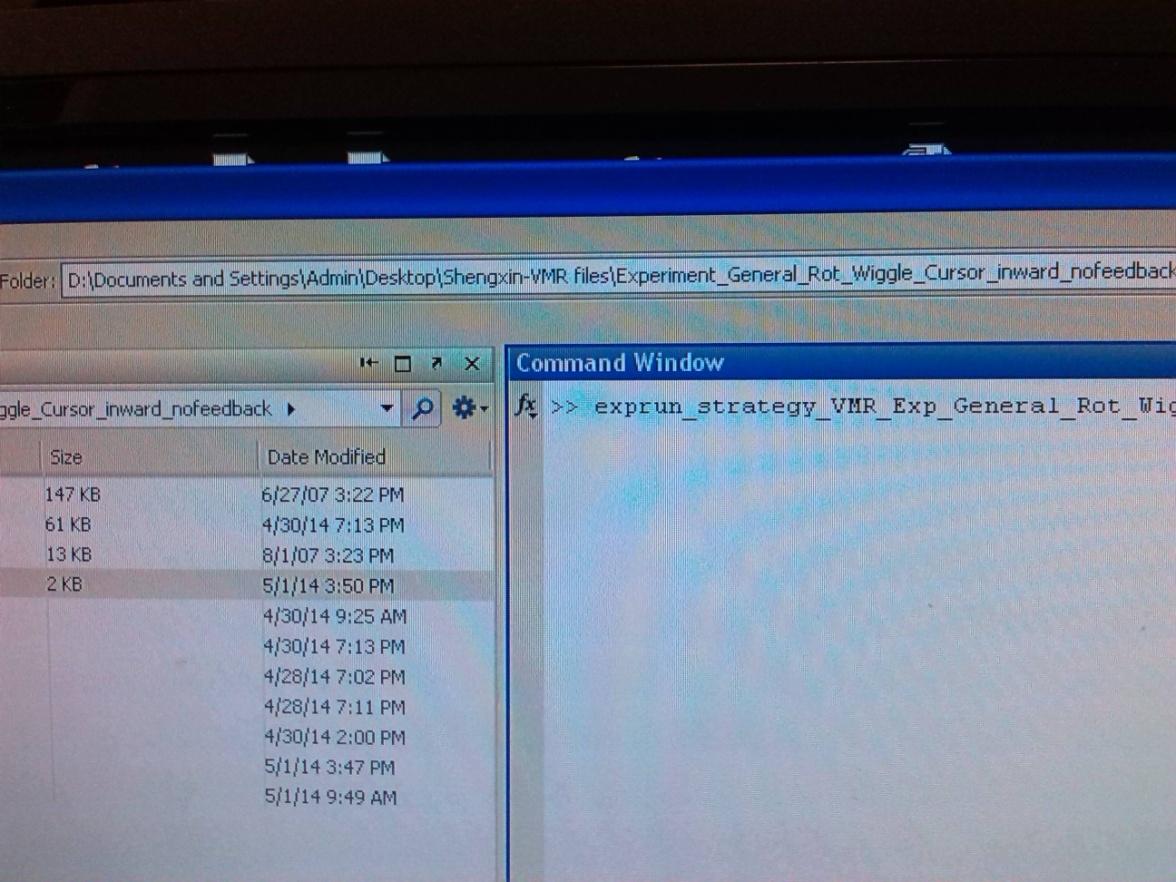
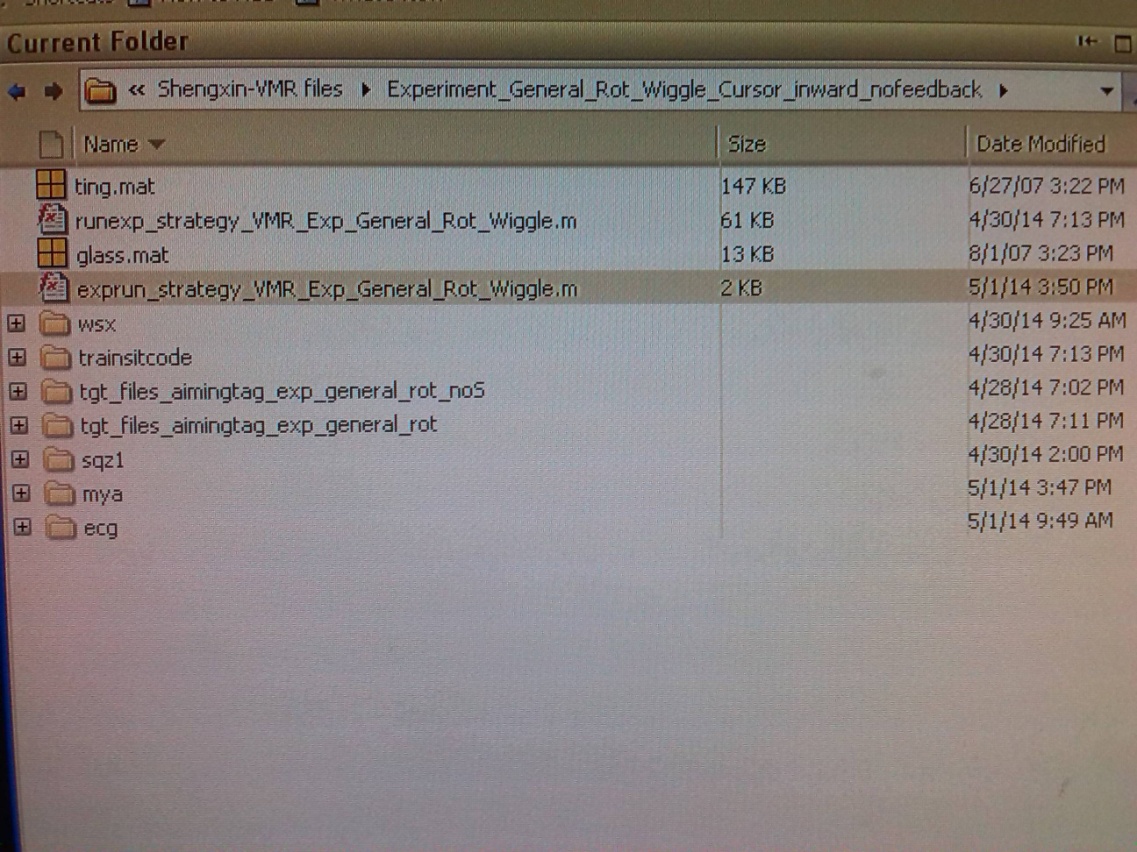
**Notes:**

“**Would you like to continue (Y/N)?** appears on the screen and means the end of experiment.

**File path:**

D:\Documents and Settings\Admin\Desktop\ Shengxin-VMR files\**Experiment\_General\_Rot\_Wiggle\_Cursor\_inward\_nofeedback**

****

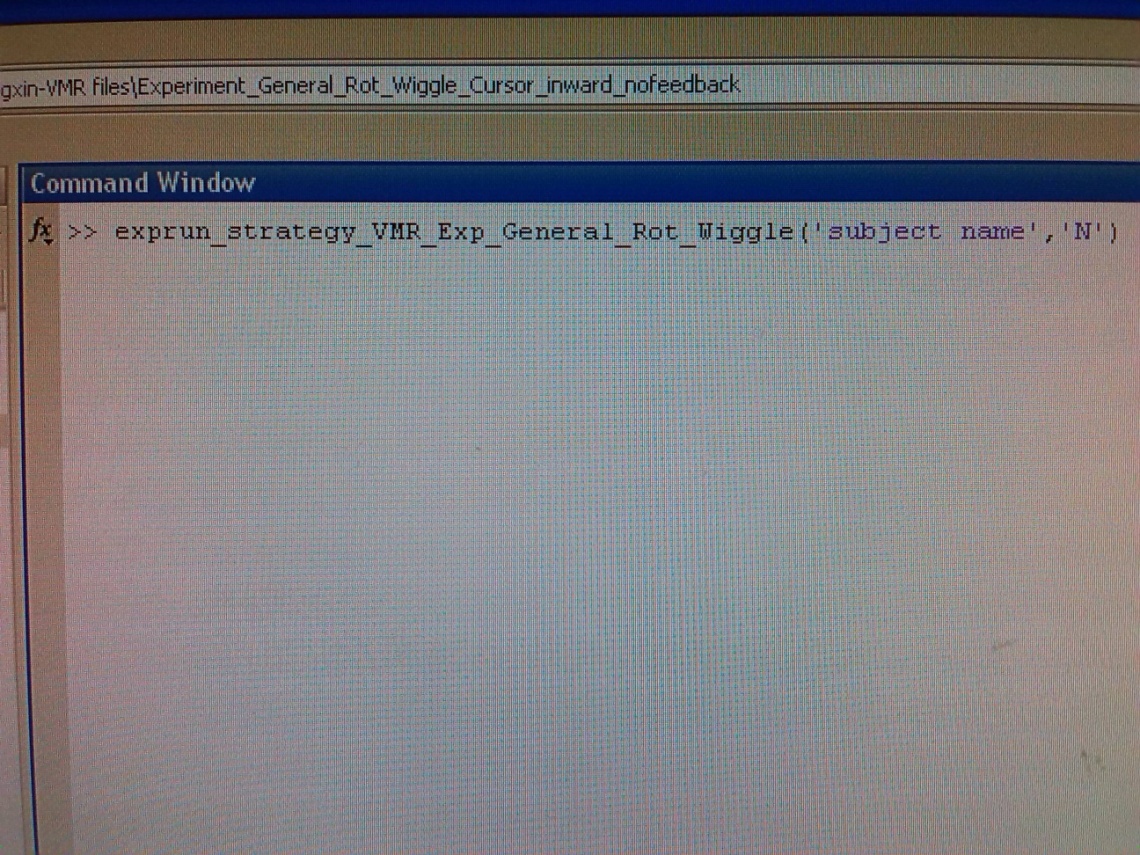
****

**Name of command run in MATLAB**

exprun\_strategy\_VMR\_Exp\_General\_Rot\_Wiggle (**'subjects’ name'**, 'N')

You only need to change the subjects’ name.

‘N’ means negative rotation and should not be changed in the two experiments for these two subjects (8:00-9:40 and 12:00-13:20)



Before start the experiment, you have to record the information of subject in the file strategy.xlsx which is on the Desktop

**Something may happen:**

If you stop experiment before block: Baseline 2, you should comment the code before Baseline 2 in the code file “exprun\_strategy\_VMR\_Exp\_General\_Rot\_Wiggle”, when you start again.

WinTabMex(0, windowHandleOrNameString [, show=1]);

- Initialize connection to tablet. You must provide either a Psychtoolbox onscreen window handle,

or the name of the titlebar of another window, e.g., the Matlab window, in 'windowHandleOrNameString'.

The optional 'show' flag if provided, can be set to 1 (default) to show, or zero to hide the window.

After a successfull init, you can call the enable function below to start data collection.

WinTabMex(1); - Shutdown the driver, close connection to tablet.

WinTabMex(2); - Clear all pending tablet events and start data collection / recording into the event queue.

WinTabMex(3); - Stop data collection / recording into the event queue.

[old, new] = WinTabMex(4 [, reqSize]); - Report, and optionally try to resize the internal event queue to 'reqSize' entries.

Report queue size before/after resize in 'old' and 'new'. You must check 'old' and 'new' for a valid size, as

this method can fail if you asked for too much. If it failed, retry with smaller 'reqSize' until success.

pkt = WinTabMex(5); - Try to fetch the oldest tablet event packet from the internal event queue.

This polls for available data, returning at most one packet from the queue:

If no new data is available, it will return an empty matrix, aka isempty(pkt) will be true.

If data is available, it will return a 8-by-1 double vector with the following elements in the different rows:

1 = x-position, 2 = y-position, 3 = z-position (in units of 1/1000th of centimeters).

4 = Button state: If you do button = uint32(pkt(4)), you'll get an integer: Each bit in the integer describes

state of one button. Mapping is device specific: E.g., bitget(button, 3) would tell you the state of the 3rd button.

5 = Serial number. A monotonically increasing serial number. Should increase without gap, otherwise you lost events!

6 = Timestamp. Time of when this tablet event was detected, in units of milliseconds. Baseline and accuracy are unknown.

7 = Status: An encoded status code: status = uint32(pkt(7)). Then bitget(status, 1) will be non-zero if the tool has left

the active area of the tablet. bitget(status, 2) will be non-zero if packets were lost due to queue-overflow.

bitget(status, 3) will be non-zero if the pen is at the margin of the active area.

8 = Changed: Info about what changed in this packet wrt. to previous packet: changed=uint32(pkt(8));

bitget(changed, 2) means that the 'Status' field has changed. bitget(changed, 7) == button changed,

bitget(changed, 8:10) means x, y or z position has changed.

9 = Normal pressure: Info about the pressure along the tablet surface normal, or pressure on the pen tip, in arbitrary units.