Programming – course 4
02/01/2014
Martin SZINTE

Using the Psychtoolbox (PART 1/2)

Plan

1. Installing de la Psychtoolbox (PTB)
2. What is the PTB?
3. The Screen function
   - Testing the PTB
   - Screen
   - Crash of the PTB
4. Help and demos of the PTB
   - Help section
   - PsychDemos
5. Essential functions of the PTB
   - FillRect / FrameRect
   - DrawLine / FillPoly
   - MakeTexture / DrawTexture
   - FillOval / FrameOval / DrawDots
   - DrawText
6. Control stimuli duration and creating moving objects
   - Flip duration
   - Creating moving object
7. Use of keyboard and mouse
   - Use of the keyboard
   - Use of the mouse

1. Installing the Psychtoolbox (PTB)

In order to install the "Psychtoolbox" you are invited to follow the clear and detailed instructions given at the following address:
http://psychtoolbox.org/wikka.php?wakka=PsychtoolboxDownload

Remark 1:
The installation of the Psychtoolbox can nowadays be made for both 32-bits and 64-bits Matlab and computers. However select wisely the installation you need in function of the different toolboxes you might use (e.g. the Eyelink toolbox works only in Matlab 32-bits under Windows).

2. What is the PTB?

The PTB is an agglomerate of Matlab files (.m files) and Matlab executable files (.mex files) written in C and C++ allowing with a little bit of practice to program more or less complex experiments assisted by computers.

This toolbox has several advantages:

- It is very flexible and allow the realization of almost all kind of stimuli (audio/visual)
- It is used by a large community of users and developers
- It is open-source and free
- The help support and the use of the FAQ is free and frequently well detailed
- A web hotline exist in case of severe problems or lack

- It is a toolbox documented by a peer-review method article. Indeed you shouldn't forget to cite the work of the Psychtoolbox initiators in your future work.

Pelli, D. G. (1997). The Video Toolbox software for visual psychophysics: Transforming numbers into movies. *Spatial Vision*, *10*, 437-442.
Brainard, D. H. (1997). The Psychophysics Toolbox. *Spatial Vision*, *10*, 433-436.

More information concerning such toolbox can be found on its website: http://psychtoolbox.org/

Remark:
This toolbox is principally used by vision scientists however together with a few built-in functions of Matlab it allows to develop any kind of experiment assisted by computers. It therefore perfectly suit experiments in language science, audition or emotion research for example…
Moreover coupled to other toolboxes the PTB allows the development of experiment using EEG, MEG, fMRI, eye and body tracking and way more.

3. The SCREEN function

TESTING THE PTB

A good way to start using the PTB is to run the "ScreenTest" function.
Such function will launch simple tests evaluating the correctness of the PTB installation.

=> ScreenTest
Your screen should suddenly turn to black and something like this should be printed in the command window.

```
***** ScreenTest: Testing Screen 0 *****

PTB-INFO: This is Psychtoolbox-3 for Apple OS X, under Matlab 64-Bit (Version 3.0.10 - Build date: May 22 2013).
PTB-INFO: Type 'PsychtoolboxVersion' for more detailed version information.
PTB-INFO: Most parts of the Psychtoolbox distribution are licensed to you under terms of the MIT License, with
PTB-INFO: some restrictions. See file 'License.txt' in the Psychtoolbox root folder for the exact licensing conditions.

PTB-INFO: Deficient Apple OS/X 10.7 or later detected: Would use fragile CoreVideo timestamping as fallback,
PTB-INFO: if beamposition timestamping would not work. Will try to use beamposition timestamping if possible.

PTB-INFO: OpenGL-Renderer is NVIDIA Corporation :: NVIDIA GeForce GT 650M OpenGL Engine :: 2.1 NVIDIA-8.18.22 310.40.05f01
PTB-INFO: Renderer has 1024 MB of VRAM and a maximum 1024 MB of texture memory.
PTB-INFO: VBL startline = 900 , VBL Endline = 925
PTB-INFO: Measured monitor refresh interval from beamposition = 16.736159 ms [59.750865 Hz].
PTB-INFO: Will use beamposition query for accurate Flip time stamping.
PTB-INFO: Measured monitor refresh interval from VBLsync = 16.736077 ms [59.751158 Hz]. (50 valid samples taken, stddev=0.052363
ms.)
PTB-INFO: Small deviations between reported values are normal and no reason to worry.
PTB-INFO: Support for fast OffscreenWindows enabled.

***** ScreenTest: Done With Screen 0 *****
```

If the toolbox was incorrectly installed you shouldn't face such message.
In such case copy the error message to Google and look for solutions (it is likely that other users faced your problems and that a simple solution exists).

SCREEN

Let's try now some basics of the PTB…
We will first open a full-screen window and progressively change the background color following Matlab colormap.
Look for "testScreen.m" in the material folder, add it to your path and execute it through the command window.
=> testScreen(1);

```
function testScreen(my_colorMap)
% ----------------------------------------------------------------------
% testScreen(my_colorMap)
% ----------------------------------------------------------------------
% Goal of the function :
% Display changing background colors on a full-screen
% ----------------------------------------------------------------------
% Input(s) :
% my_colorMap : set of colors of Matlab
%               if ==1 : winter colors
%               if ==2 : summer colors
%               if ==3 : spring colors
%               if ==4 : autumn colors
%               if ==5 : cool colors
%               if ==6 : jet colors
%               if ==7 : hsv colors
%               if ==8 : hot colors
% ----------------------------------------------------------------------
% Output(s):
% (none)
% ----------------------------------------------------------------------
% Function created by Martin SZINTE (martin.szinte@gmail.com)
% Last update : 01 / 01 / 2014
% Project : Programming courses
% Version : -
% ----------------------------------------------------------------------

% selection of a colormap
switch my_colorMap
    case 1;matCol = winter(200);
    case 2;matCol = summer(200);
    case 3;matCol = spring(200);
    case 4;matCol = autumn(200);
    case 5;matCol = cool(200);
    case 6;matCol = jet(200);
    case 7;matCol = hsv(200);
    case 8;matCol = hot(200);
end
close all                       % the use of the colormap function open by default a figure
window that we close here

scrAll = Screen('Screens');     % function sending back a list of screen number corresponding
to the number of monitor plugged to your computer
scrNum = max(scrAll);           % defines the hightest monitor number as the drawing monitor

HideCursor;                     % hides the mouse cursor
Screen('Preference', 'SkipSyncTests', 1);    % this line allows approximate timing - to use if
you do not use a physical monitor but just the screen of your laptop
[scr] = Screen('OpenWindow',scrNum,[],[]);   % function opening a window in full-screen mode

priorityLevel = MaxPriority(scr);Priority(priorityLevel);  % gives maximum priority to our
program (limit background program e.g. antivirus)

% drawing loop
for rep = 1:200
    for timeFlip = 1:10
        Screen('FillRect',scr,matCol(rep,:)*255);  % draws a rectangle of full-scren size with
a specified color
        Screen(scr, 'Flip');                       % flips the rectangle on the monitor
    end
end

ShowCursor;          % shows back the mouse cursor
Screen('CloseAll');  % closes the full-screen window

end
```

The screenTest() function uses the Screen() function of the PTB.
Screen() is the core of the PTB toolbox. Through the different input arguments that you would give it, it will do different things such as drawing rectangles or other forms, determining the monitor size or resolution, playing a video or saving screenshots.
To do so, you should define (as chain of characters) the desired sub function you wish.
In the following of this course we will try some of the sub function of Screen().

Question 1: What happen if you enter screenTest(6) in the command window?

By changing the input of the function screenTest(), you trigger a different set of color changes. Select a number between 1 and 8 in order to display one of the 8 different color sets.

Question 2: What is the role of Screen(scr,'Flip') ?

The PTB works with as double buffering window system implying that you should before displaying anything on the screen draw your object on a second virtual window before flipping it to the real one.

CRASH OF THE PTB

When you'll code your own full-screen functions, you might experience problems and crashes. Such crashes will block your computer in full-screen mode giving you no clue of how to stop or close it. To escape the full-screen mode you should use the following 3 steps procedure:

PC: => press **ALT + TAB**
MAC: => press **COMMAND + ZERO**
This first step brings your cursor back to the command window of Matlab.

PC/MAC => press **CTRL + C**
This second step kills any ongoing process (if the crash did not already kill them).

PC/MAC => type **sca**
This third step closes all screen generated by the PTB.

If this 3 steps procedure doesn't work your last option is:
PC: => press **CTRL+ALT+DEL**
MAC: => press **COMMAND+ALT+ESC**
This option will force the closing of Matlab.
The problem of such option is that you will not know what causes the crash as the command window will be cleared.

4. Help and demos of the PTB

HELP SECTION

There are different ways to access the help section of the PTB:

- To access a list of possibilities of the PTB
  type => help Psychtoolbox
- To access a list of the Screen sub-functions,
  type => Screen
- To open the help section of a specified sub-function of Screen
  type => Screen OpenWindow ?

type => Screen Flip ?
type => Screen DrawLine ?

- Through the website of the PTB
  http://docs.psychtoolbox.org/Psychtoolbox

Question 3: What are the necessary inputs of Screen('DrawLine')?

Screen('DrawLine', windowPtr [,color], fromH, fromV, toH, toV [,penWidth]);

The inputs arguments that are not described under brackets are necessary to the execution of the function. On the contrary bracketed arguments are optional. The default values of optional arguments are given in the help section of each Screen sub-function.

PSYCHDEMOS

Now if you would like to know what the PTB could do or if you would like to have some inspiration for your following experiments, you can access to a list of demos by typing:

=> help Psychdemos

Matlab will return a list of PTB demos on which you can click to have more information.
To run a demo, type its name in the command window.

Try for example:
=> GratingDemo
=> DotDemo
=> MovieDemo

To access the code of such demos (and copy part of them in your own codes), type:
=> open GratingDemo
=> open DotDemo
=> open MovieDemo

5. Essential drawing functions of the PTB

Through the Screen function of the PTB, it is possible to create several visual stimuli composed of more or less simple geometric shapes.
We will here describe some essential functions of the PTB allowing you to elaborate the stimuli of your following experiments.

FILLRECT/ FRAMERECT

These two functions draw rectangles or squares. To use them follow such schema:

```
Screen('FillRect', windowPtr [,color] [,rect] );
Screen('FrameRect', windowPtr [,color] [,rect] [,penWidth]);
```

FILLRECT draws filled rectangles/squares of a certain size and color.
FRAMERECT draws rectangles/squares made only of a frame of a pre-defined width.

Here is an example of simple utilization of these two drawing functions.
Look for "testRect.m" in the material folder, add it to your path and execute it through the command window.
=> testRect(1);

```
function testRect(type)
% --------------------------------------------------------------------
% testRect(type)
% --------------------------------------------------------------------
% Goal of the function :
% Simple function that illustrates the FillRect and FrameRect sub
% functions of Screen()
% --------------------------------------------------------------------
% Input(s) :
% type : switch of the display
%                 if type == 1  :    FILLRECT
%                 if type == 2  :    FRAMERECT
%                 else          :    FILLRECT & FRAMERECT
% --------------------------------------------------------------------
% Output(s):
% (none)
% --------------------------------------------------------------------
% Function created by Martin SZINTE (martin.szinte@gmail.com)
% Last update : 02 / 01 / 2014
% Project : Programming courses
% Version : -
% --------------------------------------------------------------------

type;

scrAll = Screen('Screens');
scrNum = max(scrAll);

HideCursor;
Screen('Preference', 'SkipSyncTests', 1);
[scr] = Screen('OpenWindow',scrNum);

priorityLevel = MaxPriority(scr);Priority(priorityLevel);

colBG = [255 255 255];              % screen background color
colRect = [0 0 0];                  % filling color of the rectangles
rectRect = [200, 200, 400, 300];    % [Left Top Right Bottom] coordinates of the first
rectangle
rectRect2 = [200, 400, 700, 500];   % [Left Top Right Bottom] coordinates of the second
rectangle
rectWidth = 20;                     % width of the frame rectangle
```

```
% Main display loop
for timeFlip = 1:200

    Screen('FillRect',scr,colBG);  % fills background color of the screen

    % display selection loop
    if type == 1
        Screen('FillRect',scr,colRect,rectRect);               % draws a filled rectangle on
the secondary screen buffer
    elseif type == 2
        Screen('FrameRect',scr,colRect,rectRect2,rectWidth);   % draws a framed rectangle on
the secondary screen buffer
    else
        Screen('FillRect',scr,colRect,rectRect);               % draws a filled rectangle on
the secondary screen buffer
        Screen('FrameRect',scr,colRect,rectRect2,rectWidth);   % draws a framed rectangle on
the secondary screen buffer
    end

    Screen(scr, 'Flip');                                       % Flips the secondary screen
buffer on the monitor

end

ShowCursor;
Screen('CloseAll');

end
```

Question 4: What is the role of the input argument "type" of testRect() function?

This argument allows us to decide what to draw on the full-screen window. If type equals 1 the program draw a black filled rectangle, if type equal 2 a black framed rectangle, and if type equal 3 a black filled and a black framed rectangle.

Question 5: Why should we put "type;" at the beginning of the function?

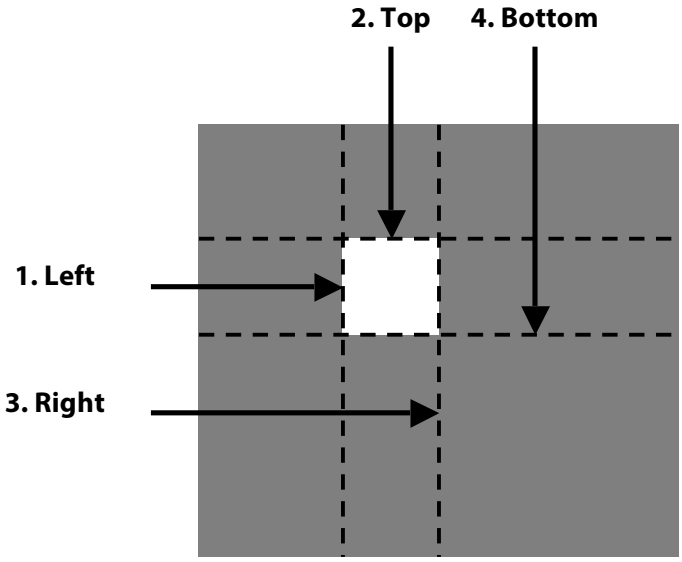
It is a good point to first call the input argument especially if the function loads a full-screen window. Indeed if you once forgot to feed the function with its necessary input argument "type" the program will crash before creating a full-screen window.

Question 6: How can we modify the size of the rectangles?

The coordinate system used by the PTB to specify the size of drawing elements is called LTBR for "Left-Top-Right-Bottom" and should be given when a "Rect" argument is need.

This means that we should transmit to the drawing function a vector of 4 elements:

- The first element determines the coordinate in pixel from the Left side of the screen to the edge of the element to draw.
- The second element determines the coordinate in pixel from the Top of the screen to the edge of the element to draw.
- The third element determines the coordinate in pixel from the Left of the screen to the edge of the element to draw.
- The forth element determines the coordinate in pixel from the Bottom of the screen to the edge of the element to draw.



Advice:
A good mnemonics way to remember such coordinate system is the following.
Think to a "LeTteR Box" and you will recall quickly the order of the coordinates to give in order to create a Rect.

Remark:
If the LTBR system stays unclear to you, a good solution could be to determine your own drawing functions using your own coordinates system and own input arguments.

DRAWLINE / FILLPOLY

It exists several sub function of Screen allowing us to draw lines and any kind of polygons.
To use them follow such schema:

```
Screen('DrawLine', wPtr [,color], fromH, fromV, toH, toV [,penWidth]);
Screen('FillPoly', wPtr [,color], pointList [, isConvex]);
```

Test these new functions by simply replacing in testRect.m the lines corresponding to the draw of rectangles.

If you don't understand the role of each input argument, look at their help section on the PTB website or by typing:
=> Screen 'DrawLine' ?
=> Screen 'FillPoly' ?

MAKETEXTURE / DRAWTEXTURE

It is possible to display in a full-screen window an image or a picture that is store in your computer. For this purpose you need to create your image or transform an existing picture in a matrix format. Such 3D (normal color) or 4D (color + transparency) matrix will specify the color of each pixel defining your image.
With such matrix you will be able to create an image texture (using "MakeTexture") and to draw the image to a full-screen window (using "DrawTexture").

Look for "testImage.m" and the picture pinkyandthebrain.jpg in the material folder, add the function to your path and execute it through the command window.
=> testImage

```
function testImage
% --------------------------------------------------------------------
% testImage
% --------------------------------------------------------------------
% Goal of the function :
% Display images or pictures in a full screen window
% --------------------------------------------------------------------
% Input(s) :
% (none)
% --------------------------------------------------------------------
% Output(s):
% (none)
% --------------------------------------------------------------------
% Function created by Martin SZINTE (martin.szinte@gmail.com)
% Last update : 02 / 01 / 2014
% Project : Programming courses
% Version : -
% --------------------------------------------------------------------

scrAll = Screen('Screens');
scrNum = max(scrAll);

HideCursor;
Screen('Preference', 'SkipSyncTests', 1);
[wPtr] = Screen('OpenWindow',scrNum);

priorityLevel = MaxPriority(wPtr);Priority(priorityLevel);

colBG = [0 0 0];                                % color of the background

pict = imread('pinkyandthebrain.jpg');          % reads the picture file and convert it to a
3d matrix
```

```
t_handle = Screen('MakeTexture',wPtr,pict);     % creates a texture and define its handle

for timeFlip = 1:1000                           % display loop

    Screen('FillRect',wPtr,colBG);              % draws a black background color
    Screen('DrawTexture',wPtr,t_handle);        % draws the texture the secondary screen
buffer
    Screen(wPtr, 'Flip');

end
ShowCursor;
Screen('CloseAll');

end
```

Remark1:
The Screen sub-functions *DrawTexture* and *MakeTexture* possess several input and output arguments. Look carefully for their roles through their help sections.

Using the same Screen sub-functions it is possible to draw any kind of stimuli such as stimuli defined by mathematical formulas.

Look for “testGabor.m” in the material folder, add the function to your path and execute it through the command window.

```
function testGabor(gab_angle,gab_contrast)
% --------------------------------------------------------------------
% testGabor(gab_angle,gab_contrast)
% --------------------------------------------------------------------
% Goal of the function :
% Simple function that draw a gabor
% --------------------------------------------------------------------
% gab_angle: gabor angle
% gab_contrast: gabor contrast
% --------------------------------------------------------------------
% Output(s):
% (none)
% --------------------------------------------------------------------
% Function created by Martin SZINTE (martin.szinte@gmail.com)
% Last update : 02 / 01 / 2014
% Project : Programming courses
% Version : -
% --------------------------------------------------------------------

gab_angle;
gab_contrast;

scrAll = Screen('Screens');
scrNum = max(scrAll);

HideCursor;
Screen('Preference', 'SkipSyncTests', 1);
wPtr = Screen('OpenWindow',scrNum,[0 0 0],[], [],2);
Screen('BlendFunction', wPtr, GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
% this last line allows the draw of transparency in the full-screen window

priorityLevel = MaxPriority(wPtr);Priority(priorityLevel);

colBG = [255/2 255/2 255/2];                % defines background color as gray

deg2pix = 40;                               % defines the size of 1 visual degree in pixels

% Gabor settings
bgluminance=0.5;                            % background luminace
frequency=2.5;                              % grating frequency
gab_phase=pi/2;                                % phase of the grating
sigma_period=1.1;                           % periods covered by one STD of gausian
angleGrat=gab_angle;                        % angle of grating in degree
gabContrast = gab_contrast;                 % Grating contrast

gabPix_X = 100;                             % Gab size X in pixels
gabPix_Y = 100;                             % Gab size Y in pixels
gabCenterX = 200;                           % Gab center X coordinate
```

```
gabCenterY = 200;                           % Gab center Y coordinate

destRect = [gabCenterX-gabPix_X/2,...       % left coordinate of the rect
            gabCenterY-gabPix_Y/2,...       % top coordinate of the rect
            gabCenterX+gabPix_X/2,...       % right coordinate of the rect
            gabCenterY+gabPix_Y/2];         % bottom coordinate of the rect

pixelsPerPeriod=deg2pix/frequency;          % pixel per period of the gabor
sigma=pixelsPerPeriod*sigma_period;         % sigma of the gabor

% Create grating (using nested function GenerateGrating)
TargetGrating = GenerateGrating(gabPix_X, gabPix_Y, angleGrat, pixelsPerPeriod, gab_phase,
gabContrast);

% Creation gaussian enveloppe (using nested function GenerateGaussian)
aperture = GenerateGaussian(gabPix_X, gabPix_Y, sigma, sigma, 0, 0, 0);
TargetShift = TargetGrating + bgluminance;                      % adds the background mean
color
TargetShift(TargetShift>1)=1; TargetShift(TargetShift<0)=0;     % makes color value in between
0 and 1
TargetCOR = TargetShift.*255;                                   % makes color value in color
system of the PTB

Target=[];
Target(:,:,1) = (TargetCOR.*1);                                 % defines R matrix column
Target(:,:,2) = (TargetCOR.*1);                                 % defines G matrix column
Target(:,:,3) = (TargetCOR.*1);                                 % defines B matrix column
Target(:,:,4) = aperture.*255;                                  % degines alpha values
(transparency)


texdistract0 = Screen('MakeTexture',wPtr,Target);               % Makes texture

for timeFlip = 1:500
    Screen('FillRect',wPtr,colBG);
    Screen('DrawTexture', wPtr, texdistract0, [], destRect, [], 0); % Draws the texture
    Screen('Flip',wPtr);
end

ShowCursor;
Screen('CloseAll');


% nested function generating grating (see in matlab)
    function grating = GenerateGrating(m,n,angle,lambda,ph,con)

    % nested function generating gaussian (see in matlab)
    function gaussian = GenerateGaussian(m, n, sigma1,sigma2, theta,xoff,yoff)

end
```