**Candiate Number: BPWF6**

**A Visual Search Task Generator**

**Experiment Overview**

This experiment is a visual search task. A target stimulus and distractor stimuli are displayed in a screen. The target stimulus is randomly either presented or not presented and the participant indicates her response via button presses. This paradigm is widely used to study attention and perception, as the difficulty of the task can be manipulated to vary the amount of cognitive effort required to search for the target. The results of visual search experiments can provide insight into how visual information is processed, including how attention is directed, how information is prioritized, and how memory is used during search.

As indicated above, the cognitive effort of visual search can be manipulated. This commonly happens based on the following characteristics:

* **Set size**: This refers to the number of items in the display. A larger set size typically makes the task more difficult, as the search time increases and the accuracy decreases.
* **Feature similarity**: This refers to the similarity of the distractors to the target. If the distractors are similar to the target, the task becomes more difficult, as it is harder to distinguish the target from the distractors.
* **Target uniqueness**: This refers to the number of items in the display that have the target feature. If the target is unique in the display (i.e., it is the only item with a certain feature), the task is typically easier, as the target can be found more quickly.

It is furthermore important that the target location is unpredictable (random). This is because if the location of the target is unpredictable, the task becomes more difficult, as the search must be performed more thoroughly to ensure that the target is found.

The present program enables the experimenter to manipulate the above parameters and more and can thus be seen as a *visual search experiment generator* that flexibly meets most needs of the typical visual search researcher.

**Description of procedure**

In this experiment, there are two conditions in a between-subject design. There are two scenarios: a) creating a new experiment and b) accessing an already existing experiment.

1. The two conditions are defined by the experimenter via a graphical user interface in which she can set the desired parameters. She also defines the number of trials per participant (this remains the same for the whole experiment, i.e., for every participant in this experiment – but, of course, can be changed across experiments). The experimenter is asked to enter the experiment name, which has to be different from former experiment names in case she is creating a new experiment. She is then also asked to enter a participant ID. Then the section for the actual participant starts, where the participant is asked to provide consent and demographic data. The Pyqt interface is then closed and the Psychopy interface is started, which first contains the instructions for the experiment, detailing characteristics of the experiment parameters and task stimuli. The participant has to press any button to resume and start the experiment. The participant can respond whether a target stimulus was present or not using the keyboard keys ‘n’ and ‘y’. Note that non-English keyboards might have to use the ‘z’ button instead of the ‘y’ button, as Psychopy assumes an English keyboard.
2. If the experimenter wants to continue with an already existing experiment, she is asked to enter the exact name of that experiment. Experiments are saved as folders in the working directory. So, if you forgot a given experiment name, you can have a look there. The experiment parameters section is then skipped (in the background, the program accesses the experiment parameters csv that is saved for every participant and uses this information to generate the experiment) and the experimenter is asked to provide the participants ID. In case the ID already exists in the csv that saves the measurement data, the program assumes that an interrupted session should be resumed, and assigns the experimental condition of the first session to this second session. The remaining procedure is the same as in a).

**Experimenter’s manual**

The experimenter just has to run the main file. She is then guided through the process via a graphical user interface and hence the working of the visual search task generator should be self-explanatory. Importantly, you can create different experiments with this generator and access previous ones. The generator allows for a high degree of modularity to answer a variety of questions within the visual search paradigm. The three classic parameters mentioned in section one can be manipulated via choosing the geometric figure serving as target or distractor stimulus, the respective colours, whether there should be two distinct types of distractor stimuli or not, whether the stimuli should move or not and how many distractor stimuli should be displayed.

**Program highlights**

The code is highly modular, allowing for manipulation of several parameters, resulting in different experiments that can be created with this code. It combines the libraries PyQt5 and Psychopy. PyQt5 is used to collect information and Psychopy is used to display the experiment. I think I combined these libraries quite parsimoniously. The PyQt5 window is created, executed and closed within a class, which allows to run the whole PyQt interface procedure via one command in the main file, calling the class. Coding the PyQt interface in a class also has the benefit that there is no confusion between the two windows I had to create (one for Psychopy and one for PyQt5).

Furthermore, this code covers a lot of functionality and I think the structure of the files makes the code readable and helps understanding it.

I also think I found a good solution to calculate unique random positions for the stimuli, avoiding a while loop and also avoiding calculating each possible position and drawing from it.

I would also like to stress how the interplay between object-oriented PyQt5 class and the assignCondition function allows you to create new experiments or access existing ones, even continuing with a given participant.

One last thing I would like to mention is that all functionalities come together in the main file, making it only necessary to run this file to execute all interfaces and the experiment flow.