PRIORITY SETTING

PRIORITY

- Modern computers have multiple software processes constantly competing for access to resources.
- How these resources are allocated moment to moment can affect the execution of your script
- Recommendation: When you are testing with Psychtoolbox, close applications other than Matlab
- Use Psychtoolbox's Priority function to assign a priority to the execution of your process

PRIORITY

- Use Priority() to set the priority level
- The higher the priority level, the less chance there is of other processes interfering w ith your script
- Available levels and their functions differ depending on your OS

PRIORITY: WINDOWS

- On Windows there are 3 levels available:
  - 0 : normal priority level
  - 1: high priority level
  - 2: real time priority level
- Using level 2 may cause problems (for example, it may disable keyboard input). Proba bly only want to use this when absolutely necessary, for example when running an int ense animation where timing really matters.

PRIORITY

- MaxPriority(windowOrScreenNum) will tell you the maximum priority allowed on your system
- Not recommended to use greater than 1 on windows

```
whichScreen = max(Screen('Screens'));
maxPriorityLevel = MaxPriority(whichScreen);
Priority(maxPriorityLevel);
```

These lines would go at the beginning of your script to set priority level for that script

GETTING RESPONSE TIME

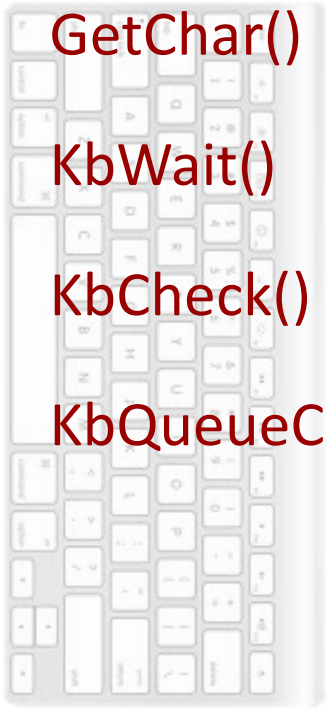
COLLECTING RESPONSES

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()

KEYBOARD RESPONSES

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

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.

Not good for getting response time!!!!!

GETCHAR

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

pressed =

p

>> pressed = GetChar()

pressed =

r

>> pressed = GetChar()

pressed =

e

>> FlushEvents;GetChar()

ans =

x
```

KBWAIT

[secs, keyCode, deltaSecs] = KbWait()

Will wait until the user presses a key, and return the time and keypress.

KBWAIT

[secs, keyCode, deltaSecs] = KbWait()

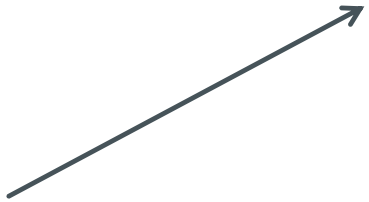
keyCode is a vector of all the keys, with a 1 for any key that was pressed.

find(keyCode) will return the index of the button(s) pressed.

That code can then be turned into a character using KbName()

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

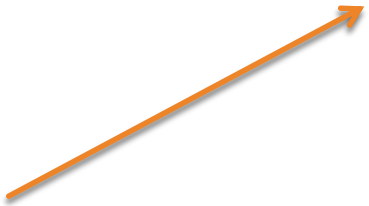
Use -1 to listen to all keyboards
Use -2 to listen to all keypad devices
Use -3 to listen to all keyboards and keypads

KBWAIT

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

KBWAIT

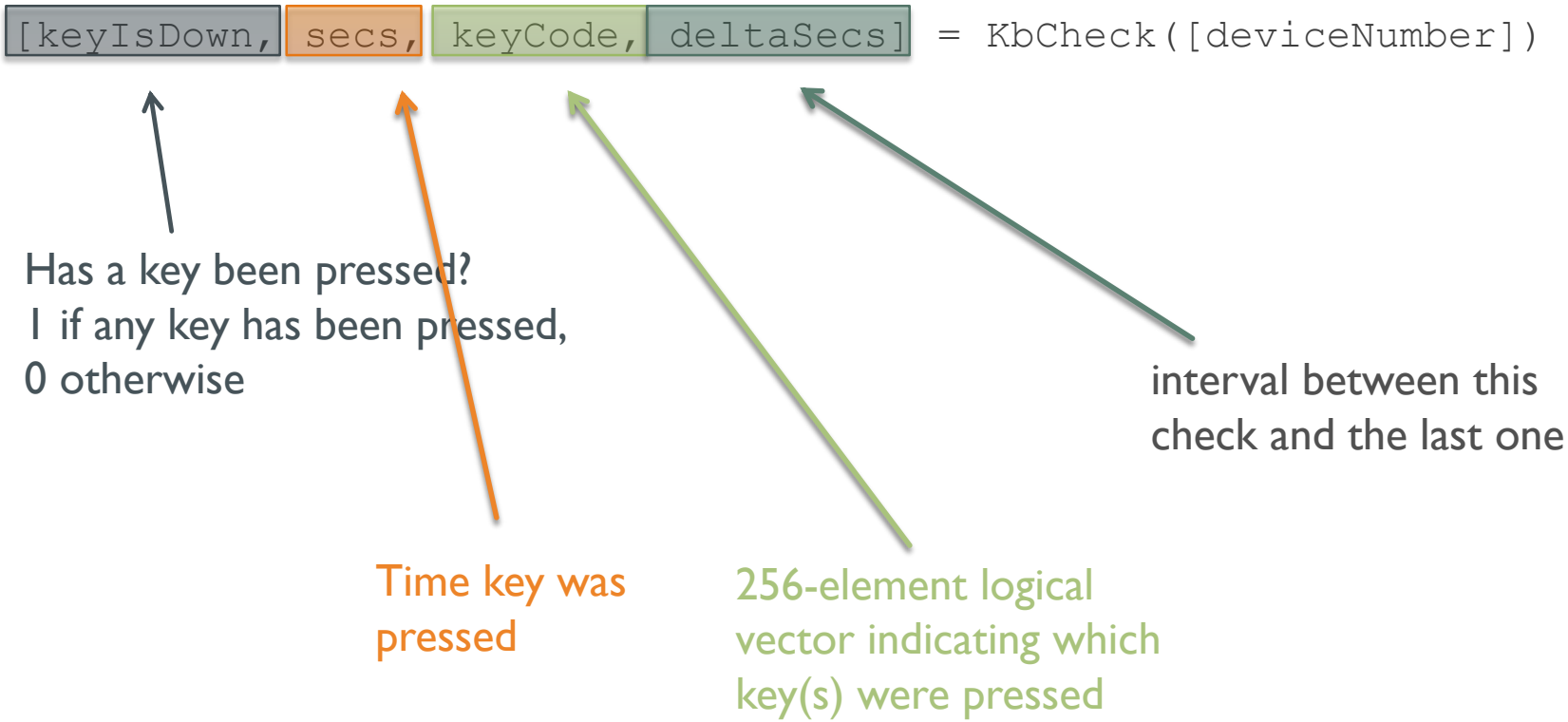
[secs, keyCode, deltaSecs] = KbWait([devicenumber] [, forWhat = 0][, untilTime=inf)



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
    Screen('Preference', 'SkipSyncTests', 1);
    WaitSecs(.5);
    startTime = GetSecs();
    keyIsDown = 0;

    [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);
        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
```

IGNORING RESPONSES

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

vector of key codes
to ignore

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

vector of key codes
to include

```
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

MOUSE CONTROL

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

mouseControl.m

PRACTICAL PSYCHOPHYSICAL EXP.

SimonDemo.m

SimonEffectPsychtoolbox.m