Psychtoolbox use on BIAC computers (and scanner)

Even if you are familiar with Psychtoolbox, getting it to work on the BIAC computers presents a few challenges. Here is a list of some of the things lab members have experienced so that we don’t have to reinvent the wheel every time.

1. Getting PTB to load at all:
   If you try to run your tested PTB code on a BIAC computer without any changes, you will notice that simple PTB commands will fail. For example, AssertOpenGL or a simple OpenWindow using the Screen function may throw errors. It turns out that PTB needs to be set up every time you open MATLAB. You can either do this by navigating to the Psychtoolbox folder on Munin, or add the following two lines of code to your program. If you have many runs that use the same function call, you can also make this a conditional so it only sets up PTB on the first run.

   ```matlab
   addpath('\Munin\Data\Programs\MATLAB\PsychToolbox\3.0.11');
   BIACSetupPsychtoolbox;
   ```

2. Having a trigger from the scanner start your experiment:
   Various lab members have had a lot of trouble with this, especially because BIAC5 and BIAC6 seem to work differently. However, code provided to us by Chris Petty seems to work on both scanners. Near the top of your code, you need to make sure you can access and send information to the daq (data acquisition device). To do that, put in the following lines:

   ```matlab
   try
daq = DaqDeviceIndex();
catch
    error('Daq device not found');
   end
   ```

   Then, right before the start of your experiment, you can paste the following code. This will take the number that the counter is at (which counts the number of TTL pulses) and wait for another to be sent out before continuing. Note that there is a pulse per slice acquisition.

   ```matlab
   curcount = DaqCIn(daq);
   while 1
     if DaqCIn(daq) > curcount
       % start your task
       break
     else
       pause(.05)
       % do short sleep here just so you’re not executing
       % the counter check a billion times
     end
   end
   ```

3. Recording onset and duration of events:
   To do fMRI analysis, it is important to know the onset time and duration of each event that occurs in your task. There are a few easy ways to do this with PTB. Firstly, if you want to manually get the time whenever you want, you can use the command GetSecs; to retrieve the current computer time. You should do this near the beginning of your task and after your disdaqs to get a sense of your start time; this start time can be subtracted out of all other timing calls in the future to get a relative time (in terms of start of experiment/scan) instead of the absolute computer time. You can also use timing output from the command Screen('Flip'). There are three timing outputs from this command; the second is the one that is closest to stimulus presentation. So, using the line: `[~ onsetTime] = Screen('Flip');` will give you the time at which the stimulus shown on the next screen is displayed (in the same format as GetSecs, so the absolute computer time). You can use this to get specific onsets of stimulus presentations. Note that both of these will be in milliseconds, so divide the output by 1000 for seconds.

   Similarly, you can use either of these two methods to get durations—simply get the timing after an event goes away (using either method) and subtract the onset time to get the duration. Both onset and duration are important for fMRI analysis.

4. Using the button box in the scanner:
   If you’re using the button box(es) for responses in the scanner, it is useful to know that the buttons do not linearly increase from left to right. If you’re only using one box, specifically, the right one, you should be fine, but the problem comes when you try to use both. Here is the coding for the buttons, from left to right.

   ```matlab
   Left hand: 9 8 7 6, Right hand: 1 2 3 4
   ```
   Also, the buttons register as if they are the numbers on the top of the keyboard, so all of them will have a KbName of 1!, 2@, etc. Also, the button presses are extra sticky in the scanner. This isn’t a problem usually when you’re looking for a response in a certain time window, but it can be an issue if you have responses advancing your screens. Add in extra wait times after those.

Other thing of note: the projected screen into the scanner is quite small, pixelwise, so try to leave extra space around your words/images when testing elsewhere. Hopefully that proved at least a little helpful! Good luck!