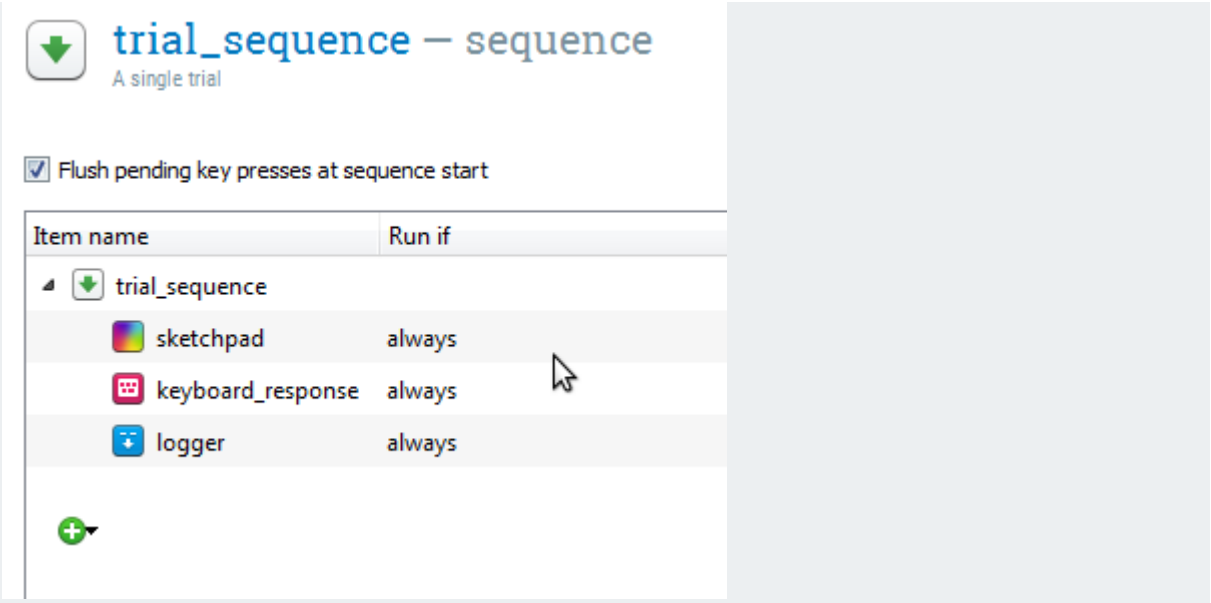

Figure 4. An example of a sequence item used as a trial sequence. (This example is not related to the experiment created in this tutorial.)

You can combine loops and sequences in a hierarchical way, to create trial blocks, and practice and experimental phases. For example, the *trial_sequence* is called by the *block_loop*. Together, these correspond to a single block of trials. One level up, the *block_sequence* is called by the *practice_loop*. Together, these correspond to the practice phase of the experiment.

Step 2: Add a block_loop and trial_sequence

The default template starts with three items: A notepad called *getting_started*, a sketchpad called *welcome*, and a sequence called *experiment*. We don't need *getting_started* and *welcome*, so let's remove these right away. To do so, right-click on these items and select 'Delete'. Don't remove *experiment*, because it is the entry for the experiment (i.e. the first item that is called when the experiment is started).

Our experiment will have a very simple structure. At the top of the hierarchy is a loop, which we will call *block_loop*. The *block_loop* is the place where we will define our independent variables (see also Background box 1). To add a loop to your experiment, drag the loop icon from the item toolbar onto the *experiment* item in the overview area.

A loop item needs another item to run; usually, and in this case as well, this is a sequence. Drag the sequence item from the item toolbar onto the *new_loop* item in the overview area. OpenSesame will ask whether you want to insert the sequence into or after the loop. Select 'Insert into new_loop'.

By default, items have names such as *new_sequence*, *new_loop*, *new_sequence_2*, etc. These names are not very informative, and it is good practice to rename them. Item names must consist of alphanumeric characters and/ or underscores. To rename an item, double-click on the item in the overview area. Rename *new_sequence* to *trial_sequence* to indicate that it will correspond to a single trial. Rename *new_loop* to *block_loop* to indicate that will correspond to a block of trials.

The overview area of our experiment now looks as in Figure 5.

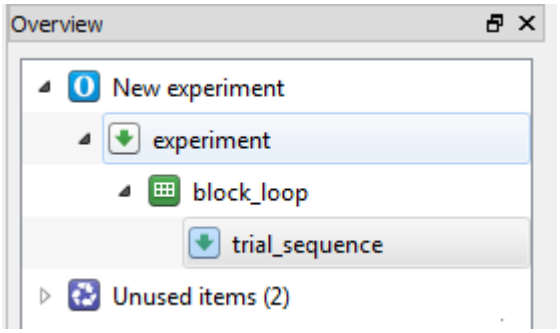

Figure 5. The overview area at the end of Step 2.

Background box 3: Unused items

Tip — Deleted items are still available in the Unused Items bin, until you select 'Permanently delete unused items' in the Unused Items tab. You can re-add deleted items to your experiment by dragging them out of the Unused Items bin into a sequence or loop.

Step 3: Import images and sound files

For this experiment, we will use images of cats, dogs, and capybaras. We will also use sound samples of meows and barks. You can download all the required files from here:

- https://osdoc.cogsci.nl/3.3/attachments/cats-dogs-capybaras/stimuli.zip

Download `stimuli.zip` and extract it somewhere (to your desktop, for example). Next, in OpenSesame, click on the 'Show file pool' button in the main toolbar (or: Menu →View → Show file pool). This will show the file pool, by default on the right side of the window. The easiest way to add the stimuli to the file pool is by dragging them from the desktop (or wherever you have extracted the files to) into the file pool. Alternatively, you can click on the '+' button in the file pool and add files using the file-selection dialog that appears. The file pool will automatically be saved with your experiment.

After you have added all stimuli, your file pool looks as in Figure 6.

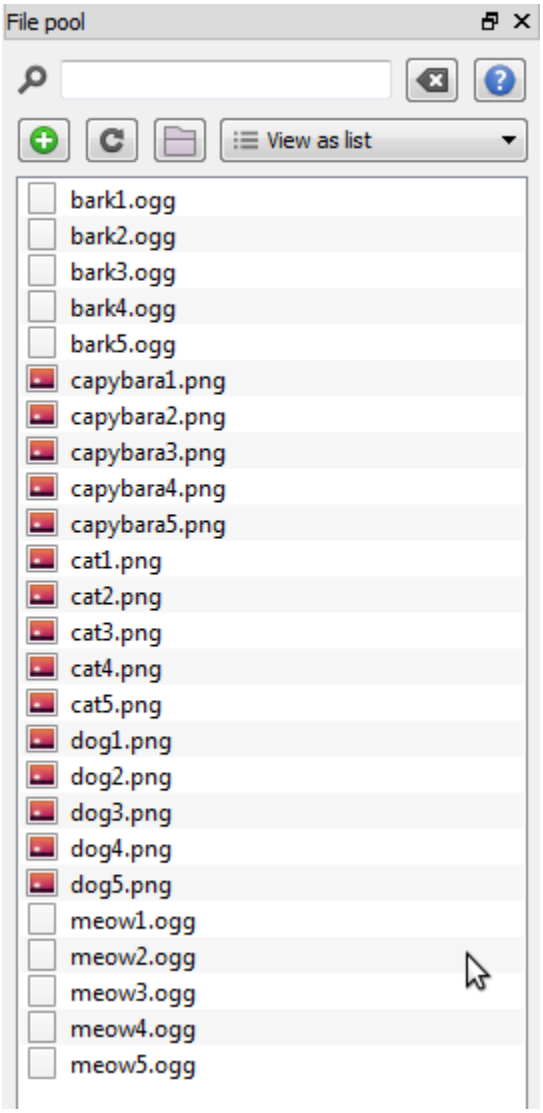

Figure 6. The file pool at the end of Step 3.

Step 4: Define the experimental variables in the block_loop

Conceptually, our experiment has a fully crossed 3×2 design: We have three types of visual stimuli (cats, dogs, and capybaras) which occur in combination with two types of auditory stimuli (meows and barks). However, we have five exemplars for each stimulus type: five meow sounds, five capybara pictures, etc. From a technical point of view, it therefore makes sense to treat our experiment as a 5×5×3×2 design, in which picture number and sound number are factors with five levels.

OpenSesame is very good at generating full-factorial designs. First, open *block_loop* by clicking on it in the overview area. Next, click on the Full-Factorial Design button. This will open a wizard for generating full-factorial designs, which works in a straightforward way: Every column corresponds to an experimental variable (i.e. a factor). The first row is the name of the variable, the rows below contain all possible values (i.e. levels). In our case, we can specify our 5×5×3×2 design as shown in Figure 7.

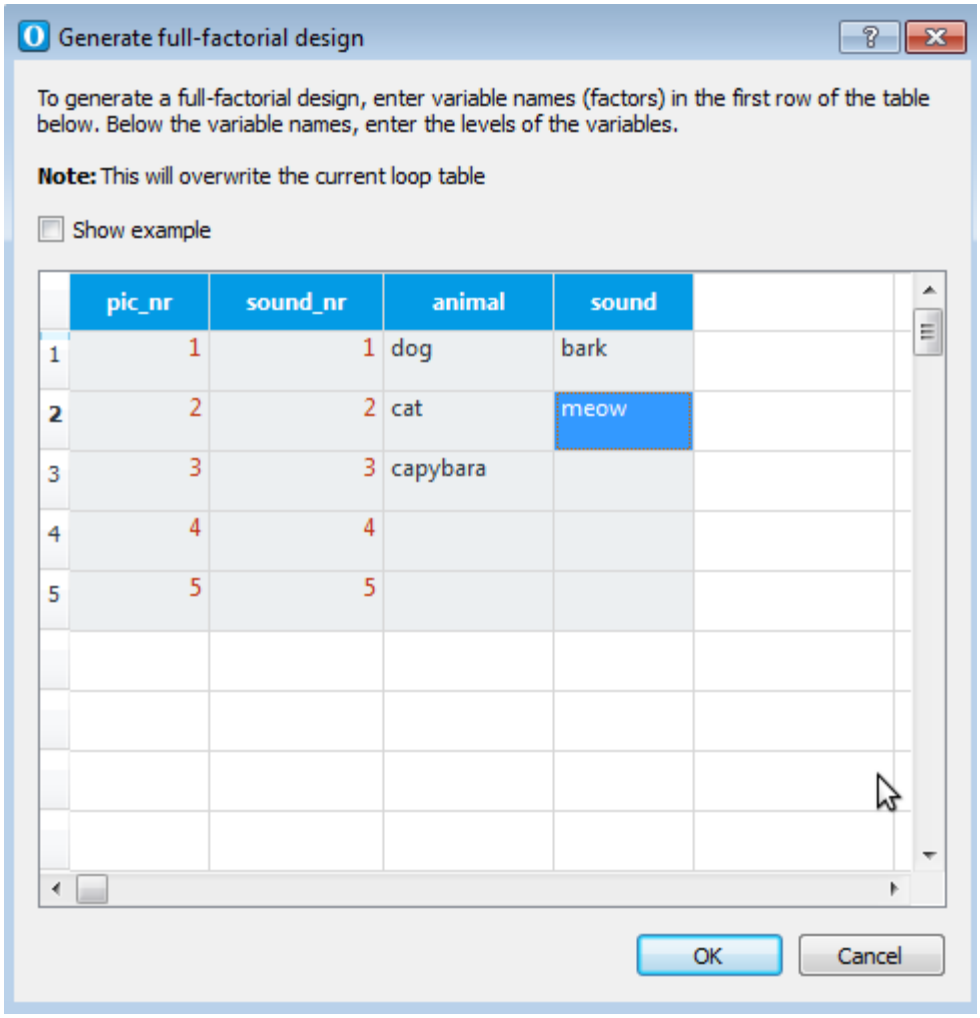

Figure 7. The loop wizard generates full-factorial designs.

After clicking 'Ok', you will see that there is now a loop table with four rows, one for each experimental variable. There are 150 cycles (=5×5×3×2), which means that we have 150 unique trials. Your loop table now looks as in Figure 8.

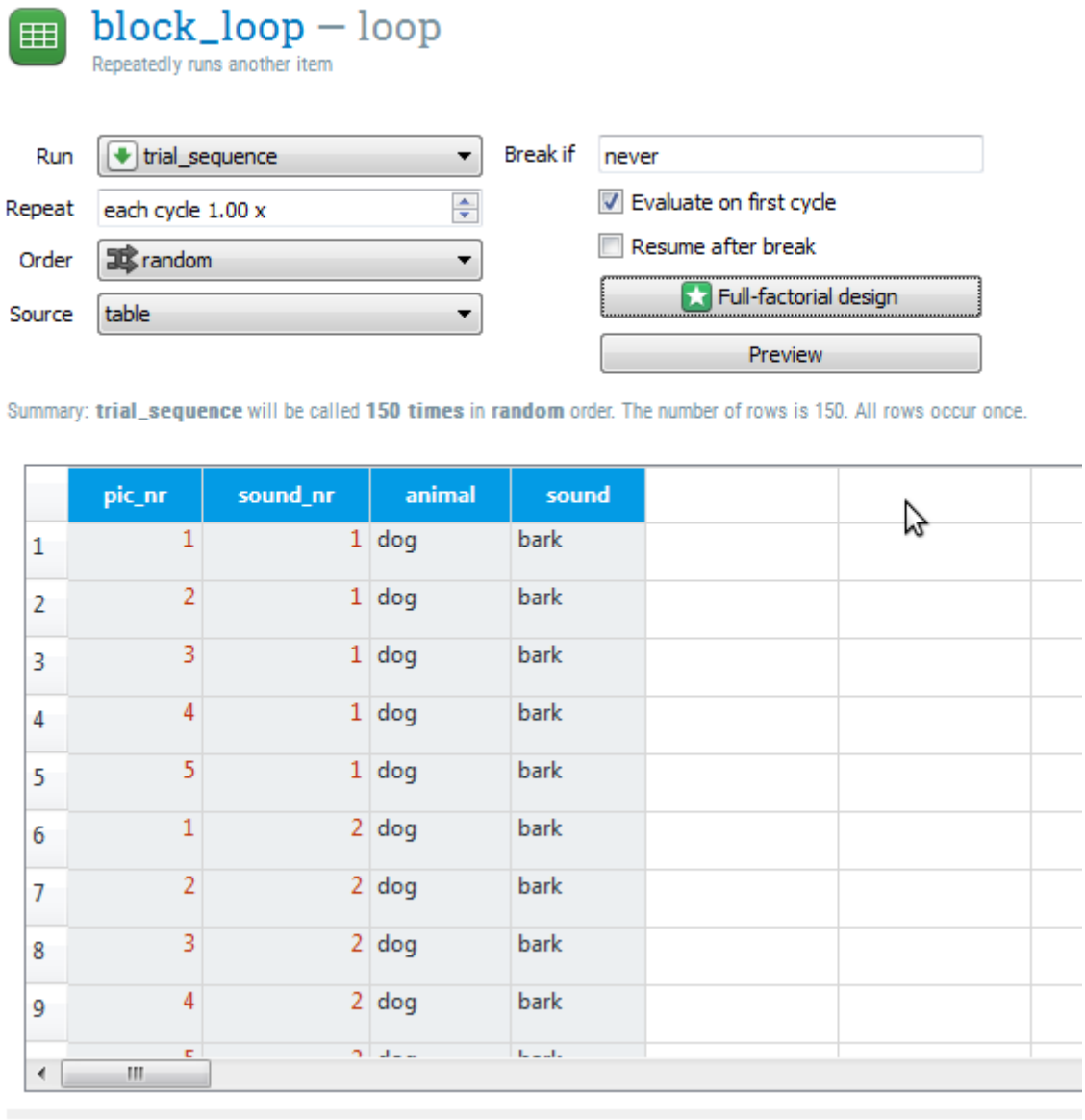

Figure 8. The loop table at the end of Step 4.

Step 5: Add items to the trial sequence

Open *trial_sequence*, which is still empty. It's time to add some items! Our basic *trial_sequence* is:

1. A sketchpad to display a central fixation dot for 500 ms
2. A sampler to play an animal sound
3. A sketchpad to display an animal picture
4. A keyboard_response to collect a response
5. A logger to write the data to file

To add these items, simply drag them one by one from the item toolbar into the *trial_sequence*. If you accidentally drop items in the wrong place, you can simply re-order them by dragging and dropping. Once all items are in the correct order, give each of them a sensible name. The overview area now looks as in Figure 9.

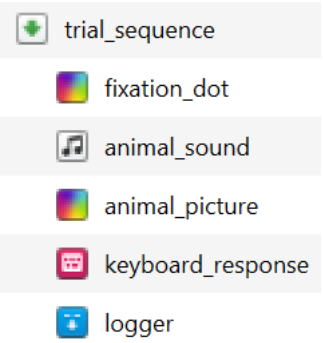

Figure 9. The overview area at the end of Step 5.

Step 6: Define the central fixation dot

Click on *fixation_dot* in the overview area. This opens a basic drawing board that you can use to design your visual stimuli. To draw a central fixation dot, first click on the crosshair icon, and then click on the center of the display, i.e. at position (0, 0).

We also need to specify for how long the fixation dot is visible. To do so, change the duration from 'keypress' to 500 ms.

The *fixation_dot* item now looks as in Figure 10.

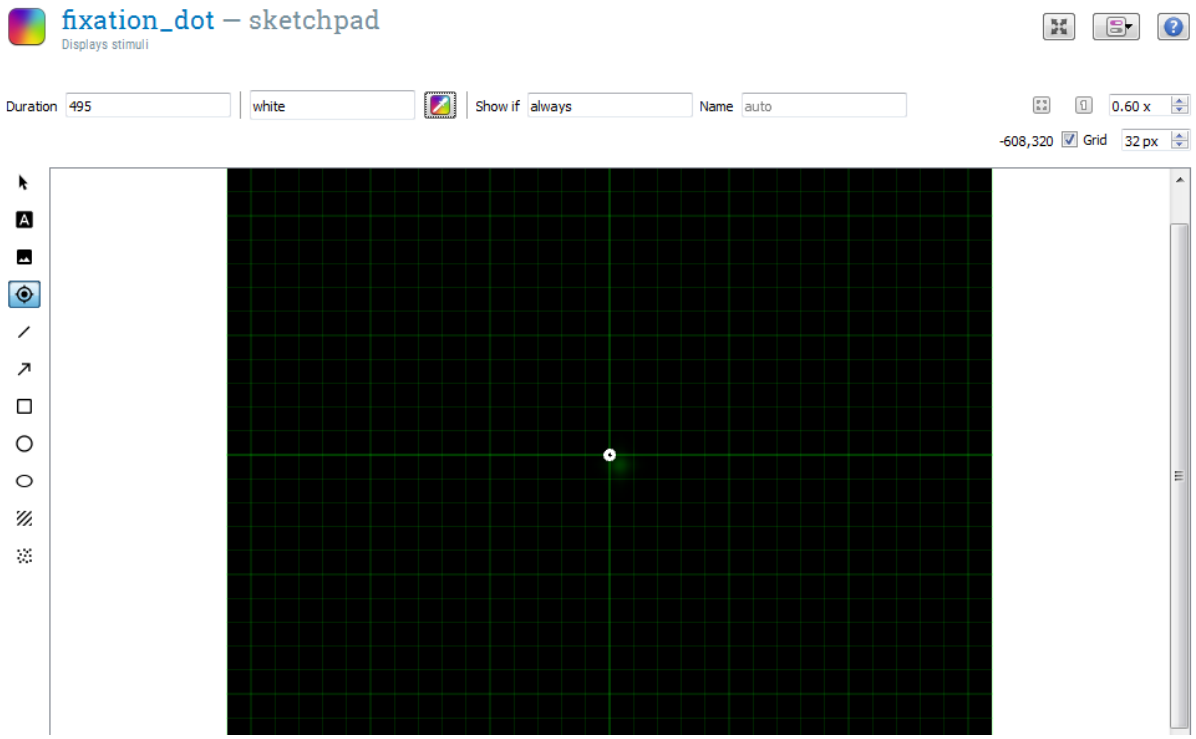

Figure 10. The *fixation_dot* item at the end of Step 6.

Step 7: Define the animal sound

Open *animal_sound*. The sampler item provides a number of options, the most important being the sound file that should be played. Click on the browse button to open the file-pool selection dialog, and select one of the sound files, such as `bark1.ogg`.

Of course, we don't want to play the same sound over-and-over again! Instead, we want to select a sound based on the variables `sound` and `sound_nr` that we have defined in the *block_loop* (Step 5). To do this, simply replace the part of the string that you want to have depend on a variable by the name of that variable between square brackets. More specifically, 'bark1.ogg' becomes '[sound][sound_nr].ogg', because we want to replace 'bark' by the value of the variable `sound` and '1' by the value of `sound_nr`.

We also need to change the duration of the sampler. By default, the duration is 'sound', which means that the experiment will pause while the sound is playing. Change the duration to 0. This does not mean that the sound will be played for only 0 ms, but that the experiment will advance right away to the next item, while the sound continues to play in the background. The item *animal_sound* now looks as shown in Figure 11.

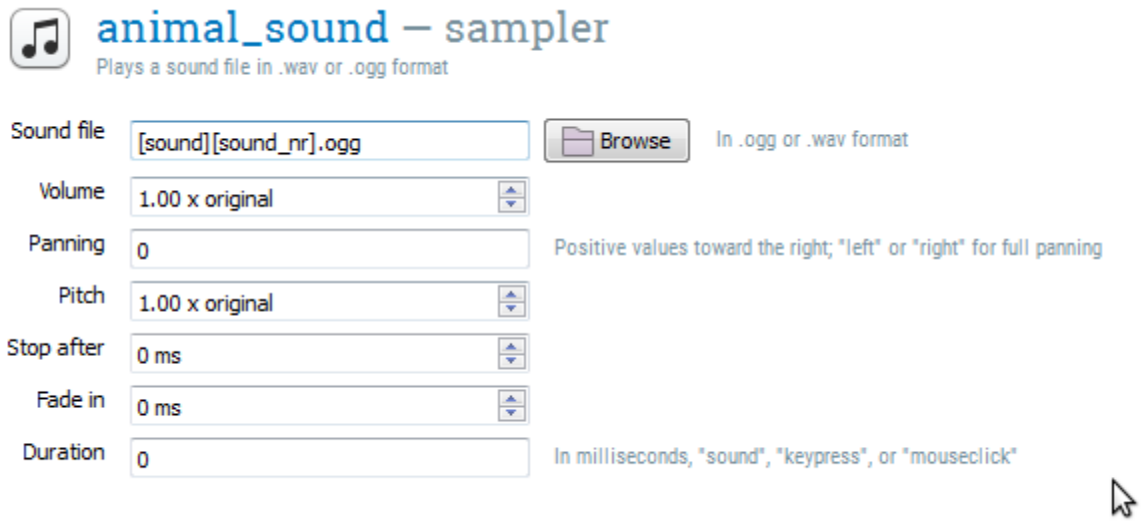

Figure 11. The item *animal_sound* at the end of Step 7.

Background box 5: Variables

For more information about using variables, see:

- https://osdoc.cogsci.nl/3.3/manual/variables

Step 8: Define the animal picture

Open *animal_picture*. Select the image tool by clicking on the button with the landscape-like icon. Click on the center (0, 0) of the display. In the File Pool dialog that appears, select `capybara1.png`. The capybara's sideways glance will now lazily stare at you from the center of the display. But of course, we don't always want to show the

same capybara. Instead, we want to have the image depend on the variables `animal` and `pic_nr` that we have defined in the *block_loop* (Step 4).

We can use essentially the same trick as we did for *animal_sound*, although things work slightly differently for images. First, right-click on the capybara and select 'Edit script'. This allows you to edit the following line of OpenSesame script that corresponds to the capybara picture:

```
draw image center=1 file="capybara1.png" scale=1 show_if=alway
s x=0 y=0 z_index=0
```

Now change the name of image file from 'capybara.png' to '[animal][pic_nr].png':

```
draw image center=1 file="[animal][pic_nr].png" scale=1 show_i
f=always x=0 y=0 z_index=0
```

Click on 'Ok' to apply the change. The capybara is now gone, replaced by a placeholder image, and OpenSesame tells you that one object is not shown, because it is defined using variables. Don't worry, it will be shown during the experiment!

Finally, set 'Duration' field to '0'. This does not mean that the picture is presented for only 0 ms, but that the experiment will advance to the next item (*response*) right away. Since *response* waits for a response, but doesn't change what's on the screen, the target will remain visible until a response has been given.

Step 9: Define the response

Open the *keyboard_response* item. This item collects a single keyboard button press.

- **Correct response** — this is where you can indicate which key is the correct response. As the correct response depends on the trial, you can leave this field empty.

  When left empty, the keyboard item will look if the correct response is declared as a variable 'correct_response' elsewhere in our experiment. We will now make this variable in the *block_loop* item. Make a new column in block_loop, called correct_response, and on each of the rows below, enter 'z' (without quotation marks) or '/', depending on whether the value for 'animal' is dog or cat, respectively. As such, we have a left- and righthanded button press for dogs and cats, respectively. For capybara's, enter the value None (we want participants to not respond upon seeing a capybara, after all). 'None' should be written with a capital N.

- **Allowed responses** is a semicolon-separated list of keys that are accepted. Let's set this to "z;/", meaning that we'll accommodate a left- and righthanded key press.
- **Timeout** indicates a duration after which the response will be set to 'None', and the experiment will continue. A timeout is important in our experiment, because participants need to have the opportunity to *not* respond when they see a capybara. So let's set the timeout to 2000.

Step 10: Define the logger

We don't need to configure the logger, because its default settings are fine; but let's take a look at it anyway. Click on *logger* in the overview area to open it. You see that the option 'Log all variables (recommended)' is selected. This means that OpenSesame logs everything, which is fine.

Background box 8: Always check your data!

The one tip to rule them all — Always triple-check whether all necessary variables are logged in your experiment! The best way to check this is to run the experiment and investigate the resulting log files.

Step 11: Finished! (Sort of …)

You should now be able to run your experiment. There is still a lot of room for improvement, and you will work on polishing the experiment below. But the basic structure is there!

Click on the 'Run fullscreen' (`Control+R`) button in the main toolbar to give it a test run.

Background box 11: Quick run

Tip — A test run is executed even faster by clicking the orange 'Run in window' button, which doesn't ask you how to save the logfile (and should therefore only be used for testing purposes).

The following steps are for you to solve by yourself!

Easy: Add an instruction and goodbye screen

- sketchpad and form_text_display items can present text
- Good instructions are short and concrete

Easy: Inspect the data

- Run the experiment once on yourself. You can reduce the number of trials by setting the Repeat value of the *block_loop* to less than one.
- Open the data file in Excel and have a look at the types of variables being logged. RT and ‘correct’ are particularly relevant when analyzing data - as are our experimental variables.

Medium: Provide feedback on every trial

- A good, unobtrusive way to provide feedback is by briefly presenting a red dot after an incorrect response, and a green dot after a correct response

- Use Run-If statements! (clicking on trial_sequence, you see a run-if column. Here, use the statement '[correct] = 1' and '[correct] = 0', for a sketchpad item providing positive and negative feedback, respectively. Make sure that the feedback displays are shown for a limited amount of time (meaning that the experiment doesn't wait for another keypress before moving on).

Difficult: Divide the trials into multiple blocks

- Add a sketchpad to the end of the trial_sequence that invites participants to take a short break
- Use a Run If statement to run this sketchpad only after every 15 trials
- You need the modulo (`%`) operator to do this, as well as the variable `count_trial_sequence`