Documentatie Experimentele Psychologie

Annabel Couzijn, Brandon Chin-A-Lien, Daniëlla Zinn

Design

Short summary of the research question and task

The goal of our experiment is to test whether stimuli previously associated with food-related images capture attention faster than other stimuli. Previous research shows that reward-associated stimuli capture attention faster (Anderson et al., 2013), both by current and prior associations (Kristjansson et al., 2010). Attention is captured persistently more by stimuli that previously rewarded the participant (Hickey et al., 2010; Hickey et al., 2011). Previous research has indicated that appetizing pictures of foods have a reward value for human participants, especially when the participants were hungry (Huffman et al., 2001). For visual attention, higher levels of hunger significantly predicted increased alertness for sweets, candy and fatty foods (Gearhardt et al., 2012).

Our Food-associated Visual Search Task (FaVST) consists of three parts: a learning trial in which neutral images or images of food are associated with different objects (i.e. symbol and color combinations), the Visual Search Task (VST) and a part in which the participant states their preference towards the different symbols and assesses their hunger.

The VST (Treisman & Gelade, 1980) requires participants to search for a ‘target’ in a field of ‘distractors’ that look similar in some aspects to the target. Originally, two different conditions were used: conjunction and disjunction search. In the conjunction condition, the target differs in color or symbol. It takes more time to find this target compared to the disjunction condition, in which the target ‘pops out’ because it differs in more dimensions from the distractors.

We are using three symbols (O, X and Δ) instead of two (T and N) and three colors (green, blue and red) instead of two (pink and green), because we use nine different images for the associations and each image needs an object to be associated with. Only the conjunction search is used, because if the target is already quickly found because of a pop-out effect, the food or reward associations would make no difference in the reaction time.

To achieve the goal of the experiment, we need to answer on the following questions:

- Will targets associated with sweets / fast food be found more quickly than targets associated with neutral foods, landscapes or targets without any association?
- Are the stimuli successfully associated with the food representations?
- Is the difference in reaction time greater when the participants are hungry relative to non-hungry participants?

List of requirements

- Main script
    - Participants must be assigned to experimental group or control group
    - Participants must be able to fill in participant number, whether their eyesight is impaired, whether they have an eating disorder or ADHD, age, their food satisation, gender and sex.
    - Sensitive information must be asked after the test.
    - Only the experimental group does the training phase. The control group completely skips this phase.
- Training phase (linking images to symbols)
    - An encoding block must be followed by a retrieval block.
    - All 9 symbols and their associated pictures (chocolate, pizza, bread, landscape 1, landscape 2, landscape 3, or nothing (x3)) must be presented in the encoding block.
    - Only show the next association in the encoding block when the participant presses F or J.
    - In the retrieval block, one of the multiple choice options must be a food picture and the other must be a landscape picture.
    - Only 6 associations (the food and landscape associated objects) are asked in the retrieval block.
    - The user must press 1 or 2 to answer the multiple choice question.
    - The training phase can only stop after two consecutive retrieval blocks in which all 6 multiple choice questions were answered correctly. Else, the encoding block is repeated, followed by another retrieval block.
    - The objects are presented in a different order in both blocks.
- Visual search task
    - Instruction screen must be shown.
    - Symbols must be shown on a screen in random posiotions in sets of 24.
    - Each round, the target color and shape must be chosen randomly.
    - Distractors must vary from the target in color, shape or both.
    - Reaction time must be measured and stored.
    - Code must be able to conduct multiple trials.
    - Pause before starting and after every 50 trials.
    - Variables must be saved in a struct.
    - A condition (i.e. a specific object as target) cannot be shown after 20 hit trials have been conducted with this condition.
    - Participants must answer 15 practice trials correctly.
    - 20% of the trials must not contain a target.
    - VST struct must store per trial:
        - participant number
        - trial number
        - reaction time
        - accuracy of the answer (correct or incorrect)
        - the presence of the target
        - what key was pressed
        - what objects were shown in every trial (target & distractors)

        - the condition of the target (food associated, neutral associated, no association (control))
        - the specific image the target was associated with (chocolate, pizza, bread etc.)
- Afterward testing the food associations
    - The three conditions must be compared (food association versus neutral association, neutral association versus control, control versus food association)
    - The placement (left or right) and comparison of specific symbol (blue circle vs green triangle) needs to be done randomly to prevent biases.
    - Of each comparison, it must be stored which condition the participant preferred

Global overview of the visual search task design

The script ‘main’ mostly consists of code relevant for the initialization and storage of data. We saw no use in casting these processes to smaller functions. Primarily, because compartmentalizing the data storage would drastically reduce comprehensibility of the overall code. Additionally main was populated by functions necessary for performing the experiment, alongside simpler functions for textboxes. The functions responsible for experimentation were association_task, VST and test_association.

The association_task function was responsible for establishing the associations between images and symbols. The function itself mainly regulates how many times encoding blocks and retrieval blocks are run. Making sure to end when all symbols can be retrieved perfectly two times in a row. Additionally, a large part of the code consists of randomizing symbol orderm, which is necessary for proper association establishment. The actual visuals seen by the participants is handled by establish_assocation. This function operates in two modes, one for encoding and the other for retrieval. The function only consists of input checks and displaying of symbols and/or images. Tracking user input and data management is done by assocation_task.

VST takes input from the symbols generated in main and performs the Treisman visual search exclusively in conjunctive form. Similar to association_task, the visuals themselves are handled by a subfunction, doe_experiment. doe_experiment itself calls treisman_conj which displays the symbols in a random order. doe_experiments’s other role is to store the information of the visual search per trial. Because we iterate through a trial many times, we decided it would be much more useful to condense all of this into one function, instead of having it all in VST.

The last major function, test_association, simply logs the user's symbol preference. Only consisting of two_symbols and keypress logging.

GUI’s are used in between tasks and for user input.

Flowcharts

Entire experiment / main

The main script starts with assigning the participant to the experimental group or the control group. After that, the images are initialized and the objects are randomly associated with the images. These associations are stored in a struct. This is followed by asking the participant for their participant number and whether they have an uncorrected visual impairment. If the answer is yes, they are redirected to a screen that tells them they cannot participate but they are thanked for their interest. If the answer is no, the experiment continues. If the participant is in the experimental group, they start with the training phase (association_task function), but if they are in the control group, they skip this phase. Then, the test phase begins and the VST function is run. This starts with practice trials, of which 15 trials have to be correct to continue with the experiment. Data about each trial (trial number, reaction time, target type, accuracy, keypress) is stored in a data struct. After the test phase, the preference test takes place (test_association) and the hunger slides appears so the participant can assess their hunger. Then, the demographical questions appear (gender, sex, age, ADHD, eating disorders). The answers from the preference test, hunger slider and demographical questions are stored. For every person, a struct called ‘data’ is stored. This contains their ppn, a struct with their VST data, a table with their preferences for which condition-associated stimuli they prefer, a struct with their symbol-image associations (including the condition, like food associated, neutral associated and control), a struct with their answers to the demographical questions and a variable determining whether they belong to the experimental group or the control group.The experiment end with a screen that thanks the participant for their participation.

The following flow charts were made to illustrate the main script:

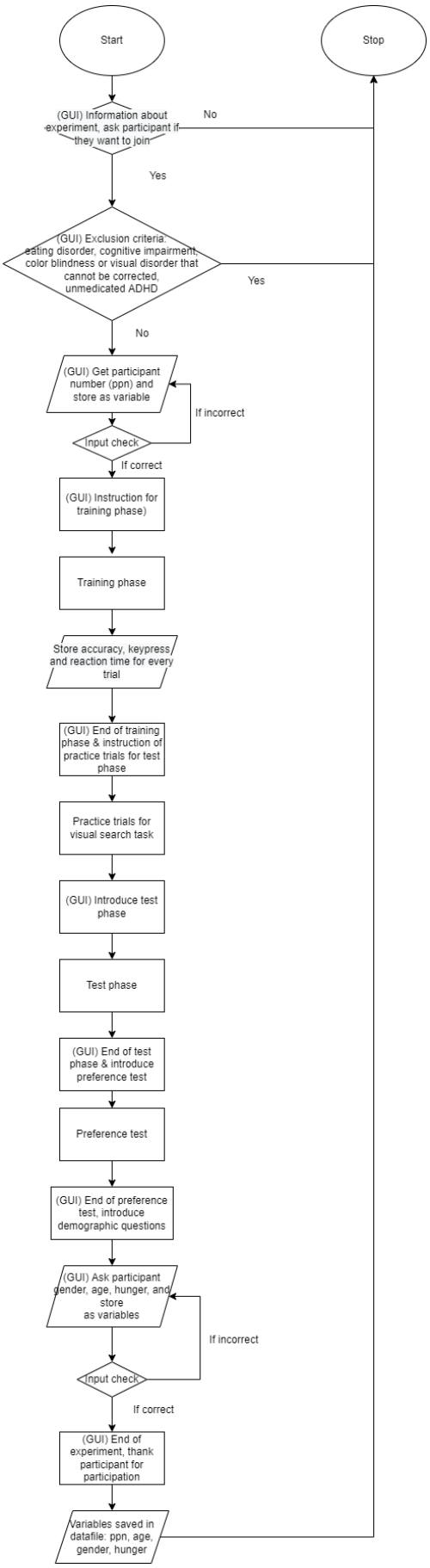

Figure 1. First version of the main script.

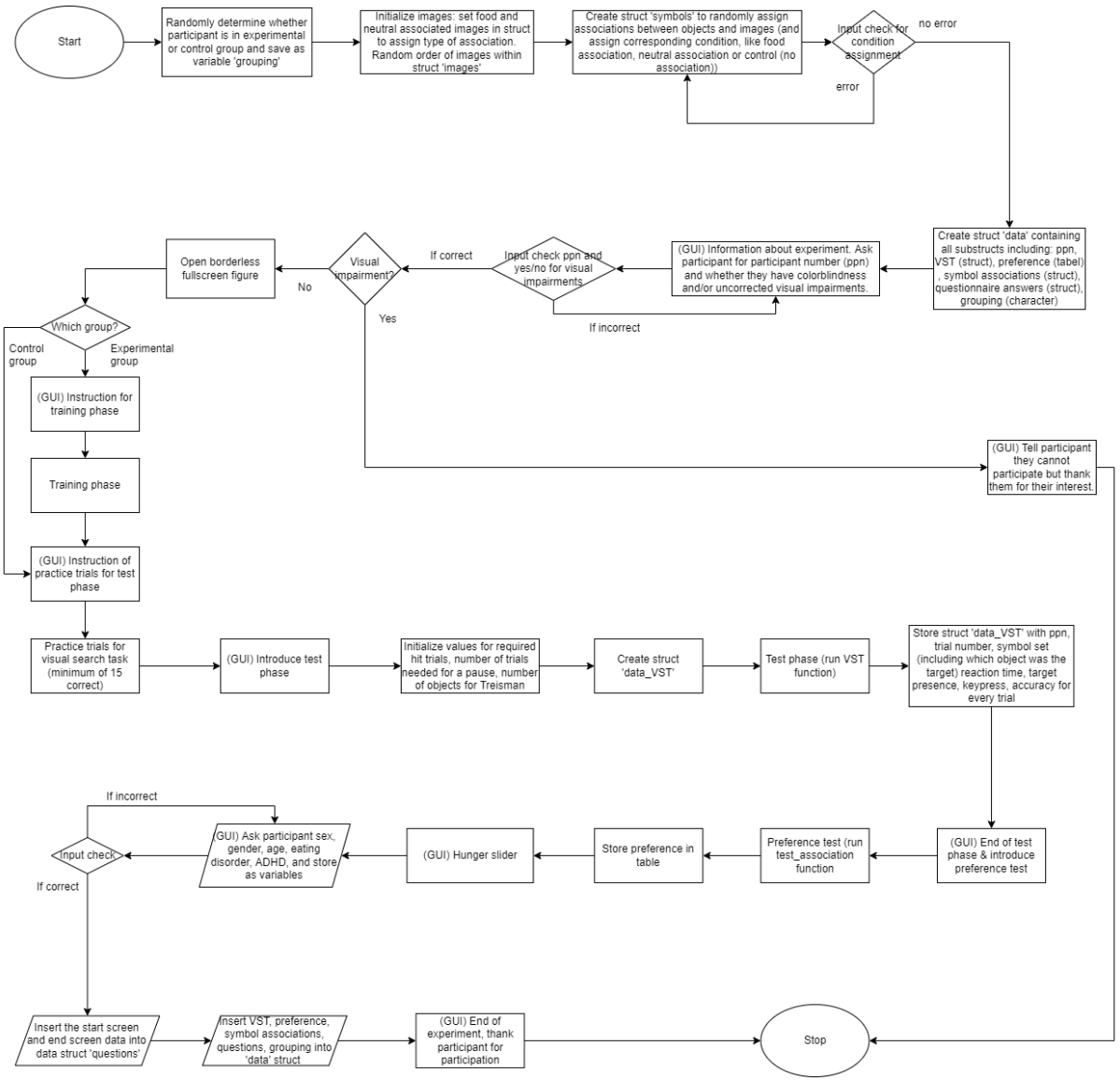

Figure 2. Final version of the main script.

Training phase / association_task

The training phase starts with randomizing the order in which the symbols are presented. Then, the encoding block starts, in which 9 associations are shown one after another (establish_association function with ‘learning’ as condition input). The next association is shown when the participant presses either J or F (it does not matter which one). After this encoding block, the retrieval block starts. In this block, 6 objects (the ones associated with either food or landscape images) are shown one after another on the left side of the screen (establish_association function with ‘testing’ as condition input). On the right side, two images are shown: a food image and a landscape image. The participant is asked which of these images is associated with the object. Depending on their answer, they receive feedback telling them if their answer was correct or incorrect. The objects are presented in a different order in both blocks. After the retrieval block, the encoding block is repeated,

followed by another retrieval block. The training phase stops after two consecutive retrieval blocks in which all 6 multiple choice questions were answered correctly.
The following flow charts were made to illustrate the training phase:

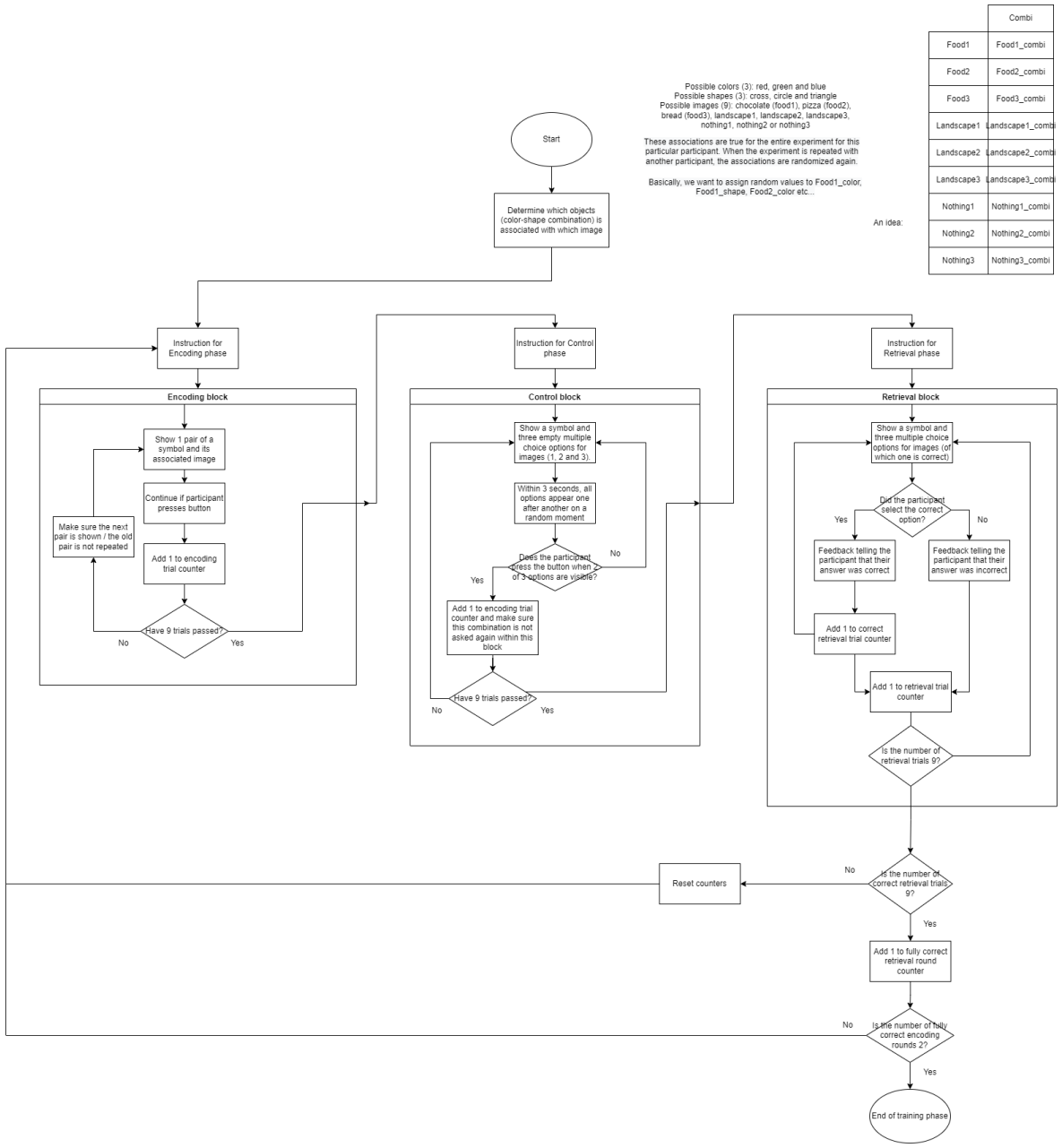

Figure 3. First version of the training phase.

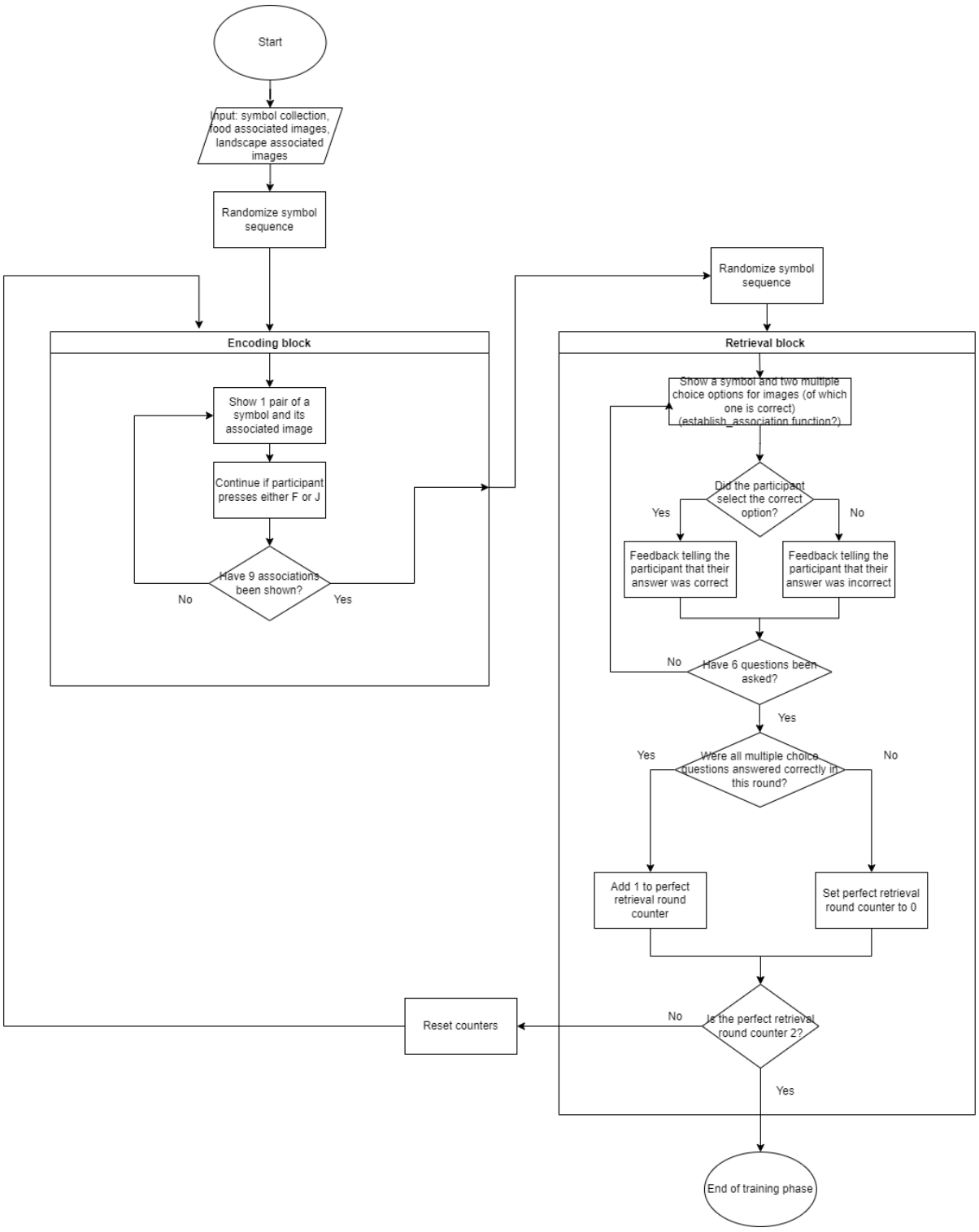

Figure 4. Final version of the training phase.

Training visuals / establish_association

The establish_association function plots the encoding en retrieval visuals.. If the input for the condition is ‘learning’, an objects is shown on the left side of the screen, and on the right ide of the screen, the corresponding image depending on the associations made in the main script for this participant. A’+’ is plotted in between these images. If the input is ‘testing’, an

objects is shown on the left side of the screen, and on the right side, two multiple choice options are shown, one of which is a food image and one is a landscape image. One of these options is associated with the objects and is therefore the correct option.

The following flow chart was made to illustrate establish_association:

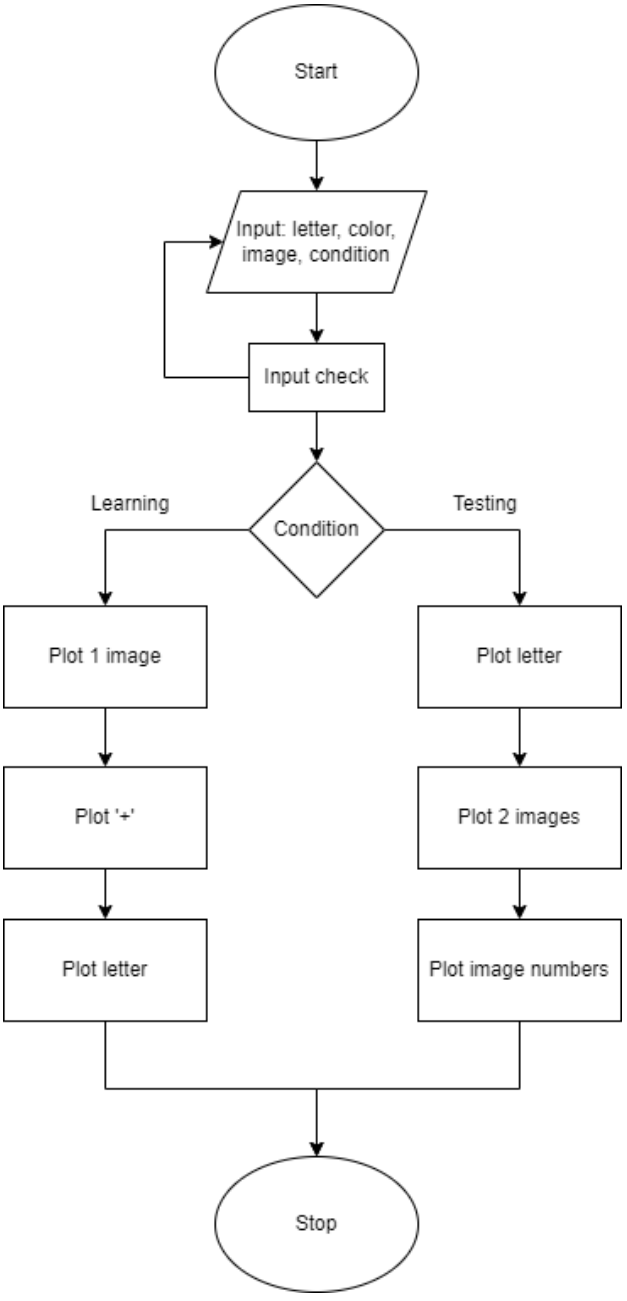

Figure 5. Flowchart of the establish_assocation function.

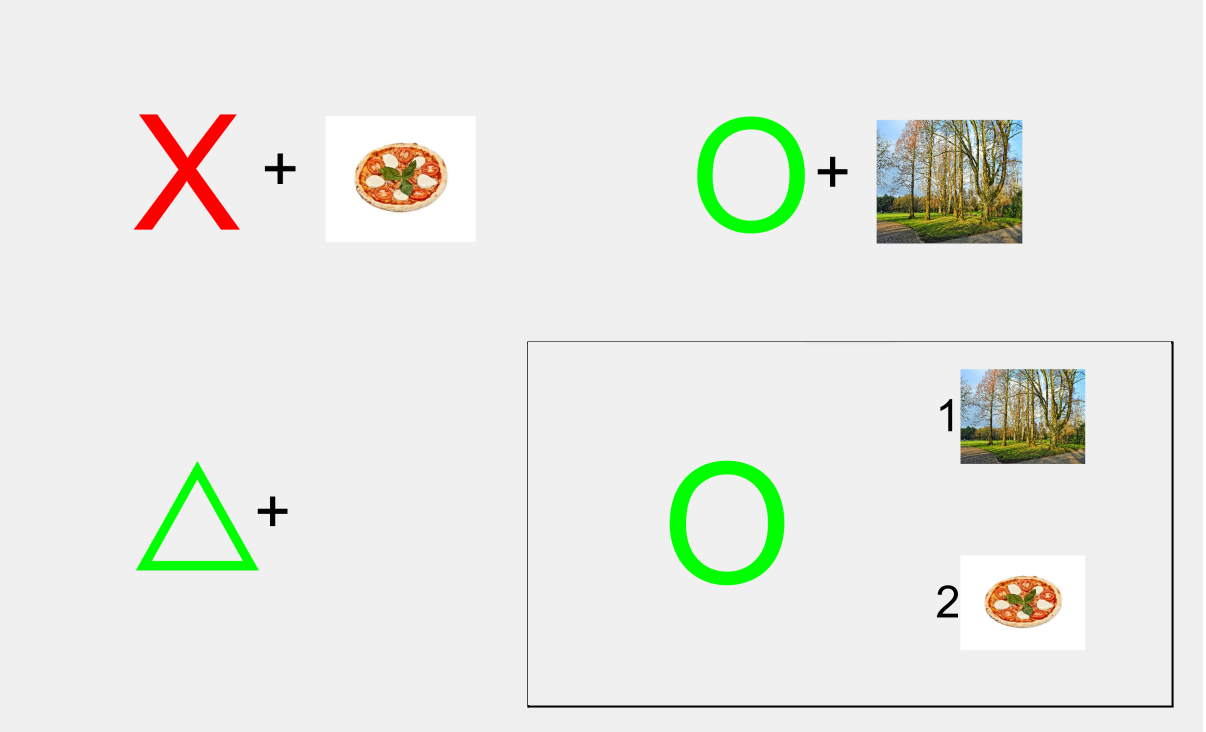

Figure 6. Examples of the visuals that establish_assocations can create.

Testing phase / VST

The VST function starts by creating a struct in which the ppn, trial number, symbol set, reaction time, target presence, keypress and accuracy for each trial will be put. Then, a matrix is formed to count the hit trials for each possible target. The practice trails are run before the real experiment: for each practice trial, a new set (of 1 target and 3 distractors) is chosen with the make_symbol_set function and consequently, the VST trial is run with doe_experiment. This continues until 15 correct practice trials have been run. After this, the real trials begin. For each trial, the target and distractor symbols are chosen randomly, the corresponding cell is chosen from the matric and the target presence is determined (80% for present target). If the trial was a hit trial (the keypress is J and the response was accurate), 1 is added to the value of the chosen cell. When a cell hits a value of 20, the corresponding target index is removed so this target will not be presented again. Every 50 trials, a pause screen appears that stops when the user pressed any key. The experiment stops when each target has had 20 hit trials (each cell has a value of 20). Finally, the data about the trials is stored in the data struct.

The following flow charts were made to illustrate the test phase:

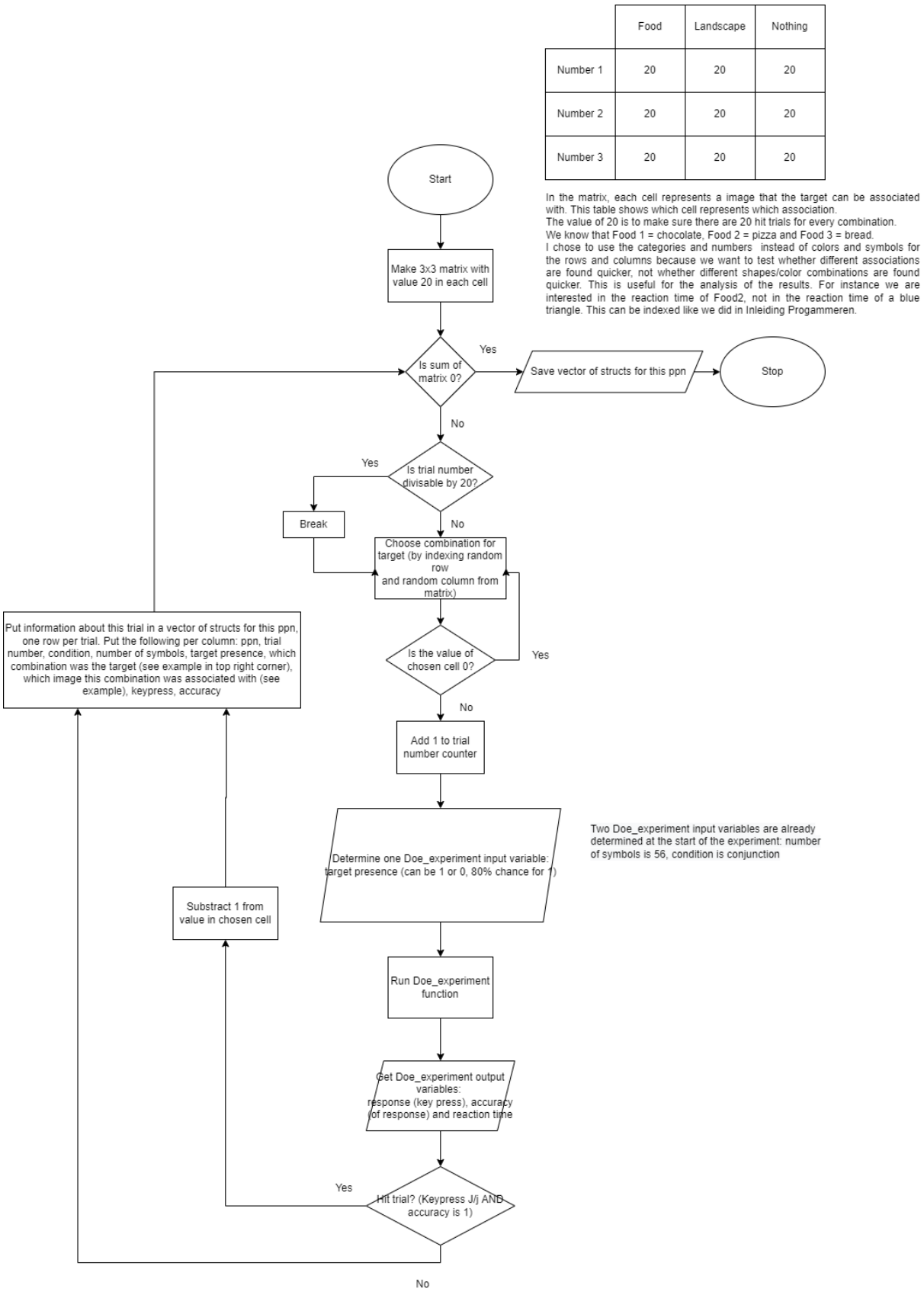

Figure 7. *First version of the test phase.*

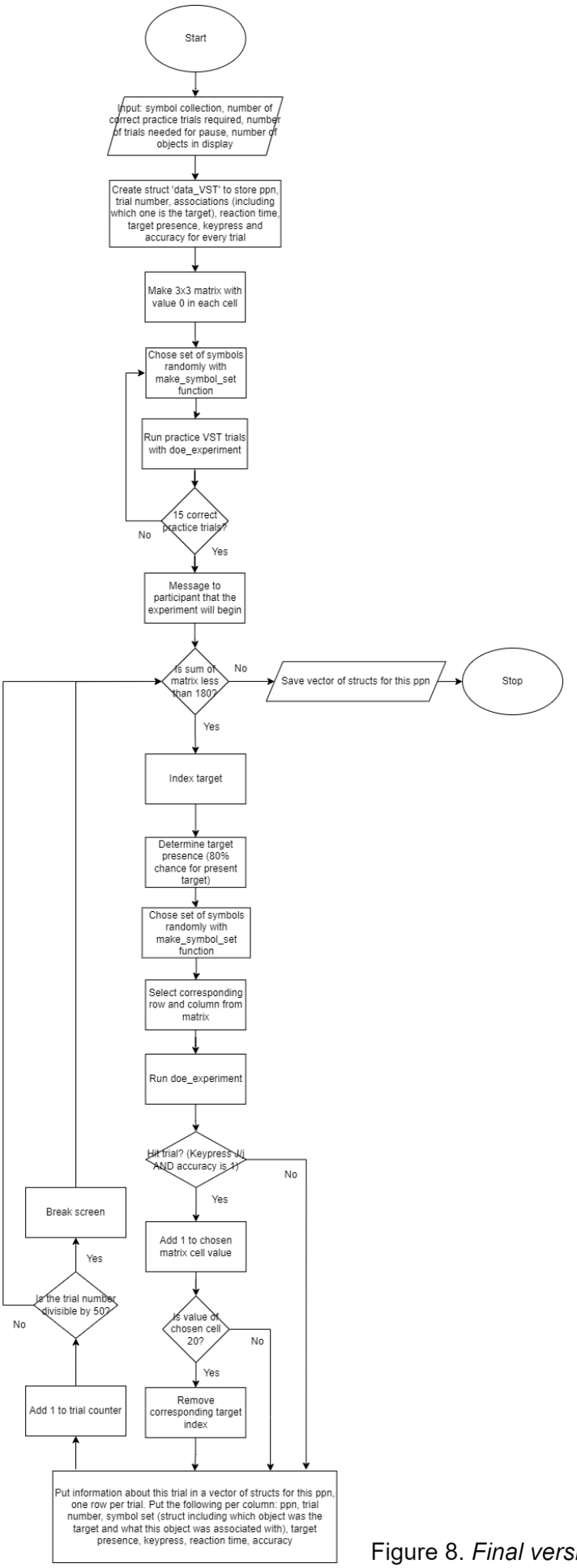

Figure 8. *Final version of the test phase.*

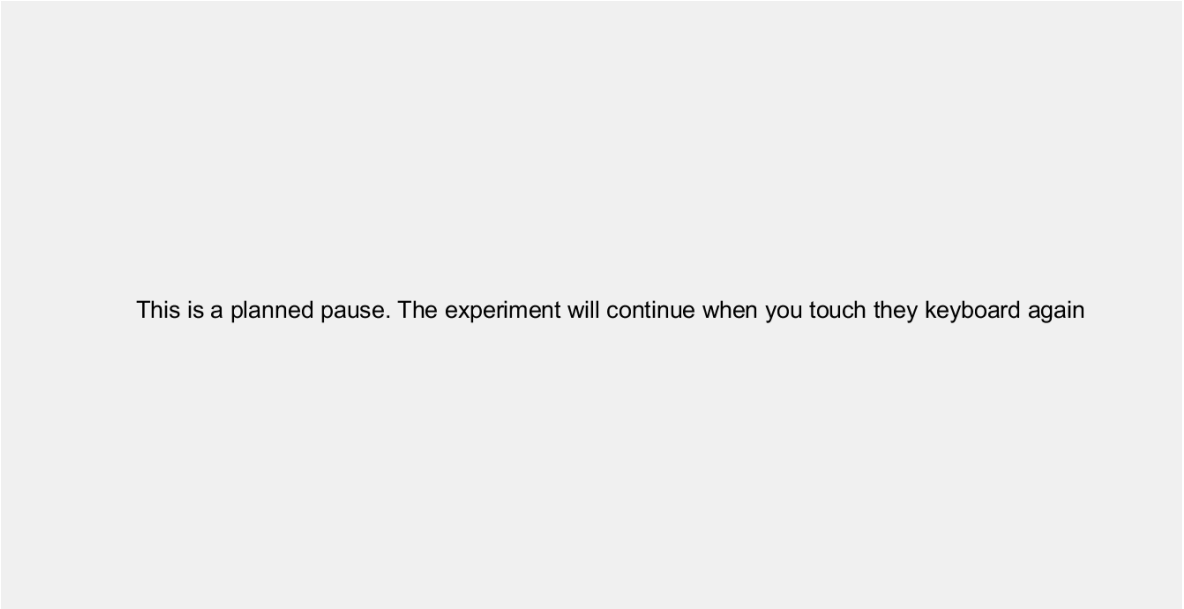

Figure 9. *Example of the screen that appears after 50 trials, to give the user a break.*

Make set of 4 objects / make_symbol_set

This function makes a set of 4 objects to be plotted into a figure. The first objects is the target.

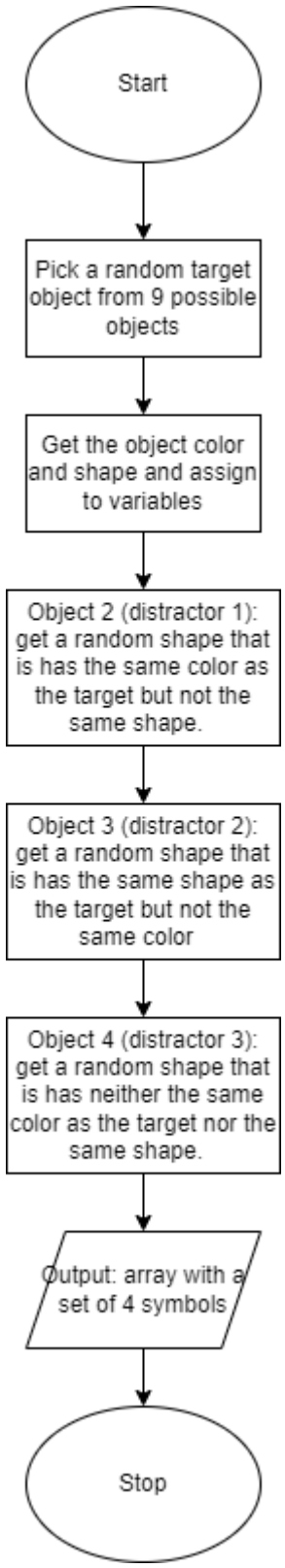

Figure 10. *Flowchart of the function make_symbol_set*

Running a trial / doe_experiment

The doe_experiment runs the treisman_conj function to make a single VST display. Then it measures the reaction time, keypress, and accuracy. After each trial, it puts the information about the trial in the struct.

The following flow charts were made to illustrate the test phase:

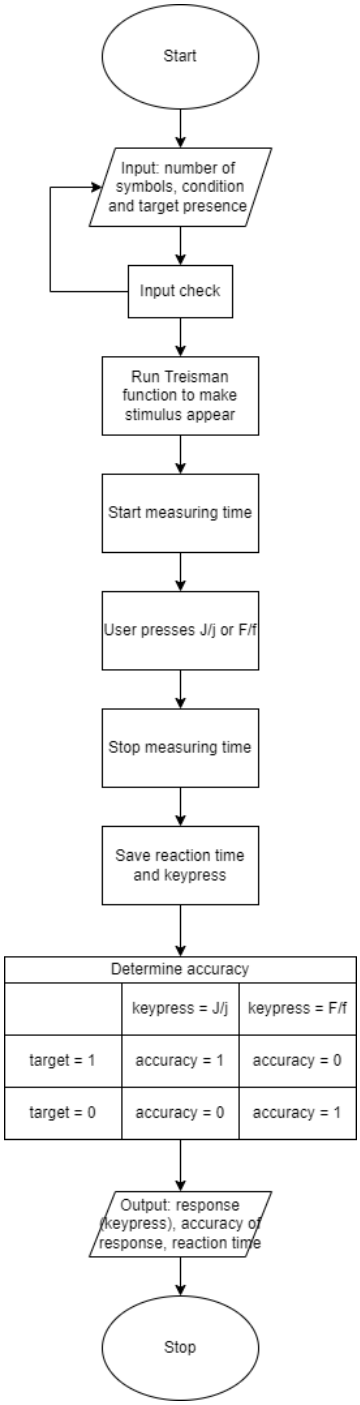

Figure 11. *First version of doe_experiment.*

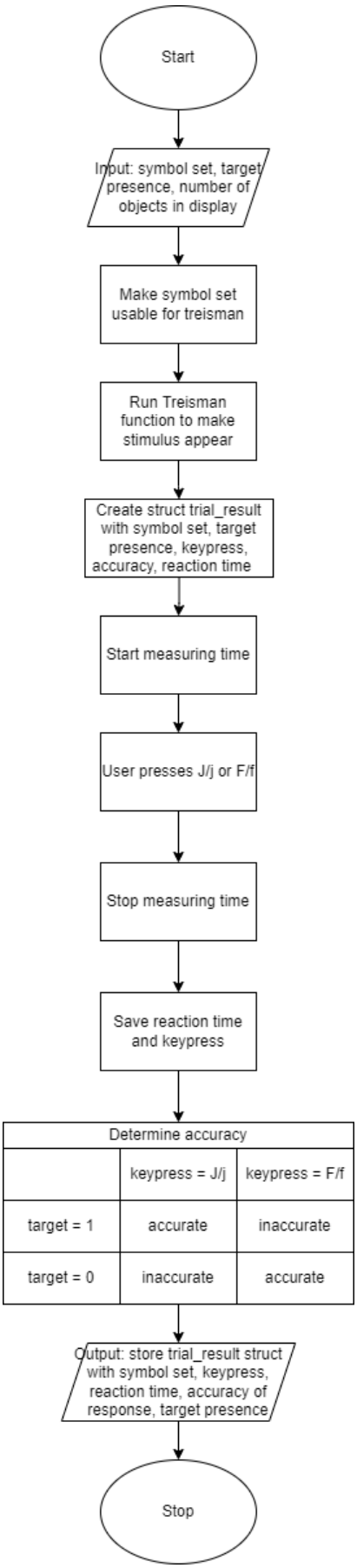

Figure 12. *Final version of doe_experiment.*

Visual search display / treisman_conj

The treisman_conj function creates a conjunction display for the visual search task, based on the target presence, number of objects and set of symbols. Is uses a distribution of t targets, (t is either 1 or 0 depending on whether the target is present or not),and n/2-t, n/4 and n/4 distractors, with n being the total amount of objects. It uses the zet_symbool_in_figuur function to place each objects onto the figure.

The following flow charts were made to illustrate treisman_conj:

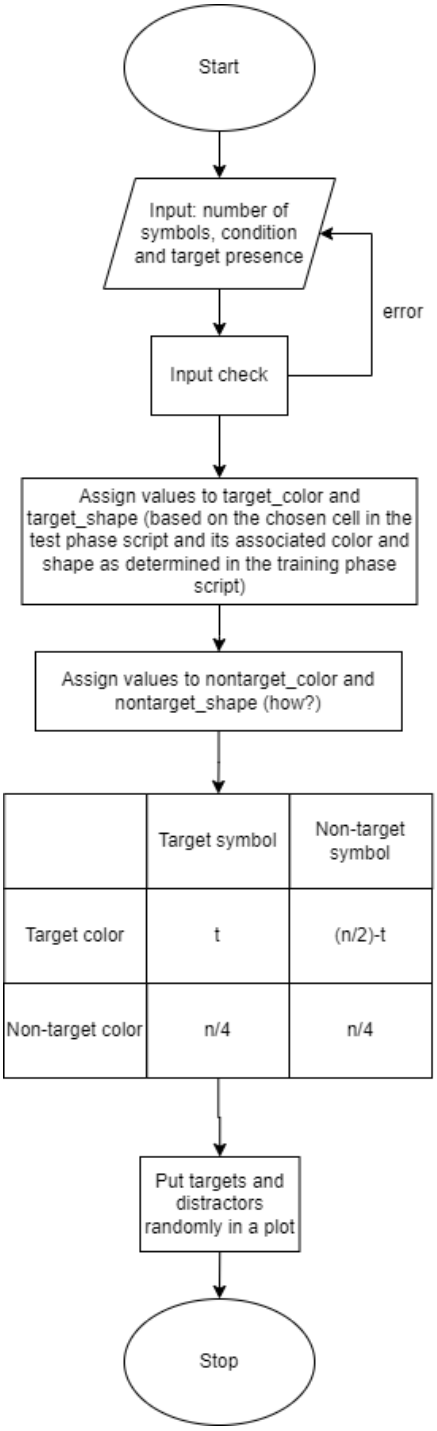

Figure 13. *First version of treisman_conj.*

	Target symbol	Non-target symbol
Target color	t	(n/2)-t
Non-target color	n/4	n/4

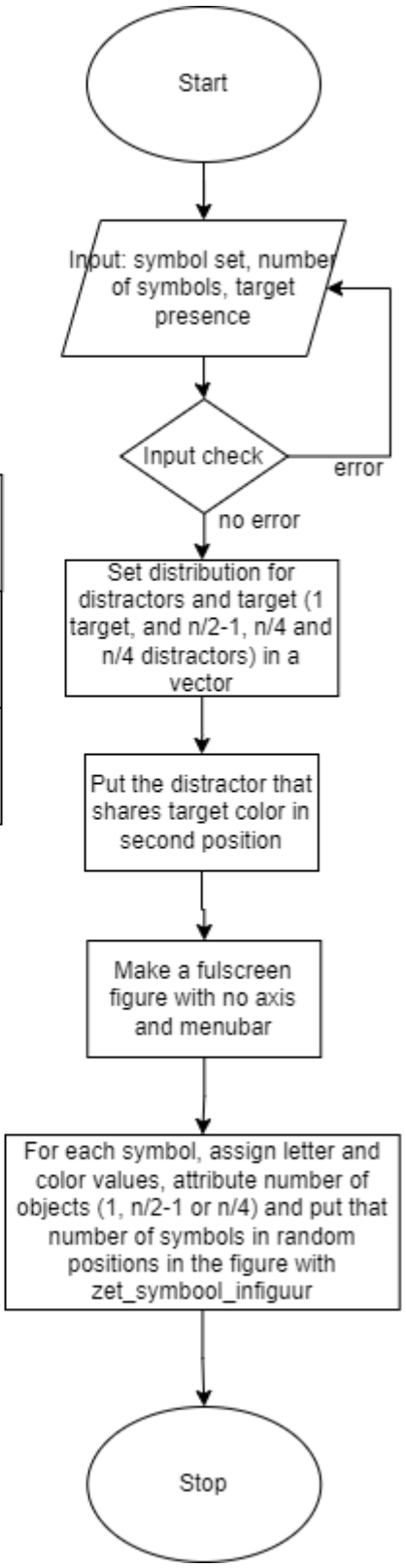

Figure 14. *Final version of treisman_conj.*

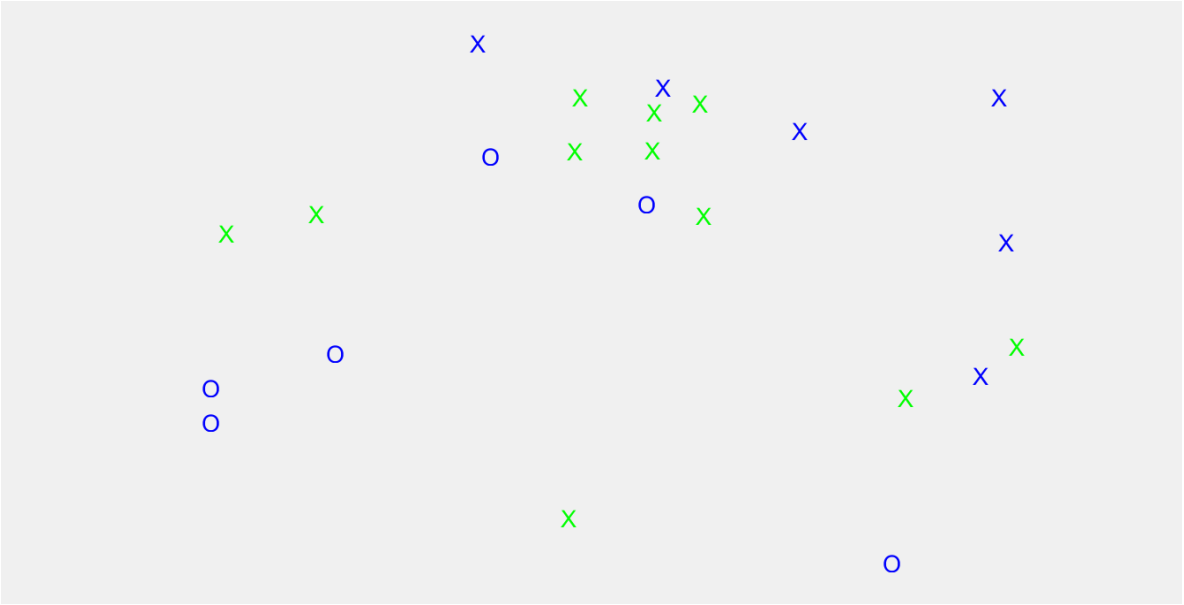

Figure 15. *An example of a VST trial.*

Preference test / test_association

The test_assocation function measures the preference of the participant for symbols from a specific condition (food associated, neutral associated and control). It does this by presenting two objects at once, both objects are from two different condition (two_symbols function). The user presses F if they prefer the object on the left and J if they prefer the object on the right. Then, the next object pair is shown and the user chooses again. The preference within each condition comparison is stored in a table.

The following flow chart was made to illustrate test_association:

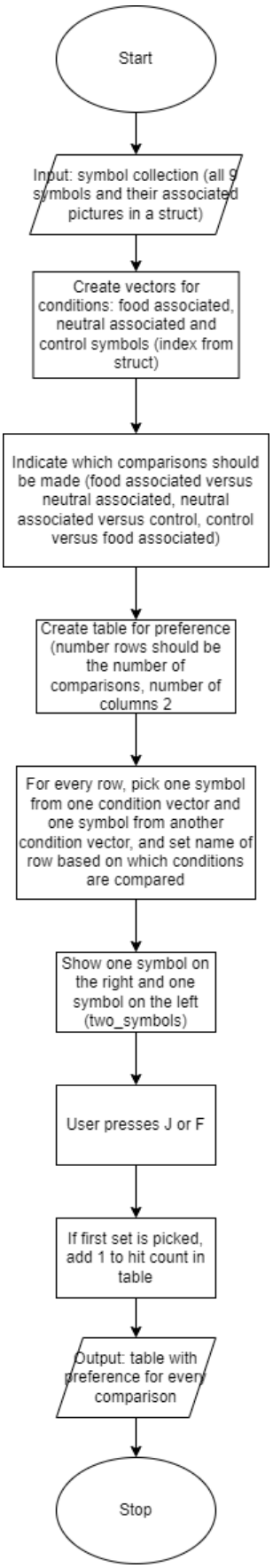

Figure 16. *Flowchart of test_association.*

Visuals preference test / two_symbols

This function plots two symbols onto the screen, so the participant can choose which character they prefer.

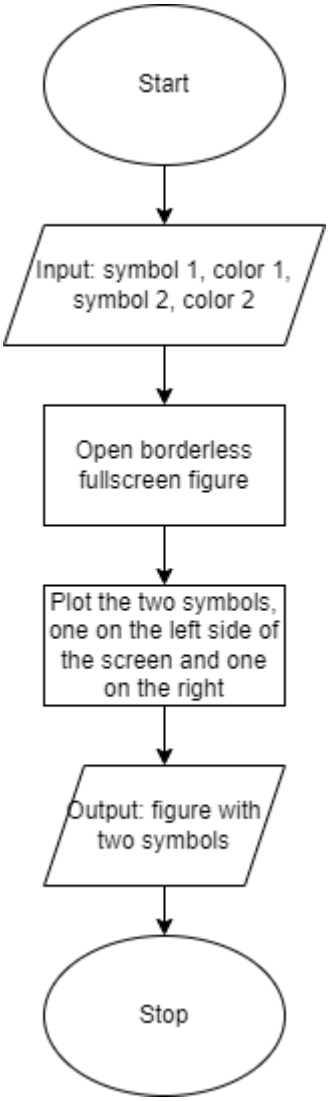

Figure 17. *Flowchart of the two_symbols function*

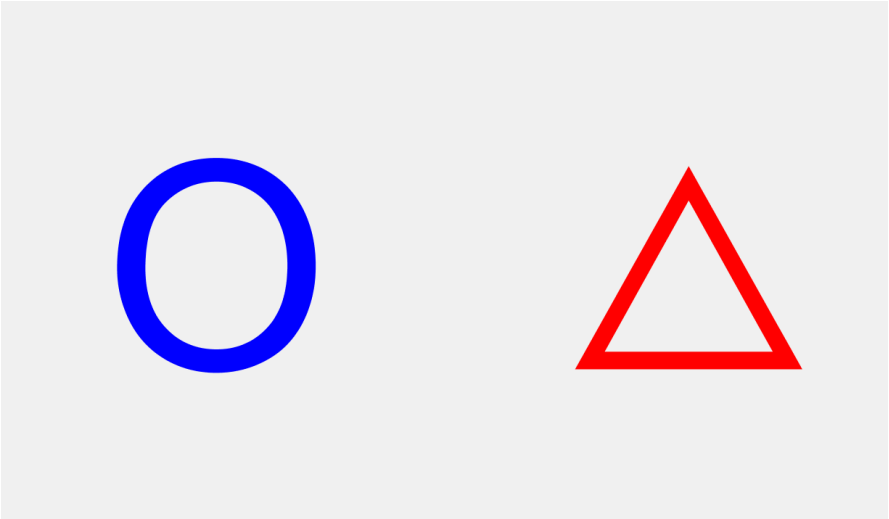

Figure 18. *An example of a screen in which the user has to state their preference towards one symbol.*

Randomize order within vector / randomize

This function randomizes the order of the conent in a dimensional cell or vector.

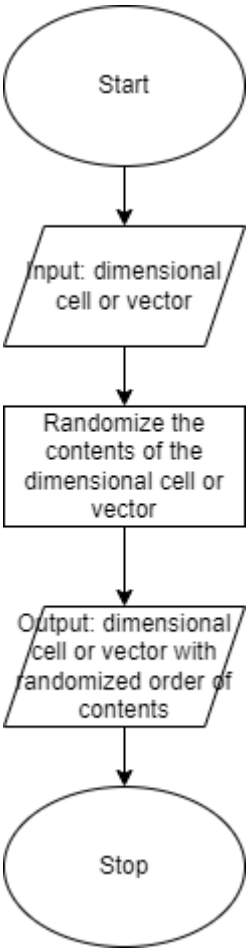

Figure 18. *A flowchart of the randomize function.*

GUI

There are six information screens part of the GUI: a start screen, three break screens, a slider to indicate food satisation (hungerslider) and an endscreen. The information on the screens, such as a title and other text is put in short functions, in order to make the script short. Every function (startscreen, endscreen, breakscreen1, breakscreen2 and breakscreen3) except for the function hungerslider, makes use of the generate_form function. This is a function that generates a single layout for those screens. All functions make use of the functions uisubmitbutton and pressed. The function pressed is made for uisubmitbutton and does the inputchecks for every input given by the user in the textboxes and closes the figure afterwards, when the “Continue” button in a screen is pushed, made by the function uisubmitbutton and puts the values in the map that are asked for in generate_form.

uisubmitbutton & pressed

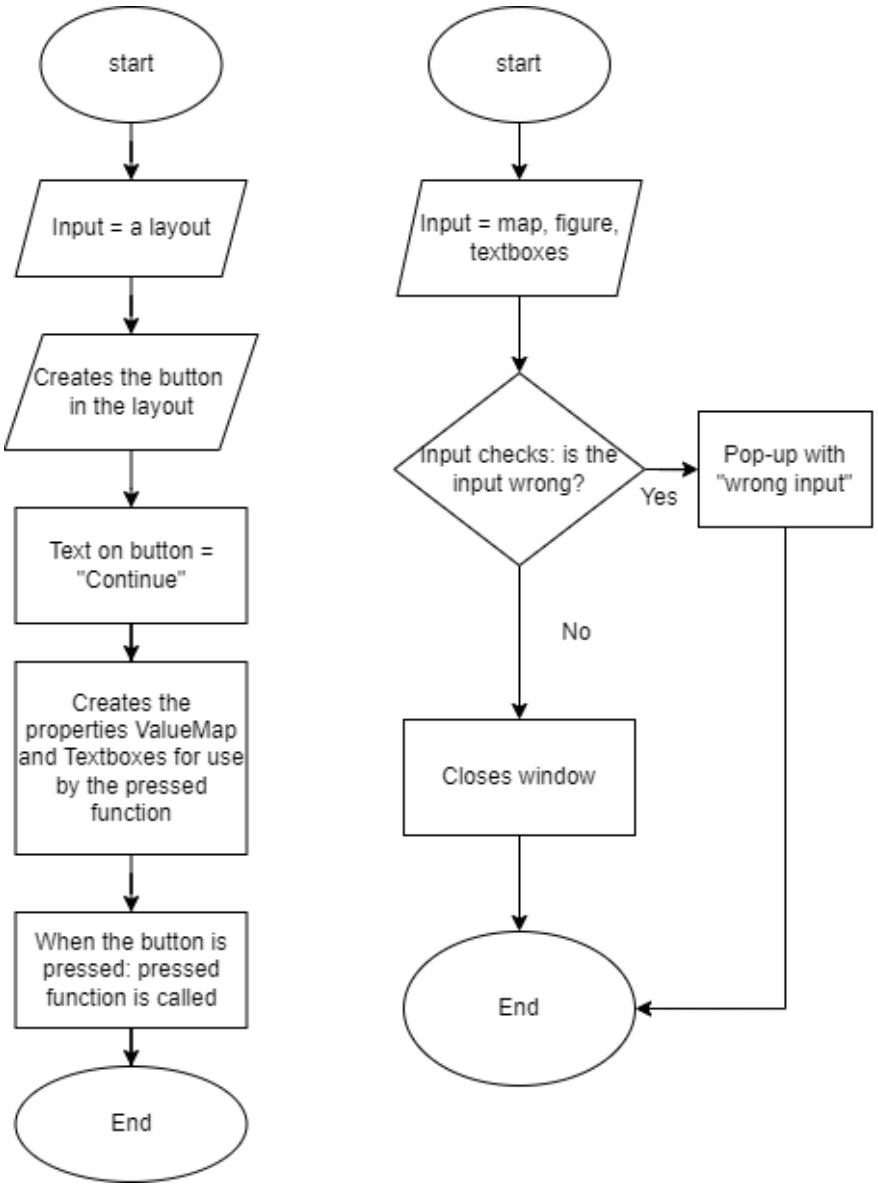

Figure 19. The uisubmitbutton function on the left and the pressed function on the left.

generate_form

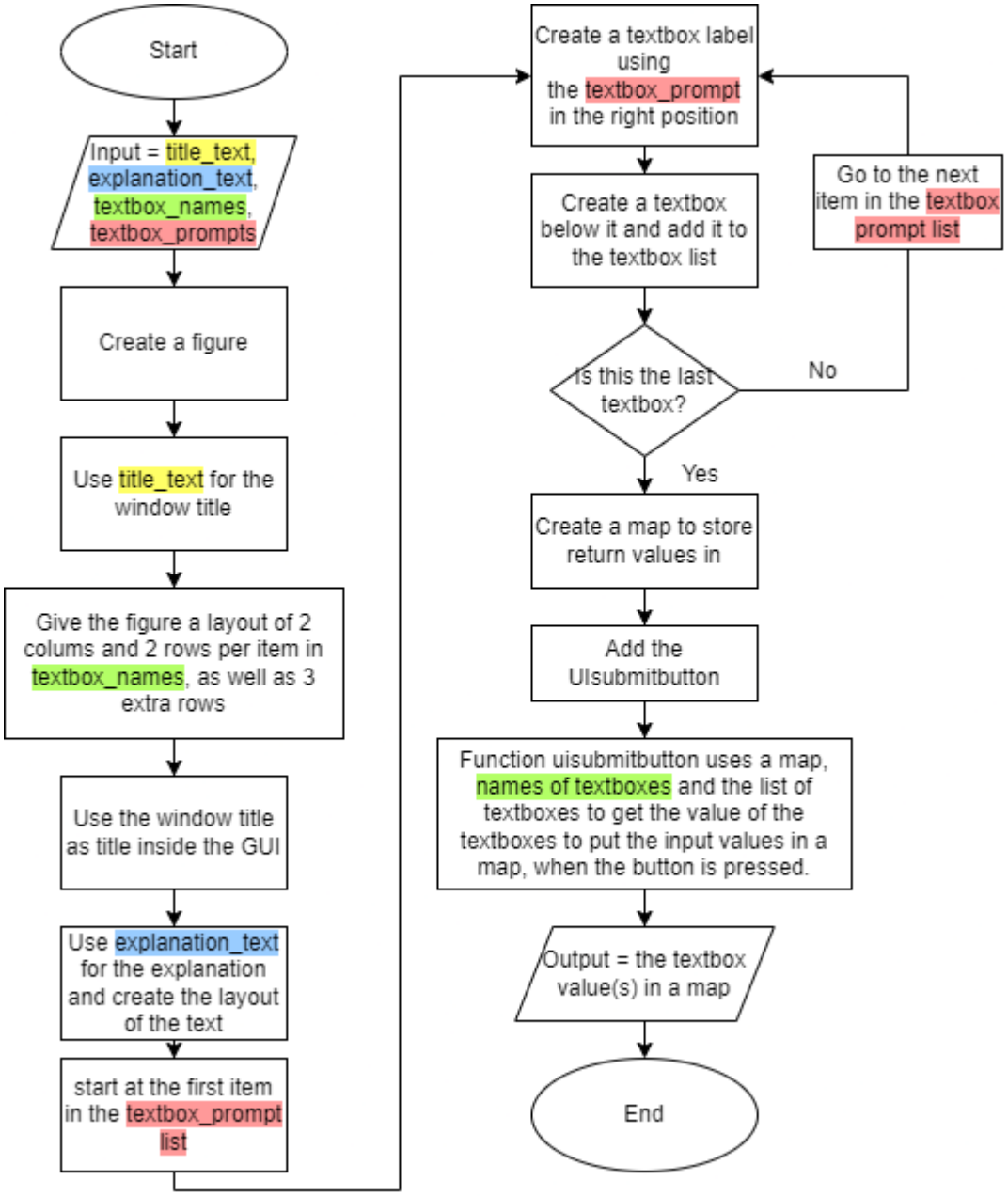

Figure 20. The function generate_form. Colors are used to indicate where the different inputs play a role.

hungerslider, endscreen, startscreen, breakscreen1, breakscreen2, breakscreen3

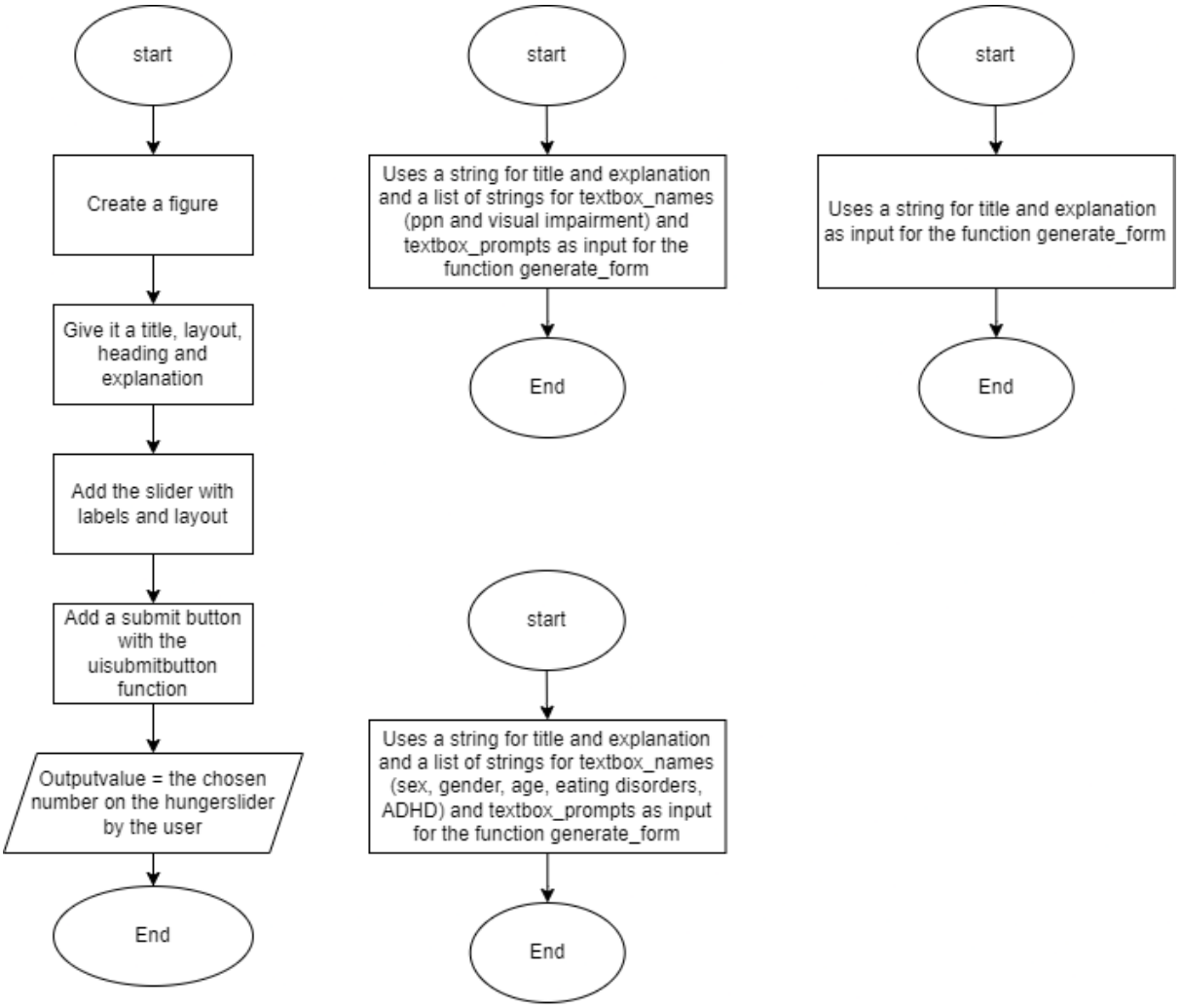

Figure 21. 4 flowcharts: hungerslider on the left, endscreen (above) and startscreen (under) in the middle and breakscreen1, breakscreen2 and breakscreen3 have the same flowchart (on the right).

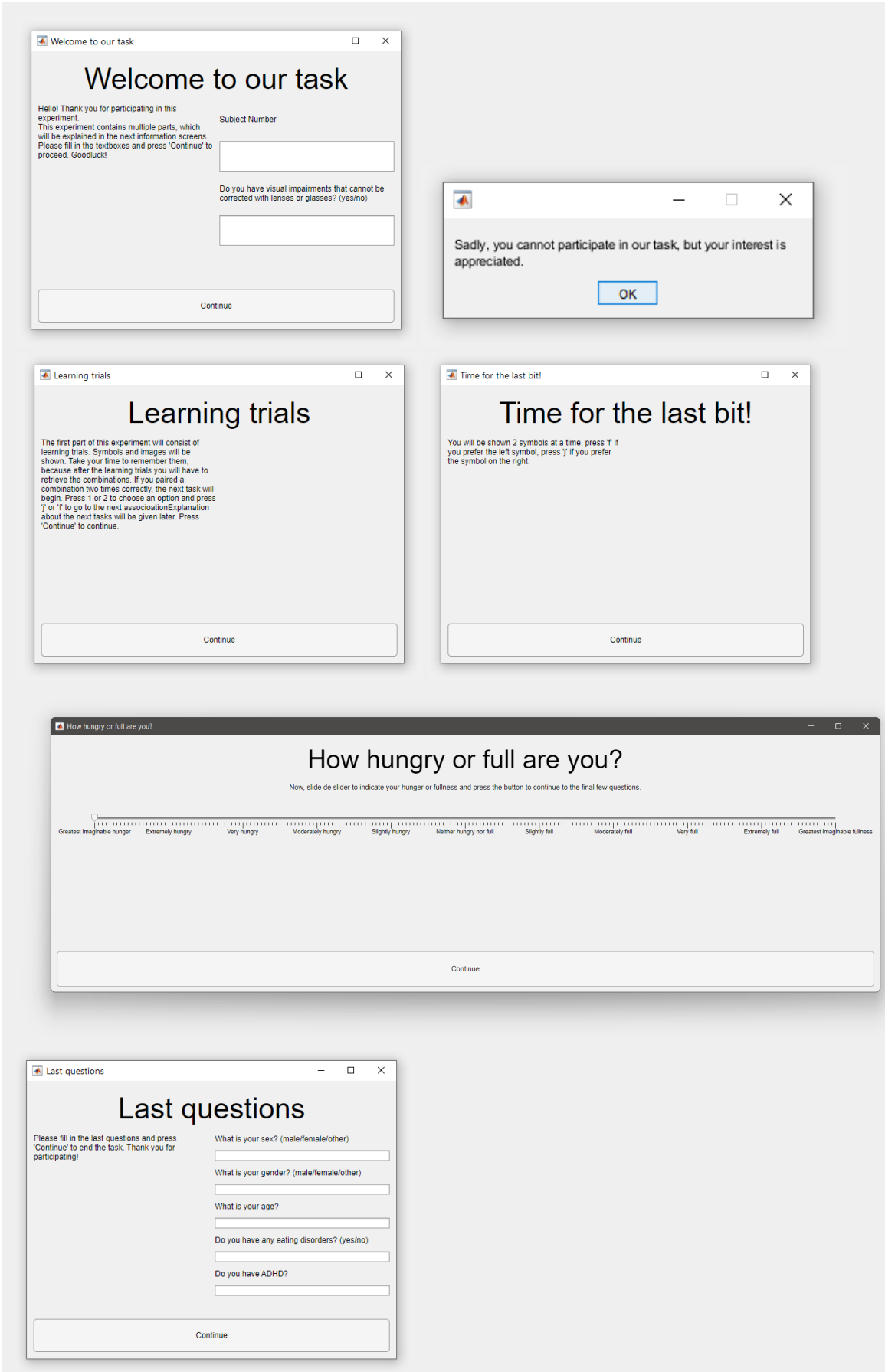

Figure 22. Several GUI screens used in our experiment.

Implementing

See attachment for the code.

Evaluation

The association task had to be tested primarily on two fronts: making sure symbols are randomized without randomizing their associations and images and making sure to log the correct answers during retrieval. Initially, when a symbol was put in a random order its associated image was also randomized. The testing of this bug was done by repeatedly running the learning trials. When a symbol retained its associated images across trials, we knew the bug was fixed.

two_symbols was a smaller function. The implementation was quite simple, consisting of simply plotting two symbols largely on the screen. Most testing consisted of figuring out the coordinates to properly center the symbols. We roughly measured the distance between both symbols and the edges of the screen. And testing was concluded when this distance was about equal.

treisman_conj was adapted from Inleiding Progammeren's Treisman_exp. The main differences were that instead of passing a type, like conjunction, an entire symbol set was passed. This symbol set consisted of 4 symbols each having its own letter and color. Because we had to keep track of which symbols were already presented, we had to be able to determine which symbols from the set became the target. It was decided that the first symbol became the target, thus being presented only once or not at all. The symbol that shared the target color had to be assigned the largest population to make sure conjunction worked properly. We tested this by passing several symbol sets to the function until the distribution was proper.

doe_experiment.m was mostly adapted from the visual search task used in Inleiding Programmeren. The main difference was being that symbol sets had to be passed instead of types. This was easily implemented and when doe_experiment could produce a proper treisman graph based on the input and return the output variables we were satisfied.

make_symbol_set its purpose was to get passed the complete set of symbols and return only four symbols, whereby only two colors and two letters are used. Additionally, the first two and last two symbols each share their color. The function relied on picking two random colors and two random letters and then selecting the matching symbols from the complete symbol collection using logical vectors. make_symbol_set was tested by making sure each symbol produced matched the requirements. For example, making sure the second symbol matched the first in color before moving to the code of the first symbol. When all prerequisites were met for the fourth symbol testing was concluded.

test_association was a simpler function and mostly consisted of two_symbols. Similar to doe_experiment the only code present in the function was to pull from the set of collective symbols pass this to a function that produces visual results and then track user input. Input checks were tested by pressing all buttons on the keyboard and making sure only f and j would trigger an unpause. Data storage was tested by randomly picking between two symbols and logging our results, and using the MatLab working space we could see if the stored data matched our inputs.

VST was also mostly adapted from visual search task. Most of the testing consisted in making sure the hit matrix was properly tracking which symbol was passed. We ran the VST with a lower amount of required hits to make sure the code properly logged correct hits and stored this data.

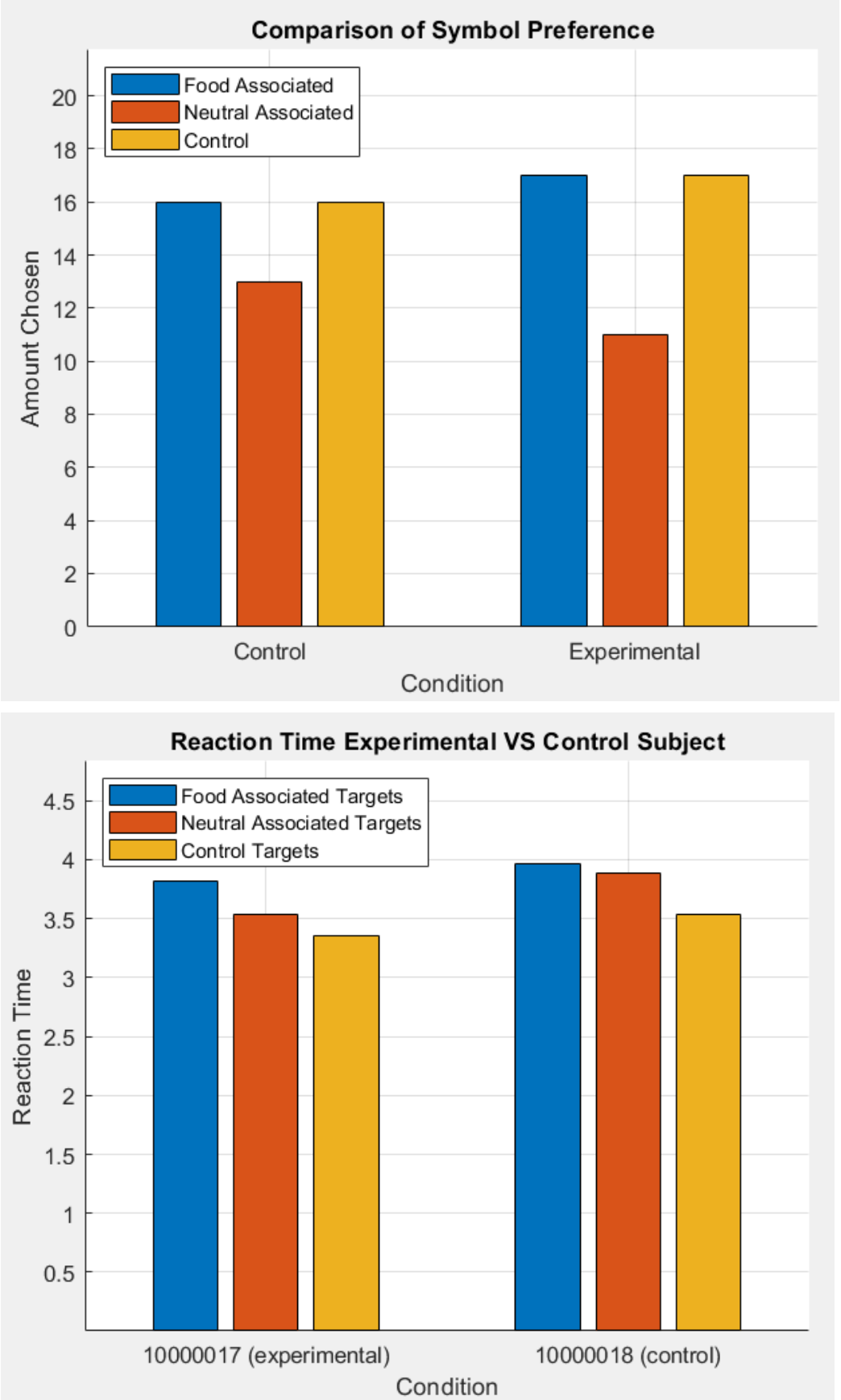

Figure 23. *Plotted data from two participants, one from the experimental group (left) and one from the control group (right). The graphs above show the amount of time a condition was preferred over another. The graphs under show the reaction time per association for the target.*

The program performs as intended. All major tasks are properly performed with proper data output. All participants had all the intended data points stored (see figure 23) and no major issues with data export were experienced.

Feedback

Feedback from participants

- The green symbols were popping out less than the red and blue ones according to the participants. They were more difficult to see, so this should be a darker color the next time.
- According to one participant the Δ's and X's were more difficult to distinguish relative to the O's and any other symbol.
- Make food pictures more arousing. This can be done by making them bigger and making the white borders around it less big to integrate it better in interface.

Known bugs

- For every task: if you click too fast on a button, the command window will appear in front of the screen of the experiment.
- Some symbols can overlap in the VST. This makes it possible that the target might overlap with another symbol.

Limitations

- The largest limitation is the inability to track which symbols are associated with which image during the experiment. This leads to difficulty when ensuring the program is working properly. The only way that data can be monitored is by running the program in sections
- Secondly, whether a control trial or experimental trial is run depends on either a hard coded variable or a randomizer. Preferably we would like to add a HUD for the researcher which can determine which condition is ran.
- Another limitation is the layout of the GUI. While the function generate_form makes the code much shorter by having one function to do the layout of every GUI, it is also a limitation. If you make the continue button larger for example, the continue button will be larger in every GUI screen that uses this function.

Changes in the flow charts

- Main
    - We moved sensitive questions to the end of the experiment instead of the beginning
    - Some exclusion critera are no longer exlcusion criteria, but they are asked at the end of the experiment for further analysis
    - First it was noted that the keypress, accuracy and reaction time should be stored for the training task, but this was a simple misplacement of the box. There is now no output of the training phase (in whcih the associations are learned)
    - In the first version of the flow chart, it was not clear yet that the control group should skip the training phase
    - First it was not noted that the output of the test phase should be a struct containing data about each trial.

- Association_task
    - The most important change is that we got rid of the control block because the coding was getting too difficult and we were running out of time
    - Another adjustment is that the assigning of the color and shapes based on the associations is now done in a more soft-coded manner
- VST
    - Instead of counting down from 20 to 0, we are now counting up from 0 to 20 in the cells of the matrix
    - The breaks are now every 50 trials instead of 20
- Doe_experiment
    - There are actually no input checks in this function so we removed it from the flow chart
    - The output is more elaborated (it is now clear that the data should be put in a struct)
- Treisman
    - The target/distractor colors and shapes are no longer determined by target_color, target_shape, nontarget_color and nontarget_shape variables, but by indexing from the symbol set struct.
    - The distractor that shares the target color is now put into the second position so it has n/2-t objects

Reflection assignment

The concepts were mainly translated into code by first assessing the methodology traditionally used in experiments. Afterwards for many functions flowcharts or sketches were utilized to illustrate how each function could be split into subfunctions. Subfunctions could largely be grouped into visual and non-visual groups. The visual functions were the first functions to be developed.

No functions were specifically difficult to implement. The most time consuming parts were mostly ensuring proper usage of data structs. One example being the usage of structs and cell array and making sure that when passing to functions the input is compatible. A concrete problem was when several functions were not working because the content with cells were surrounded by brackets. After lots of testing it turned out that using double brackets (“ “) initialized different data types then single brackets (‘ ‘). These kinds of issues are things you always deal with when learning a new language and cannot really be prevented. Obviously, we will make sure to not make this mistake again, however similar mistakes are inevitable and a natural part of the learning experiences.

Some functions that were slightly more time consuming were make_symbol_set and the GUI functions. make_symbol_set is meant to produce symbol sets usable for a visual search trial, however the entire process consisted of lots of logical vectors and was extremely time consuming. It was by no means an impossible function to implement but had we taken a step back and thought about implementation and which logical vectors to use it would have been less of a struggle.

The GUI functions had their own struggles. Part of it was used from an earlier assignment (for “inleiding programmeren”), but since we wanted to have multiple GUI screens, using the

same long code would create a huge script. In the end this has been solved by creating the function generate_form and using it for the layout for breakscreen1, breakscreen2, breakscreen3, startscreen and endscreen. The breakscreens were not a struggle, though getting the layout to fit the endscreen and the startscreen was not easy, though it worked in the end, due to a lot of trying. Only the function hungerslider had its own written layout, since it was too different from the other screens.

References

Anderson, B. A., Laurent, P. A., and Yantis, S. (2013). Reward predictions bias attentional selection. Frontiers in Human Neuroscience, 7. https://doi.org/10.3389/fnhum.2013.00262

Gearhardt, A. N., Treat, T. A., Hollingworth, A., & Corbin, W. R. (2012). The relationship between eating-related individual differences and visual attention to foods high in added fat and sugar. Eating Behaviors, 13(4), 371–374. https://doi.org/10.1016/j.eatbeh.2012.07.004

Hickey, C., Chelazzi, L., & Theeuwes, J. (2010). Reward Changes Salience in Human Vision via the Anterior Cingulate. Journal of Neuroscience, 30(33), 11096–11103. https://doi.org/10.1523/jneurosci.1026-10.2010

Hickey, C., Chelazzi, L., & Theeuwes, J. (2011). Reward has a residual impact on target selection in visual search, but not on the suppression of distractors. Visual Cognition, 19(1), 117–128. https://doi.org/10.1080/13506285.2010.503946

Huffman, Elizabeth & Chabris, Christopher & Ariely, Dan & Aharon, Itzhak & Kaplan, Lee & Breiter, Hans. (2022). Pictures of Food Have Reward Value that Varies According to Appetitive State. http://www.chabris.com/Huffman2001.pdf

Kristjansson, A., Sigurjonsdottir, O., & Driver, J. (2010). Fortune and reversals of fortune in visual search: Reward contingencies for pop-out targets affect search efficiency and target repetition effects. Attention, Perception & Psychophysics, 72(5), 1229–1236. https://doi.org/10.3758/app.72.5.1229

Treisman, A. M., & Gelade, G. (1980). A feature-integration theory of attention. Cognitive Psychology, 12(1), 97–136. https://doi.org/10.1016/0010-0285(80)90005-5