Experiment 1 code guide:

Line 107 to 146: parameters that have to be set manually depending on the specifics of the recording session

Line 146 to 156: checking the number of inputs of the function

After, the different functions start to be called. From now on, I will describe the most important functions

getSubjectDetails: This function asks for input of the participants in the command window.

initConstantsParameters: This function sets all the constant parameters required for the experiment to work appropriately. Some of the variables in this function can be changed in needed, but some should not be changed

initPsychtooblox: This function initialize PTB, it opens a window from which different information are fetched (refresh rate, size…)

Then, some functions are found in conditional statements, depending on what the parameters are set to above:

turnOffPhotoTrigger: This function turns the photodiode square OFF

Then, in the section: %% initilizing instruments, the different recording devices are initialized: EEG, Eyetracker…

Once the different devices are initialized, different small functions are called: showmessage (showing a message on screen to the participants), savecode (which saves the code of the experiment in the participant folder, to always be able to trace which code was used for whom) and loadTextureFromHD (loads the different stimuli in memory).

[miniBlocks, TriggerMatrix] = createMiniBlocks(): this function creates a matrix in which all the different trials information are found (stimuli, duration, orientation…). It also creates a TriggerMatrix. This matrix contains the successive LTP triggers for each trials.

Then, there are the showInstructionsFunctions which present the instructions on the screen

Then there is a statement to run the practice. It is a run of the experiment that won’t be recorded where participants can see what it will look like and can practice.

Then, the runMiniBlock starts. Begore it starts, a bunch of triggers are being sent, to mark the onset of the experiment itself. But the runMiniBlocks function is where its at: it is the experiment loop, where all the stimuli will be shown and so on.

Then, once the runMiniBlocks function is over, we are done with the experiment, so a message is shown to the participant to say goodbye, end triggers are sent, then we do a safe exit, which closes the window safely, gives back the priority…

I will now describe the big functions in more details:

# initConstantsParameters

As previously mentioned, this function sets all the parameters that will remain constant during the experiment. The experiment is full of constraints, so there are many things that need to be set before we can actually start.

# PARAMETERS THAT MAY BE ALTERED

First, there are the parameters that can be altered. For example, you have the experiment name, that will be added to the saved files. If you want the files to be called differently in the end, then you should change the name here. For the file postfix, if for example you have jpg instead of png, that is where you should let the script know.

But then, we are directly diving into the number of the different stimuli in the different categories. There, it gets a little complicated. In this experiment, we have 4 stimuli categories, 3 Durations, 3 orientations and three task demands. When you do the math, you lend by 4\*3\*3\*3= 108 combinations. We want things to be fully controlled and balanced between the combinations of conditions. So for example, we want to have the same number of faces presented for 0.5 seconds in the right orientation in the task relevant condition as the number of letters in the same conditions. This means that we have to be extremely careful on how the different trials are set. You can go check the pre-registration to know the different counts of stimuli in the different conditions. Remember that we want to avoid randomly sampling things, because that means you don’t have control over what happens. So say you respect what I just said above, but you do so by always having the same stimulus in the same conditions, then you have a problem, too. This is why we have such a complex randomization mechanism.

So at this stage, a few things are set:

NUMBER\_OF\_TOTAL\_TRIALS = 1440 \* trial\_mod: we want to have 1440 trials in total. If you are in ECOG, you have half of that, so 1440\*0.5 for example.

NUMBER\_OF\_NON\_TARGET\_SETS\_PER\_CAT = 16 \* trial\_mod; We have 4 categories of stimuli. For each miniblocks, 16 task relevant and 16 task irrelevant stimuli will be shown (plus the targets). In each stimuli categories, we have 20 stimuli. So if you do the math again, you have 1280 non targets stimuli in total (16\*2\*4\*10 = 1280). Again, since we want to have an equal number of stimuli of the different categories, we have 1280/4 = 320 stimuli per categories. But since we have only 20 stimuli per category, that means we will need to repeat them a certain amount of time: 320/20 = 16. So here, we are setting the number of repetitions of every given stimulus we will have throughout the experiment.

NUMBER\_OF\_NON\_TARGETS\_PER\_CAT\_PER\_MB = 8: Here again, you just have to do the math. We want to have the same number of non targets of each category in a given miniblock. We have 32 non targets and 4 categories: 32/4=8.

Then, the section TARGETS begin. There will then be a section called non-targets and irrelevants. What happens in this sections is that we are setting the different NUMBERS of stimuli in the different combination of conditions. As above mentioned, we have a certain total amount of targets in the different conditions. We want everything to be balanced. So before selecting the actual stimuli (that will happen later on), we want to make sure that the numbers are fully balanced.

# TARGETS

As the comments says in the beginning, we are setting the total number of stimuli shown during the experiment that are targets per category (faces, objects…). To the risk of being repetitive, we want to have everything as nicely balanced as we can. The problem is, we can attain perfection because we have 3 different durations, for a paired number of stimuli. Therefore, there will be one too much of something. If you check the preregistration, we want to have a total of 160 targets (hence the total 1280 non targets). Since we have 2 different kinds of blocks: face objects, letters false fonts, we have 80 of each. So for example, if we consider the letter false fonts case, we have a total of 80 targets throughout the experiment. There should be a balance of targets in the different conditions. We want to have half of them in the center orientation, 25% in the right and 25% in the left orientation. So we will have 20 letters in the center orientation, 10 in the right and 10 in the left orientation, which adds up to 40, same for the false fonts, so 40 + 40 = 80. So far so good, pretty straight forward right? But then, things get a little messy. We have 3 different durations, and we want to have a third of the stimuli in each of them. The thing is that we want this third to be equally distributed within the different orientations. So before, I said we have 20 letters in the straight orientation. We want to have a third of these 20 in the short duration, another third in intermediate and the last third in the long duration. 20/3 doesn’t fall even, which means the best we can do is something like 7 6 7, or 6 7 7 or 7 7 6 (notice that these are the different combinations you can have, this is important: you have three different versions that maximize the balance). Same for the right ones, we have 10, so 10/3 = 3+3+4, or any other combinations of these three numbers (obviously we could have 8 1 1 but that would be way too uneven). So you can see that there will be a slight imbalance. There is not much to be done here, you can fall perfectly and have things balanced. That means that for each participant, you will have these slight imbalance present. But the one thing you can do is having it a bit more evenly distributed across participants. So imagine you have for one participant the case where you have the following: target letters front short = 7, target letters front intermediate = 7, target letters front long = 6. You should try and make sure that we don’t have exactly the same across all participants. For the next participant, it would be nice to have something different. The best thing to do in that case is to try to have the same amount of the different combinations across participants. So a third of what I just described, a third of 6 7 7 and a third of 7 6 7. So we want a third of the participant to have 7 7 6, a third to have 6 7 7 and another third to have 7 6 7 (sorry for the super long explanation, I am always terrible at explaining maths). This is what we are trying to do here. At this point you may have noticed that we are not considering the different categories. We have three different versions of the number of targets in the different duration orientation combinations, but we have 4 different stimuli categories. We decided that within a participant, we will have the same version across categories. So if for a given participant we have the 7 6 7 version, this means that this is the distribution we have for the targets of all different categories. The reason for that is to maximize the within participant consistency.

So what is going on here is that we are setting the counts of the targets in the different category to match what I just described. Then, to have this balance of the different number combination across participants, what we can do is participant 1 have the version 1, participant 2 the version 2, participant 3 the version 3, and then participant the version 1 again. However, this was not the best solution in our case, because down the line there is an other randomization mechanism also having version, 2 in that case. The problem if we have one that goes like 1 2 3 1 2 3 1 2 3 and the other going like 1 2 1 2 1 2 1 2 1 2, that means that things end up intersecting (see table 1).

Table 1: Version balance explanation. As you can wsee, there is a quite a lot of intersect between the third and half random. With the third random however, things are more independent from another. I am not entirely sure things are orthogonal though

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Third random | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 | 1 |
| Half random | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| Third random improve | 1 | 1 | 2 | 2 | 3 | 3 | 1 | 1 | 2 | 2 |

So what we are doing instead is having the version follow the following: 1 1 2 2 3 3. In other words, the two first participants get version 1, the participants 3 and four get version 2 and so on. As you can see in table 1, this renders the other randomization we have relatively independent.

This is exactly what is happening in this bit of the code:

% The nr of trials for each duration comes in three versions:

% 1 1 2 2 3 3 1 1 2 2 3 3 etc

u = mod(int64(subNum),6);

g = round(u/2);

j = mod(g,3);

y = j -3;

v = floor(abs(double(y)/double(3)))\*3;

version\_duration = j + int64(v);

Depending on the subNum, a version will be attributed, such that participant 1 and 2 get version 1, participant 3 and four get version 2 and so on…

Then, we have the NUM\_TARGETS\_PER\_CAT which sets the total number of targets we have per categories (we have 40 in the normal versions). We then have to subdivide it into the different conditions combinations. We will have 20 targets in the center orientation. So then, we have to subdivided it into the duration conditions. So we have the following statement:

v = [7, 6, 7];

NUM\_TARGETS\_CENTER\_DUR\_05\_PER\_CAT = v(version\_duration);

v = [7, 7, 6];

NUM\_TARGETS\_CENTER\_DUR\_1\_PER\_CAT = v(version\_duration);

v = [6, 7, 7];

NUM\_TARGETS\_CENTER\_DUR\_15\_PER\_CAT = v(version\_duration);

What is happening there is that we have the v variable containing the three possible numbers of stimuli in a given combination of conditions. So say, we got from about the version duration 2, we will get in NUM\_TARGETS\_CENTER\_DUR\_05\_PER\_CAT= 6, in NUM\_TARGETS\_CENTER\_DUR\_1\_PER\_CAT=7 and in NUM\_TARGETS\_CENTER\_DUR\_15\_PER\_CAT=7. So we have 6 center short, 7 center intermediate and 7 center long.

Then, the same is done for the left and right orientation. For the ECOG, we have the same rationale, just with lower numbers. That’s it, we now have different numbers for the targets.

# NON-TARGETS and IRRELEVANTS

Then, the same needs to be done for the non-targets and task irrelevant. We have two kinds of blocks: the one with faces and objects targets, and the ones with letters and false fonts targets. So the category that is task relevant is going to be irrelevant in the other kind of blocks and vice versa. But the number of stimuli in both conditions is the same. As I already mentioned, we have 320 stimuli per category in the non target conditions, so we have 160 task relevant and 160 task irrelevant. So again, we will need to divide these guys to get the correct numbers in the combinations of duration and orientation. So again, we will have half in the center orientation, which is why we have the statement: NUM\_NON\_TARGETS\_CENTER\_PER\_CAT\_PER\_MB\_TYPE = (160/n\_mb\_types) \* trial\_mod

This is equal to 80 per task relevance conditions. Which needs to be further divided in third, giving us the following possible versions: 27 27 26 or 26 27 27 or 27 26 27. So then, we are selecting which version the same way we did for the targets.

# IRRELEVANTS

For the task irrelevant, it is exactly the same.

Then, we are done for this part. What happens next is that we set a few additional parameters: number of orientation and categories, number of unique stimuli…

Then, the number of mini block per run is set, as well as the total number of runs. This will be useful down the line for when we have the breaks and so on. We also set the number of tragets per mini block. Even though we setted the number of different targets overall, there should still be variation across miniblocks. If we were to always have 2 target, then at some points, participant could learn it and stop being attentive whenever they already saw the second stimulus.

Then, for the ECOG, we selected 10 stimuli that will be the targets, since the experiment is shorter.

Then, the different durations parameters are being set: jitters, stimuli durations…

Then a few text parameters are set, I don’t think you need to worry about that. Same for the messages, except if you want to add some.

Then, later on, you have the differnet trigger codes..

You have a nested loop, leading to something called the matrixLUT, I don’t think you need to worry about it, it is for the audio triggers of the ECOG.

# Sound parameters

This section sets the different sounds parameters for the ECOG audio triggers.

# PARAMETERS THAT SHOULD NOT BE ALTERED, BUT SHOULD BE USED AS REFERENCE

So here, are the name says, you should NOT change anything, that would lead to problems at some point. For example, if you start messing around with the stimuli coding, without changing the stimulus fetching, you won’t be showing the stimuli you want and so on.

Here, many constant are being sent, some of them are meant to make the code easily readable. TRUE=1, FALSE=0 for example and son. I think this part is relatively straight forward, if you have any question, feel free to ask me directly.

# Practice parameters :

Loading the practice matrix

# PSEUDORANDOMIZATION OF NON-TARGETS FOR EACH MINIBLOCK