Homework 9

Do Q1 now: Q1 is posted now (10 points)

Q2 posted will be posted later this week (30 points)

due Mon Dec 5 in class

Homework 10

posted on/before Mon Dec 5

due Wed Dec 14 at noon

in lieu of a final exam

Homework 9

clarification: in general, when using PsychoPy,
if you are not using the mouse, it should be turned off
(then turned back on when you exit)

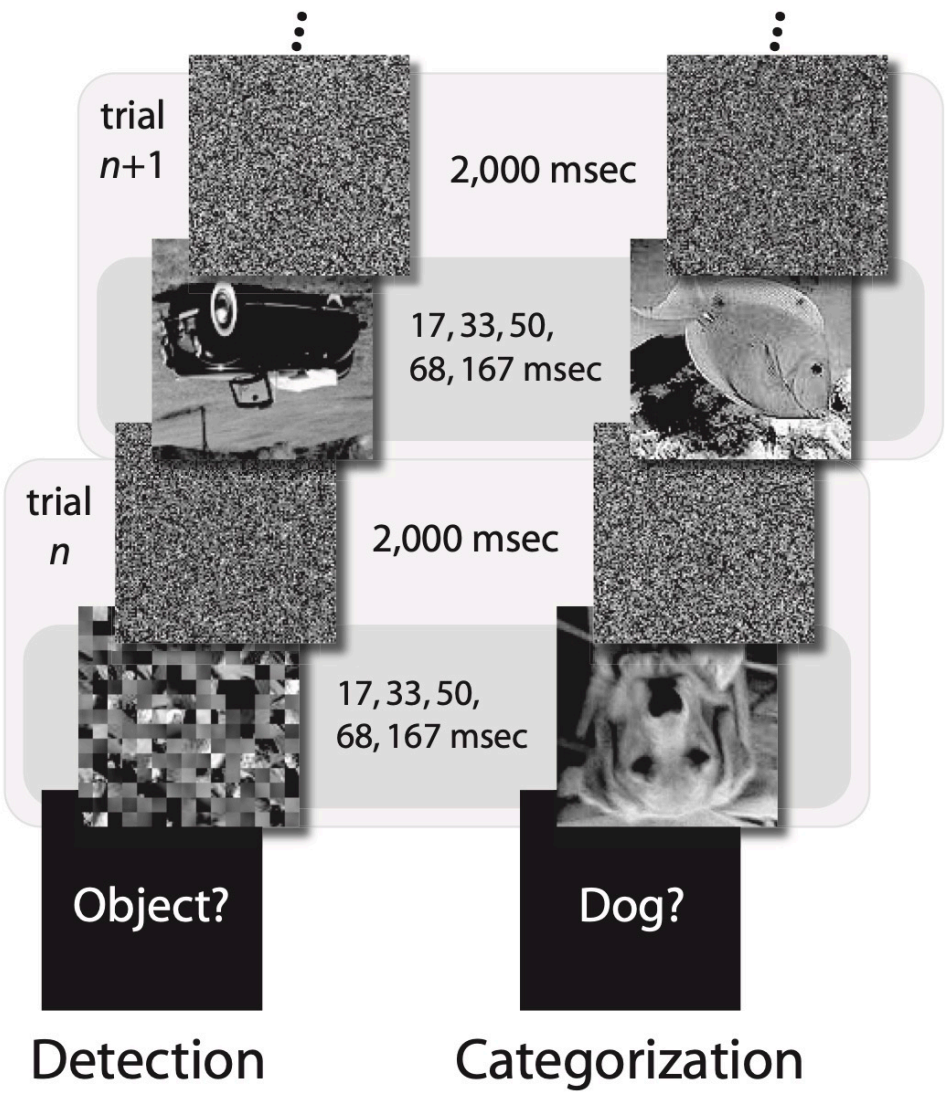
download from Brightspace

`Psychopy.zip` (Python files)

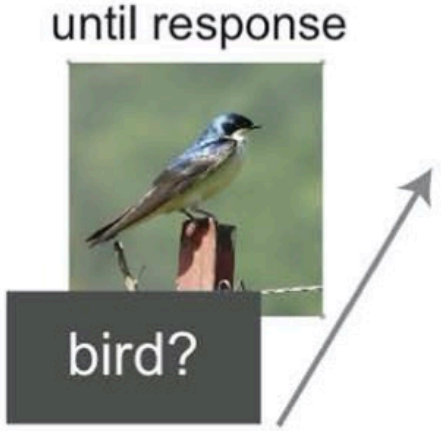
Timing

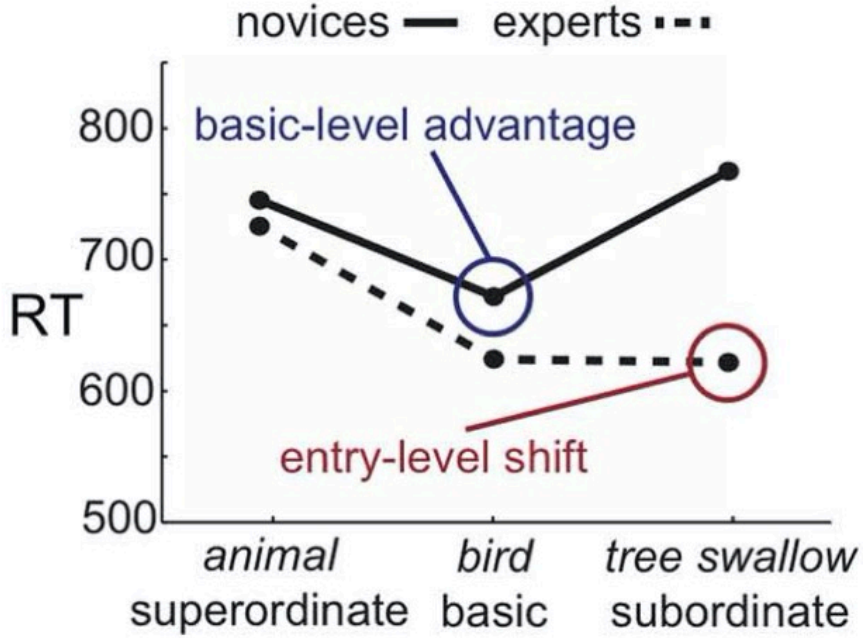
examples

1) manipulating time



2) measuring time





manipulating timing in PsychoPy

manipulating timing in PsychoPy

coarse timing

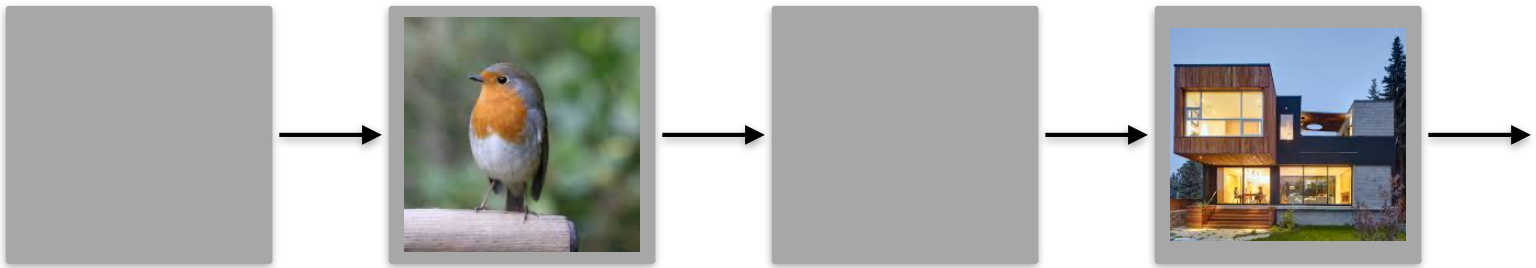
```
mywin.flip()

... do stuff ...
... write stuff to screen ...

core.wait(1.0)

mywin.flip()
```

fine for coarse timing between long events or pauses between trials



manipulating timing in PsychoPy

coarse timing

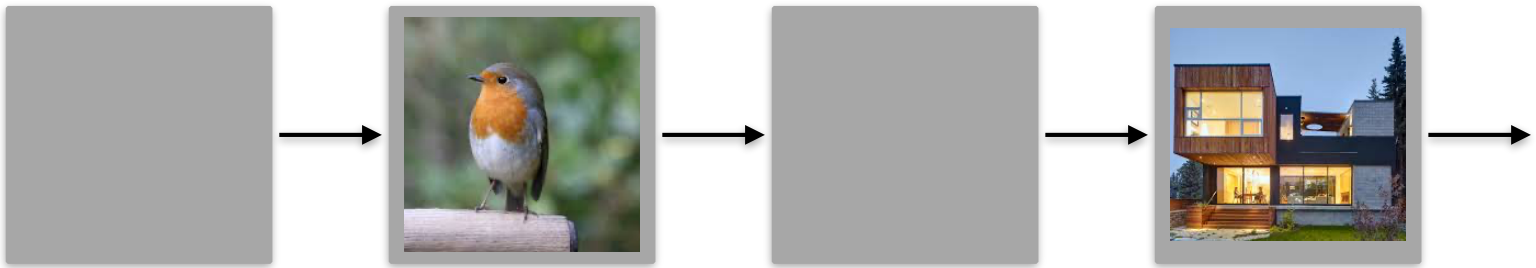
```
mywin.flip()

... do stuff ...
... write stuff to screen ...

core.wait(1.0)

mywin.flip()
```

cannot do anything while you `wait` (cannot change stimulus, cannot get a response, cannot do processing)



manipulating timing in PsychoPy

coarse timing

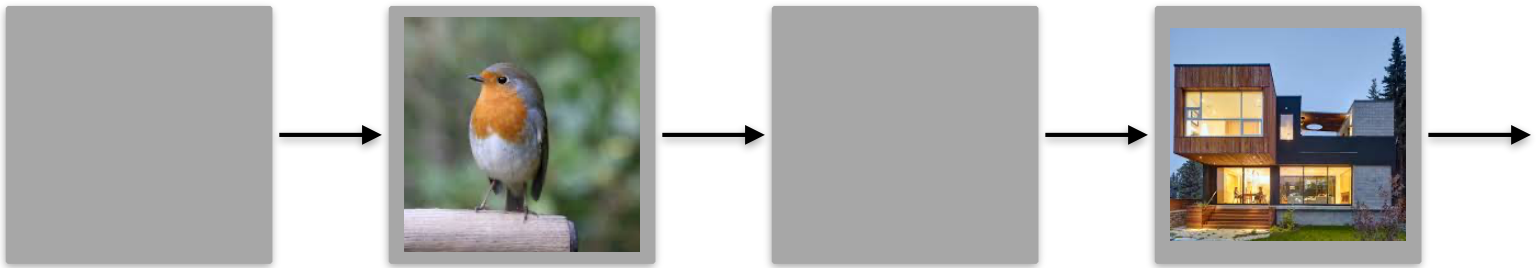
```
mywin.flip()

... do stuff ...
... write stuff to screen ...

core.wait(1.0)

mywin.flip()
```

also, a screen refresh only occurs based on the refresh rate
60Hz = 60 refresh per sec
is 1/60 sec per refresh
is 16.667 ms per refresh



manipulating timing in PsychoPy

coarse timing

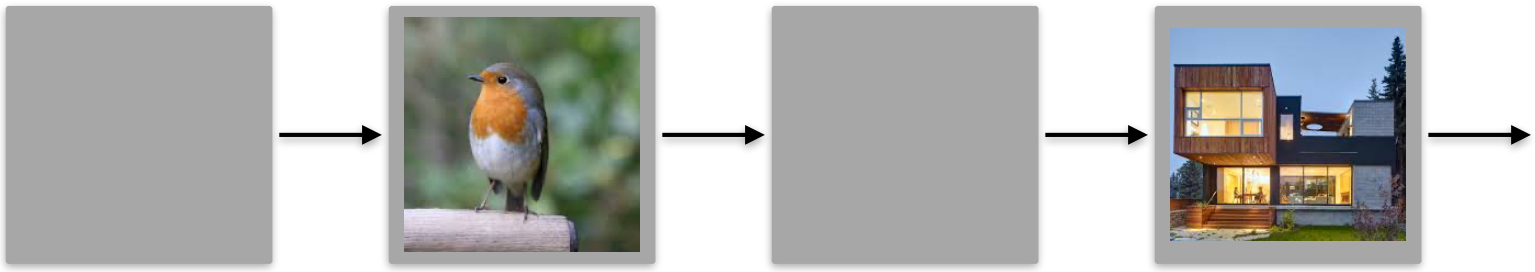
```
mywin.flip()

... do stuff ...
... write stuff to screen ...

core.wait(1.0)

mywin.flip()
```

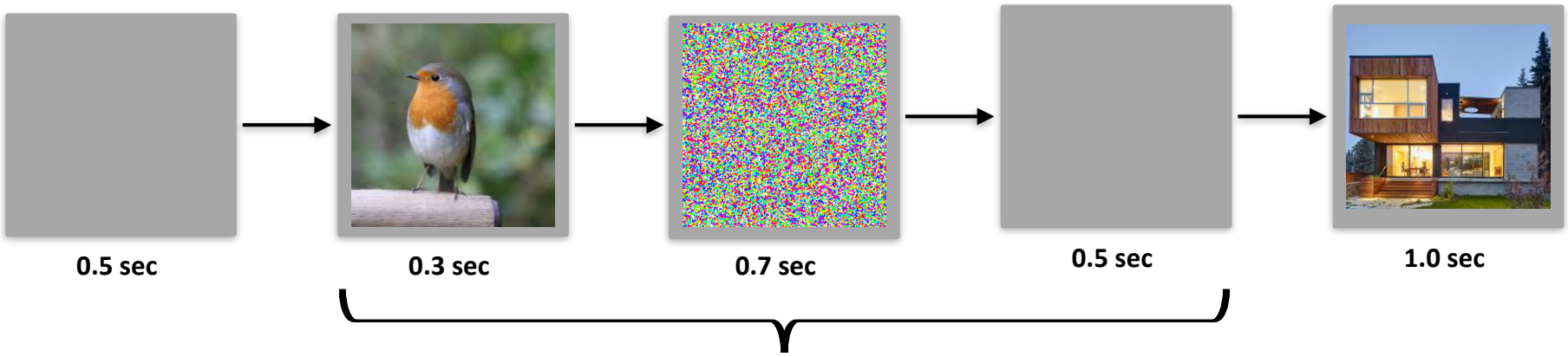
also, a screen refresh only occurs based on the refresh rate
120Hz = 120 refresh per sec
is 1/120 sec per refresh
is 8.333 ms per refresh



manipulating timing in PsychoPy

better timing (that also allows something else to happen)

imagine this an an old-new recognition memory test



respond *old* or *new* any time
during this interval

manipulating timing in PsychoPy

`test8.py`

clocks in PsychoPy

```
fixDur   = .5; stimDur  = .5; blankDur = 1.0

expClock   = core.Clock(); trialClock = core.Clock()

.
.
.

trialClock.reset()

.
.
.

while trialClock.getTime() < fixDur:
    core.wait(.0001)
```

clocks in PsychoPy

set event durations

```
fixDur   = .5; stimDur  = .5; blankDur = 1.0

expClock   = core.Clock(); trialClock = core.Clock()

.
.
.

trialClock.reset()

.
.
.

while trialClock.getTime() < fixDur:
    core.wait(.0001)
```

clocks in PsychoPy

```
fixDur   = .5; stimDur  = .5; blankDur = 1.0

expClock   = core.Clock(); trialClock = core.Clock()
```

create clocks

```
.
.
.

trialClock.reset()

.
.
.

while trialClock.getTime() < fixDur:
    core.wait(.0001)
```

clocks in PsychoPy

```
fixDur   = .5; stimDur  = .5; blankDur = 1.0

expClock   = core.Clock(); trialClock = core.Clock()

.
.
.

trialClock.reset()
```

reset a clock

```
.
.
.

while trialClock.getTime() < fixDur:
    core.wait(.0001)
```

clocks in PsychoPy

```
fixDur   = .5; stimDur  = .5; blankDur = 1.0

expClock   = core.Clock(); trialClock = core.Clock()

.
.
.

trialClock.reset()

.
.
.

while trialClock.getTime() < fixDur:
    core.wait(.0001)
```

simple use of a clock

clocks in PsychoPy

```
fixDur   = .5; stimDur  = .5; blankDur = 1.0

expClock   = core.Clock(); trialClock = core.Clock()

.
.
.

trialClock.reset()

.
.
.

while trialClock.getTime() < fixDur:
    core.wait(.0001)
```

avoid overtaxing your computer while maintaining ms timing

logging in PsychoPy

printing to the console can be time consuming, so instead it is recommended to use PsychoPy logging

```
from psychopy import logging

logging.console.setLevel(logging.WARNING)
logging.setDefaultClock(expClock)
lastLog = logging.LogFile('logfile.log',
                level=logging.INFO, filemode='w')

mywin.logOnFlip(level=logging.EXP, msg='msg')
mywin.flip()
```

sets up logging

https://www.psychopy.org/api/logging.html

logging in PsychoPy

printing to the console can be time consuming, so instead it is recommended to use PsychoPy logging

```
from psychopy import logging

logging.console.setLevel(logging.WARNING)
logging.setDefaultClock(expClock)
lastLog = logging.LogFile('logfile.log',
                level=logging.INFO, filemode='w')

mywin.logOnFlip(level=logging.EXP, msg='msg')
mywin.flip()
```

particularly useful logging

https://www.psychopy.org/api/logging.html

timing for `test8a.py`

```
experiment start:  5.75000e-06
fix on 0:          0.0177
fix off 0:         0.5139
stimulus on 0:     0.5139
stimulus off 0:    1.0359
blank end 0:       2.0511
-----------------------------------
fix on 1:          2.0511
fix off 1:         2.5472
stimulus on 1:     2.5472
stimulus off 1:    3.0693
blank end 1:       4.0809
-----------------------------------
fix on 2:          4.0809
fix off 2:         4.5806
stimulus on 2:     4.5806
stimulus off 2:    5.1026
blank end 2:       6.1080
-----------------------------------
```

0.5000 { 0.0000 – 0.5000
0.5000 { 0.5000 – 1.0000
1.0000 { 1.0000 – 2.0000
2.0000 – 2.5000
2.5000 – 3.0000
3.0000 – 4.0000
4.0000 – 4.5000
4.5000 – 5.0000
5.0000 – 6.0000

not exactly 0.5sec

timing for `test8a.py`

```
                experiment start:  5.75000e-06
0.5000  0.0000  fix on 0:          0.0177
        0.5000  fix off 0:         0.5139
0.5000  0.5000  stimulus on 0:    0.5139
        1.0000  stimulus off 0:   1.0359
1.0000  2.0000  blank end 0:      2.0511
                ---------------------------------
        2.0000  fix on 1:          2.0511      not start at 2sec
        2.5000  fix off 1:         2.5472
        2.5000  stimulus on 1:    2.5472
        3.0000  stimulus off 1:   3.0693
        4.0000  blank end 1:      4.0809
                ---------------------------------
        4.0000  fix on 2:          4.0809
        4.5000  fix off 2:         4.5806
        4.5000  stimulus on 2:    4.5806
        5.0000  stimulus off 2:   5.1026
        6.0000  blank end 2:      6.1080
                ---------------------------------
```

timing for `test8a.py`

```
                experiment start:  5.75000e-06
0.0000          fix on 0:          0.0177
0.5000          fix off 0:         0.5139
0.5000          stimulus on 0:     0.5139
1.0000          stimulus off 0:    1.0359
2.0000          blank end 0:       2.0511
                ---------------------------------
2.0000          fix on 1:          2.0511
2.5000          fix off 1:         2.5472
2.5000          stimulus on 1:     2.5472
3.0000          stimulus off 1:    3.0693
4.0000          blank end 1:       4.0809
                ---------------------------------
4.0000          fix on 2:          4.0809
4.5000          fix off 2:         4.5806
4.5000          stimulus on 2:     4.5806
5.0000          stimulus off 2:    5.1026
6.0000          blank end 2:       6.1080
                ---------------------------------
```

0.5000 = 0.0000 → 0.5000

0.5000 = 0.5000 → 1.0000

1.0000 = 1.0000 → 2.0000

timing errors growing

timing for `test8.py`

desired timing

actual timing

for example, in an fMRI experiment, you either need to sync the experiment with when brain scans are done or make sure that the timing of the experiment lines up with the timing of the scans

relative vs. absolute timing

relative timing **(relative to previous event)** *likely to accumulate timing errors*

absolute timing **(from the start of the experiment)**

absolute timing in PsychoPy

`test9.py`

test9.py

```
# initial (blank) flip to sync up timing better
mywin.flip()
expClock.reset()

for i in range(len(fnames)*Nrep):
    # draw fixation
    fixation.draw()

    # wait until start of trial
    while expClock.getTime() < expDelay + i*trialDur:
        core.wait(.0001)

    mywin.logOnFlip(level=logging.EXP, msg=f'fix on {i}')
    mywin.flip()

    # get read to draw image to screen after fixation
    mypic[i % len(fnames)].draw()

    # wait for fixation duration
    while expClock.getTime() < expDelay + i*trialDur + fixDur:
        core.wait(.0001)

    # display image
    mywin.logOnFlip(level=logging.EXP, msg=f'fix off / stimulus on {i}')
    mywin.flip()

    while expClock.getTime() < expDelay + i*trialDur + fixDur + stimDur:
        # later we will add looking for responses
        core.wait(.0001)

    # image off
    mywin.logOnFlip(level=logging.EXP, msg=f'stimulus off {i}')
    mywin.flip()
```

```
# duration parameters
fixDur   = .5
stimDur  = .5
blankDur = 1.0
trialDur = fixDur+stimDur+blankDur
```

blank period

fixation point on

stimulus on

absolute timing in PsychoPy

```
experiment start:  2.50000e-06
fix on 0:          0.01621
fix off 0:         0.50006
stimulus on 0:     0.52186
stimulus off 0:    1.02184
---------------------------------
fix on 1:          2.02174
fix off 1:         2.50015
stimulus on 1:     2.52178
stimulus off 1:    3.02211
---------------------------------
fix on 2:          4.02197
fix off 2:         4.50004
stimulus on 2:     4.52221
stimulus off 2:    5.02220
---------------------------------
```

still not perfect

absolute timing in PsychoPy

```
experiment start:  2.50000e-06
fix on 0:          0.01621
fix off 0:         0.50006
stimulus on 0:     0.52186
stimulus off 0:    1.02184
-----------------------------------
fix on 1:          2.02174
fix off 1:         2.50015
stimulus on 1:     2.52178
stimulus off 1:    3.02211
-----------------------------------
fix on 2:          4.02197
fix off 2:         4.50004
stimulus on 2:     4.52221
stimulus off 2:    5.02220
-----------------------------------
```

no growing errors

other timing challenges

you're running a precisely-timed experiment and in the middle

... dropbox starts syncing files

... the operating system starts a background task

... you left 1/2 dozen tabs open on Safari

... and had your email app open and it does a download

while on a multi-core system, it is less likely that multiple jobs will interfere, it is possible, leading to timing delays

experimental computers

- use full-screen PsychoPy rather than a window
- shut down all other applications
- remove background applications loaded at startup
- schedule virus scanners for overnight
- turn off power management options
- reboot computer from time to time
- have sufficient computer RAM (virtual memory is slow)
- use a real graphics card (ATI or NVIDIA) with sufficient video memory (VRAM)

MacOS

The stimulus presentation on MacOS used to be very good, up until version 10.12. In MacOS 10.13 something changed and it appears that a form of triple buffering has been added and, to date, none of the major experiment generators have managed to turn this off. As a result, since MacOS 10.13 stimuli appear always to be presented a screen refresh period later than expected, resulting in a delay of 16.66 ms in the apparent response times to visual stimuli.

Windows 10

In Windows, triple buffering is something that might be turned on by default in your graphics card settings (look for 3D, or OpenGL, settings in the driver control panel to turn this off). The reason it gets used is that it often results in a more consistent frame rate for games, but having the frame appear later than expected is typically bad for experiments!

Linux

In Linux, again, timing performance of the visual stimuli depends on the graphics card driver but we have also seen timing issues arising from the Window Compositor and with interactions between compositor and driver. The real complication here is that in Linux there are many different window compositors (Compiz, XFwm, Enlightenment,. . .), as well as different options for drivers to install (e.g. for nVidia cards there is a proprietary nVidia driver as well as an open-source “Nouveau” driver which is often the default but has worse performance). Ultimately, you need to test the timing with hardware and work through the driver/compositor settings to optimise the timing performance.

https://www.psychopy.org/PsychoPyManual.pdf (Sec 2.8)

experimental computers (for timing)

- use computers with Linux or Windows

- can put a Linux or Windows partition on a Mac (and boot into it) - do not want to use VMware or Parallels

confirm timings

- use a photometer to measure onset/offset of test stimuli

- (also used to test luminance, colors, when precision needed)

set priority in PsychoPy

priority determines what runs on a computer (what is given priority to run) - multiple users, multiple apps, multiple things running within an app, operating system functions, automated tasks, scheduled tasks

```
from psychopy.iohub import devices
```

```
p = devices.getPriority()
```

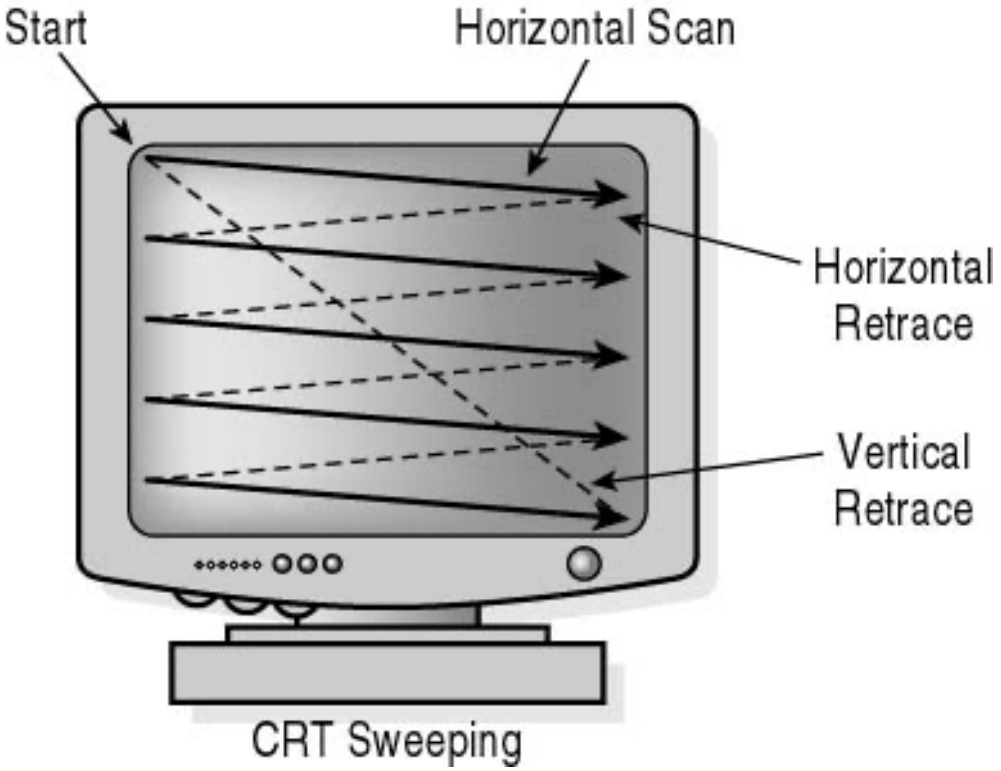
returns 'normal', 'high', or 'realtime'

```
devices.setPriority('high')
```

not supported on Mac OSX

https://www.psychopy.org/api/iohub/device/computer.html

monitor refreshes



Cathode Ray Tube (CRT)
literal sweep of electron beam
(refreshed per second)

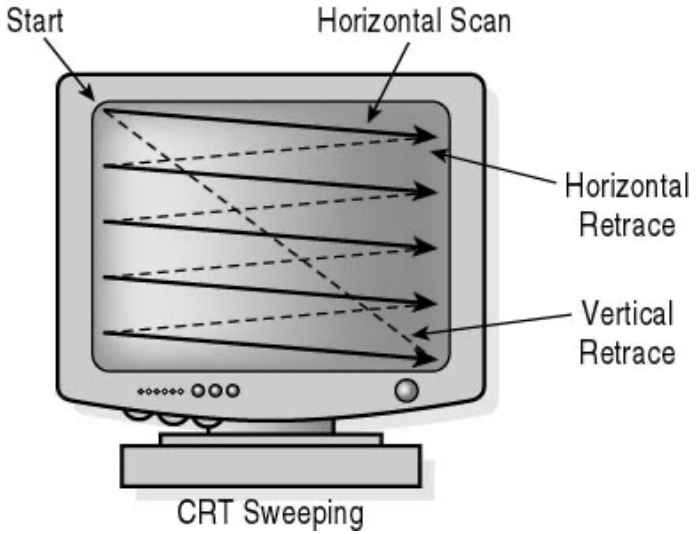
LCD, LED, OLED
changes of pixel elements per second

monitor refreshes

`win.flip()` can only occur when there is a refresh

60Hz = 60 refreshes per second

1/60 second per refresh = 16.667ms per refresh



can only have timings that are integer multiples of the refresh times

get monitor refresh rate (and other information)

```
from psychopy import logging, info
```

(assumes `mywin` has been defined)

```
mywin.recordFrameIntervals = True
logging.console.setLevel(logging.DEBUG)

runInfo = info.RunTimeInfo(win=mywin,
                refreshTest='grating', verbose=True,
                userProcsDetailed=True)
print(runInfo)

mywin.close()
core.quit()
```

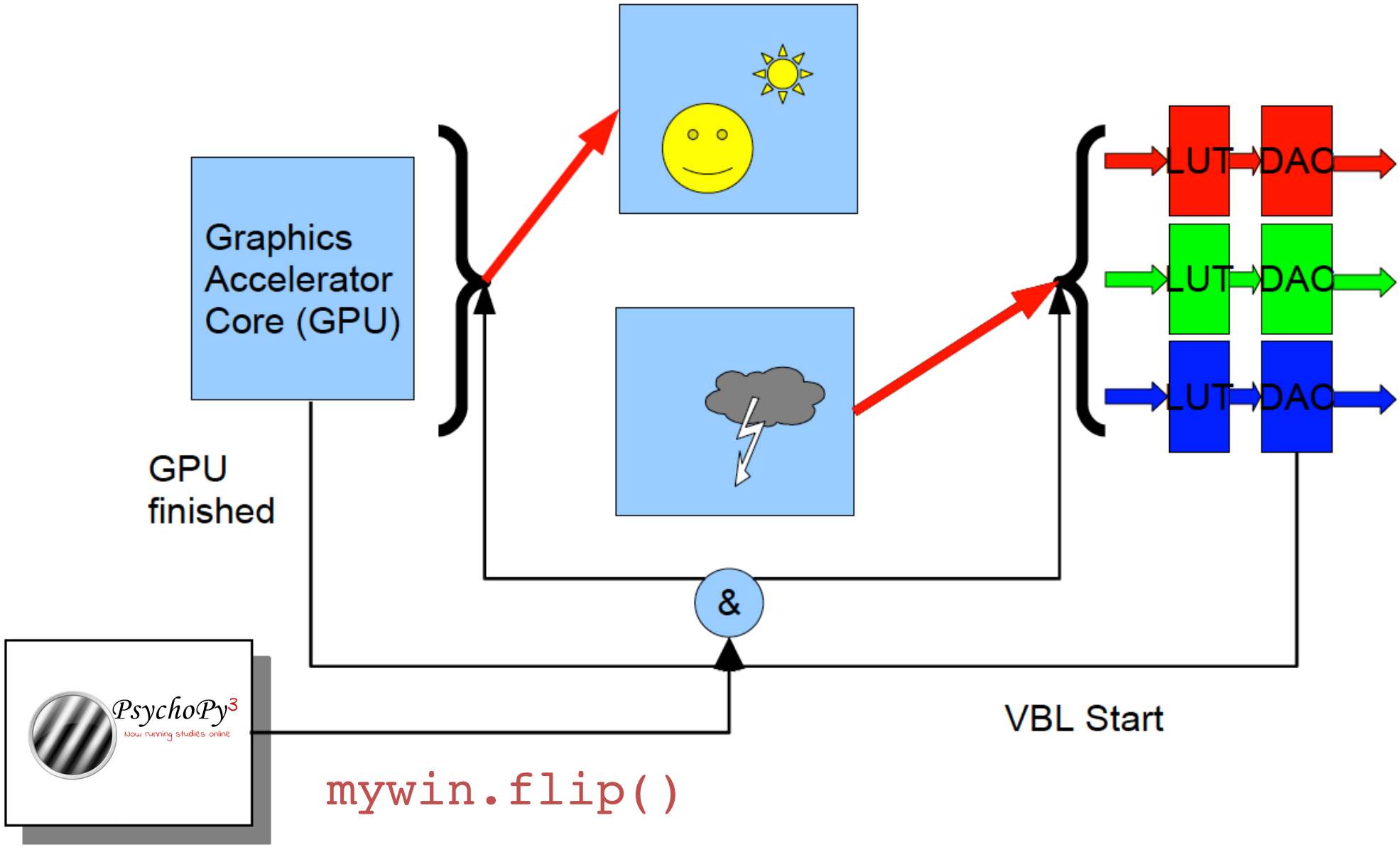
no substitute for confirming timings with external measures

get monitor info

```
get_monitor_info = True
```

test10.py

recall


Graphics Accelerator Core (GPU)
LUT
DAC
LUT
DAC
LUT
DAC
GPU finished
&
PsychoPy3
Now running studies online
VBL Start
mywin.flip()