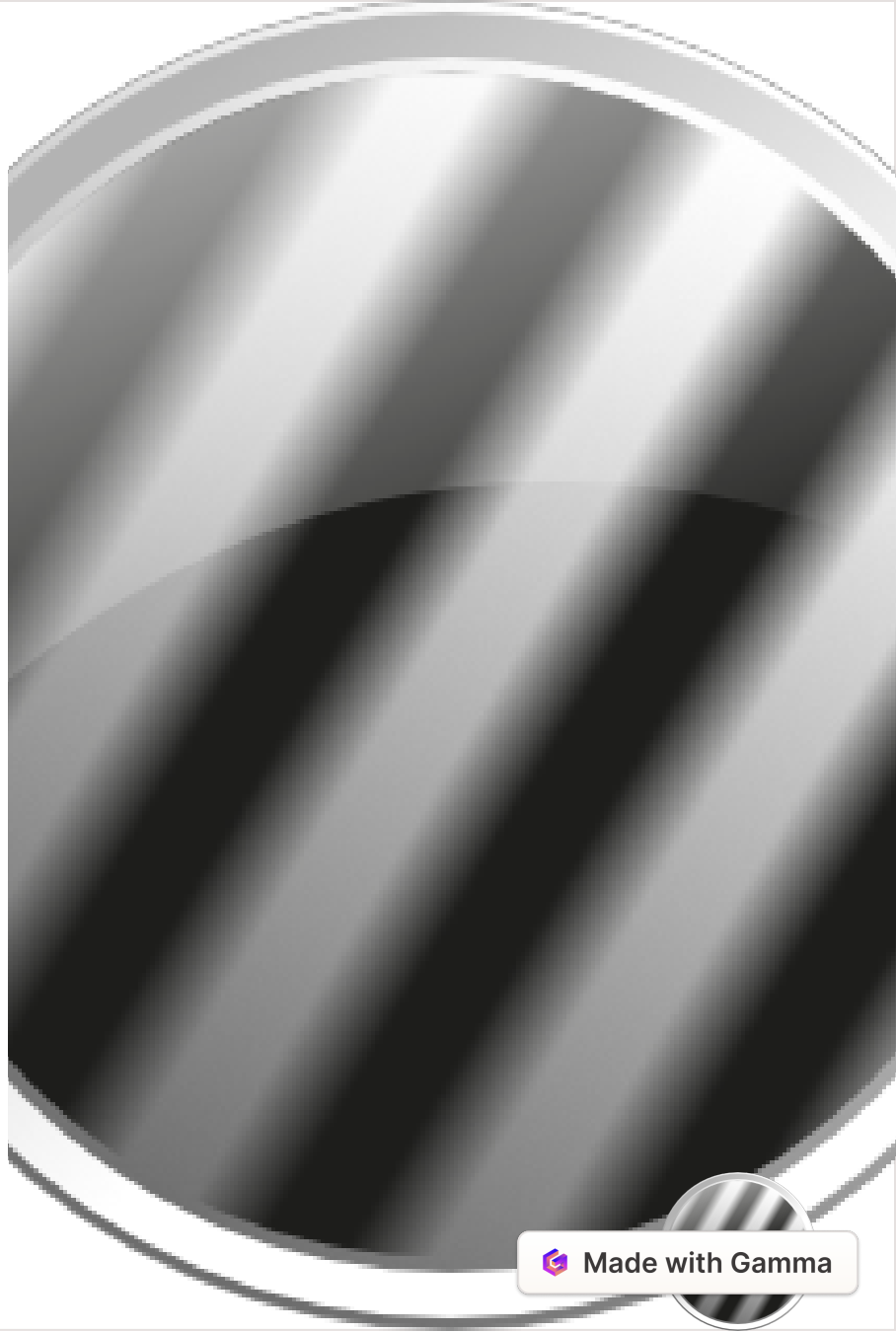
Creating Block Designs PsychoPy

In this session, we will explore how we build complex designs in PsychoPy, including blocked designs, counterbalancing and more.

by Becca Hirst



Session content

- Implementing block designs in PsychoPy
- Counterbalancing
- Branched designs
- Adaptive designs (e.g. psychophysics staircases)

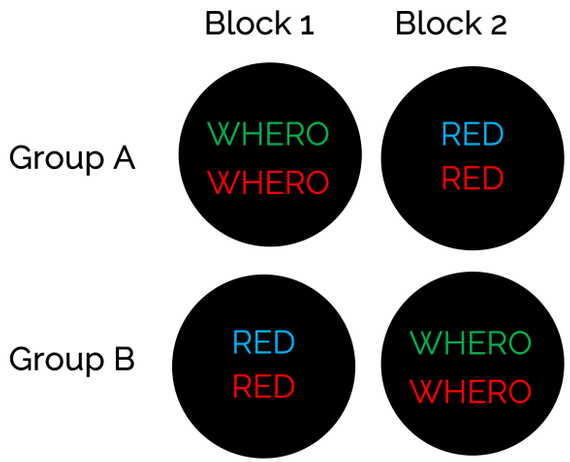
What is a "complex" design

Often in an experiment you want to present several "blocks" of trials. For example, in a bilingual Stroop task you might have a block of stimuli in English (**red, green**, **blue** …) and a block of trials in another language (**red, kākāriki**, **kikorangi**). You might want to present these blocks either as:

- **Randomised block designs:** two (or more) blocks of trials, in a random order.
- **Counterbalanced designs:** group A sees one block first, group B sees the other block first.

Other forms of complex design include:

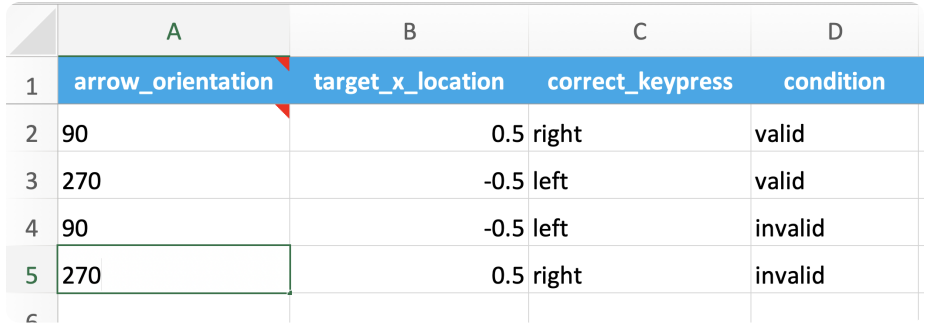
- **Branched designs:** Where we present a certain part of our experiment (or not) based on previous answers.
- **Adaptive designs**, where a task gets gradually harder, based on previous responses (this is a bit different, so we will cover that separately).



What makes a single "block"

At the moment, we have a spreadsheet listing a set of trials. This can be considered one "**block**" of trials.

(see previous session materials for a reminder on what these columns relate to!)



	A	B	C	D
1	arrow_orientation	target_x_location	correct_keypress	condition
2	90	0.5	right	valid
3	270	-0.5	left	valid
4	90	-0.5	left	invalid
5	270	0.5	right	invalid

Column headers: Parameters that change trial-by-trial.

Row values: The value of each parameter on each trial.

Making several blocks

To make several blocks, we need to first make a spreadsheet per block.

	A	B	C	D
1	arrow_orientation	target_x_location	correct_keypress	condition
2	90	0.5	right	valid
3	270	-0.5	left	valid
4	90	0.5	right	valid
5	270	-0.5	left	valid

A block of "valid" trials

Here the cue always points in the direction of the target.

	A	B	C	D
1	arrow_orientation	target_x_location	correct_keypress	condition
2	90	-0.5	left	invalid
3	270	0.5	right	invalid
4	90	-0.5	left	invalid
5	270	0.5	right	invalid

A block of "invalid" trials

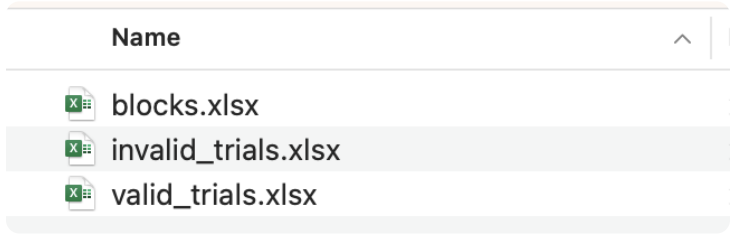
Here the cue always points in the opposite direction to the target.

	A
1	block_file
2	valid_trials.xlsx
3	invalid_trials.xlsx

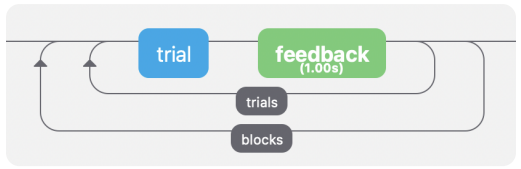
A file listing our blocks

This spreadsheet contains a column listing the trials to present in each block.

We therefore have 3 spreadsheets in total listed as below:



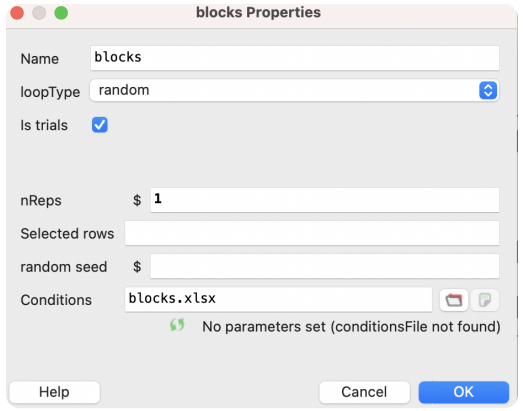
Adding blocks in PsychoPy



Add a nested loop

When we added a loop around our "trial" routine. This was because we were repeating that trial several times (with different parameters).

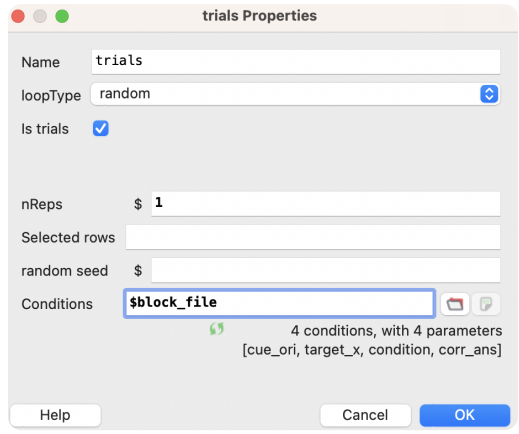
A block can be considered repeating a set of trials several times (with a different spreadsheet each time). So, we can therefore add a second loop around our trials.



Set up your blocks loop

The blocks loop takes in our blocks spreadsheet.

> nReps here corresponds to the number of times you repeat the two blocks.

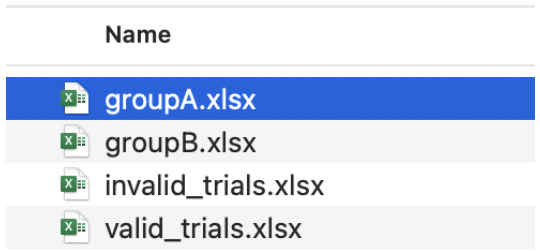


Update your trials loop

The trials loop now takes the variable `block_file` from our blocks spreadsheet (so the set of trials changes on each block.

Because your "blocks" loop loopType is set to "random" this means that you now have a randomised block design. The order of the blocks will be random.

Counterbalancing in PsychoPy



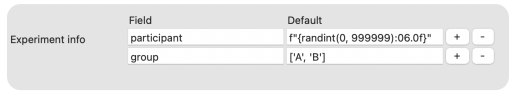
Make one block file per group

Here the order of rows for groupA.xlsx is:

- valid_trials.xlsx
- invalid_trials.xlsx

For group B the order of rows are:

- invalid_trials.xlsx
- valid_trials.xlsx

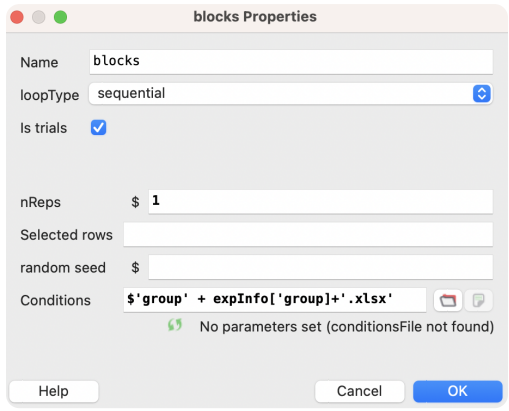


Add a "group" field to your Experiment Settings

Select the cog icon to access experiment settings. Add a field called group.

- group : `['A', 'B']`

This will present a drop down at the start of your task to select a Group.



Update your blocks loop

You can access the answer given in the startup dialogue using `expInfo['group']`. By using `$'group'+expInfo['group']+'.xlsx'` we make a value that will be either "groupA.xlsx" or "groupB.xlsx"

Set the loopType to sequential.

Because your "blocks" loop loopType is set to "sequential" this means that you now have a design where group A will see one order of blocks and group B will see another order of blocks.

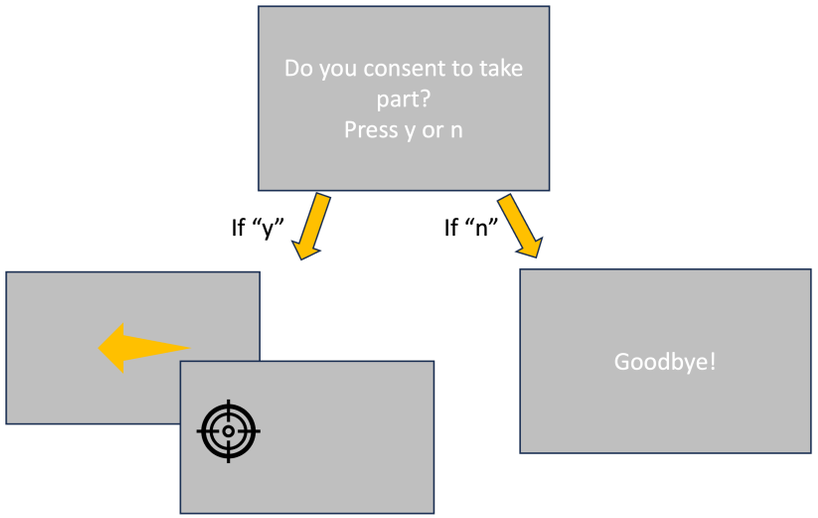
Exercise

Now you know how to access information from the experiment settings via `expInfo`. Add some text to your instructions routine to say "Hello" to the participant. For example if my participant ID is "Becca" I want the text to read "Hello Becca".

Branched designs

Imagine you only want to show part of your experiment based on a response given earlier in the experiment.

Example: I might have a "consent" routine. If the participant presses the "y" key I show my experiment. If they press the "n" key I do not show my experiment.

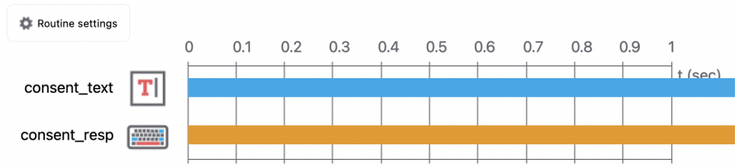


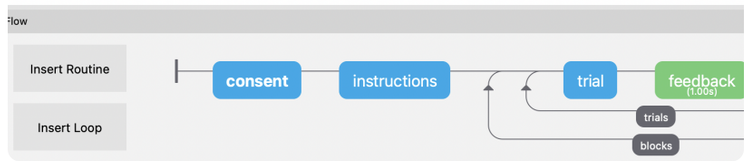
Add a "consent" routine

Here my "consent_resp" allows the "y" and "n" keys.

My consent_text reads as below:

Do you consent to take part?
Press y or n

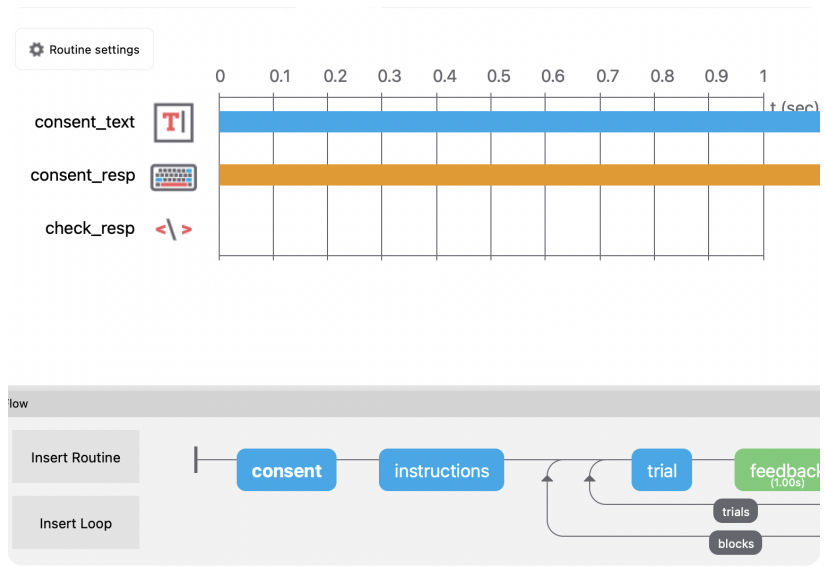




Add a code component

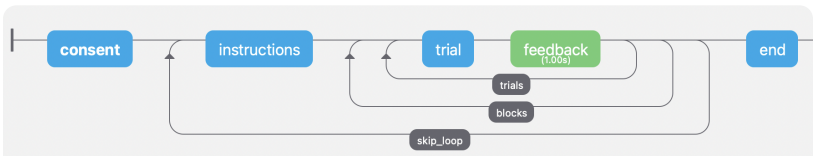
In the "end routine" of your code component add:

```
if consent_resp == 'y':
    show_task = 1
else:
    show_task = 0
```

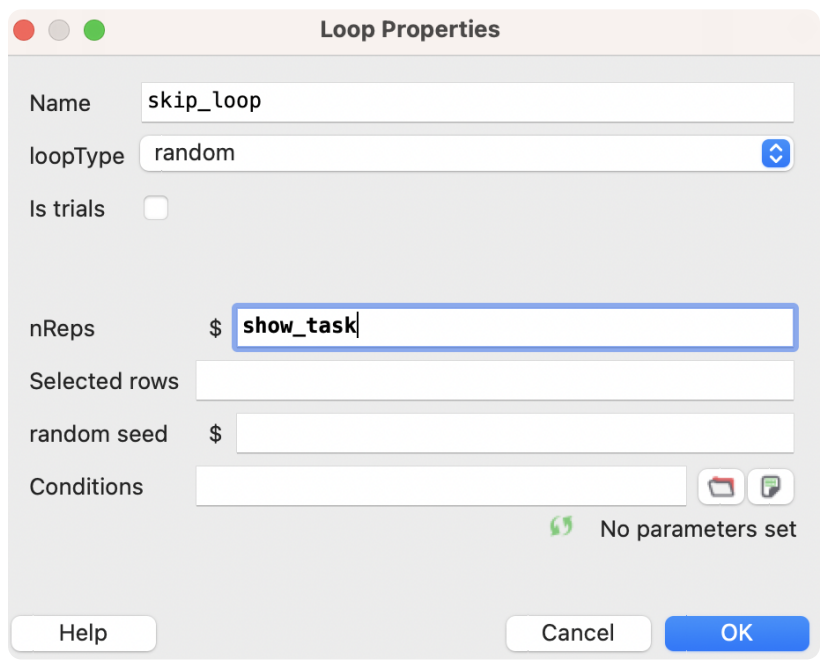


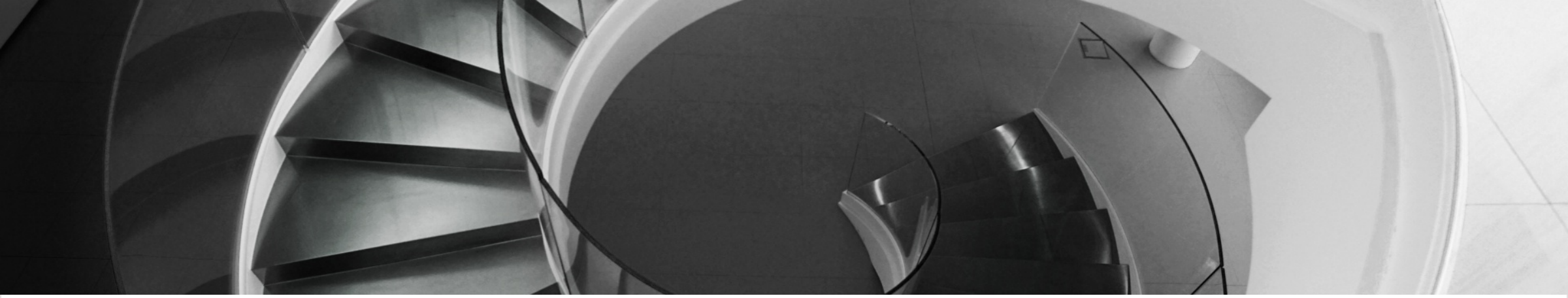
Add a loop around everything you would skip

Your flow should look like this:



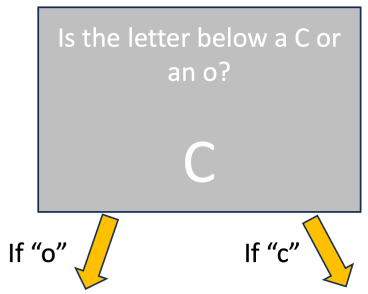
Using `show_task` means if this value is 0, everything in the loop is skipped. If it is 1 then everything in that loop is presented.

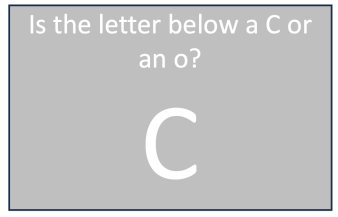


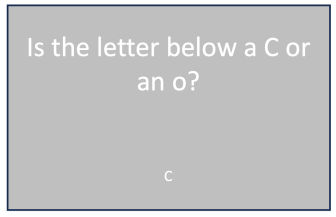


Adaptive designs

"Adaptive designs" means something changing depending on previous responses. One example of this is a psychophysics staircase.



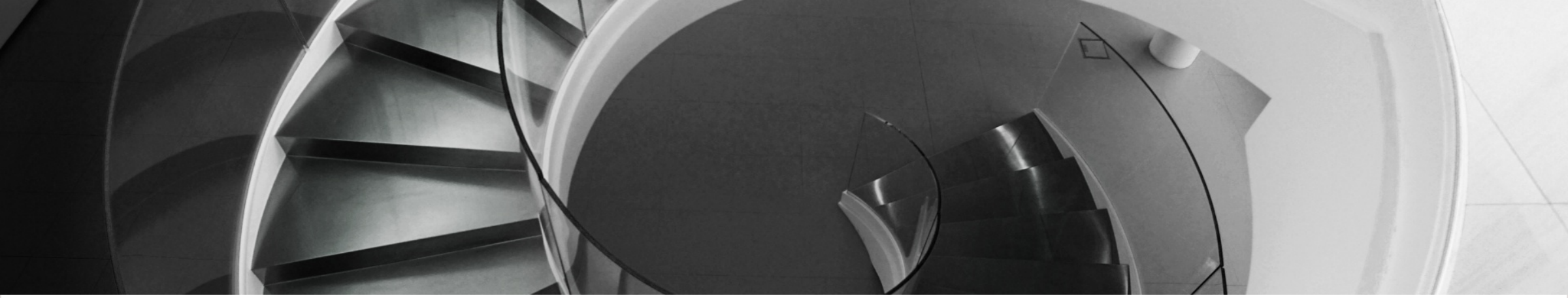




ⓘ We don't always need a staircase to make something adaptive, a lot of the time this can be done in code. This example is just to demonstrate the "staircase" loopType.

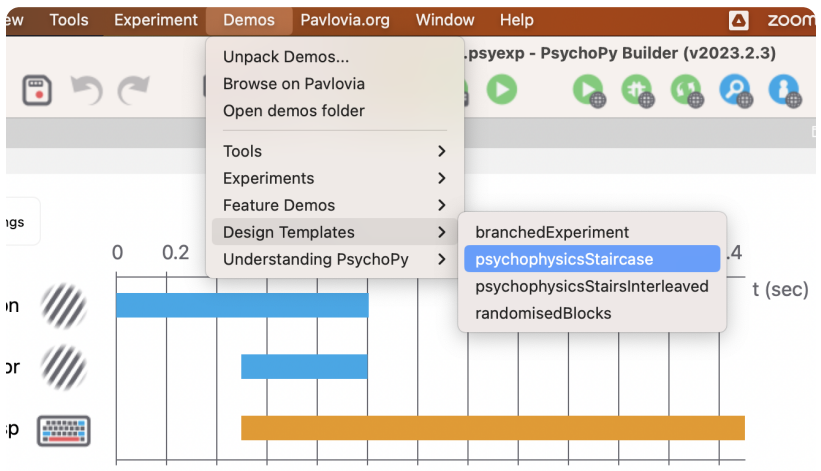
⚠ Staircase loopTypes are not yet supported for online use with pavlovia.org. If you want to use a staircase online this can be done using code. Take a look at **this demo**.

In this example, the letter gets smaller if the participant is incorrect, and larger if correct.

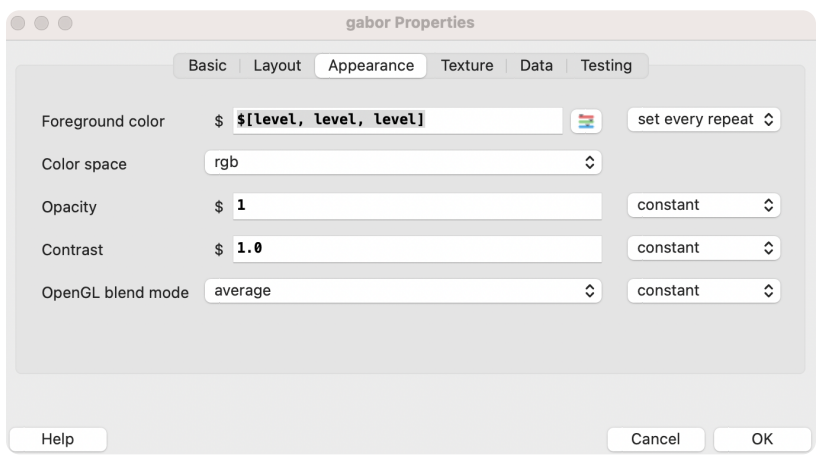


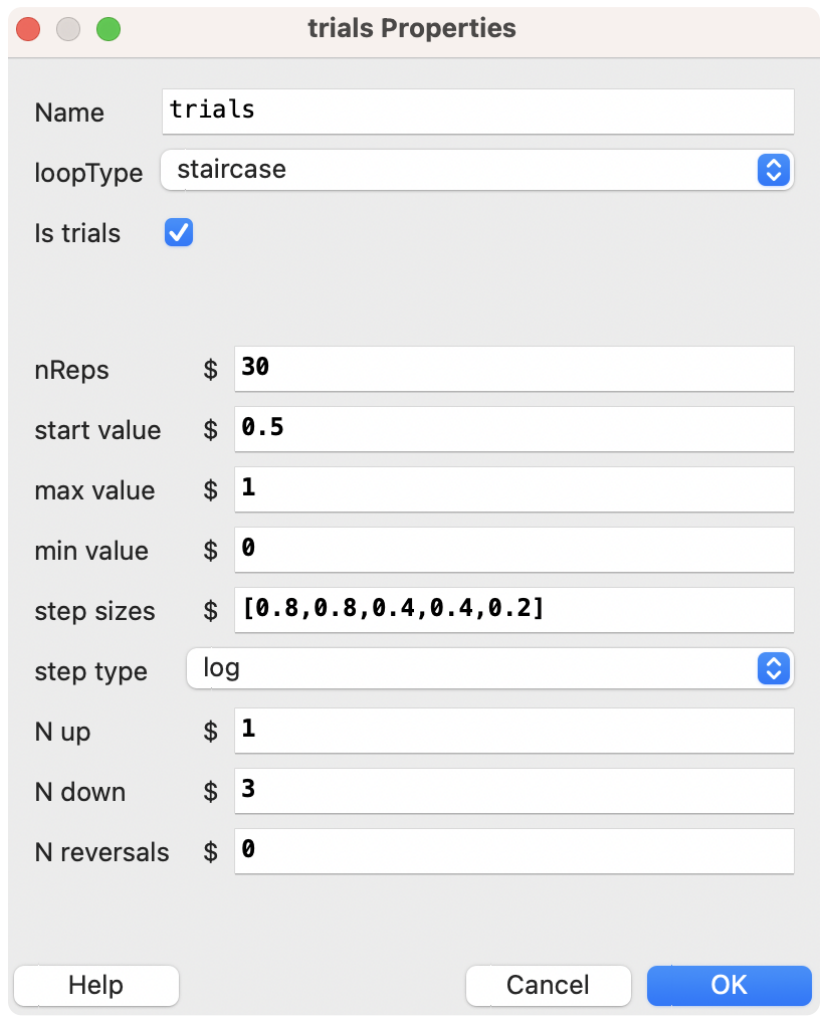
Staircase loopTypes

To look at an example staircase - take a look at the staircase example in the PsychoPy app.

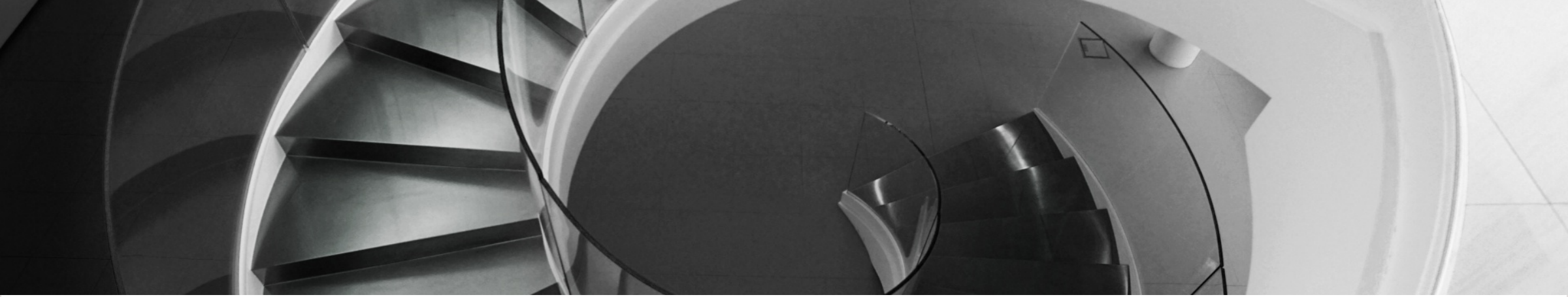


This experiment uses the value `level` adapted by the staircase to update the color of a gabor based on `resp.corr`.





The loop settings indicate the parameters of the psychophysics staircase. `start value` will be the starting value of `level` on every reversal this will update based on the current `step size` (in this case using log steps).



You don't have to use staircase loops to be adaptive!

Remember, if you don't want to use a staircase loop you can just use code components.

That's all for this session!