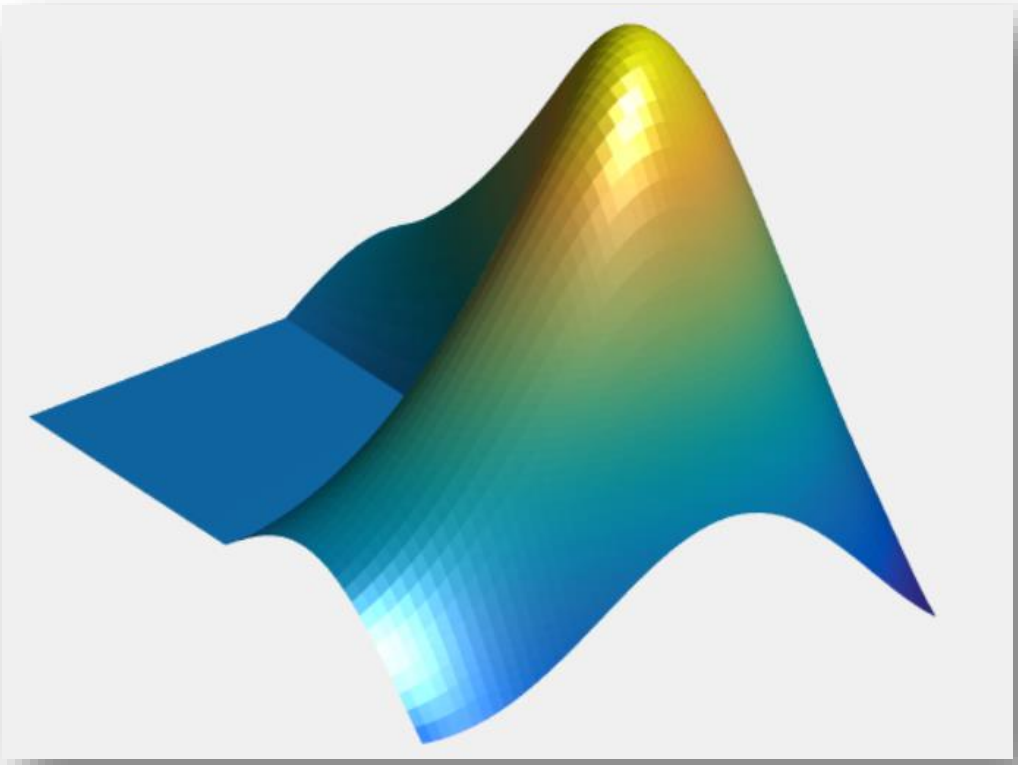
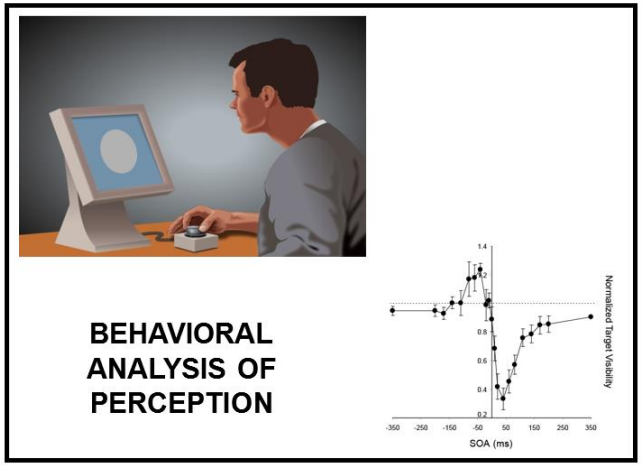
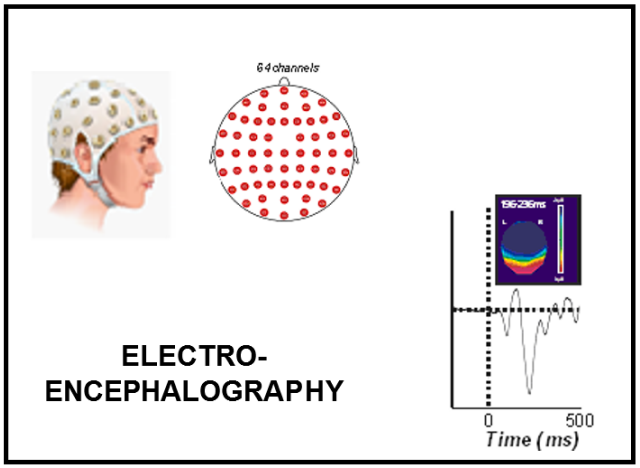
PsychToolbox: Part III

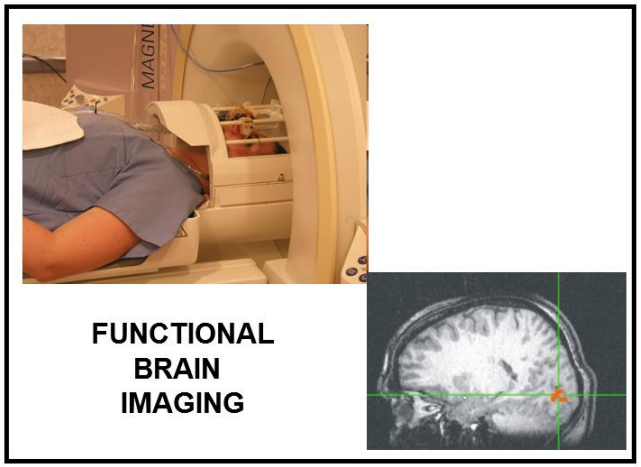


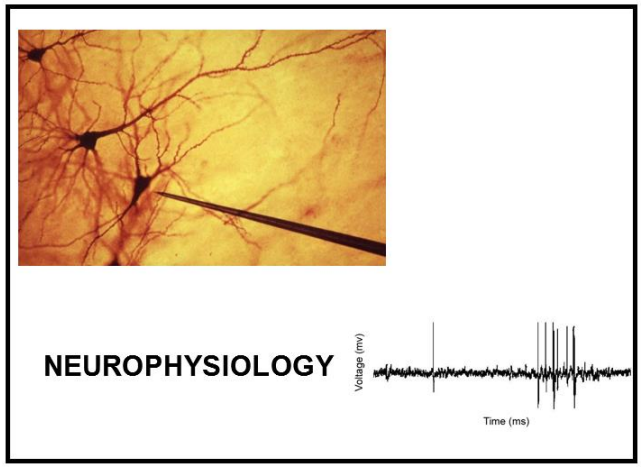
NSC 546 Computing for Neuroscience

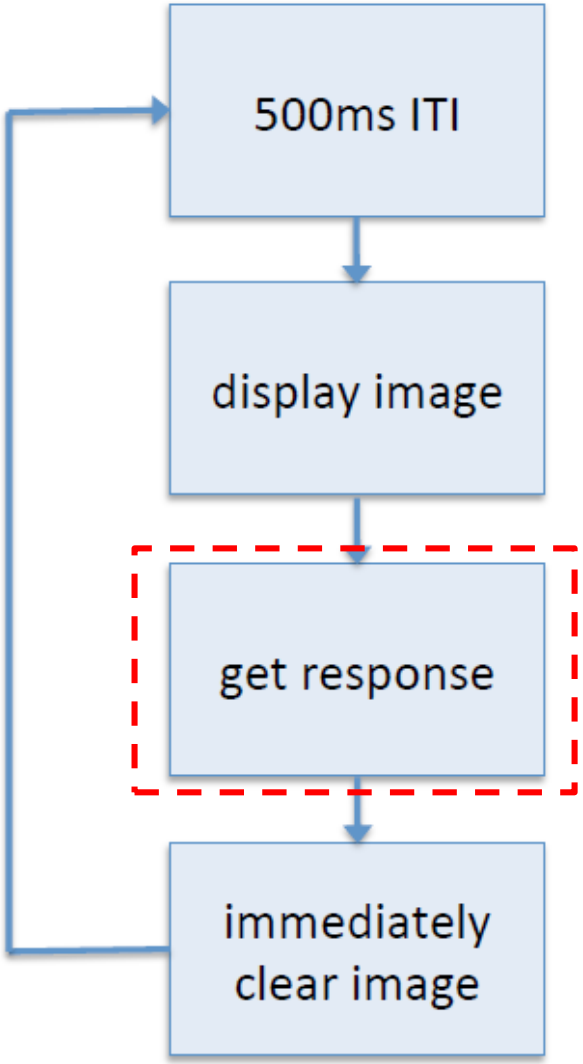
A Typical Trial of an Experiment











Do not use

```
Nscreen = 0;
[wPtr, rect] = Screen('OpenWindow', Nscreen);

pause;          # keypress will not be recorded

pause(.25);     # timing is not precise

x = input('Enter session :');
                # keypress will not be recorded

Screen('CloseAll');
```

These commands work fine outside of PsychToolbox of course.

Do not use

```
Nscreen = 0;
[wPtr, rect] = Screen('OpenWindow', Nscreen);

pause;          # keypress will not be recorded

pause(.25);     # timing is not precise

x = input('Enter session :');
                # keypress will not be recorded

Screen('CloseAll');
```

These commands work fine outside of PsychToolbox of course.

Do not use

```
[ch, when] = GetChar();
```

`GetChar` is a PsychToolbox command to get characters, but the timing (`when`) should not be trusted

KbWait

```
Nscreen = 0;
[wPtr, rect] = Screen('OpenWindow', Nscreen);

I = imread('IMG_3630.jpg');
txtImg = Screen('MakeTexture', wPtr, I);
Screen('DrawTexture', wPtr, txtImg);
[VBLT SOTime] = Screen(wPtr, 'Flip', 0);
KbWait;

Screen('CloseAll');
```

KbWait

`KbWait` is fine for something like

Press any key to start next block

It SHOULD NOT be used to measure response times since it checks the keyboard every 5ms, so there will be a 5ms uncertainty in measured RTs.

http://docs.psychtoolbox.org/KbWait

KbCheck

Try this on the command line:

```
>> KbCheck();
```

And this:

```
>> pause(.25); KbCheck();
```

KbCheck

Try this on the command line:

```
>> KbCheck();
```

The "enter" key is still down when this start.

You're not quick enough to lift your finger before KbCheck sees it and returns 1.

And this:

```
>> pause(.25); KbCheck();
```

After the 250ms delay, your finger is almost certainly off the "enter" key and returns 0.

KbCheck

KbCheck does what the name suggests: It Checks the keyboard. It does not wait for a key to be pressed. Polling must be used to get a response from a keyboard.

```
t1 = GetSecs;
while true
    [touch,tpress,keyCode] = KbCheck;

    if touch
        fprintf('Key being pressed\n');
        fprintf('RT = %dms\n', round((tpress-t1)*1000));
        break;
    end

    WaitSecs(0.0005);
end
```

KbCheck

By itself, this will loop forever.

```
t1 = GetSecs;
while true
    [touch,tpress,keyCode] = KbCheck;

    if touch
        fprintf('Key being pressed\n');
        fprintf('RT = %dms\n', round((tpress-t1)*1000));
        break;
    end

    WaitSecs(0.0005);
end
```

KbCheck

However, this will break out of the while loop.

```
t1 = GetSecs;
while true
    [touch,tpress,keyCode] = KbCheck;

    if touch
        fprintf('Key being pressed\n');
        fprintf('RT = %dms\n', round((tpress-t1)*1000));
        break;
    end

    WaitSecs(0.0005);
end
```

KbCheck

This is an alternative

```
t1 = GetSecs;
keeplooping = true;
while keeplooping
    [touch,tpress,keyCode] = KbCheck;

    if touch
        fprintf('Key being pressed\n');
        fprintf('RT = %dms\n', round((tpress-t1)*1000));
        keeplooping = false;
    end

    WaitSecs(0.0005);
end
```

KbCheck

Returns "1" if ANY key is down (regular key, shift key, alt key, ctrl key).
Returns "0" otherwise.

```
t1 = GetSecs;
keeplooping = true;
while keeplooping
    [touch,tpress,keyCode] = KbCheck;

    if touch
        fprintf('Key being pressed\n');
        fprintf('RT = %dms\n', round((tpress-t1)*1000));
        keeplooping = false;
    end

    WaitSecs(0.0005);
end
```

KbCheck

Without this, especially if priority is high, you can overload your computer. So you'll have ½ ms resolution, which is fine.

```
t1 = GetSecs;
keeplooping = true;
while keeplooping
    [touch,tpress,keyCode] = KbCheck;

    if touch
        fprintf('Key being pressed\n');
        fprintf('RT = %dms\n', round((tpress-t1)*1000));
        keeplooping = false;
    end

    WaitSecs(0.0005);
end
```

KbCheck

time the key was pressed as returned by **GetSecs**

```
t1 = GetSecs;
keeplooping = true;
while keeplooping
    [touch,tpress,keyCode] = KbCheck;

    if touch
        fprintf('Key being pressed\n');
        fprintf('RT = %dms\n', round((tpress-t1)*1000));
        keeplooping = false;
    end

    WaitSecs(0.0005);
end
```

KbCheck

RT in ms relative to when the **KbCheck** loop started

```
t1 = GetSecs;
keeplooping = true;
while keeplooping
    [touch,tpress,keyCode] = KbCheck;

    if touch
        fprintf('Key being pressed\n');
        fprintf('RT = %dms\n', round((tpress-t1)*1000));
        keeplooping = false;
    end

    WaitSecs(0.0005);
end
```

KbCheck

256-element array with "1" for any key that's pressed.

```
t1 = GetSecs;
keeplooping = true;
while keeplooping
    [touch,tpress,keyCode] = KbCheck;

    if touch
        fprintf('Key being pressed\n');
        fprintf('RT = %dms\n', round((tpress-t1)*1000));
        keeplooping = false;
    end

    WaitSecs(0.0005);
end
```

Decoding keyCode

View the table of keyCode numbers and keyboard keys

```
% get mapping from keyCode # to keyboard key
table = KbName('KeyNames');

for i=1:length(table)
    fprintf('%3d : %s\n', i, table{i});
end
```

Decoding keyCode

```
lkey = KbName('d');
rkey = KbName('k');

ListenChar(2);
while true
   [touch,tpress,keyCode] = KbCheck;

   if ( keyCode(lkey)==1 || keyCode(rkey)==1 )
      break;
   end
end
ListenChar(0);
```

what is this doing?

what is this doing?

Decoding keyCode

Monitoring Special Characters

```
% special keys

clear all;  close all;  clear mex;

%ListenChar(2);
keeplooping=true;
while keeplooping
   [touch,tpress,keyCode]=KbCheck;

   if ( keyCode(KbName('esc')) )       keeplooping=false;
   elseif ( keyCode(KbName('space')) )     fprintf('Space Bar Hit!\n');keeplooping=false;
   elseif ( keyCode(KbName('up')) )   fprintf('Up Arrow Hit!\n');keeplooping=false;
   elseif ( keyCode(KbName('down')) ) fprintf('Down Arrow Hit!\n');keeplooping=false;
   elseif ( keyCode(KbName('right')) ) fprintf('Right Arrow Hit!\n');keeplooping=false;
   elseif ( keyCode(KbName('left')) ) fprintf('Left Arrow Hit!\n');keeplooping=false;
   elseif ( keyCode(KbName('left_shift')) ) fprintf('Left Shift Hit!\n');keeplooping=false;
   elseif ( keyCode(KbName('right_shift')) ) fprintf('Right Shift Hit!\n');keeplooping=false;
   end
end
%ListenChar(0);

fprintf('Done!\n');
```

More Keyboard Notes

Do not simply "ignore" invalid keys.

```
lkey = KbName('d');     rkey = KbName('k');

ListenChar(2);
t1 = GetSecs;
while true
        [touch,tpress,keyCode] = KbCheck;

        if ( keyCode(lkey)==1 || keyCode(rkey)==1 )
                rt = round((tpress-t1)*1000);
                break;
        end
end
ListenChar(0);
```

this "ignore" everything but the two valid keys – while fine for accuracy data, a problem for RTs

More Keyboard Notes

Do not simply "ignore" invalid keys.

```
lkey = KbName('d');     rkey = KbName('k');

ListenChar(2);
t1 = GetSecs;
badkey = false;
while true
        [touch,tpress,keyCode] = KbCheck;

        if ( keyCode(lkey)==1 || keyCode(rkey)==1 )
                rt = round((tpress-t1)*1000);
                break;
        elseif touch
                badkey = true;
        end
end
ListenChar(0);
```

Better!

KbCheck

KbCheck only checks keyboard when it's called

```
t1 = GetSecs;
while true
   [touch,tpress,keyCode] = KbCheck;

   if touch
      fprintf('Key being pressed\n');
      break;
   end

   WaitSecs(0.0005);
end
```

KbCheck

```
t1 = GetSecs;
while true
   [touch,tpress,keyCode] = KbCheck;

   if touch
      fprintf('Key being pressed\n');
      break;
   end

   WaitSecs(0.0005);
end
```

so if a key is pressed and released during this interval, it will be missed by **KbCheck**

KbCheck

```
t1 = GetSecs;
while true
   [touch,tpress,keyCode] = KbCheck;

   if touch
      fprintf('Key being pressed\n');
      break;
   end

   WaitSecs(0.0005);
end
```

not a problem for keyboard keypresses and if the loop only monitors for keypresses

KbCheck

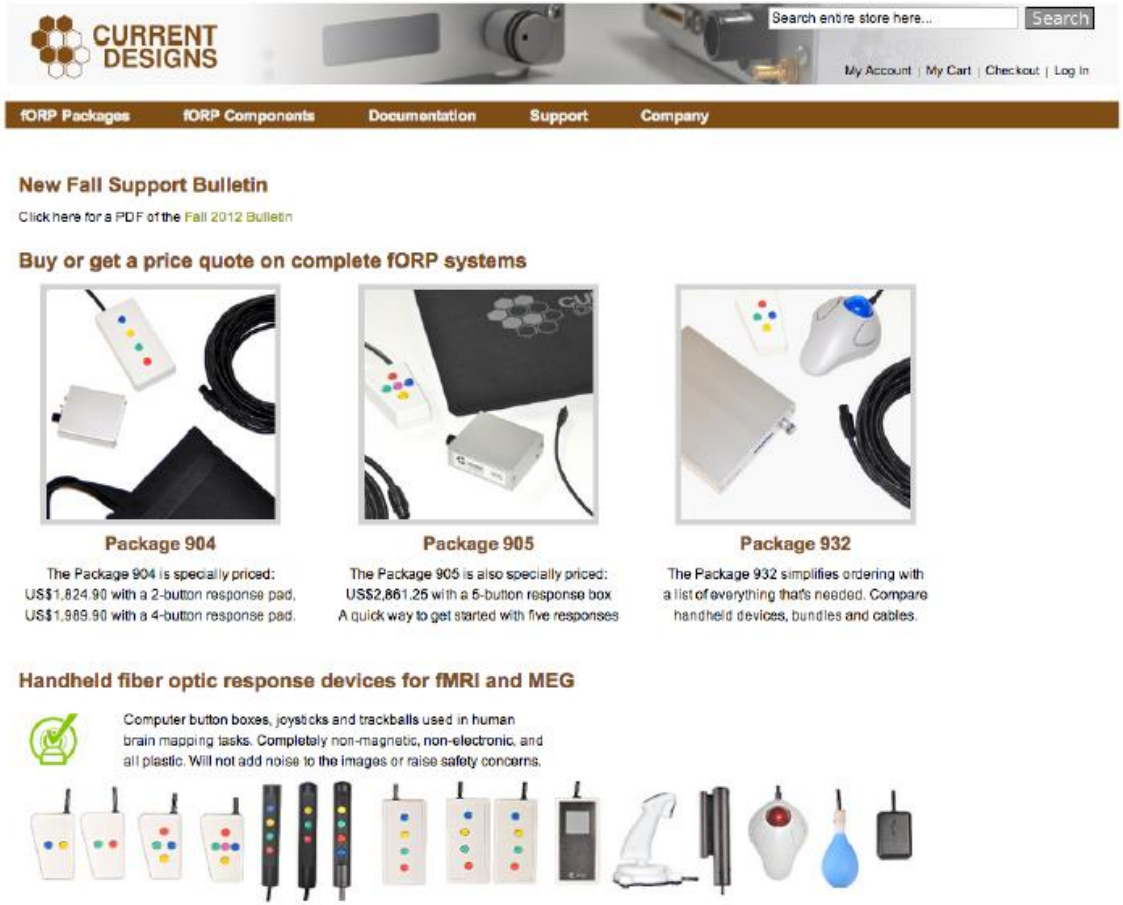
```
t1 = GetSecs;
while true
   [touch,tpress,keyCode] = KbCheck;

   % COMPLEX IMAGE PROCESSING HERE

   WaitSecs(0.0005);
end
```

it could be a problem if there is a lot of time-consuming processing in this loop

KbCheck



some response boxes (e.g., for fMRI or MEG) translate a physical button press into a signal that's read as a keyboard keypress for only a few milliseconds

KbCheck versus KbQueue

KbCheck

checks the current state of the keyboard

KbQueue

keys are added to a queue in the background (probably via interrupts)

`KbQueue` routines create and act on the keyboard queue, not on the keyboard state directly

the queue can only hold 30 events

but only the first and last keypress press and release times are recorded

KbQueue commands

```
KbQueueCreate();
 ⋮
KbQueueStart();
 ⋮
while true
   [pressed, firstPress, firstRelease, ...
         lastPress, lastRelease] = KbQueueCheck();
   if pressed
      fprintf('You pressed key %i which is %s\n', ...
            find(firstPress), KbName(firstPress));
   end
end
KbQueueStop();
 ⋮
KbQueueRelease();
```

it takes a bit of time to create the queue

KbQueue commands

```
KbQueueCreate();
 ⋮
KbQueueStart();
 ⋮
while true
   [pressed, firstPress, firstRelease, ...
         lastPress, lastRelease] = KbQueueCheck();
   if pressed
      fprintf('You pressed key %i which is %s\n', ...
            find(firstPress), KbName(firstPress));
   end
end
KbQueueStop();
 ⋮
KbQueueRelease();
```

for a created queue, start delivering keys to it

KbQueue commands

```
KbQueueCreate();
 ⋮
KbQueueStart();
 ⋮
while true
   [pressed, firstPress, firstRelease, ...
         lastPress, lastRelease] = KbQueueCheck();
   if pressed
      fprintf('You pressed key %i which is %s\n', ...
            find(firstPress), KbName(firstPress));
   end
end
KbQueueStop();
 ⋮
KbQueueRelease();
```

stop delivering keys to the queue

KbQueue commands

```
KbQueueCreate();
 ⋮
KbQueueStart();
 ⋮
while true
   [pressed, firstPress, firstRelease, ...
         lastPress, lastRelease] = KbQueueCheck();
   if pressed
      fprintf('You pressed key %i which is %s\n', ...
            find(firstPress), KbName(firstPress));
   end
end
KbQueueStop();
 ⋮
KbQueueRelease();
```

delete the queue

KbQueue commands

```
KbQueueCreate();
 ⋮
KbQueueStart();
 ⋮
while true
    [pressed, firstPress, firstRelease, ...
          lastPress, lastRelease] = KbQueueCheck();
    if pressed
       fprintf('You pressed key %i which is %s\n', ...
             find(firstPress), KbName(firstPress));
    end
end
KbQueueStop();
 ⋮
KbQueueRelease();
```

like **KbCheck** but returns first and last key pressed

KbQueue commands

```
KbQueueCreate();
 ⋮
KbQueueStart();
 ⋮
while true
   [pressed, firstPress, firstRelease, ...
         lastPress, lastRelease] = KbQueueCheck();
   if pressed
      fprintf('You pressed key %i which is %s\n', ...
            find(firstPress), KbName(firstPress));
   end
end
KbQueueStop();
 ⋮
KbQueueRelease();
```

now, the **`firstPress`** (and **`lastPress`**) arrays have a time at the index location of when a key was pressed

Sound commands

```
% sound parameters
load handel;
nrchannels = 2; % Two channels
deviceid=13;

[Y,FS,NBITS]=wavread('numinout.wav'); % reading the sound file

% Perform basic initialization of the sound driver:
freq=44100; % check this frequency with the hardware
InitializePsychSound(0);
pahandle = PsychPortAudio('Open', deviceid, [], 2, freq, nrchannels);
volume=PsychPortAudio('Volume',pahandle,.20);

% Fill the audio playback buffer with the audio data 'wavedata':
PsychPortAudio('FillBuffer', pahandle, Y');

% Start playing the sound
ts1 = PsychPortAudio('Start', pahandle, 1, 0, 0);

% Stop playback:
PsychPortAudio('Stop', pahandle);

% Close the audio device at the end of the experiment
PsychPortAudio('Close', pahandle);
```