PSYC2301 Project: Perception of Differences in Shape Size

Identification:

* Title: Perception of Differences in Shape Size
* Programmer: Adam Tyson
* Date: February 2011
* Language: Matlab v7.10

Problem to be solved:

This function of this program is to determine whether the colour of a shape, and the shape itself play a role in a subjects’ ability to judge its size. The program is designed to test a subjects’ perception of the difference in size of two shapes. The subject will be presented with a shape in the centre of the screen of a random size, and other similar shapes, of different sizes will also appear around the centre shape. After a brief delay, one of them will change to a specified colour. The subject will then be required to respond as to whether they think the coloured shape is larger or smaller than the central one.

The user can specify the colour of the shape, red, green, blue, or all three, as well as the shape itself, circle, square, triangle, or again, all three. Also the user can specify how many blocks of each stimulus condition (such as red circle, or green square) should be displayed, and how many trials make up each block. For each block of trials, the average response time, and the number of correct responses will be recorded. These results from each set of blocks from each subject will be exported to Microsoft Excel, and saved along with the subject’s details, such as name, age, etc.

Solution:

To write this function, I generated functions to carry out different tasks, such as display the stimulus, and write the results. Most of the functions are called from the main script, shape\_size\_main\_program.m. The script begins by clearing the command window, the workspace, and closing any open figures, to prevent any stored variables from being called in the program. Then two functions are called, the first, subject\_info.m collects information about the subject, such as name, and age, to go with the experimental data. The second, exp\_info, collects information about the experiment, to determine how it is carried out, such as what stimulus should be displayed, and how many trials should make up each block.

Two functions are then called, trial\_conds.m and set\_cols\_shapes.m which assign certain variables for the displaystim.m function to display the correct stimulus when required. Another function, functionsdata.m is also called to assign variables for the experiment to be passed to the function, displaystim.m, such as the size of the screen, the font size, which stimulus should be displayed, and how many trials should be run.

A conditional statement then determines whether the experiment should be run in development mode or not. If yes, the Psychtoolbox sync tests are skipped, and the warnings are supressed, to make testing the program easier. The experiment will run in full screen by default, but will run in a small window if development mode is required, to make it easier to leave the program should an error occur.

The Psychtoolbox screen function is then used to open a window, the mouse cursor is hidden for the experiment, and keyboard output to the command window is supressed during the experiment. Another conditional statement then follows, so if a practice block is required, the function practiceinstructions.m is called to display instructions to the subject about it, and the displaystim.m function is called to display one block of 5 trials to the user, of red circles. The data from this block is not saved; it is just to let the subject get used to responding to the stimuli before the main experiment begins.

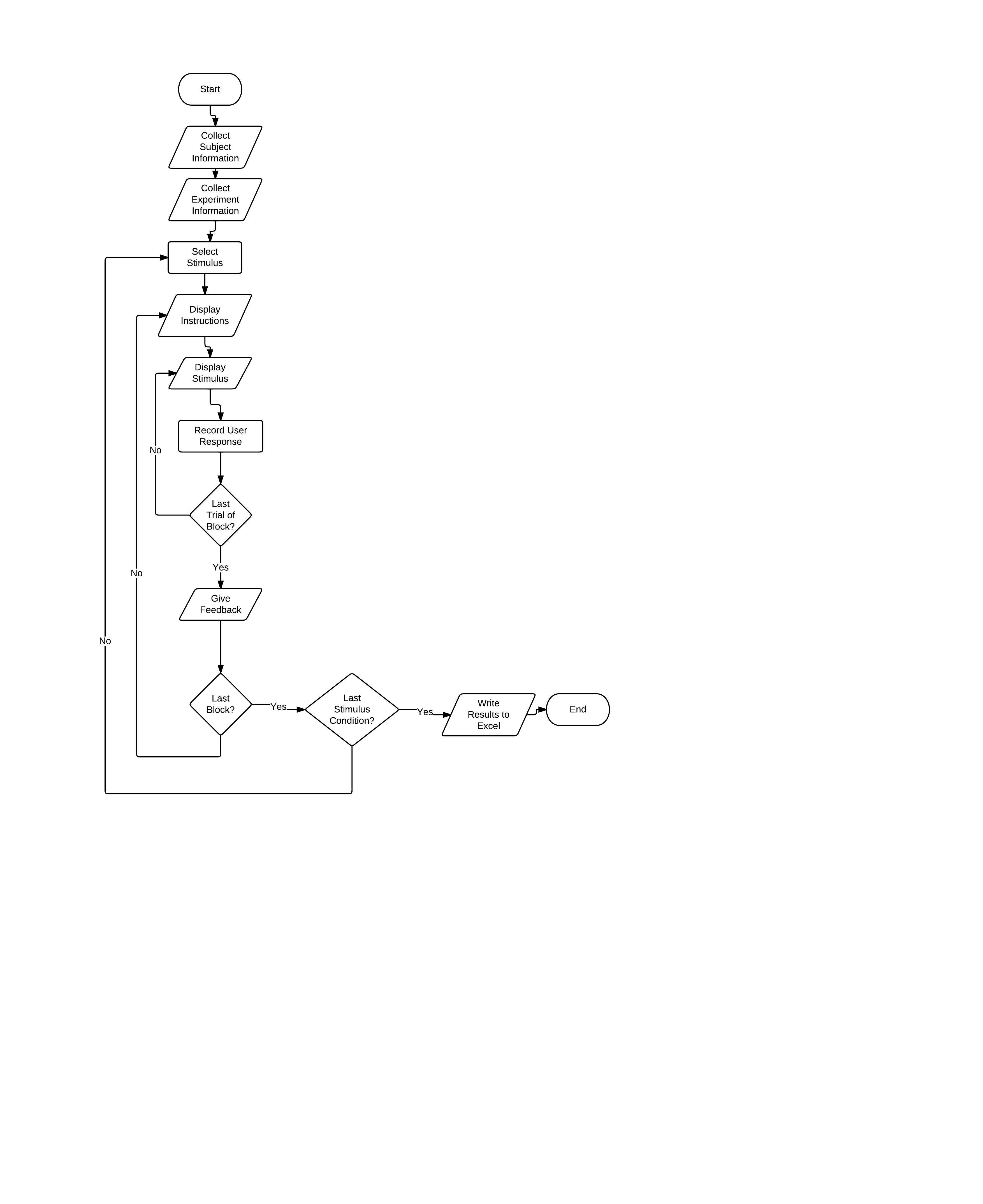
Two loops are then set up, for the displaystim.m function. One repeats the function for the required amount of blocks, and the other cycles through all the possible stimulus conditions. Inside these loops, the displayinfo.m, displayinstructions.m and the displaystim.m functions are called. The function, displayinfo.m is used to assign display variables for the other two functions, for each stimulus condition. Then displayinstructions.m uses these variables to display information to the subject about the block, and how to respond to the stimulus. The information is also passed to the displaystim.m function. This function takes the experimental information as input, displays one block of trials, and returns the results of that block, as the number of correct responses, and the average response time.

The displaystim function firstly sets certain variables, such as initialising the number of correct responses to 0, and sets other variables to allow for correct positioning of the shapes. After this, the function then displays the stimulus. This consists of a shape in the centre of the screen, surrounded by the same shape, of different sizes. After a pause of half a second, one of the outer shapes will change colour. The subject must then respond as to whether the coloured shape is larger or smaller than the central shape, responding with up on the keyboard for larger, and down for smaller. This is repeated for the number of trials in the block. Then the results of the block are displayed to the subject, and passed back to the main script. The loops in the main script then repeat this for each block of each stimulus condition. The results from each block are then saved as two 2D arrays, results\_correct, and results\_avtime

The screen function is then called to close the on screen window, at the end of the experiment, and the main script then calls a function, writeresults.m to write the saved results to a Microsoft Excel file, named after the subject. Two functions, title\_names.m and results\_titles.m are nested within writeresults.m to write titles for the saved data. The saved Excel files contains one sheet with the subject, and the experimental details, one sheet with the number of correct responses for each block, and a third sheet with the average reaction times for each block.

See Fig 1 for program structure.

Fig 1: Flowchart of the structure of the program designed to run the shape size experiment



Program Listing

%shape\_size\_main\_program.m

%Requires the user to input the subject and experimental information,

%displays the stimulus for each stimulus condition to the subject, then

%records and exports response data along with the subject information to

%Excel

clear all; % clear workspace

close all; % closes all figures

clc; % clears command window

subject\_info=collect\_subject\_info; % calls function to collect subject information

clc %clears command window for next set of inputs

exp\_info=collect\_exp\_info; % calls function to collect experiment information

clc %clears command window again

trialconds=trial\_conds(exp\_info); %calls function to set a value (0, 1 or 2) corresponding to each possible stimulus type, to determine whether or not they are displayed

[colours, shapes] =set\_cols\_shapes; % calls function to set the colour and shape data for each possible stimulus type

[sizeofscreen, numoftrials, bkgrndshapecol, fs, lw, screen] = functionsdata(exp\_info); %calls function to set certain variables

try %run script as long as no errors occur

if exp\_info(6)==121; %if response is y (for development mode)

Screen('Preference', 'SkipSyncTests', 1); %skip sync tests for development

Screen('Preference', 'SuppressAllWarnings', 1); %suppress warnings for development (enable to monitor PTB problems)

sizeofscreen=[20 20 820 620]; % sets Psychtoolbox screen size to small window for development

end

[screen1,rect]=Screen('OpenWindow',screen, 0, sizeofscreen); %open screen of set dimensions on screen 0, (outside displaystim function to keep window open between blocks)

HideCursor; %hide mouse cursor for experiment

ListenChar(2); %suppress keyboard output to command window

width=rect(3); % assigns width (in pixels) of the screen to width

height=rect(4); % assigns height (in pixels) of the screen to height

if exp\_info(5)==121; %if practice block required

practiceinstructions(screen1, fs, width, height) % call function to display information about the practice block to the subject

displaystim(5, bkgrndshapecol, [255 0 0], 1, fs, lw, screen1, rect, width, height); %calls function to display stimuli, for one practice block of 5 red circles

end

stim\_counter=0; % initalises variable

for a=1:9; %loop for 1 to 9 (number of possible stimuli)

if trialconds(a)==2; %if both colour and shape required

shape=shapes(a); %sets required shape for function

colour=colours(a); %sets required colour for function

stim\_counter=stim\_counter+1; % counts how many different stimulus have been displayed (to allow correct text to be shown in displaystim function

for blox=1:exp\_info(1); %sets loop to repeat for number of blocks

[shapetype, shapecolour, colourtext] = displayinfo(shape, colour); %call function to assign display variables

displayinstructions(blox, stim\_counter, screen1, fs, width, height, shapetype, colourtext); %call function to display information to the subject

[num\_correct, avtime] = displaystim(numoftrials, bkgrndshapecol, shapecolour, shape, fs, lw, screen1, rect, width, height); %calls function to display stimuli, and returns results

blk\_res\_correct(blox)=num\_correct; % sets the blox'th element of the variable as the number of correct responses from the block

blk\_res\_avtime(blox)=avtime; % sets the blox'th element of the variable as the average response time from the block

pause(0.5) %pause before next stimulus condition

end

results\_correct(a,:)=blk\_res\_correct; %sets the ath row of the matrix, as the vector of the number of correct responses

results\_avtime(a,:)=blk\_res\_avtime; %sets the ath row of the matrix, as the vector of the average response time

end

end

ListenChar(0); %Reset back to default setting, allows keyboard input to the command window

ShowCursor; % show mouse cursor again

Screen('Close', screen1); %close screen

writeresults(subject\_info,exp\_info, results\_correct, results\_avtime) %calls function to write results to excel

catch % if error occurs

Screen('CloseAll'); % close screen

Priority(0); % reset priority back to lowest

ShowCursor

end

function [subject\_info] = collect\_subject\_info

% collect information about the subject as a cell, for export to MS Excel,

% to go with the experimental data from each subject.

subject\_info{1}=input('Enter subject number: '); %collects basic information to go with results for analysis of experimental data

subject\_info{2}=input('Enter subject name: ','s');

subject\_info{3}=input('Enter subject age: ','s');

subject\_info{4}=input('Enter subject gender (m/f): ' , 's');

subject\_info{5}=input('Enter subject handedness (r/l/a): ' , 's');

subject\_info{6}=input('Enter session number: ');

end



function [exp\_info] = collect\_exp\_info

%collect information about the experiment as a 1D array

exp\_info(1)=input('Enter number of blocks per colour: '); % how many blocks per colour should be presented?

exp\_info(2)=input('Enter number of trials per block: '); %how many trials should be presented in each block?

exp\_info(3)=input('Which colours should be displayed? red(r), green(g), blue(b), or all(a)? ' , 's'); %which colours of shapes should be presented?

exp\_info(4)=input('Which shapes should be displayed? circles(c), squares(s), triangles(t), or all(a)? ', 's'); %which shapes should be presented?

exp\_info(5)=input('Practice sessions? (y/n) ','s'); % should practice sessions be displayed before each set of blocks?

exp\_info(6)=input('Development mode? (y/n) ','s'); % If y, then runs in development mode (display in small window, and skips sync tests in PTB)

end



function [trialconds] = trial\_conds(exp\_info)

%work out which stimulus needs to be displayed

%add 1 when stimulus condtion is required. Eventually in the main script,

%only conditions of value 2 will be displayed to the subject

trialconds=zeros(1, 9); %inialises all 9 possible stimulus conditions to 0

if exp\_info(3)==114 % if response is r, add 1 to the red stimulus conditions

trialconds(1:3)=trialconds(1:3)+1;

elseif exp\_info(3)==103 % if response is g, add 1 to the green stimulus conditions

trialconds(4:6)=trialconds(4:6)+1;

elseif exp\_info(3)==98 % if response is b, add 1 to the blue stimulus conditions

trialconds(7:9)=trialconds(7:9)+1;

elseif exp\_info(3)==97 % if response is a, add 1 to all stimulus conditions

trialconds=trialconds+1;

end

if exp\_info(4)==99 % if response is c, add 1 to the circle stimulus conditions

trialconds(1:3:7)=trialconds(1:3:7)+1;

elseif exp\_info(4)==115 % if response is s, add 1 to the square stimulus conditions

trialconds(2:3:8)=trialconds(2:3:8)+1;

elseif exp\_info(4)==116 % if response is t, add 1 to the triangle stimulus conditions

trialconds(3:3:9)=trialconds(3:3:9)+1;

elseif exp\_info(4)==97 % if response is a, add 1 to all stimulus conditions

trialconds=trialconds+1;

end



function [colours, shapes] =set\_cols\_shapes

%sets the colour and shape data for each possible stimulus type,

% to be passed to the function displaystim

colours(1:3)=1; %1 for red, 2 for green, 3 for blue

colours(4:6)=2;

colours(7:9)=3;

shapes(1:3:7)=1; %1 for circle, 2 for square, 3 for triangle

shapes(2:3:8)=2;

shapes(3:3:9)=3;

end



function [sizeofscreen, numoftrials, bkgrndshapecol, fs, lw, screen] = functionsdata(exp\_info)

%functionsdata.m

%assigns variables for later functions

sizeofscreen=[]; % sets default screen size to fill screen with window, [] for max dimensions

numoftrials=exp\_info(2); %number of trials per block

bkgrndshapecol=128; % set colour of background (non tested) and centre shapes to grey

fs=15; %set font size to 15

lw=4; %set shape line width to 4

screen=0; %set main screen to 0, (can be changed to use screen other than main monitor)

end



function practiceinstructions(screen1, fs, width, height)

%practiceinstructions.m

%displays instructions to the subject for the practice block

Screen('TextFont',screen1, 'Courier New'); %Choose font

Screen('TextSize',screen1, fs ); %Choose font size

Screen('DrawText', screen1, ('The practice block is about to begin,'), width/10, height/10, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Please read the instructions carefully.'), width/10, (height/10)+50, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Press any key to continue.'), width/10, (height/10)+100, [240 248 255]);%Draw text

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %Flip onto screen

KbWait;

pause(0.5);

Screen('TextFont',screen1, 'Courier New'); %Choose font

Screen('TextSize',screen1, fs ); %Choose font size

Screen('DrawText', screen1, ('Instructions:'), width/100, height/100, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Please look at the centre circle,'), width/100, height/100+40, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Press up or down when you see the red circle around it'), width/100, (height/100)+60, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Please press up, if the red circle is larger,'), width/100, (height/100)+90, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Press down if the red circle is smaller'), width/100, (height/100)+110, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Press any key when ready'), width/100, (height/100)+180, [240 248 255]);%Draw text

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %Flip onto screen

KbWait;

pause(1);

end



function [crrct, avtime] = displaystim(numoftrials, bkgrndshapecol, shapecolour, shape, fs, lw, screen1, rect, width, height)

%displaystim

%displays stimulus block depending on input arguements supplied

z=[0 0 0 0 0 0 0 1]; % set all but the last element of z to 0, the 8th to 1, to randomly select one shape to be coloured

crrct=0; %inialise crrct

shapspac=height/(3\*(2^0.5)); %assigns shapespacing value to ensure shapes at 45,135,225 degrees etc are the same distance from the centre as the orthogonal shapes

for trials=1:numoftrials; % sets up loop for each trial

a=((randperm(9))\*2)+10; %assign random permutation of shape sizes from 12 to 28 to a

%also makes sure that the test shape is a different size than the centre

%shape, as they are both selected from the same selection of sizes

b=randperm(8);%assigns random permutation of numbers 1 to 8 to b

%use the z array and b to select a random shape to be the test shape

%%%%%%%%%%%%%%%%%%%%%%%%%%%% CIRCLE %%%%%%%%%%%%%%%%%%%%%%%%%

if shape==1;

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-a(1)) (height/2-a(1)) (width/2+a(1)) (height/2+a(1))], lw ); %draws random size square in the centre of the screen

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-a(2)) (height/6-a(2)) (width/2+a(2)) (height/6+a(2))], lw ); %0 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2+shapspac-a(3)) (height/2-shapspac-a(3)) (width/2+shapspac+a(3)) (height/2-shapspac+a(3))], lw ); %45 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2+ height/3-a(4)) (height/2-a(4)) ((width/2)+height/3+a(4)) (height/2+a(4))], lw ); %90 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2+shapspac-a(5)) (height/2+shapspac-a(5)) (width/2+shapspac+a(5)) (height/2+shapspac+a(5))], lw ); %135 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-a(6)) ((5\*height)/6-a(6)) (width/2+a(6)) ((5\*height)/6+a(6))], lw ); %180 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-shapspac-a(7)) (height/2+shapspac-a(7)) (width/2-shapspac+a(7)) (height/2+shapspac+a(7))], lw ); %225 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2- height/3-a(8)) (height/2-a(8)) (width/2-height/3+a(8)) (height/2+a(8))], lw ); % 270 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-shapspac-a(9)) (height/2-shapspac-a(9)) (width/2-shapspac+a(9)) (height/2-shapspac+a(9))], lw ); % 315 degrees

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %flip to screen

pause(0.5)

%%%%%%%%%

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-a(1)) (height/2-a(1)) (width/2+a(1)) (height/2+a(1))], lw ); %draws random size square in the centre of the screen

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-a(2)) (height/6-a(2)) (width/2+a(2)) (height/6+a(2))], lw ); %0 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2+shapspac-a(3)) (height/2-shapspac-a(3)) (width/2+shapspac+a(3)) (height/2-shapspac+a(3))], lw ); %45 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2+ height/3-a(4)) (height/2-a(4)) ((width/2)+height/3+a(4)) (height/2+a(4))], lw ); %90 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2+shapspac-a(5)) (height/2+shapspac-a(5)) (width/2+shapspac+a(5)) (height/2+shapspac+a(5))], lw ); %135 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-a(6)) ((5\*height)/6-a(6)) (width/2+a(6)) ((5\*height)/6+a(6))], lw ); %180 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-shapspac-a(7)) (height/2+shapspac-a(7)) (width/2-shapspac+a(7)) (height/2+shapspac+a(7))], lw ); %225 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2- height/3-a(8)) (height/2-a(8)) (width/2-height/3+a(8)) (height/2+a(8))], lw ); % 270 degrees

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-shapspac-a(9)) (height/2-shapspac-a(9)) (width/2-shapspac+a(9)) (height/2-shapspac+a(9))], lw ); % 315 degrees

%%%%%%

if z(b(1))==1

Screen('FrameOval',screen1,shapecolour, [(width/2-a(2)) (height/6-a(2)) (width/2+a(2)) (height/6+a(2))], lw ); %0 degrees

elseif z(b(2))==1

Screen('FrameOval',screen1,shapecolour, [(width/2+shapspac-a(3)) (height/2-shapspac-a(3)) (width/2+shapspac+a(3)) (height/2-shapspac+a(3))], lw ); %45 degrees

elseif z(b(3))==1

Screen('FrameOval',screen1,shapecolour, [(width/2+ height/3-a(4)) (height/2-a(4)) ((width/2)+height/3+a(4)) (height/2+a(4))], lw ); %90 degrees

elseif z(b(4))==1

Screen('FrameOval',screen1,shapecolour, [(width/2+shapspac-a(5)) (height/2+shapspac-a(5)) (width/2+shapspac+a(5)) (height/2+shapspac+a(5))], lw ); %135 degrees

elseif z(b(5))==1

Screen('FrameOval',screen1,shapecolour, [(width/2-a(6)) ((5\*height)/6-a(6)) (width/2+a(6)) ((5\*height)/6+a(6))], lw ); %180 degrees

elseif z(b(6))==1

Screen('FrameOval',screen1,shapecolour, [(width/2-shapspac-a(7)) (height/2+shapspac-a(7)) (width/2-shapspac+a(7)) (height/2+shapspac+a(7))], lw ); %225 degrees

elseif z(b(7))==1

Screen('FrameOval',screen1,shapecolour, [(width/2- height/3-a(8)) (height/2-a(8)) (width/2-height/3+a(8)) (height/2+a(8))], lw ); % 270 degrees

elseif z(b(8))==1

Screen('FrameOval',screen1,shapecolour, [(width/2-shapspac-a(9)) (height/2-shapspac-a(9)) (width/2-shapspac+a(9)) (height/2-shapspac+a(9))], lw ); % 315 degrees

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

elseif shape==2

%%%%%%%%%%%%%%%%%%%%%%%%%%%% SQUARE %%%%%%%%%%%%%%%%%%%%%%%%%

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-a(1)) (height/2-a(1)) (width/2+a(1)) (height/2+a(1))], lw ); %draws random size square in the centre of the screen

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-a(2)) (height/6-a(2)) (width/2+a(2)) (height/6+a(2))], lw ); %0 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2+shapspac-a(3)) (height/2-shapspac-a(3)) (width/2+shapspac+a(3)) (height/2-shapspac+a(3))], lw ); %45 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2+ height/3-a(4)) (height/2-a(4)) ((width/2)+height/3+a(4)) (height/2+a(4))], lw ); %90 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2+shapspac-a(5)) (height/2+shapspac-a(5)) (width/2+shapspac+a(5)) (height/2+shapspac+a(5))], lw ); %135 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-a(6)) ((5\*height)/6-a(6)) (width/2+a(6)) ((5\*height)/6+a(6))], lw ); %180 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-shapspac-a(7)) (height/2+shapspac-a(7)) (width/2-shapspac+a(7)) (height/2+shapspac+a(7))], lw ); %225 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2- height/3-a(8)) (height/2-a(8)) (width/2-height/3+a(8)) (height/2+a(8))], lw ); % 270 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-shapspac-a(9)) (height/2-shapspac-a(9)) (width/2-shapspac+a(9)) (height/2-shapspac+a(9))], lw ); % 315 degrees

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %flip to screen

pause(0.5)

%%%%%%

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-a(1)) (height/2-a(1)) (width/2+a(1)) (height/2+a(1))], lw ); %draws random size square in the centre of the screen

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-a(2)) (height/6-a(2)) (width/2+a(2)) (height/6+a(2))], lw ); %0 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2+shapspac-a(3)) (height/2-shapspac-a(3)) (width/2+shapspac+a(3)) (height/2-shapspac+a(3))], lw ); %45 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2+ height/3-a(4)) (height/2-a(4)) ((width/2)+height/3+a(4)) (height/2+a(4))], lw ); %90 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2+shapspac-a(5)) (height/2+shapspac-a(5)) (width/2+shapspac+a(5)) (height/2+shapspac+a(5))], lw ); %135 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-a(6)) ((5\*height)/6-a(6)) (width/2+a(6)) ((5\*height)/6+a(6))], lw ); %180 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-shapspac-a(7)) (height/2+shapspac-a(7)) (width/2-shapspac+a(7)) (height/2+shapspac+a(7))], lw ); %225 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2- height/3-a(8)) (height/2-a(8)) (width/2-height/3+a(8)) (height/2+a(8))], lw ); % 270 degrees

Screen('FrameRect',screen1,bkgrndshapecol, [(width/2-shapspac-a(9)) (height/2-shapspac-a(9)) (width/2-shapspac+a(9)) (height/2-shapspac+a(9))], lw ); % 315 degrees

%%%%%%

if z(b(1))==1

Screen('FrameRect',screen1,shapecolour, [(width/2-a(2)) (height/6-a(2)) (width/2+a(2)) (height/6+a(2))], lw ); %0 degrees

elseif z(b(2))==1

Screen('FrameRect',screen1,shapecolour, [(width/2+shapspac-a(3)) (height/2-shapspac-a(3)) (width/2+shapspac+a(3)) (height/2-shapspac+a(3))], lw ); %45 degrees

elseif z(b(3))==1

Screen('FrameRect',screen1,shapecolour, [(width/2+ height/3-a(4)) (height/2-a(4)) ((width/2)+height/3+a(4)) (height/2+a(4))], lw ); %90 degrees

elseif z(b(4))==1

Screen('FrameRect',screen1,shapecolour, [(width/2+shapspac-a(5)) (height/2+shapspac-a(5)) (width/2+shapspac+a(5)) (height/2+shapspac+a(5))], lw ); %135 degrees

elseif z(b(5))==1

Screen('FrameRect',screen1,shapecolour, [(width/2-a(6)) ((5\*height)/6-a(6)) (width/2+a(6)) ((5\*height)/6+a(6))], lw ); %180 degrees

elseif z(b(6))==1

Screen('FrameRect',screen1,shapecolour, [(width/2-shapspac-a(7)) (height/2+shapspac-a(7)) (width/2-shapspac+a(7)) (height/2+shapspac+a(7))], lw ); %225 degrees

elseif z(b(7))==1

Screen('FrameRect',screen1,shapecolour, [(width/2- height/3-a(8)) (height/2-a(8)) (width/2-height/3+a(8)) (height/2+a(8))], lw ); % 270 degrees

elseif z(b(8))==1

Screen('FrameRect',screen1,shapecolour, [(width/2-shapspac-a(9)) (height/2-shapspac-a(9)) (width/2-shapspac+a(9)) (height/2-shapspac+a(9))], lw ); % 315 degrees

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%% TRIANGLE %%%%%%%%%%%%%%%%%%

elseif shape==3;

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(1) (height/2)+a(1); (width/2) (height/2)-a(1); (width/2)+a(1) (height/2)+a(1) ], lw ); %draws random size square in the centre of the screen

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(2) (height/6)+a(2); (width/2) (height/6)-a(2); (width/2)+a(2) (height/6)+a(2); ], lw ); %0 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(3)+shapspac (height/2)+a(3)-shapspac; (width/2)+shapspac (height/2)-shapspac-a(3); (width/2)+shapspac+a(3) (height/2)+a(3)-shapspac; ], lw ); % 45 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(4)+height/3 (height/2)+a(4); (width/2)+height/3 (height/2)-a(4); (width/2)+(height/3)+a(4) (height/2)+a(4)], lw ); % 90 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)+shapspac-a(5) (height/2)+shapspac+a(5); (width/2)+shapspac (height/2)+shapspac-a(5); (width/2)+shapspac+a(5) (height/2)+shapspac+a(5) ], lw ); % 135 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(6) (5\*height/6)+a(6); (width/2) (5\*height/6)-a(6); (width/2)+a(6) (5\*height/6)+a(6); ], lw ); % 180 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-shapspac-a(7) (height/2)+shapspac+a(7); (width/2)-shapspac (height/2)+shapspac-a(7); (width/2)-shapspac+a(7) (height/2)+shapspac+a(7) ], lw ); % 225 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(8)-height/3 (height/2)+a(8); (width/2)-height/3 (height/2)-a(8); (width/2)-(height/3)+a(8) (height/2)+a(8) ], lw ); % 270 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-shapspac-a(9) (height/2)-shapspac+a(9); (width/2)-shapspac (height/2)-shapspac-a(9); (width/2)-shapspac+a(9) (height/2)-shapspac+a(9); ], lw ); % 315 degrees

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %flip to screen

pause(0.5)

%%%%%%

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(1) (height/2)+a(1); (width/2) (height/2)-a(1); (width/2)+a(1) (height/2)+a(1) ], lw ); %draws random size square in the centre of the screen

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(2) (height/6)+a(2); (width/2) (height/6)-a(2); (width/2)+a(2) (height/6)+a(2); ], lw ); %0 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(3)+shapspac (height/2)+a(3)-shapspac; (width/2)+shapspac (height/2)-shapspac-a(3); (width/2)+shapspac+a(3) (height/2)+a(3)-shapspac; ], lw ); % 45 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(4)+height/3 (height/2)+a(4); (width/2)+height/3 (height/2)-a(4); (width/2)+(height/3)+a(4) (height/2)+a(4)], lw ); % 90 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)+shapspac-a(5) (height/2)+shapspac+a(5); (width/2)+shapspac (height/2)+shapspac-a(5); (width/2)+shapspac+a(5) (height/2)+shapspac+a(5) ], lw ); % 135 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(6) (5\*height/6)+a(6); (width/2) (5\*height/6)-a(6); (width/2)+a(6) (5\*height/6)+a(6); ], lw ); % 180 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-shapspac-a(7) (height/2)+shapspac+a(7); (width/2)-shapspac (height/2)+shapspac-a(7); (width/2)-shapspac+a(7) (height/2)+shapspac+a(7) ], lw ); % 225 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-a(8)-height/3 (height/2)+a(8); (width/2)-height/3 (height/2)-a(8); (width/2)-(height/3)+a(8) (height/2)+a(8) ], lw ); % 270 degrees

Screen('FramePoly',screen1,bkgrndshapecol, [(width/2)-shapspac-a(9) (height/2)-shapspac+a(9); (width/2)-shapspac (height/2)-shapspac-a(9); (width/2)-shapspac+a(9) (height/2)-shapspac+a(9); ], lw ); % 315 degrees

%%%%%%

if z(b(1))==1

Screen('FramePoly',screen1,shapecolour, [(width/2)-a(2) (height/6)+a(2); (width/2) (height/6)-a(2); (width/2)+a(2) (height/6)+a(2); ], lw ); %0 degrees

elseif z(b(2))==1

Screen('FramePoly',screen1,shapecolour, [(width/2)-a(3)+shapspac (height/2)+a(3)-shapspac; (width/2)+shapspac (height/2)-shapspac-a(3); (width/2)+shapspac+a(3) (height/2)+a(3)-shapspac], lw ); % 45 degrees

elseif z(b(3))==1

Screen('FramePoly',screen1,shapecolour, [(width/2)-a(4)+height/3 (height/2)+a(4); (width/2)+height/3 (height/2)-a(4); (width/2)+(height/3)+a(4) (height/2)+a(4)], lw ); % 90 degrees

elseif z(b(4))==1

Screen('FramePoly',screen1,shapecolour, [(width/2)+shapspac-a(5) (height/2)+shapspac+a(5); (width/2)+shapspac (height/2)+shapspac-a(5); (width/2)+shapspac+a(5) (height/2)+shapspac+a(5) ], lw ); % 135 degrees

elseif z(b(5))==1

Screen('FramePoly',screen1,shapecolour, [(width/2)-a(6) (5\*height/6)+a(6); (width/2) (5\*height/6)-a(6); (width/2)+a(6) (5\*height/6)+a(6); ], lw ); % 180 degrees

elseif z(b(6))==1

Screen('FramePoly',screen1,shapecolour, [(width/2)-shapspac-a(7) (height/2)+shapspac+a(7); (width/2)-shapspac (height/2)+shapspac-a(7); (width/2)-shapspac+a(7) (height/2)+shapspac+a(7) ], lw ); % 225 degrees

elseif z(b(7))==1

Screen('FramePoly',screen1,shapecolour, [(width/2)-a(8)-height/3 (height/2)+a(8); (width/2)-height/3 (height/2)-a(8); (width/2)-(height/3)+a(8) (height/2)+a(8) ], lw ); % 270 degrees

elseif z(b(8))==1

Screen('FramePoly',screen1,shapecolour, [(width/2)-shapspac-a(9) (height/2)-shapspac+a(9); (width/2)-shapspac (height/2)-shapspac-a(9); (width/2)-shapspac+a(9) (height/2)-shapspac+a(9); ], lw ); % 315 degrees

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

FlushEvents('keyDown');

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %flip to screen

KbName('UnifyKeyNames'); %Used To Unify keyboard keys across computers

startSecs = GetSecs; %read time (absolute)

while ~KbCheck;

end % wait for a key press

[~, Time, keyCode ] = KbCheck; %Store details about the keypressed

RelTime = Time -startSecs; % Compute time difference

Screen('PutImage', screen1, 0, rect); % clear screen to black

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %flip to screen

for i=1:8 %for 1 to 8 (each surrounding square)

if z(b(i))==1; %if square is the coloured one

halfwidthcol=a(i+1); %then assign size of coloured square to halfwidthcol (i+1, to ignore first, centre square)

end

end

if halfwidthcol>a(1)&&keyCode(38)==1; %if larger and correct, increase score by one

crrct=crrct+1;

elseif halfwidthcol<a(1)&&keyCode(40)==1; % if smaller and correct, increase score by one

crrct=crrct+1;

end

totaltime(trials)=RelTime; %assign each reaction time to different element of the array, totaltime

pause(1); %pause for one second (change to waitsecs????)

end

avtime=mean(totaltime); %assign the average time taken to a label

Screen('TextFont',screen1, 'Courier New'); %Choose font

Screen('TextSize',screen1, fs ); %Choose font size

Screen('DrawText', screen1, (['you got ' num2str(crrct) ' out of ' num2str(numoftrials) ' correct']), width/10, height/10, [240 248 255]);%Draw text eg(you got 4 out of 5 correct)

Screen('DrawText', screen1, (['your average reaction time was ' num2str(avtime) ' seconds']), width/10, (height/10)+100, [240 248 255]);%Draw text eg( your average reaction tme was 2 seconds)

Screen('DrawText', screen1, ('Press any key'), width/10, (height/10)+300, [240 248 255]);%Draw text eg( your average reaction tme was 2 seconds)

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %Flip onto screen

KbWait; %wait for keypress

pause(0.5)

end



function [shapetype, shapecolour, colourtext] = displayinfo(shape, colour)

%displayinfo.m

%assigns variables for displayinstructions.m and displaystim.m

if shape==1; % for each shape

shapetype='circle'; %assign string info for instructions

elseif shape==2

shapetype='square'; %assign string info for instructions

elseif shape==3;

shapetype='triangle'; %assign string info for instructions

end

if colour==1 %for each colour

shapecolour=[255 0 0]; %set colour of shape to red

colourtext='red'; %assign string info for instructions

elseif colour==2

shapecolour=[0 255 0];%set colour of shape to green

colourtext='green';%assign string info for instructions

elseif colour==3

shapecolour=[0 0 255]; %set colour of shape to blue

colourtext='blue';%assign string info for instructions

end

end



function displayinstructions(blox, stim\_counter, screen1, fs, width, height, shapetype, colourtext)

%displayinstructions.m

% displays information to the subject

if blox==1&&stim\_counter==1; %if first block, display text

Screen('TextFont',screen1, 'Courier New'); %Choose font

Screen('TextSize',screen1, fs ); %Choose font size

Screen('DrawText', screen1, ('The main experiment is about to begin'), width/100, height/100, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Please read the instructions carefully.'), width/100, height/100+40, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Press any key to continue'), width/100, height/100+100, [240 248 255]);%Draw text

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %Flip onto screen

elseif blox==1&&stim\_counter~=1; %if first block of a new stimulus type, display text

Screen('TextFont',screen1, 'Courier New'); %Choose font

Screen('TextSize',screen1, fs ); %Choose font size

Screen('DrawText', screen1, ('Thankyou, please take a break'), width/100, height/100, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('The stimulus will now change,'), width/100, height/100+40, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('please read the instructions carefully.'), width/100, height/100+60, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Press any key to continue'), width/100, height/100+100, [240 248 255]);%Draw text

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %Flip onto screen

end

KbWait;%wait for subject response

pause(0.3);

if blox>1; %if not the first block, display text

Screen('TextFont',screen1, 'Courier New'); %Choose font

Screen('TextSize',screen1, fs ); %Choose font size

Screen('DrawText', screen1, ('Thankyou, please take a short break before the next block.'), width/100, height/100, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Press any key when ready'), width/100, height/100+40, [240 248 255]);%Draw text

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %Flip onto screen

KbWait;

pause(0.3);

end

Screen('TextFont',screen1, 'Courier New'); %Choose font

Screen('TextSize',screen1, fs ); %Choose font size

Screen('DrawText', screen1, ('Instructions:'), width/100, height/100, [240 248 255]);%Draw text

Screen('DrawText', screen1, (['Please look at the centre ' shapetype ',']), width/100, height/100+40, [240 248 255]);%Draw text

Screen('DrawText', screen1, (['Press up or down when you see the ' colourtext ' ' shapetype ' around it']), width/100, (height/100)+60, [240 248 255]);%Draw text

Screen('DrawText', screen1, (['Please press up, if the ' colourtext ' ' shapetype ' is larger,']), width/100, (height/100)+90, [240 248 255]);%Draw text

Screen('DrawText', screen1, (['Press down if the ' colourtext ' ' shapetype ' is smaller']), width/100, (height/100)+110, [240 248 255]);%Draw text

Screen('DrawText', screen1, ('Press any key when ready'), width/100, (height/100)+180, [240 248 255]);%Draw text

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %Flip onto screen

KbWait;

pause(0.5);

Screen('TextFont',screen1, 'Courier New'); %Choose font

Screen('TextSize',screen1, fs ); %Choose font size

Screen('DrawText', screen1, ('Get Ready'), width/10, height/3, [240 248 255]);%Draw text

Screen('DrawingFinished', screen1); %tell PTB that no further drawing commands fill follow before flip

Screen(screen1, 'Flip'); %Flip onto screen

pause(1);

end



function writeresults(subject\_info,exp\_info, results\_correct, results\_avtime)

%writeresults.m

% writes subject data and results to an xls file named after the subject

[subject\_titles, exp\_titles] = title\_names; %calls function to get titles for the subject and experiment information for excel

[resultsx, resultsy] = results\_titles; %calls function to get titles for the experimental data for excel

xlswrite([subject\_info{2} num2str(subject\_info{6})],subject\_titles,1,'A1'); %write subject titles information to excel

xlswrite([subject\_info{2} num2str(subject\_info{6})],subject\_info,1,'A2'); %write subject information to excel sheet 1

xlswrite([subject\_info{2} num2str(subject\_info{6})],exp\_titles,1,'A4'); %write experimental titles information to excel

xlswrite([subject\_info{2} num2str(subject\_info{6})],exp\_info,1,'A5'); %write experimental information to the same sheet in excel sheet 2

xlswrite([subject\_info{2} num2str(subject\_info{6})],resultsx,2,'A1'); %write results titles for each block to excel

xlswrite([subject\_info{2} num2str(subject\_info{6})],resultsy,2,'A2'); %write results titles for each block to excel

xlswrite([subject\_info{2} num2str(subject\_info{6})],results\_correct,2,'B2'); %write number of correct results for each block to excel sheet 3

xlswrite([subject\_info{2} num2str(subject\_info{6})],resultsx,3,'A1'); %write results titles for each block to excel

xlswrite([subject\_info{2} num2str(subject\_info{6})],resultsy,3,'A2'); %write results titles for each block to excel

xlswrite([subject\_info{2} num2str(subject\_info{6})],results\_avtime,3,'B2'); %write reaction times for each block to excel

end



function [subject\_titles, exp\_titles] = title\_names

%writes cells that contain the titles for the excel sheets

subject\_titles{1}='subject number'; %sets the titles of the subjects info

subject\_titles{2}='subject name';

subject\_titles{3}='subject age';

subject\_titles{4}='subject gender';

subject\_titles{5}='subject handedness';

subject\_titles{6}='session number';

exp\_titles{1}='number of blocks'; % sets the titles of the experiment info

exp\_titles{2}='number of trials';

exp\_titles{3}='colours';

exp\_titles{4}='shapes';

exp\_titles{5}='practice y/n';

exp\_titles{6}='full screen y/n';

end



function [resultsx, resultsy] = results\_titles

%writes titles for the excel information for reaction times

resultsx{1}='block'; % information accross the top of the results sheet

resultsx{2}=1;

resultsx{3}=2;

resultsx{4}='etc';

resultsy1{1}='red circles'; %information for down the side of the results sheet

resultsy1{2}='red squares';

resultsy1{3}='red triangles';

resultsy1{4}='green circles';

resultsy1{5}='green squares';

resultsy1{6}='green triangles';

resultsy1{7}='blue circles';

resultsy1{8}='blue squares';

resultsy1{9}='blue triangles';

resultsy=resultsy1'; %transposes resultsy (to make column vector)

end

List of Variables:

Stimulus Display Variables:

sizeofscreen : A 4x1 vector representing the size of the screen

numoftrials : Number of trials to be displayed in each block

bkgrnshapecol : Colour vector for the colour of the centre, and the background (non tested) shapes

fs : Size of text for instructions and subject information

lw : width of shape line for stimulus

colour: an integer passed to the displaystim function corresponding to the colour to be displayed for that block

shape: an integer passed to the displaystim function corresponding to the shape to be displayed for that block

rect : vector of screen dimensions

screen1 : screen number for display of stimulus

Result Variables:

num\_correct : number of correct responses from a stimulus block

avtime : average response time from each stimulus block

blk\_res\_correct : vector of the correct responses from all the blocks of one stimulus condition

blk\_res\_avtime : vector of the average response times from all the blocks of one stimulus condition

Exported Variables:

results\_correct : array of all the correct responses from all the blocks of all stimulus conditions

results\_avtime : array of all the average response times from all the blocks of all stimulus conditions

subject\_info : 1x6 cell array of subject information

exp\_info 1x6 array of experiment information

Titles for the information when exported to excel:

subject\_titles

exp\_titles

resultsx

resultsy

Other Variables:

trialconds : 1x9 vector of integers between 0 and 2, representing whether or not a stimulus condition should be displayed

rect: screen dimensions

colours : 1x9 vector of integers, corresponding to colour information for each stimulus condition

shapes : 1x vector of integers, corresponding to colour information for each stimulus condition

stim\_counter : count of how many different stimulus conditions have been displayed

Input Data:

Experimental information and subject information must be entered by the user, along with responses from the subject.

Processing:

In the displaystim function, it is required that the tested shape and the central shape are of different sizes, so that one is bigger than the other.

a=((randperm(9))\*2)+10; %assign random permutation of shape sizes from 12 to 28 to a

%also makes sure that the test shape is a different size than the centre

%shape, as they are both selected from the same selection of sizes

Screen('FrameOval',screen1,bkgrndshapecol, [(width/2-a(1)) (height/2-a(1)) (width/2+a(1)) (height/2+a(1))], lw ); %draws random size square in the centre of the screen

In the input argument of the screen function, the call to a(1) selects a random number from a, which forms the basis of the size of the shape.

Output Data:

The program displays instructions to the user, as well as information between blocks, and a summary of their results after each block. It displays the stimulus once for each trial, and exports all the results, along with subject and experimental information to a Microsoft Excel file.

Computer Environment:

Hardware: Dell Desktop, with Intel Pentium chip.

Software: Matlab v7.10 with Psychophysical toolbox.

Verification:

The main script was run several times, with different experimental variables, such as different numbers of blocks and trials. All permutations of stimuli were displayed, and the program was run with and without practice sessions and also with and without development mode. It was confirmed that the instructions and the stimulus were displayed in every case, and that the subject information, experimental information and the results were exported correctly to Excel.

The program only allows the shapes to be circles, squares or triangles, and only permits the test colour to be red, green or blue. The shapes can be altered by changing

the displaystim function. The test colours can be easily changed by altering the shapecolour vector, other variables can also be altered easily, such as line width, font size, and background shape colour.

User Guide

To run the program, open MATLAB (any recent version with Psychtoolbox installed), and open the script shape\_size\_main\_program.m, using the debug menu, run the program. The program displays full instructions to the user, about what information to enter, and also full instructions to the subject about what to do. The results will be saved as a Microsoft Excel file named after the subject name and subject number, in the same folder as the program.