دانشکده روانشناسی و علوم تربیتی


# MATLAB for Brain and Cognitive Psychology (Response collection)

Presented by:
Ehsan Rezayat, Ph.D.
Faculty of Psychology and Education, University of Tehran.
Institute for Research in Fundamental Sciences (IPM), School of Cognitive Sciences,
emails: rezayat@ut.ac.ir, rezayat@ipm.ir, erezayat.er@gmail.com

# Today: Steps in the Psychophysics Lab

- Generating Stimuli
- Visual Display
- Stimulus presentation
- Response collection

# Collecting responses

# PsychHID

- Interaction with USB devices is accomplished through PsychHID (Human Interface Device)
- Even internal keyboards are accessed this way

# Listing devices

```
devices = PsychHID('Devices');
```

- Returns a structure array where each element describes a single device
- PsychHID only checks for USB devices on startup. If you plug in a device after starting matlab it wont be recognized by PsychHID, *even if you can see its input on the screen.* You need to either restart Matlab or issue **clear PsychHID** to renumerate the connected devices.

# Psychtoolbox Response Monitoring

- GetChar()
- KbWait()
- KbCheck()
- KbQueueCheck()

- GetMouse()
- GetClicks()
- GetMouseWheel()
- SetMouse()
- ShowCursor()
- HideCursor()

- GamePad()

# Psychtoolbox Response Monitoring

- PsychRTBox()

http://lobes.usc.edu/RTbox/

# Other response input methods

- Ask()
- GetEchoNumber()
- GetEchoString()
- GetNumber
- GetString

# Keyboard responses

- GetChar()
- KbWait()
- KbCheck()
- KbQueueCheck()

# Keyboard responses

CURRENT DESIGNS

MRI response interface that delivers the keypresses from the button box and the triggers from the scanner is a keyboard device.

# GetChar

[ch, when] = GetChar()

GetChar can return characters that were type *before* you called it!
As long as listening is turned on, GetChar will be listening. It will then return all the keys pressed since it started listening, in order. If there are none left in the queue, it will wait for a new one.

Use FlushEvents() to clear the queue and to start listening. You can also call ListenChar() to turn listening on and off directly.

# GetChar

```
>> FlushEvents()
>> pressed = GetChar()

pressed =

p

>> pressed = GetChar()

pressed =

r

>> pressed = GetChar()

pressed =

e

>> FlushEvents;GetChar()

ans =

x
```

# Ascii code

| Dec | Hex | Oct | Chr | Dec | Hex | Oct | HTML | Chr | Dec | Hex | Oct | HTML | Chr | Dec | Hex | Oct | HTML | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NULL | 32 | 20 | 040 | &#032; | Space | 64 | 40 | 100 | &#064; | @ | 96 | 60 | 140 | &#096; | ` |
| 1 | 1 | 001 | Start of Header | 33 | 21 | 041 | &#033; | ! | 65 | 41 | 101 | &#065; | A | 97 | 61 | 141 | &#097; | a |
| 2 | 2 | 002 | Start of Text | 34 | 22 | 042 | &#034; | " | 66 | 42 | 102 | &#066; | B | 98 | 62 | 142 | &#098; | b |
| 3 | 3 | 003 | End of Text | 35 | 23 | 043 | &#035; | # | 67 | 43 | 103 | &#067; | C | 99 | 63 | 143 | &#099; | c |
| 4 | 4 | 004 | End of Transmission | 36 | 24 | 044 | &#036; | $ | 68 | 44 | 104 | &#068; | D | 100 | 64 | 144 | &#100; | d |
| 5 | 5 | 005 | Enquiry | 37 | 25 | 045 | &#037; | % | 69 | 45 | 105 | &#069; | E | 101 | 65 | 145 | &#101; | e |
| 6 | 6 | 006 | Acknowledgment | 38 | 26 | 046 | &#038; | & | 70 | 46 | 106 | &#070; | F | 102 | 66 | 146 | &#102; | f |
| 7 | 7 | 007 | Bell | 39 | 27 | 047 | &#039; | ' | 71 | 47 | 107 | &#071; | G | 103 | 67 | 147 | &#103; | g |
| 8 | 8 | 010 | Backspace | 40 | 28 | 050 | &#040; | ( | 72 | 48 | 110 | &#072; | H | 104 | 68 | 150 | &#104; | h |
| 9 | 9 | 011 | Horizontal Tab | 41 | 29 | 051 | &#041; | ) | 73 | 49 | 111 | &#073; | I | 105 | 69 | 151 | &#105; | i |
| 10 | A | 012 | Line feed | 42 | 2A | 052 | &#042; | * | 74 | 4A | 112 | &#074; | J | 106 | 6A | 152 | &#106; | j |
| 11 | B | 013 | Vertical Tab | 43 | 2B | 053 | &#043; | + | 75 | 4B | 113 | &#075; | K | 107 | 6B | 153 | &#107; | k |
| 12 | C | 014 | Form feed | 44 | 2C | 054 | &#044; | , | 76 | 4C | 114 | &#076; | L | 108 | 6C | 154 | &#108; | l |
| 13 | D | 015 | Carriage return | 45 | 2D | 055 | &#045; | - | 77 | 4D | 115 | &#077; | M | 109 | 6D | 155 | &#109; | m |
| 14 | E | 016 | Shift Out | 46 | 2E | 056 | &#046; | . | 78 | 4E | 116 | &#078; | N | 110 | 6E | 156 | &#110; | n |
| 15 | F | 017 | Shift In | 47 | 2F | 057 | &#047; | / | 79 | 4F | 117 | &#079; | O | 111 | 6F | 157 | &#111; | o |
| 16 | 10 | 020 | Data Link Escape | 48 | 30 | 060 | &#048; | 0 | 80 | 50 | 120 | &#080; | P | 112 | 70 | 160 | &#112; | p |
| 17 | 11 | 021 | Device Control 1 | 49 | 31 | 061 | &#049; | 1 | 81 | 51 | 121 | &#081; | Q | 113 | 71 | 161 | &#113; | q |
| 18 | 12 | 022 | Device Control 2 | 50 | 32 | 062 | &#050; | 2 | 82 | 52 | 122 | &#082; | R | 114 | 72 | 162 | &#114; | r |
| 19 | 13 | 023 | Device Control 3 | 51 | 33 | 063 | &#051; | 3 | 83 | 53 | 123 | &#083; | S | 115 | 73 | 163 | &#115; | s |
| 20 | 14 | 024 | Device Control 4 | 52 | 34 | 064 | &#052; | 4 | 84 | 54 | 124 | &#084; | T | 116 | 74 | 164 | &#116; | t |
| 21 | 15 | 025 | Negative Ack. | 53 | 35 | 065 | &#053; | 5 | 85 | 55 | 125 | &#085; | U | 117 | 75 | 165 | &#117; | u |
| 22 | 16 | 026 | Synchronous idle | 54 | 36 | 066 | &#054; | 6 | 86 | 56 | 126 | &#086; | V | 118 | 76 | 166 | &#118; | v |
| 23 | 17 | 027 | End of Trans. Block | 55 | 37 | 067 | &#055; | 7 | 87 | 57 | 127 | &#087; | W | 119 | 77 | 167 | &#119; | w |
| 24 | 18 | 030 | Cancel | 56 | 38 | 070 | &#056; | 8 | 88 | 58 | 130 | &#088; | X | 120 | 78 | 170 | &#120; | x |
| 25 | 19 | 031 | End of Medium | 57 | 39 | 071 | &#057; | 9 | 89 | 59 | 131 | &#089; | Y | 121 | 79 | 171 | &#121; | y |
| 26 | 1A | 032 | Substitute | 58 | 3A | 072 | &#058; | : | 90 | 5A | 132 | &#090; | Z | 122 | 7A | 172 | &#122; | z |
| 27 | 1B | 033 | Escape | 59 | 3B | 073 | &#059; | ; | 91 | 5B | 133 | &#091; | [ | 123 | 7B | 173 | &#123; | { |
| 28 | 1C | 034 | File Separator | 60 | 3C | 074 | &#060; | < | 92 | 5C | 134 | &#092; | \ | 124 | 7C | 174 | &#124; | \| |
| 29 | 1D | 035 | Group Separator | 61 | 3D | 075 | &#061; | = | 93 | 5D | 135 | &#093; | ] | 125 | 7D | 175 | &#125; | } |
| 30 | 1E | 036 | Record Separator | 62 | 3E | 076 | &#062; | > | 94 | 5E | 136 | &#094; | ^ | 126 | 7E | 176 | &#126; | ~ |
| 31 | 1F | 037 | Unit Separator | 63 | 3F | 077 | &#063; | ? | 95 | 5F | 137 | &#095; | _ | 127 | 7F | 177 | &#127; | Del |

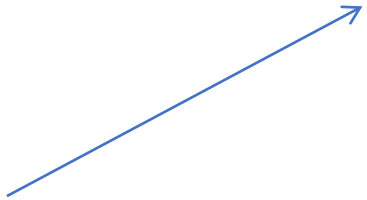
asciicharstable.com

# GetChar

- Don't use GetChar() for timing!
- No, really, don't use GetChar for response times!

# KbWait

[secs, keyCode, deltaSecs] = KbWait([devicenumber] [, forWhat = 0][, untilTime=inf)



which device are we listening to?
use PsychHID('Devices') to list all devices

GetKeyboardIndices() will return the device numbers
of all keyboard devices

**Use -1 to listen to all keyboards**
**Use -2 to listen to all keypad devices**
**Use -3 to listen to all keyboards and keypads**

# KbWait

- When you press a key, you press it and then release it

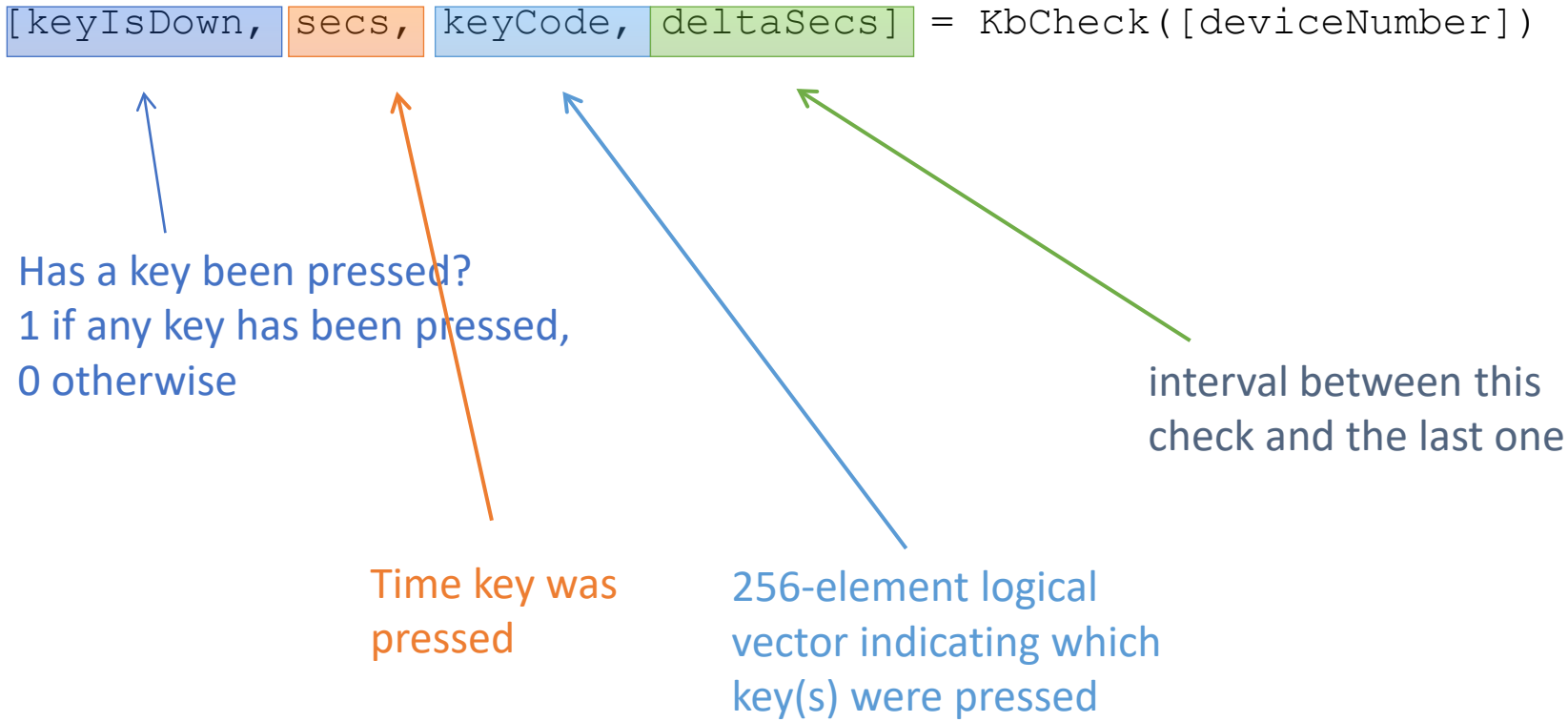
# KbWait

[secs, keyCode, deltaSecs] = KbWait([devicenumber] [, forWhat = 0][, untilTime=inf)

0: Default.  Listen for key down
1: Listen for key release
2: Wait until all keys are released, THEN wait for key down
3: Wait until all keys are released, then wait for a full key press and release

Stop waiting when we  get to this time

# KbCheck


[keyIsDown, secs, keyCode, deltaSecs] = KbCheck([deviceNumber])
Has a key been pressed?
1 if any key has been pressed,
0 otherwise
Time key was pressed
256-element logical vector indicating which key(s) were pressed
interval between this check and the last one

```
function getKeypress

WaitSecs(.5);
startTime = GetSecs();
keyIsDown = 0;

while ~keyIsDown
    [ keyIsDown, pressedSecs, keyCode ] = KbCheck(-1);
end

pressedKey = KbName(find(keyCode));
reactionTime = pressedSecs-startTime;

fprintf('\nKey %s was pressed at %.4f seconds\n\n',pressedKey,reactionTime);


end
```

# Use KbCheck to break out of an animation loop

```
function getKeypress2

WaitSecs(.5);
startTime = GetSecs();
keyIsDown = 0;

%open the screen and set up initial color
[wPtr, rect] = Screen('OpenWindow',max(Screen('Screens')));
bgColor = 255;
Screen('FillRect',wPtr,bgColor);
Screen('Flip',wPtr);
increment = -1;


while ~keyIsDown
    [ keyIsDown, pressedSecs, keyCode ] = KbCheck(-1);

    %change the background color
    bgColor = bgColor + increment;
    if bgColor <= 0 || bgColor >= 255
        increment = -increment;
    end
    Screen('FillRect',wPtr,bgColor);
    Screen('Flip',wPtr);

end

pressedKey = KbName(find(keyCode));
reactionTime = pressedSecs-startTime;

fprintf('\nKey %s was pressed at %.4f seconds\n\n',pressedKey,reactionTime);


end
```

Collect a response only while the stimulus is visible

```
function getKeypress3
%Present a stimulus and wait for a response while the stimulus is visible

WaitSecs(.5);
startTime = GetSecs();
keyIsDown = 0;
stimDuration = 5;

%open the screen
[wPtr, rect] = Screen('OpenWindow',max(Screen('Screens')));

%Read in pic and create texture
faceData = imread('sadface.jpg');
faceTexture = Screen('MakeTexture',wPtr,faceData);

%Draw it
Screen('DrawTexture',wPtr,faceTexture);
stimTime = Screen('Flip',wPtr);

while GetSecs() <= stimTime + stimDuration
    [ keyIsDown, pressedSecs, keyCode ] = KbCheck(-1);
    if keyIsDown
        responseKey = KbName(find(keyCode));
        responseTime = pressedSecs-startTime;
    end
end

%Blank screen for 5 seconds, not recording responses now
Screen('Flip',wPtr);
WaitSecs(5);

fprintf('\nKey %s was pressed at %.4f seconds\n\n',responseKey,responseTime);
clear Screen;

end
```

Continue collecting responses after the stimulus goes away

```
function getKeypress4
%Continue to collect responses after the stimulus is gone

WaitSecs(.5);
startTime = GetSecs();
keyIsDown = 0;
stimDuration = 5;
responseDuration = 10;

%open the screen
[wPtr, rect] = Screen('OpenWindow',max(Screen('Screens')));

%Read in pic and create texture
faceData = imread('sadface.jpg');
faceTexture = Screen('MakeTexture',wPtr,faceData);

%Draw it
Screen('DrawTexture',wPtr,faceTexture);
stimTime = Screen('Flip',wPtr);

while GetSecs() <= stimTime + responseDuration
    [ keyIsDown, pressedSecs, keyCode ] = KbCheck(-1);
    if keyIsDown
        responseKey = KbName(find(keyCode));
        responseTime = pressedSecs-startTime;
    end
    if GetSecs - stimTime >= stimDuration
        %stimulus has been up for as long as it should be, so we erase it
        Screen('Flip',wPtr);
    end
end

fprintf('\nKey %s was pressed at %.4f seconds\n\n',responseKey,responseTime);
clear Screen;

end
```

# Ignoring responses

```
DisableKeysForKbCheck([disablekeys])
```

vector of key codes
to ignore

```
RestrictKeysForKbCheck([enablekeys])
```

```
function waitForScannerTrigger

WaitSecs(.5);

%find key code for trigger key, which is a 5
triggerCode = KbName('5%');
keyIsDown = 0;

%Make sure no keys are disabled
DisableKeysForKbCheck([]);

%wait for trigger
while 1
    [ keyIsDown, pressedSecs, keyCode ] = KbCheck(-1);
    if keyIsDown
        if find(keyCode)==triggerCode
            break;
        end
    end
end

%Record trigger time for future reference
triggerTime = pressedSecs;
fprintf('Trigger detected\n');

%Now disable 5 key for the rest of the script
DisableKeysForKbCheck([triggerCode]);

%Now get a new response, ignoring triggers
WaitSecs(.5);
fprintf('Waiting for response.\n');
keyIsDown = 0;
while ~keyIsDown
    [ keyIsDown, pressedSecs, keyCode ] = KbCheck(-1);
end

pressedKey = KbName(find(keyCode));
reactionTime = pressedSecs-triggerTime;

fprintf('\nKey %s was pressed at %.4f seconds\n\n',pressedKey,reactionTime);

end
```

waiting for a specific response

waiting for any response EXCEPT certain keys

# Keyboard responses

- GetChar()
- KbWait()
- KbCheck()
- KbQueueCheck()

# KbQueueCheck

- An alternative set of commands for collecting keypresses:
  - KbQueueCreate
  - KbQueueStart
  - KbQueueStop
  - KbQueueCheck
  - KbQueueWait
  - KbQueueFlush
  - KbQueueRelease

# KbQueueCheck

- Advantages of KbQueueCheck:
  - Sometimes detects really brief responses that KbCheck can miss
  - Very accurate time recording
  - Records presses and releases both
- Disadvantages:
  - Difficulty in recording multiple presses of the same key
  - May not deal well with many rapid keypresses

# Steps to using KbQueue

- KbQueueCreate([deviceNumber]) to create the queue.
- KbQueueStart() to start listening
- KbQueueStop() to stop listening (does not clear the queue)
- KbQueueCheck() to check the values recorded while the queue was active
- KbQueueFlush() to empty the queue
- KbQueueRelease() to destroy the queue object

# KbQueueCheck

```
[pressed, firstPress, firstRelease, lastPress, lastRelease] =
KbQueueCheck()
```

has a key been pressed?

array indicating when each key was first pressed

array indicating when each key was first released

```
function waitForScannerTrigger

WaitSecs(.5);

%find key code for trigger key, which is a 5
triggerCode = KbName('5%');
keyIsDown = 0;

%Make sure no keys are disabled
DisableKeysForKbCheck([]);

%wait for trigger
while 1
    [ keyIsDown, pressedSecs, keyCode ] = KbCheck(-1);
    if keyIsDown
        if find(keyCode)==triggerCode
            break;
        end
    end
end

%Record trigger time for future reference
triggerTime = pressedSecs;
fprintf('Trigger detected\n');

%Now disable 5 key for the rest of the script
DisableKeysForKbCheck([triggerCode]);

%Now get a new response, ignoring triggers
WaitSecs(.5);
fprintf('Waiting for response.\n');
keyIsDown = 0;
while ~keyIsDown
    [ keyIsDown, pressedSecs, keyCode ] = KbCheck(-1);
end

pressedKey = KbName(find(keyCode));
reactionTime = pressedSecs-triggerTime;

fprintf('\nKey %s was pressed at %.4f seconds\n\n',pressedKey,reactionTime);

end
```

# Other keyboard response functions

- GetNumber()
- GetString()
- GetEchoString(wPtr,message,x,y)
- Ask(wPtr, message)

# Mouse responses

- GetMouse()
- GetClicks()
- GetMouseWheel()
- SetMouse()
- ShowCursor()
- HideCursor()

# Mouse responses

```
[x,y,buttons] = GetMouse([windowPtrOrScreenNumber][, mouseDev])
```

vector of three numbers, one for each mouse button
0 = not pressed
1 = pressed

which mouse device

```
function mouseWait

buttons = 0;

while ~buttons
    [x,y,buttons] = GetMouse();
end

fprintf('You clicked button %d\n',find(buttons));

end
```

# Mouse responses

```
[clicks,x,y,whichButton] = GetClicks([windowPtrOrScreenNumber]
[, interclickSecs][, mouseDev])
```

Use this to wait for a click and record where the user clicked, and how many clicks they made (e.g. double-click).

```
wheelDelta = GetMouseWheel([mouseIndex])
```

Use this to get the position of the mouse scroll wheel

# Controlling the mouse

- SetMouse(x,y) to move the mouse to a location
- HideCursor() to hide the mouse pointer
- ShowCursor() to show the mouse pointer

# Other input devices

- GamePad()

- Type Gamepad in the command window for help, or Gamepad Subcommand? for help with a subcommand

# Gamepad

- Gamepad('GetButton',gamepadIndex, buttonIndex) to get status of buttons
- Gamepad('GetAxis',gamepadIndex,axisIndex) to get joystick position
- Gamepad('GetBall',gamepadIndex,ballIndex) to get trackball info

# Assignment #9

- Create a function called yourinitials_session9()
  - The function will take one input, radius, which will determine the radius of a circle
  - Draw a black circle in the center of the screen. Using KbCheck, wait for the user to press a key. If the user presses R, the ball will turn red; if they press G the ball should turn green; B will turn the ball blue.
  - The ball will begin moving towards the mouse position. Only move the ball 2 pixels each frame, do not jump right to the location of the mouse. The ball will follow the mouse around the screen until the user clicks the mouse, when the program will end and the screen will clear.
  - While the ball is moving, the user may press R, G, or B to change the color of the circle accordingly.