ASSIGNMENT 9

In this assignment, you are going to practice displaying stimuli in a grid layout by producing a *visual marking* experiment. During this assignment and in the future, there might be some questions you have to answer in plain English (or Dutch). Put the answers to these questions in a *notepad* item which you place as the first item in your experiment.

Once you are finished with this assignment, please save the file as

`Asgn_8_group_`**`<here your group number>`**`.osexp`

(*for example* `Asgn_8_group_2.osexp`)

before submitting it to canvas.vu.nl. Good luck!

Make sure that your notepad starts with the following information:

```
# <name of student 1>
# <name of student 2>
```

In today´s assignment, we are going to implement an experimental paradigm that is often used to study *visual marking* (Watson & Humphreys, 1997). This is a phenomenon which suggests that the visual system actively prioritizes the selection of new visual information by top-down deprioritization of old visual information. Plainly speaking, when an observer is looking for a target among distractors (for instance a letter E among other letters) and a new set of items appears during this process, the observer will often search through this new set first before continuing with inspection the old items (if the target hasn't been found yet). This is evidenced by the finding that the time it takes to find the target is considerably shorter if it is hidden among the new set of items, than when it is hidden in the old set, or even when the two sets are presented at the same time.

In the classic visual marking paradigm, participants had to search for a yellow E among a varying number of yellow F's and green E's and had to indicate whether this yellow E was *present* (by pressing **z**) or *absent* (by pressing **m**). The search display was presented in two ways (which varied *per block*)

- *Conjunction* – Both green letters and yellow letters appeared at the same time (i.e. were presented simultaneously)
- *Gap* – The green letters are presented first, and 1000 ms later the yellow letters appeared.

The other independent variables were manipulated *within-blocks* (and thus differed *per trial*):

• Target presence	-	*absent* or *present*
• Set size (no. of items on display)	-	8, 16 or 32 (total of green + yellow, each 50%)

It is important to note that in the *gap* presentation condition, the target was always part of the second set that was presented (i.e. all yellow letters) and never part of the first set (all the green letters) if it was present.

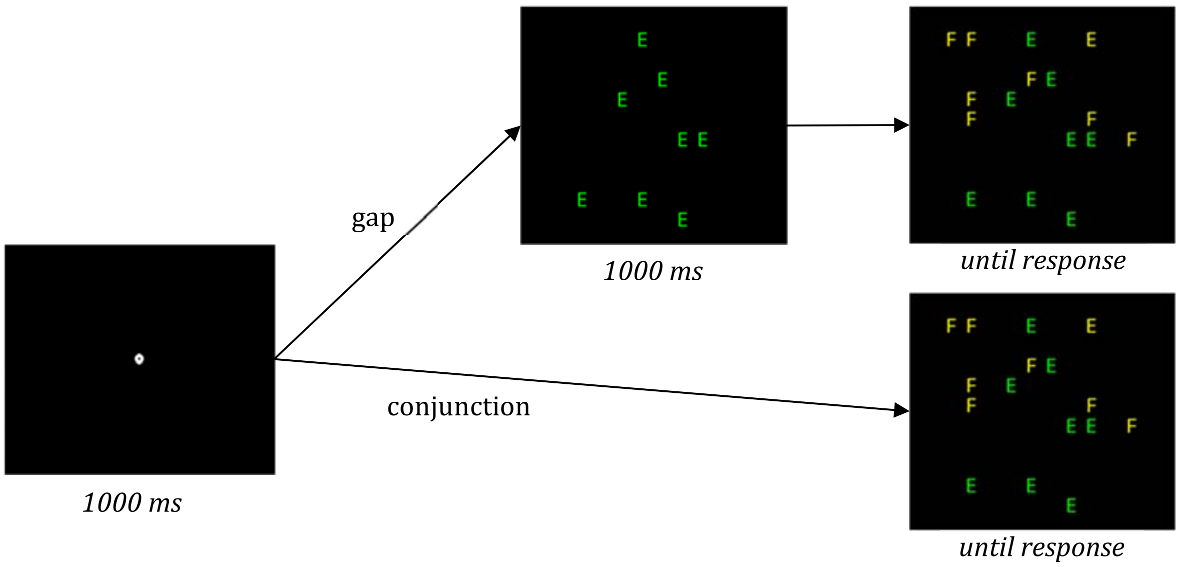

Figure 1: graphical representation of displays in a trial as presented in a gap and conjunction block

Figure 1 depicts an example of the displays as shown in a trial for each of the two presentation types. The results of the experiment are shown in Figure 2. As can be seen, participants were overall much faster in finding the target in the *gap* condition than the conjunction condition. What we can furthermore see, is that search times increase *linearly* with the number of items that are shown in the display in the conjunction condition, while the number of items in the display has less influence on the time it takes to find the target in the gap condition. Participants seem to be able to limit their search only to the newly appeared letters and effectively ignore the other distractor letters that were presented earlier.

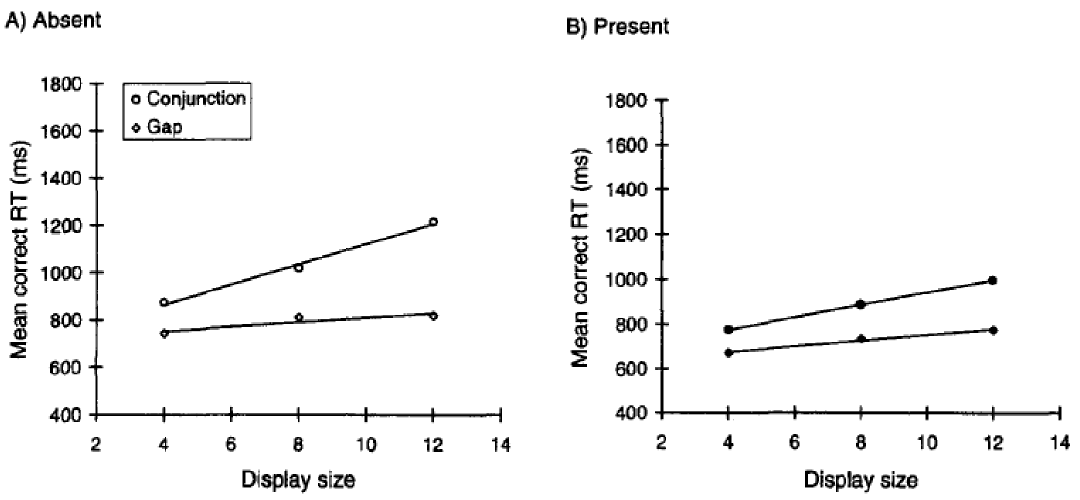

Figure 2. Mean correct reaction times (RTs) as a function of condition and display size for absent trials (Panel A) and present trials (Panel B)

Today you are going to implement the visual marking paradigm. You will use the same letter sets and colors and follow the same structure of the experiment.

General Setup

1) Start by creating an empty OpenSesame experiment. You can choose to use the **Extended template** option in the New Experiment window. This saves you the steps of creating the basic structure of the experiment yourself, as this template already contains practice and experimental structures, which each contain the necessary block and trial loops. In addition, response and logger items have also been added to appropriate locations.
   Rename your experiment to *Visual Marking.* Also, make sure your **Back-end** is set to *psycho.*

2) The next step is to put the independent variables at the appropriate places in the experiment.
   a. Create the *between-block* variable **search_type** in the practice- and experimental loop, both with the values *gap* and *conjunction*.
   b. Create the *within-block* variables **target_presence**, with values *present* and *absent* and **set_size**, with values *8*, *16* and *32* in block_loop. Use the variable wizard for this, as you will want to fully cross these factors (i.e. specify each possible combination). If you successfully did this, block_loop should now have 6 rows (or cycles).
   c. Create the **correct_response** variable in block_loop and set it to 'm' in the rows in which target_presence is "absent" and to 'z' in rows in which it is "present".

3) Change the slides for the introduction, end of the practice and end of the experiment. In addition, create a slide that you put at the top of *block_sequence* in which you announce to participant which block type they are going to perform. You can use the value of variable *search_type* you created at 2a to display in your message.

Setting up the trial sequence

4) Place an inline_script item in the trial_sequence and rename the script item to *stimulus_display.* Create a canvas called fixCanvas and draw a fixation dot. Later you will present the fixCanvas for 1000 ms (in the run phase).

5) Make sure there are a *keyboard_response* and a *logger* item in your trial_sequence. Change the 'allowed_response' field in *keyboard_response* to 'z;m'

Setting up the prepare phase of the inline script

6) Copy the code that calculates the positions in a grid layout from the slides. Use this code to make a function called makeGrid(). The function should have the following properties:
   a. It should take <u>three</u> obligatory and <u>one</u> optional argument(s)
      i. *grid_dimension* – a number indicating a single dimension of the grid. (e.g. if grid_dimension = 5 then size of the grid will become 5x5)
      ii. *x* – the x coordinate around which the grid is drawn
      iii. *y* – the y coordinate around which the grid is drawn
      iv. *linelength* – the space between each grid position (*optional*: default value: 50)
   b. It should return a list with tuples that each represent a grid position (x,y).

7) Perform the other basic steps to create a trial sequence:

- Import any modules you might require
- Retrieve the required variables from the loops
- Create two canvases to draw your stimuli on (and store it in the variables *canvas1*, and *canvas2*)

8) Use your function from 6) to create a grid of **10x10** centered at the center of the screen. Store the list of positions which your function returns in a variable called *gridPositions*. For the *linelength* you can use the default value. Check whether your function works!

Drawing the stimuli to the canvas

Now we did setup all the basic elements of the experiment, it is time to draw the stimuli on canvas1 and canvas2. One easy way to draw the stimuli is shown in the pseudo code below:

```
Randomize the gridpositions
For i = 0 : ½ set size
    Draw the green E's on gridPositions[i] to canvas1

Copy canvas1 to canvas2
For I = ½ set size : set size
    Draw the yellow F's on gridPositions[i] to canvas2
If the target is present then draw the yellow E on gridPositions[-1] to canvas 2
```

9) We want to draw letters at randomly determined positions on each trial. The easiest way to realize this is to randomize the *gridPositions*. To do that, use the *shuffle* function from the random module. Check whether this worked properly!

10) Create a *for*-loop that will draw green E's at random grid positions on *canvas1*. Note that the *gridPositions* are already shuffled! The number of green E's should be half of the value of *setsize* as specified in *block_loop* (as the other half will be yellow F's that you will draw next).

11) Copy *canvas1* to *canvas2* so that you don't have to draw the green E's again. Then, create a *for*-loop that will draw yellow F's at random grid positions on *canvas2*. The number of yellow F's should be half of the value of *setsize* as specified in *block_loop*

12) The drawing part is almost finished. You just need to add the target letter at a random position on *canvas2.* However, you only do that if the value of *target_presence* is 'present' or don't do this if its value is 'absent'. A possible position for the target is the last position in the *gridPositions*. You can use -1 to refer to the last element in the *gridPositions*.

Showing the canvas(es) on screen in the run phase

13) The final step is to show the canvases on the screen in the run phase. This has to be done differently as well depending on the value of *presentation* in experimental_loop. If you are in a *conjunction* block, you just want to show the canvas with all the letters. This can be easily done by only showing the final canvas, *canvas2.*

14) In a *gap* block, you will still need to show *canvas2* after you have shown *canvas1* for a second. Write the code that does so. (**Hint:** you always want to show *canvas1* and thus you don't need to put this inside an if-statement that checks for the value of *presentation*. *Only* when you need to show *canvas2* you will need to sleep for 1000ms and then show it).

Final touch

15) Make sure there are 2 practice blocks in total, one for the gap and one for the conjunction condition. The experimental_loop should have these two 2 cycles repeated 3 times. In block_loop you should have 6 cycles repeated 10 times. Why is it a bad idea to have more cycles than you have variable combinations (and thus have empty fields) and why is it thus better to use the *Repeat* field to determine the number of trials in a block?