The Swedish National Core Facility for Magnetoencephalography (NatMEG)

Presentation code template

Resting state experiment

Author

Judith Recober

January 16, 2025

Head of the core facility

Christoph Pfeiffer

1 Introduction

This provided code example is for a resting state recording. The code presents a fixation cross while the MEG system records the brain activity. This experiment has 2 blocks: one where the participant needs to be with the eyes open and another where the participant needs to be with the eyes closed.

The presentation code is used to show and control the experiment. For programming a single experiment, different files are needed:

- The main PCL code that will control all the experiment.
- The subroutines PCL file where some functions are defined
- The EXP file where the software is connected with the hardware
- The SC file where the different stimulus are defined

2 Complementary files

2.1 PCL subrutines

Presentation, as most of any other programming language, has predetermined functions that we can use. For example:.present(); will display something, perhaps an image, on the screen.

However, in this code, you will notice that other functions used are not from presentation per se. There is a .pcl file in the same folder that has different defined functions that will be used in this main code. These codes provide what are called "subroutines", which are a certain sequence of operations which form a whole function and that are put together forming a little module. In this case, the functions are:

- **kss:** In our code we use kss() to show the sleepiness questions to the participant.
- **ShowInstructions:** We use ShowInstructions() to show text instructions to the participant on the screen.
- **ShowAltText:** We use ShowAltText() to show a certain text on our screen (not the participant's screen).
- **WaitForAllButtonpress:** Used to wait for any button press in the keyboard to continue with the code.

It is always good practice to organize the code in this way: main code + functions/subroutines. In this way, if there is a set of instructions that you know you will use repeatedly in your code/codes, you can create a function to perform that task and then call the function. For example: during my experiment I will need to evaluate the sleepiness of my participant at least 3 times, then it is a good opportunity to write a function that you can re-use.

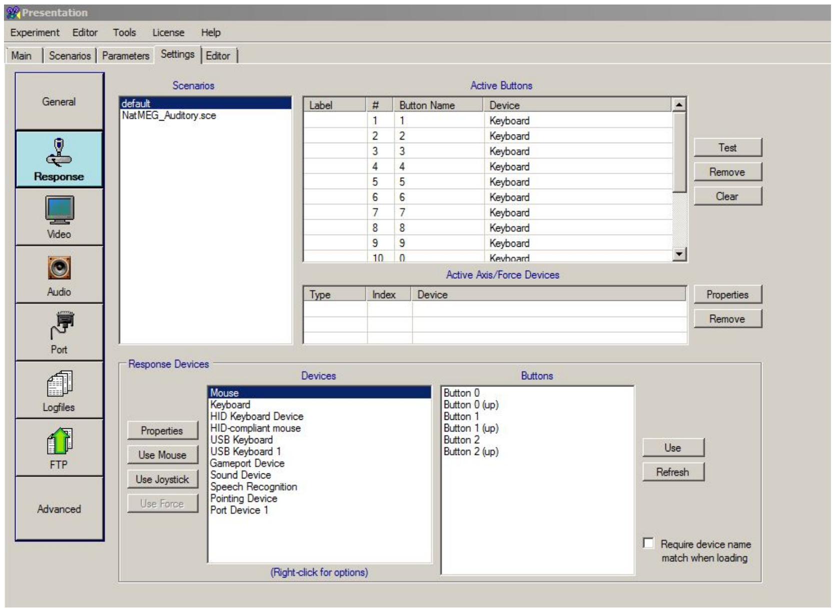
2.2 EXP files

he experiment file is a file that helps to establish connections between the software and the hardware. The main examples are:

1. Define the available buttons from the keyborad: The number of buttons and the type of button that the user

can click on the keyboard for the code to identify them will be defined in this file. Each button that the user can press on the keyboard will be assigned to a number. For example, if the user clicks the button for number one 1 on the keyboard (that is, the button name), that will be button 1 in the code. If the user clicks on enter on the keyboard, that will be button 12 in the code (see figure 1.

2. Defining the input and output ports. The number and type of input and output ports needs to be defined. Each port will be again associated with a number. In our case we have a card with ports which is called PCle-6509 and we know that port0 slot in that card corresponds to the MEG port. Thus, we will assign this port with a number, in this case number 1, and in the code we will use that number to refer to the MEG-port (see figure 2.



Scenarios

default
NatMEG_Auditory.sce

Active Buttons

Label	#	Button Name	Device
	1	1	Keyboard
	2	2	Keyboard
	3	3	Keyboard
	4	4	Keyboard
	5	5	Keyboard
	6	6	Keyboard
	7	7	Keyboard
	8	8	Keyboard
	9	9	Keyboard
	10	0	Keyboard

Active Axis/Force Devices

Type	Index	Device

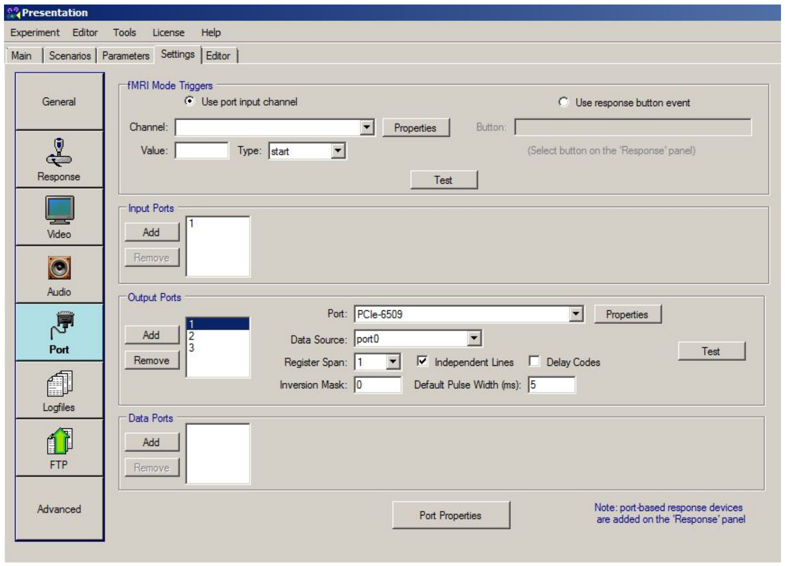
Response Devices

Devices: Mouse, Keyboard, HID Keyboard Device, HID-compliant mouse, USB Keyboard, USB Keyboard 1, Gameport Device, Sound Device, Speech Recognition, Pointing Device, Port Device 1

Buttons: Button 0, Button 0 (up), Button 1, Button 1 (up), Button 2, Button 2 (up)

☐ Require device name match when loading

Figure 1: mapping of the buttons in the EXP file



fMRI Mode Triggers
[✓ Use port input channel] Use response button event
Channel:
Value:
Type: start
Button:
(Select button on the 'Response' panel)

Input Ports: 1

Output Ports: 1, 2, 3
Port: PCle-6509
Data Source: port0
Register Span: 1
☑ Independent Lines
☐ Delay Codes
Inversion Mask: 0
Default Pulse Width (ms): 5

Data Ports

Note: port-based response devices are added on the 'Response' panel

Figure 2: Ports definition

2.3 SCE code

PCL codes are used to control your experiment in general; Blocks that it will have, when I am going to present something, send triggers to the recording systems, among others. SCE files are used to define certain stimulus that afterwards will be used and displayed in the PCL code. For example: image, video, sound, and others.

First, is also important that you notice some variables used for default settings of text and background colors, available buttons and including files are needed for the code:

```
scenario = "CAPSI_resting_state";
pcl_file = "CAPSI_resting_state.pcl";

default_background_color = 0,0,0; # black
default_text_color = 255,255,255; # white

default_font = "arial";
default_font_size = 48;
default_text_align = align_left;

This are the buttons from the keybord to be inluded
active_buttons = 15;
button_codes = 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15;
```

2.3.1 Fixation cross image

If you go to the SCE file you will see the following:

```
#fixation cross (in color black)
picture {
 box { height = 5; width = 60; color = 255,255,255; } b_Horiz; x = 0; y = 0;
 box { height = 60; width = 5; color = 255,255,255; } b_Verti; x = 0; y = 0;
} p_Fixation;
```

In this case we define a stimulus which is a picture. The default background color is black and on top of our background we will show this picture. This picture consists on 2 rectangles, which are the same but oriented differently (notice that height and width are switched), thus forming a cross. Both of the rectangles are white so the cross is white.

It is important to notice that "picture " in the beginning defines the type of stimulus that your will define and " p-Fixation;" in the end is the name that you give to this stimulus so that afterwards you can call it in the main PCL code. See the presentation manual for further explanation on how to define different stimulus.

2.3.2 Text in the screen

To show text in the screen, we have a function in the subroutines PCL file and also a stimulus defined in this SCE code. These two functions are connected to show text in the screen in the main PCL file. Now we will explain the subroutines function.

ShowAltText is a function defined in the subrountines file:

```
sub ShowAltText( string messageA, string messageB, string messageC )
begin
  t_Alt3La.set_caption( messageA );
  t_Alt3La.redraw();
  t_Alt3Lb.set_caption( messageB );
  t_Alt3Lb.redraw();
  t_Alt3Lc.set_caption( messageC );
  t_Alt3Lc.redraw();
  p_Alt3L.present();
end;
```

This funtion always needs 3 inputs (3 strings to be more specific). Thus, you always need to call the function as: ShowAltText("something", "something", "something");

Then, notice that in the SCE file there is this code:

```
#alt screen (3 lines)
picture {
 display_index = 2;
 text { display_index = 2; caption = " "; } t_Alt3La; x = 0; y = 80;
 text { display_index = 2; caption = "Press [ENTER] to confirm or [Esc] to abort. "; }
        t_Alt3Lb; x = 0; y = 0;
 text { display_index = 2; caption = " "; } t_Alt3Lc; x = 0; y = -80;
} p_Alt3L;
```

This stimulus is a picture, however, instead of having 2 rectangles, it has text. The text that will be shown is: t-Alt3La, t-Alt3Lb and t-Alt3Lc. These three text variables are the same that were defined in the ShowAltText() function (they have the same names). Thus, the ShowAltText() function receives as inputs 3 strings that will be used to define the text string that our stimulus will use to present.

To show instructions to the participant, we also use a very similar method but with a different function and different stimulus. This is the function in the subroutines folder:

```
sub ShowInstructions( string messageA, string messageB, string messageC )
begin
  t_Info3La.set_caption( messageA );
  t_Info3La.redraw();
  t_Info3Lb.set_caption( messageB );
  t_Info3Lb.redraw();
  t_Info3Lc.set_caption( messageC );
  t_Info3Lc.redraw();
  p_Info3L.present();
  WaitForAllButtonpress();
end;
```

And this is the stimulus:

```
And this is the stimulus:
#info screen (3 lines)
picture {
 text { caption = " "; } t_Info3La; x = 0; y = 80;
 text { caption = "Press [ENTER] to confirm or [Esc] to abort. "; } t_Info3Lb; x = 0; y = 0;
 text { caption = " "; } t_Info3Lc; x = 0; y = -80;
} p_Info3L;
```

The main difference is that by default, this text will be shown in the participant's screen. When we need text in our screen we need to specify display-index = 2; as we did in the stimulus for showing text in our screen.

3 Main PCL file

3.1 Initial parameters

You need to think about what parameters are important for the interface of your experiment.

- First of all, if you want to show any text instructions you may want to have different language options.
- Then, in order to run your experiments, you probably will need extra files. It is good practice that you split your code in different parts and write a "main" file that helps you to control all the experiment.
- Ports: Presentation can handle different external devices. Ports can be used to send data markers / triggers so that stimuli events can be synchronized from your Presentation script with other devices, for example a MEG recording. When a marker is sent from Presentation, recorded time is synchronized with the external device. A data marker is defined by a number ranging between 1 and 255. This marker is sent to your data recording system, and represents a single data point within your recorded data. Every event that you are interested in synchronizing within your Presentation script can be associated with an individual marker, such that you can easily process your data after recording. In this case the output ports where we want to send triggers for data synchronization are: the MEG port, the OPM port (so both MEG systems we are currently studying).

```
##################### PARAMETER THAT CAN BE CHANGED BETWEEN EXPERIMENTS ######################
# Add parameter that you may want to change between experiments, so that you get easy access to them
string language = "eng"; #language can be "eng" or "sv"

##################### FILES THAT NEED TO BE INLCUDED ######################
# First of all, we need to include all the necessary files that will be used in this experiment
include "subroutines.pcl";

begin; #beggining of our experiment

##################### PORTS DEFINITIONS ######################
# in the app to manage the exeriments, there are output ports. We need to know this
# so that the output of the code goes to the correct output port of the system
int myMEGport = 1;
output_port MEGport = output_port_manager.get_port(myMEGport);
output_port OPMport = output_port_manager.get_port(3);

default.present(); #this function presents the settings in the screen.
parameter_window.remove_all(); #This function removes any previously configured parameters
#from the parameter window to start again a new experiment
```

3.2 Prepare your experiment

In order to start running the experiments, you may need to define different variables like strings or lists or counters.

In this experiment, we mainly want to show a fixation cross while recording the user in this resting state. We need to repeat this for 2 blocks: one where the participant needs to be with the eyes open, and another one with the eyes closed. The main variable we need to define in this case are:

(1) exp-duration: control the total length of each block (2) RestingStates: It is a string type of array where the first column is used so that WE know which block are we recording and the second column is for the participant to follow the corresponding instructions depending on the block type.

Normally, it is useful to have an array type of variable so that you control which block are you recoding and thus, also show the correspoding instructions to the participant. If you do not run different blocks (for example, you just do eyes closed), then, this is not necessary.

```
##################### PREPARING YOUR EXPERIMENT ######################
int exp_duration = 30; # Total time: 2 blocks, 5 minutes per block (300 seconds each)
```

```
## Add definitions of resting state eyes open and eyes closed


array<string> RestingStates[2][2];  # create empty array with two rows and 2 col


# THIS IS FOR US TO KNOW WHICH BLOCK ARE WE IN [X][1] (first column)
RestingStates[1][1] = "Eyes open";  # THIS IS FOR US TO KNOW WHICH BLOCK ARE WE IN
RestingStates[2][1] = "Eyes closed";


# THIS IS FOR THE PARTICIPANT TO SHOW THE CORRESPONDING INSTRUCTIONS [X][2] (second column)
if language == "sv" then
 RestingStates[1][2] = "ögonen öppna under tiden, titta på korset,\n och sitt stilla och avslappnat";
 RestingStates[2][2] = "ögonen stängda och sitt stilla och avslappnat";
else
 RestingStates[1][2] = "eyes open meanwhile, look at the cross,\n and sit still and relaxed";
 RestingStates[2][2] = "eyes closed and sit still and relaxed";
end;
```

3.3 Main experiment loop

so the main structure of the code for the 2 blocks of this experiment is the following:

1. loop through the 2 blocks (eyes open and eyes closed)
   (a) show the corresponding instructions in the corresponding language
   (b) present the fixation cross
   (c) wait 5 seconds before the trigger is send to the MEG and OPM ports
   (d) show text in our screen for the progress
   (e) send the triggers to the ports
      i. loop until exp-duration per each block and show the progress and send sync triggers
   (f) send the "off" trigger to the ports
   (g) show the kss questionaire at the end of the block

```
###>> Resting state block ***************************************************************
loop int block=1; until block > 2 begin;


 if language == "sv" then
  ShowInstructions(" Under kommande sex minuter kommer vi att mäta aktiviteten \n\n",
        "medans du vilar och gör ingenting särskilt \n\n", "Håll " + RestingStates[block][2]);
 else
```

```
  ShowInstructions(" For the next six minutes we will measure the activity \n\n",
        "while resting and doing nothing in particular\n\n", "Hold on" + RestingStates[block][2]);
 end

 p_Fixation.present(); #present fixation cross and wait for a button press
 wait_interval(5000); #add 5s before sending trigger

 ShowAltText("Recording:", "resting state", RestingStates[block][1]); # show progress for us
 MEGport.send_code(16,40); # send "on" trigger
 OPMport.send_code(16,40); # send "on" trigger
 loop int dsec=1; until dsec > exp_duration begin; # 30*10 sec = 5 min
  wait_interval(10000);   #10 sec
  MEGport.send_code(64,40); # send sync trigger
  OPMport.send_code(64,40); # send "sync trigger

  ShowAltText(" Block " + string(block), RestingStates[block][1], " status: " + string(dsec)
        + " / " + string(exp_duration)); # show progress for us

  dsec=dsec+1;
 end;
 MEGport.send_code(32,40); # send "off" trigger
 OPMport.send_code(32,40); # send "off" trigger
 ShowAltText("Finished:", "resting state", RestingStates[block][1]);

 wait_interval(5000);

 kss_routine("eng");
 block = block + 1;
end; # block loop
#kss_routine("eng");

endOfTest.present();
end;
```

4 Recap

So in general, when you need to do a presentation code for an experiment you need to worry about:

- Variables that you may want to change in different experiments with different times/participants: example:

language of instructions, a variable that controls if you want to ask or not some questions depending on the participant...

- ports and triggers to the MEG and OPM and maybe others.
- Instructions: This is a very important part since you need to make sure the participant understands the task. Think about the language, variables to store your instructions and use some functions (you can use the ones we are already using in this code example) to show the instructions in the screen.
- Stimuli: In this case, our only stimuli was a fixation cross on the screen. Thus, this image was defined in the .sce code. However, in other experiments you may want to set others like: sound, video, tactile sensation...
- Experiment structure: Think about how do you want to structure your experiments for your recordings. For this, we normally use loop structures to repeat different blocks (maybe each block has different conditions). Also, think about variables like counters or strings to store data and to control your experiment.

Normally the experiments also follow a similar pattern:

1. Define all your ports, include subroutines, initialize (and if needed define) the variables that you will need in your code
2. Prepare the first loop over the blocks of your entire experiment session
3. show instructions to the participant
4. send on triggers
5. loop until the end of your experiment lenght while you send sync triggers and show the progress in your screen
6. triggers off and finish
7. kss