PsyScript tutorial 2: Face continued

This tutorial covers the following subjects:

- the 'Commands' button
- more and better logging
- the 'text' stimulus
- looping
- blank lines
- providing feedback to the subject

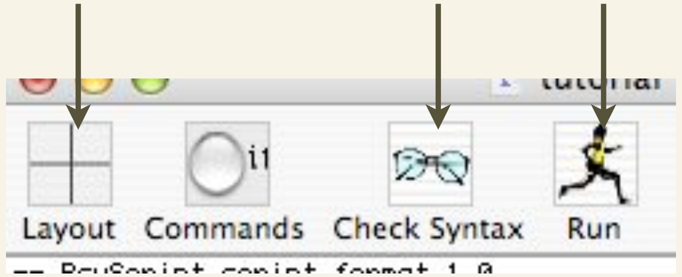
Prerequisites

This tutorial won't make much sense unless you've completed tutorial 1.

Logging reactions

In the first tutorial you produced a script which recorded two details about a click: what the subject clicked on and how long they took to do so. That's a great start for logging results but it's not really good enough for professional work. PsyScript allows you to log a number of other things that happen in a script and that's what we're going to look at next.

To get there, you're going to learn about the 'Commands' button. You used three buttons in the last tutorial:

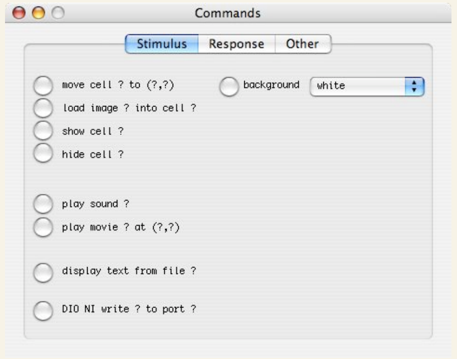


Now you learn about the button you haven't used yet: the 'Commands' button.

Open the script you saved when you did the first tutorial and click the 'Commands' button.

The commands panel

You should see a window that looks something like this:



```
move cell ? to (?,?)
load image ? into cell ?
show cell ?
hide cell ?

play sound ?
play movie ? at (?,?)

display text from file ?

DIO NI write ? to port ?
```

background: white

This window shows you all the commands you can use in your script. You don't need a manual or reference book to see them all, they're all right there, including the syntax for each command.

You can see from the top of the window that the commands are split into three categories: Stimulus, Response and Other. If you click on each of those words you'll see the commands useful for that type of thing. So far you've used the first three stimulus commands (all three needed to display an image) and two of the response commands.

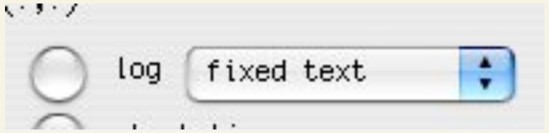
Some commands have question marks in in places. These indicate a piece of the command which changes depending on what you want the command to do. Parts of other commands are listed in square brackets. These parts of the command are only used some of the time, other times they're not needed and can be left out.

Your log file currently has only the name of a cell and a number of seconds in. It should really log a lot more about the experiment, perhaps which subject was being tested and the time and date of the experiment. So you're going to insert those extra commands.

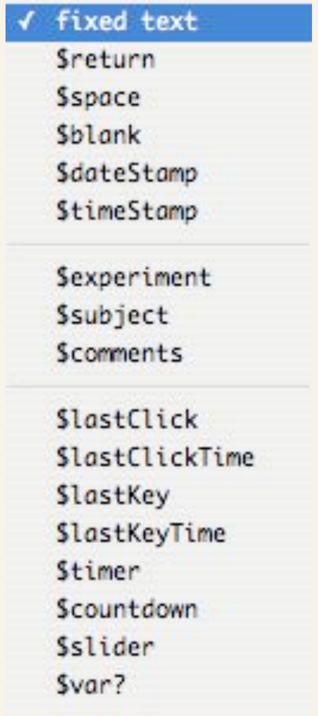
First, make sure you're working at the right place in the script. Bring your script window to the front and put the text cursor at the beginning of the 'move' line. You should see a flashing caret there.

```
proc main
|  move cell A to (-98,68)
```

Now bring the 'Commands' window to the front and click on the 'Response' tab so you can see the log command.



We don't want to log fixed text at the moment, we want to log the experiment name and the subject name. So click on the popup menu where it says fixed text and look at the options:



These six logable items aren't related to the subject's reactions. The first one allows you to log any fixed text you want. The next three are used for formatting the log file. The last two find out the date and time.

You get a chance to type in each of these three things when the experiment is finished. You'll almost always use `$subject` and may or may not use the others.

These things all change depending on how the user reacted to the stimuli you presented. These are the main things you'll want to use in a log file. You've already used two of these in your script: the top two.

I think you need the experiment and subject code, then you should log the time and date. Since it's just a tutorial, feel free to vary this if you have a better idea.

So first pick `$experiment` from the list and copy `log $experiment` from the window into your script. Then do the same thing for `$subject`, and then for $return. The `$return` item makes the log file start a new line. It's useful for splitting up responses as you'll see later.

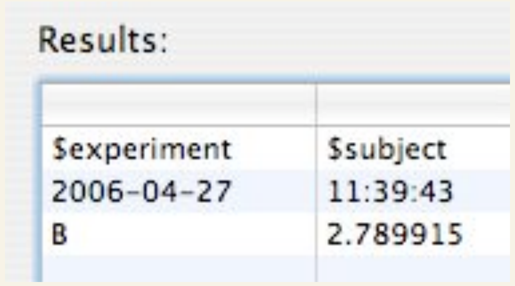
Then log `$dateStamp`, `$timeStamp` and another `$return`. You're done with the 'Commands' window now so you can close it. Put a blank line (blank lines are ignored) to the script to separate the logging from the image commands and your main procedure should now begin

```
proc main
  log $experiment
  log $subject
  log $return
  log $dateStamp
  log $timeStamp
  log $return

  move cell A to (-98,68)
```

okay ?

Save your script and run it, click on an eye, and see what you get. I got



$experiment	$subject
2006-04-27	11:39:43
B	2.789915

Notice that the `$experiment` and `$subject` haven't been replaced with the real details: it doesn't know what to put there yet because you haven't typed it in. You can type those things in in the lower left corner of the dialogue and if you save the results into a file you'll see that the file contains the details you typed in.

You now know what all four of the buttons are for, and you now know how to log all the details of your experiments. You also know how to find the syntax commands I haven't explained yet. That's pretty-much all you need. From now on, it's all minor details.

The 'text' stimulus: giving instructions

A useful step in designing an experiment is to work out what instructions you want to give to the subject. Our experiment at the moment doesn't show any instructions at all, but it would be nice if it did: it would save the experimenter from having to give a little speech to each subject before they begin the experiment. So let's put that in.

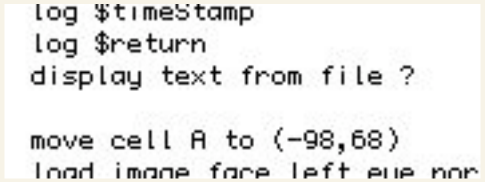
I've made up a text file with the instructions in for you. If you open the file called `clickOnEye.txt` you should see what it says. There's a convenient PsyScript command which does everything you need to display a text file: the `display text` command. You're going to use the Command window to put that in.

First, let's get the cursor into the right place: it should go before the log commands at the beginning of the script, but after the `proc main` line which tells PsyScript where the main procedure starts.

Now let's get the windows we'll need. Move your windows around until you can see both the Command window and the script window. Now make sure that the Command window is at the front (if it isn't, click on it). Then make sure you've selected the 'Stimulus' tab of the command window, so you can see the `display text` command.

Next, get the cursor to the right place. You're going to want to insert the new command immediately after the setup instructions, but before the first move cell instruction, so put the text cursor at the beginning of the blank line separating the two.

The instruction you're going to use is the `display text from file ?` one. As you can see there's a question mark in it. This indicates a part of the command that you'll have to fill in yourself because the program can't know what you want there. But you can copy most of the command out by pressing the round button next to it. Click on that button and the command will be inserted into your script, wherever the cursor currently is.



```
log $timeStamp
log $return
display text from file ?

move cell A to (-98,68)
load image face left eye nor
```

The program needs to know the name of the file you want it to use, and the question mark indicates where you put that name. Select the question mark and replace it with the name of the file you want to use:

```
display text from file clickOnEye.txt
```

If you select the icon in the Finder and do a 'Copy' then you can paste the name into the script instead of having to type it !

You're finished with the command window for now so you can close it. Then run your script to test it.

The experiment should now start off by displaying the contents of the text file. You'll see a scroll bar on the right side of the text: if there is so much text in the text file that it can't be displayed on one page then the scroll bar will be enabled and you'll be able to scroll through the text. Hit the 'OK' button when you're finished reading and the experiment should continue.

Although the `display text` instruction should save a lot of effort when giving simple instructions to the subject, you can't change the format of the text in this window: no bold text, no underlining, no change in fonts or font size. If you need anything but plain black text, make a picture of the text however you want it to appear and use image instructions to display it.

Looping

So far you've logged the results of one click. Not a very inspiring experiment: most experiments require the subject to do a little more than that. One way of extending the experiment would be to repeat the 'wait for a click and log it' sequence in the script however many times you wanted it to work:

```
wait for a click in AB
log $lastClick
log $lastClickTime
log $return

wait for a click in AB
log $lastClick
log $lastClickTime
log $return

wait for a click in AB
[…]
```

but that's not very interesting. Instead you're going to change the experiment a bit. At the moment it just waits for the subject to click on one eye, then stops. We're going to let the subject click on any of the four pictures, keep going if he or she clicks on an eye, but stop if he or she clicks on the nose or mouth.

To make this happen you're going to use a command structure: 'repeat while'. First find the line that reads

```
wait for a click in AB
```

and change it to

```
wait for a click
```

You need this change because now you want the program to register all clicks: nose (C) and mouth (D) as well as the eyes (A and B). Once you've done this you need to put a 'repeat while' loop around the logging part, so find the three log commands:

```
log $lastClick
log $lastClickTime
log $return
```

and surround them with a loop:

```
repeat while $lastClick is in AB
  log $lastClick
  log $lastClickTime
  log $return
end repeat
```

We want the script to keep repeating something while the subject is clicking the eyes. As soon as the subject clicks on something else, the `repeat while` loop will stop working.

You could pick that `repeat while` command up from the 'Commands' window if you liked. It's in the 'Other' section. It happens that I remembered the syntax so I just typed it in.

Don't worry too much about the indentation (the amount of space to the left of each instruction). If you click on the 'Check Syntax' button (the one that looks like a pair of glasses) PsyScript will make everything line up correctly for you. Since the `log` commands are inside the loop, it will indented them further than the loop instructions.

If you look at the script now you'll find there's one thing left to do: once PsyScript gets into the loop it'll go around forever logging results. You don't want that. You want to make sure that after logging the first click it asks for another click before checking to see if it's finished yet. So change this

```
wait for a click
repeat while $lastClick is in AB
  log $lastClick
  log $lastClickTime
  log $return
end repeat
```

to this

```
wait for a click
repeat while $lastClick is in AB
  log $lastClick
  log $lastClickTime
  log $return

  wait for a click
end repeat
```

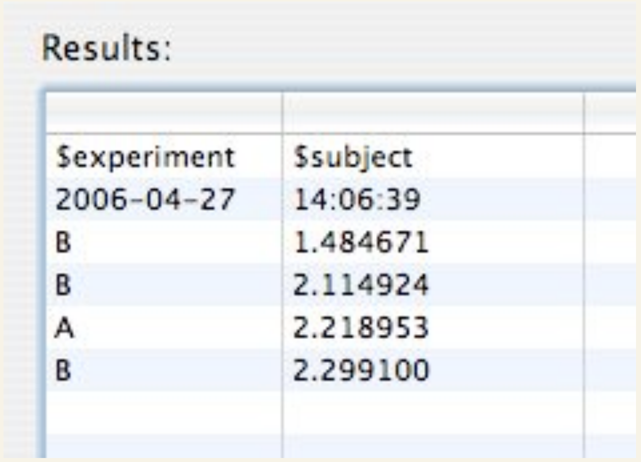
The blank line is ignored, it just groups the commands so you can see what's going on. You could insert a comment line as well: any line which starts with -- (two minus signs) is a comment line: you can put one whatever you like and PsyScript will skip right past it.

```
wait for a click
repeat while $lastClick is in AB
  log $lastClick
  log $lastClickTime
  log $return

  -- We've dealt with the last click; now let's get another.
  wait for a click
end repeat
```

Remember that if things don't look like they're indenting properly, click the 'Check syntax' button and PsyScript will do it for you.

I think you're ready to try this new program now. It's a good idea to save your script first, just in case something bad happens. And remember that you won't be able to tell you've clicked on anything: your script will keep running until you've clicked on the nose or mouth. I got


Results:

$experiment	$subject
2006-04-27	14:06:39
B	1.484671
B	2.114924
A	2.218953
B	2.299100

You can see that the last click wasn't logged (and you can see why by reading the script).

This experiment could actually be used for something: you could see how fast you can click, or how accurately you could judge a gap of five seconds.

Feedback

I noticed two things wrong with the experiment. The first was that the instructions are now wrong. I'll let you put those right yourself: you can either edit the existing instrucution file or make a new one, put the new instructions into it, and change the 'display text' command in the script to display the new file instead of the old one.

The second thing was that I couldn't really tell when I'd clicked on an eye. It would be nice to have some sort of feedback. I might make the script play a sound but we're going to use those in another tutorial – this one's about images. So I think I'll make the eyes change shape instead. I've provided one alternative picture for each eye. Mine are just narrower eyes but you could change them to be winks or bloodshot eyes or something.

The first decision is where in the script this should happen. Ideally it should be as soon as the script has decided that one of the eyes has been clicked. That would be immediately under the 'repeat' line. The first thing to do would be to figure out which eye was clicked. You can use an 'if' command to do this. Pick it out from the 'Commands' window (it's in the 'Other' section) or just type it: change this:

```
repeat while $lastClick is in AB
  log $lastClick
```

to this:

```
repeat while $lastClick is in AB
  if $lastClick is A
   else
  end if

  log $lastClick
```

(I added the blank line myself). What you can see here is a way of distinguishing between clicks on the A image and clicks on the B image. The 'repeat until' line makes sure that clicks on C or D never get into the loop so that leaves only A and B. The A clicks will make PsyScript execute the commands between the 'if' and the 'else' lines, and the only thing that's left – B clicks – will make PsyScript execute the commands between the 'else' and 'end if' lines.

So we want script lines to change the left eye above the 'else' and script lines to change the right eye below it.

How to provide the feedback ? Well, change the picture, wait a little while to let the subject see the change, then change it back again. Here's the code:

```
 if $lastClick is A
     load image face left eye narrow.png into cell A
     wait for .5 seconds
     load image face left eye normal.png into cell A
    else
     load image face right eye narrow.png into cell B
     wait for .5 seconds
     load image face right eye normal.png into cell B
   end if
```

Put the six new lines into the right places in the script and try it. And if it works, save your changes and take a well-earned rest. Oh, I provided a few more stimulus files in case you felt like playing with the tutorial more: there are some eyebrows to wiggle and a 'stop' button to use instead of making the subject click on the nose or mouth to stop.

Further

You now know how to write a script, how to present visual stimuli, and how to log results. You know where to find the rest of the commands and how to work out the correct syntax for them. If I tell you that the 'Help' menu actually does something useful you could probably work out the rest from there.

But there is another tutorial anyway. It introduces a command which plays sounds, the use of the keyboard (instead of the mouse) for allowing the user to respond to stimuli), and shows you how to use tables to make a list of stimuli. Your mission, should you decide to accept it, is to do tutorial 3 after a suitable break.