Homework 9

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

download from Brightspace

`Psychopy.zip` (Python files)

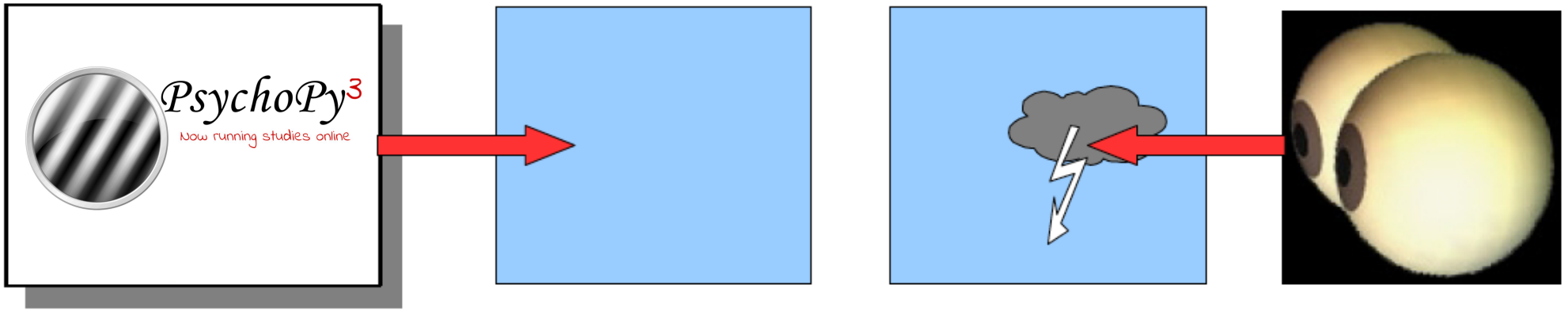
another example

```
while True:
    grating.setPhase(0.05, '+')
    grating.draw()
    fixation.draw()
    mywin.flip()

    if len(event.getKeys()) > 0:
        break
    event.clearEvents()
```

flip to make back buffer visible

https://www.psychopy.org/api/visual/gratingstim.html#psychopy.visual.GratingStim

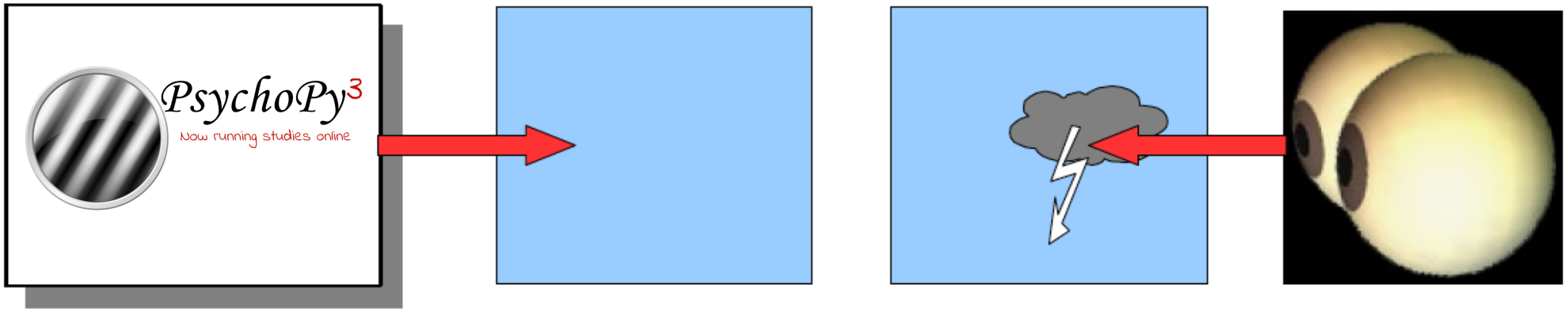
behind the scenes



Invisible
Backbuffer
(VRAM)

User-Visible
Frontbuffer
(Display)

behind the scenes

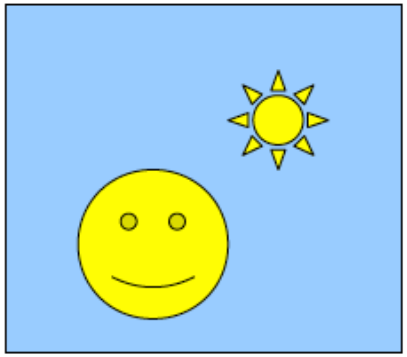


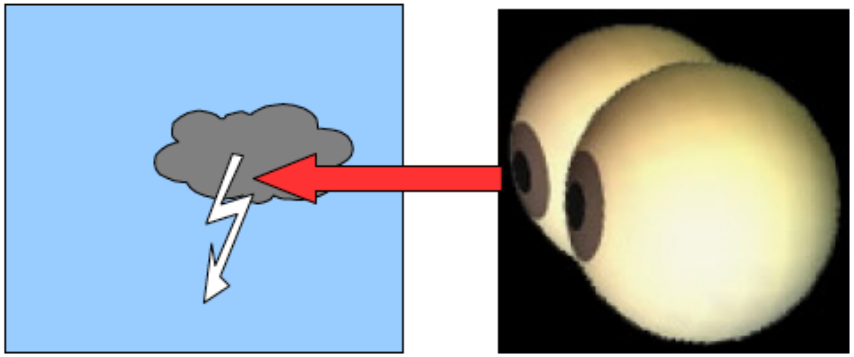
Invisible
Backbuffer
(VRAM)

User-Visible
Frontbuffer
(Display)

information currently
displayed on the monitor

behind the scenes



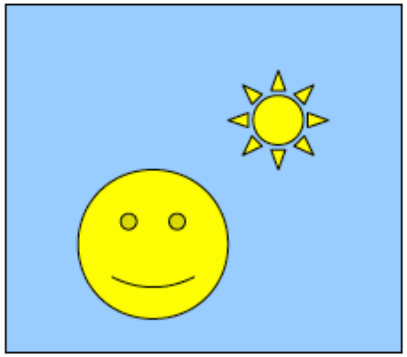


Invisible
Backbuffer
(VRAM)

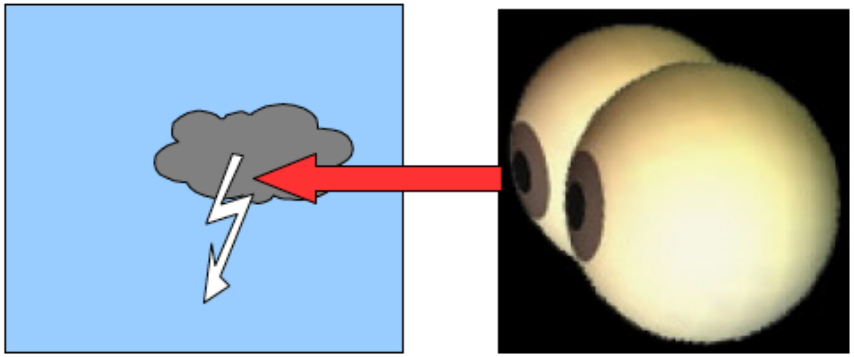
User-Visible
Frontbuffer
(Display)

you can write new information to the backbuffer

behind the scenes



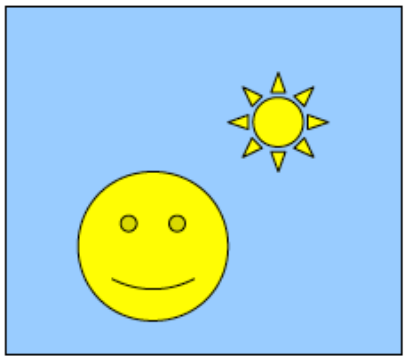
Invisible
Backbuffer
(VRAM)



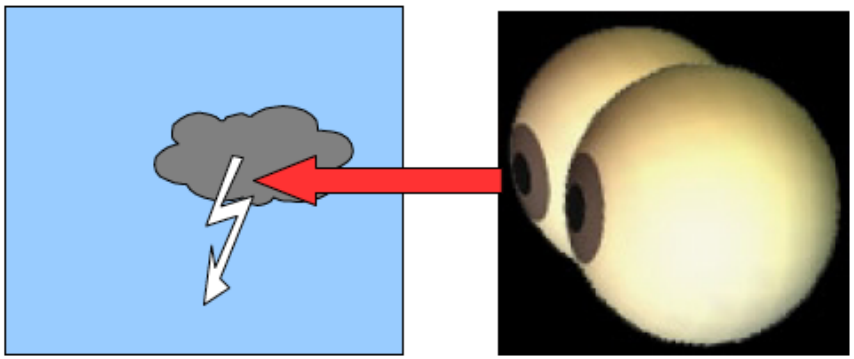
User-Visible
Frontbuffer
(Display)

writing from computer memory (RAM) to graphics memory (VRAM) takes time – OpenGL performs that copying in the background while Python and PsychoPy let you do other things

behind the scenes



Invisible
Backbuffer
(VRAM)



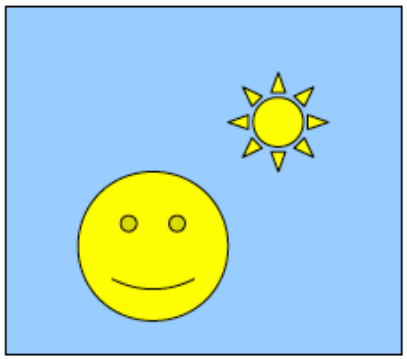
User-Visible
Frontbuffer
(Display)

whatever the name of object is

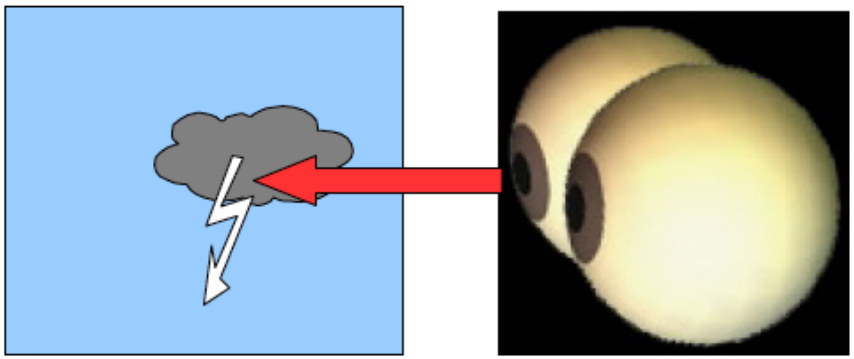
issue `mywin.flip()` command to flip the invisible back buffer to the visible front buffer

graphics hardware waits to flip to avoid artifacts

behind the scenes



Invisible
Backbuffer
(VRAM)

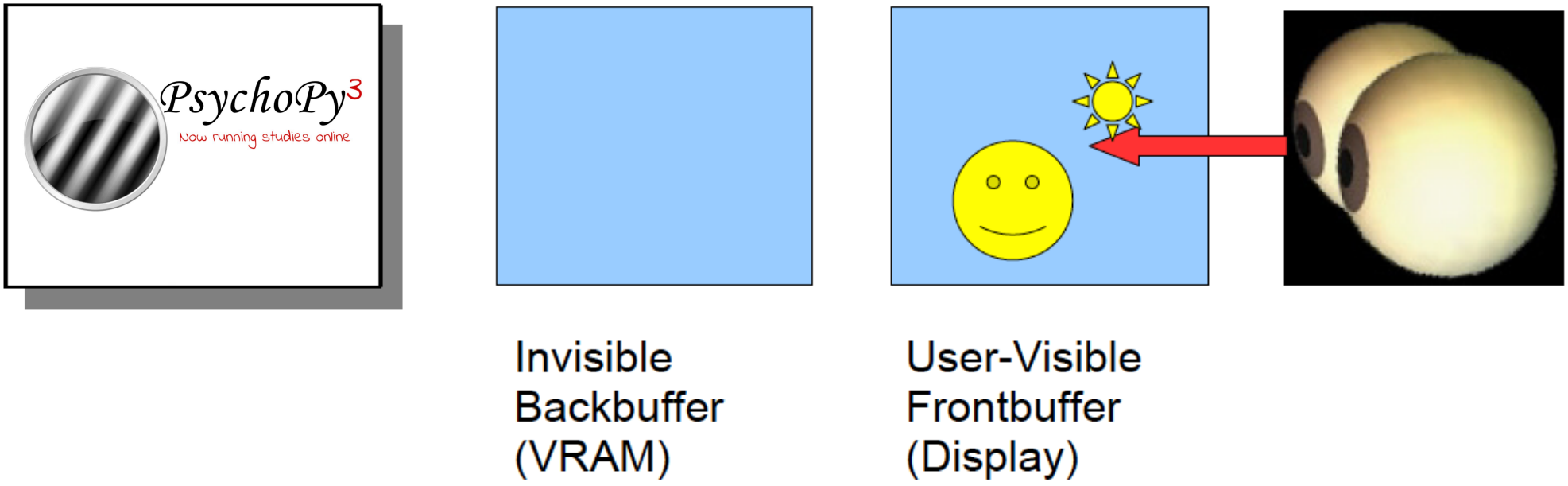


User-Visible
Frontbuffer
(Display)

1) the backbuffer must be filled completely
2) the monitor must not be in the midst of a retrace

if you don't wait, the image will appear to "tear"

behind the scenes

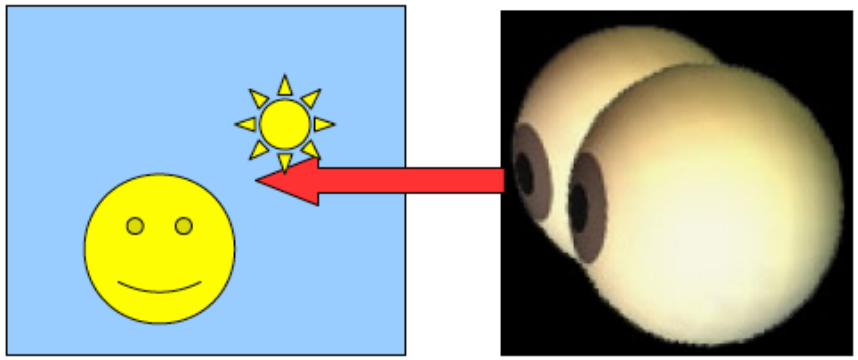


subject perceives a "tear-free" monitor update

and you get sub-ms accurate timestamp of when the monitor change occurred

behind the scenes

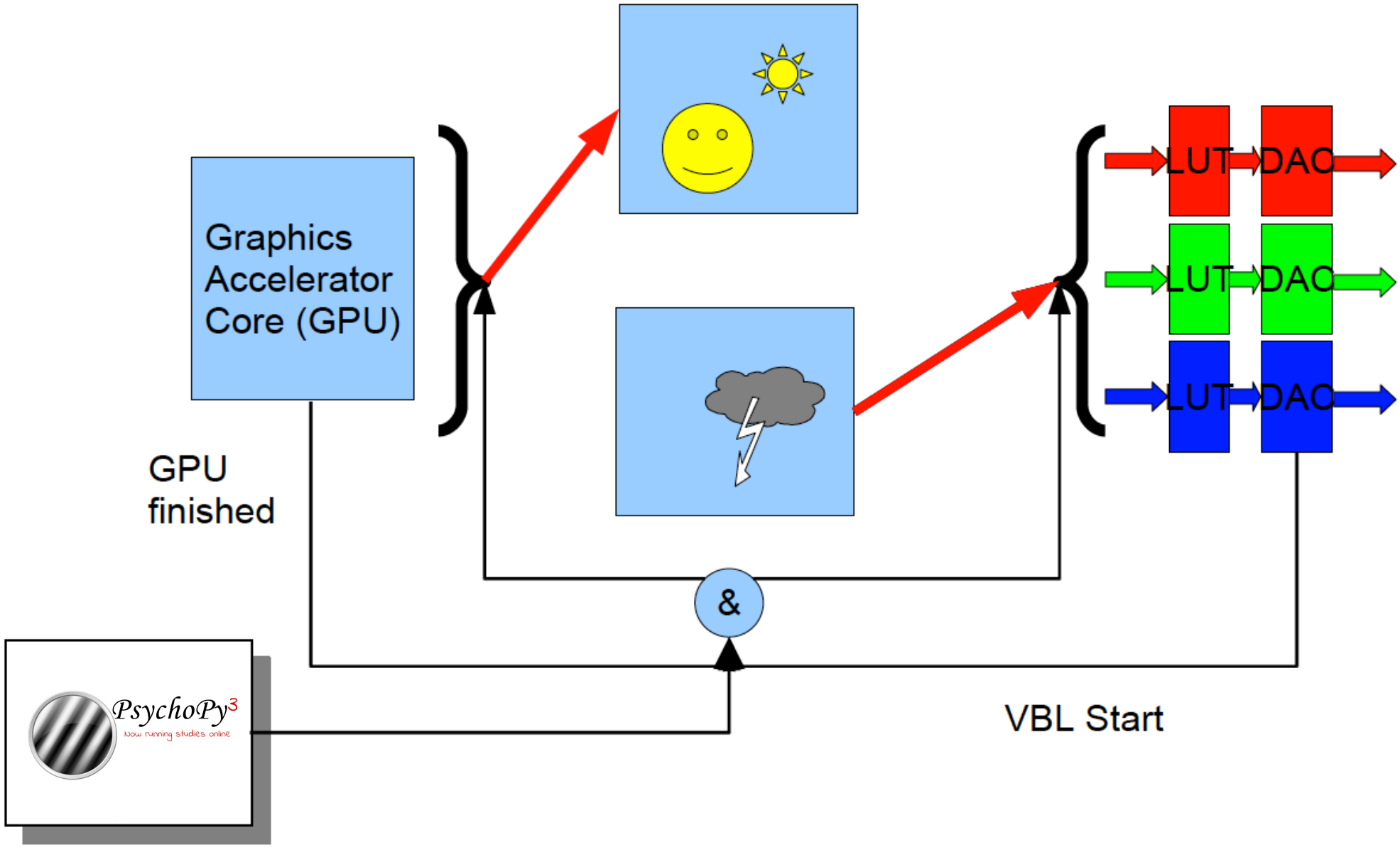
Invisible
Backbuffer
(VRAM)



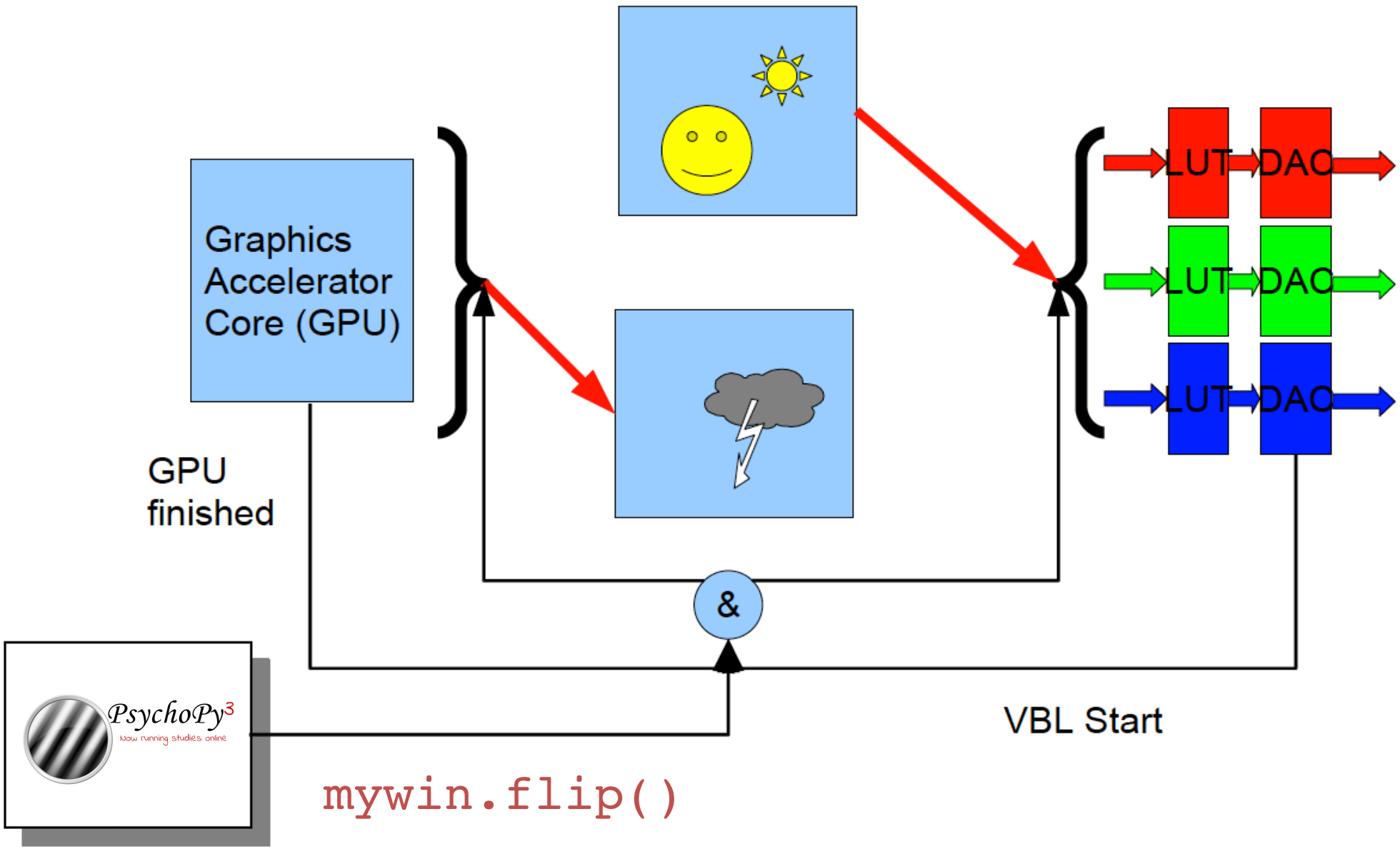
User-Visible
Frontbuffer
(Display)

backbuffer is cleared to background for next update

behind the scenes



behind the scenes



PsychoPy basics

`test3.py`

PsychoPy basics

some robust programming methods

```
if __name__ == '__main__':
    try:
        mywin = visual.Window(size=[800, 400], screen=0,
                  fullscr=False, allowGUI=True,
                  monitor="testMonitor", units='height',
                  color='gray')

        main(mywin)

        mywin.close()
        core.quit()

    except Exception as ex:
        mywin.close()
        print(ex)
        core.quit()
```

in case of an error, closes down gracefully

PsychoPy basics

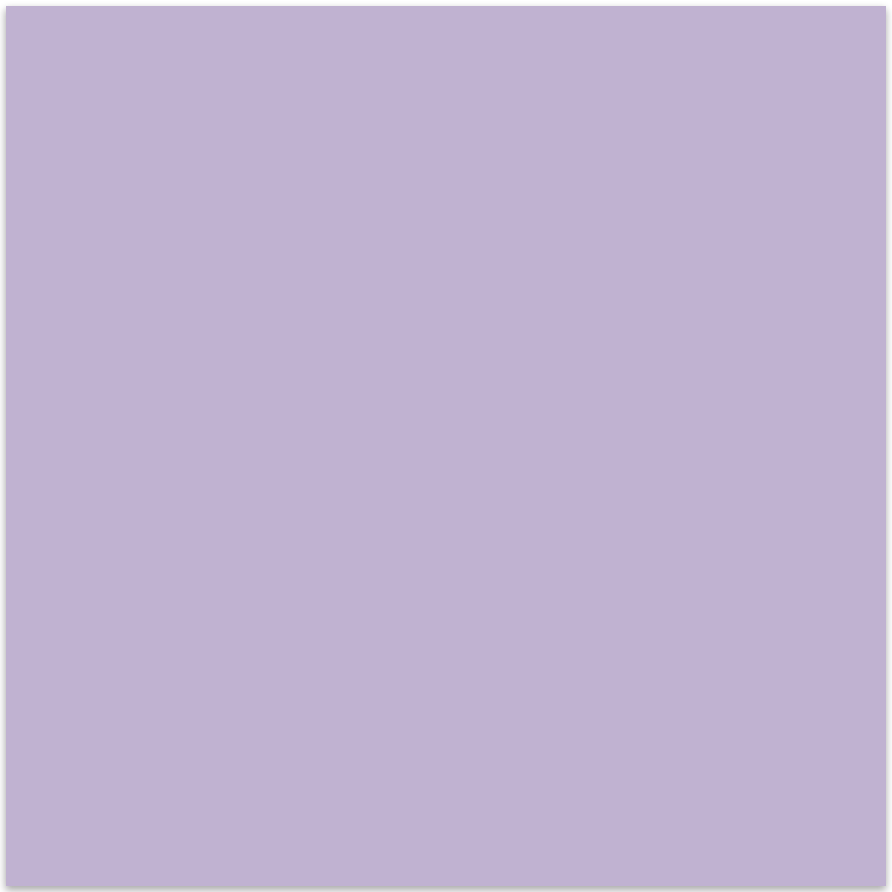
Window units

```
mywin = visual.Window(size=[800, 400], screen=0,
                      fullscr=False, allowGUI=True,
                      monitor="testMonitor", units='height',
                      color='gray')
```

`units = 'height'` normalized window units where y ranges from -.5 to .5 and x range is determined by aspect ratio (e.g. -.5 to .5 for a square screen).

PsychoPy basics

`units = 'height'` normalized window units where y ranges from -.5 to .5 and x range is determined by aspect ratio (e.g. -.5 to .5 for a square screen).



PsychoPy basics

`units = 'height'` normalized window units where y ranges from -.5 to .5 and x range is determined by aspect ratio (e.g. -.5 to .5 for a square screen).

(-1,+.5) (+1,+.5)

(-1,-.5) (+1,-.5)

PsychoPy basics

Window units

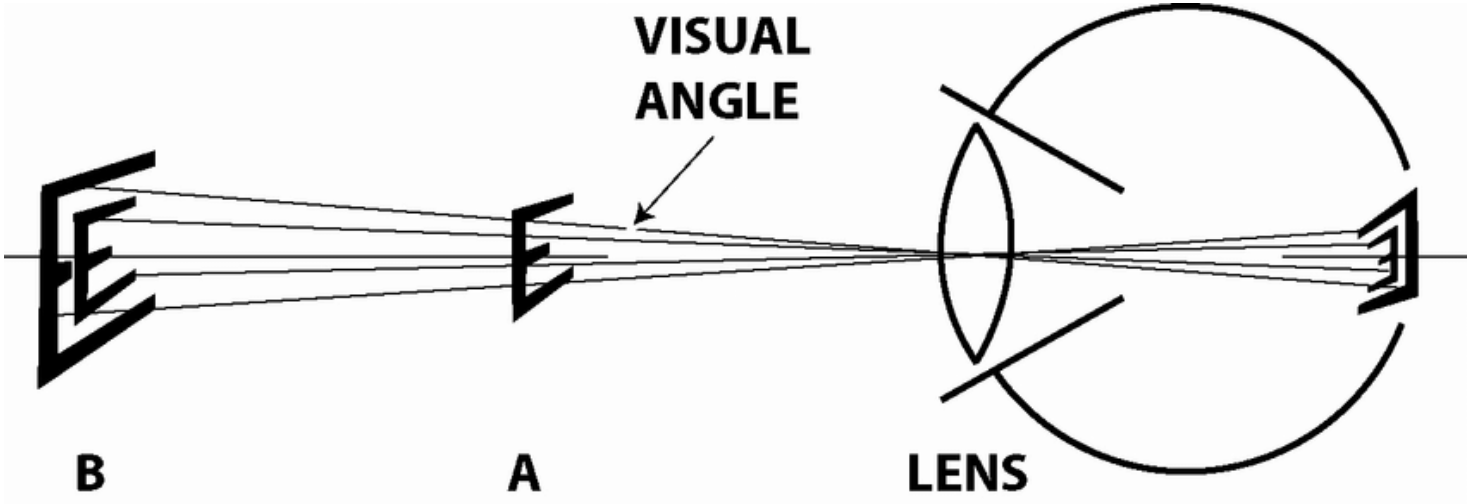
```
mywin = visual.Window(size=[800, 400], screen=0,
                      fullscr=False, allowGUI=True,
                      monitor="testMonitor", units='height',
                      color='gray')
```

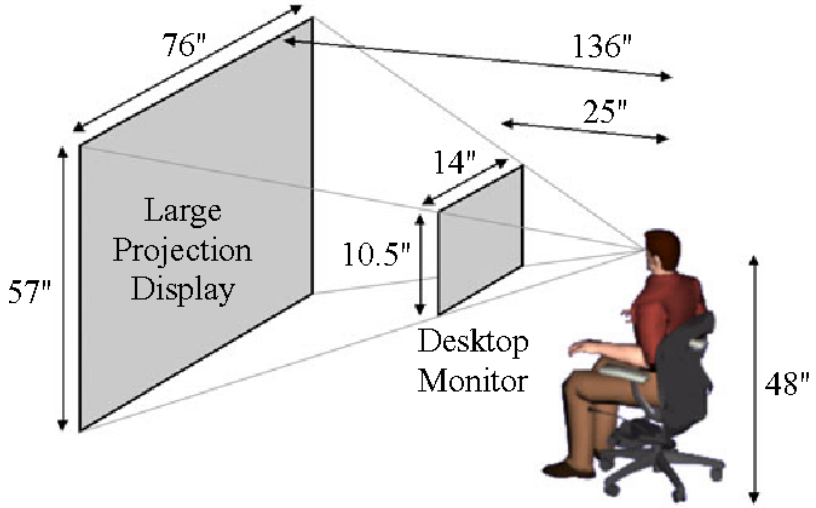
`units = 'height'`	normalized window units where y ranges from -.5 to .5 and x range is determined by aspect ratio (e.g. -.5 to .5 for a square screen).
`units = 'norm'`	fully normalized window units where both x and y values range from -1 to 1 : does not maintain aspect ratio
`units = 'cm'`	defines all positions and sizes in real world cm
`units = 'deg'`	defines all positions and sizes in real world degrees of visual angle

cm & deg require monitor information (pixels, screen size, and viewing distance), which can be set up in the monitor center

visual angle

`units = 'deg'` defines all positions and sizes in real world degrees of visual angle





PsychoPy basics

```
def main(mywin):
    msg = visual.TextStim(win=mywin, text='hello', color='red', font='times',
                          height=.1, pos=[-.2,+.2])
    msg.autoDraw = False
    msg.draw()
    mywin.flip()

    core.wait(1.0)

    mywin.flip()

    core.wait(1.0)

    msg.text = 'world'
    msg.color = 'black'
    msg.height = .2
    msg.pos = [+.2,-.3]
    msg.draw()
    mywin.flip()

    core.wait(1.0)
```

writes a message to Window

PsychoPy basics

```
def main(mywin):
    msg = visual.TextStim(win=mywin, text='hello', color='red', font='times',
                          height=.1, pos=[-.2,+.2])
    msg.autoDraw = False
    msg.draw()
    mywin.flip()

    core.wait(1.0)

    mywin.flip()

    core.wait(1.0)

    msg.text = 'world'
    msg.color = 'black'
    msg.height = .2
    msg.pos = [+.2,-.3]
    msg.draw()
    mywin.flip()

    core.wait(1.0)
```

draws a blank screen

PsychoPy basics

```
def main(mywin):
    msg = visual.TextStim(win=mywin, text='hello', color='red', font='times',
                          height=.1, pos=[-.2,+.2])
    msg.autoDraw = False
    msg.draw()
    mywin.flip()

    core.wait(1.0)

    mywin.flip()

    core.wait(1.0)

    msg.text = 'world'
    msg.color = 'black'
    msg.height = .2
    msg.pos = [+.2,-.3]
    msg.draw()
    mywin.flip()

    core.wait(1.0)
```

writes a new message to Window

PsychoPy basics

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()
```

PsychoPy basics

need to specify window

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()
```

PsychoPy basics

what text

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()
```

PsychoPy basics

color

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()
```

PsychoPy basics

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()
```

font

PsychoPy basics

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()
```

height in Window units

```
mywin = visual.Window(size=[800, 400], screen=0,
                      fullscr=False, allowGUI=True,
                      monitor="testMonitor", units='height',
                      color='gray')
```

`units = 'height'` normalized window units where y ranges from -.5 to .5 and x range is determined by aspect ratio (e.g. -.5 to .5 for a square screen).

PsychoPy basics

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()
```

position in Window units

```
mywin = visual.Window(size=[800, 400], screen=0,
                      fullscr=False, allowGUI=True,
                      monitor="testMonitor", units='height',
                      color='gray')
```

`units = 'height'` normalized window units where y ranges from -.5 to .5 and x range is determined by aspect ratio (e.g. -.5 to .5 for a square screen).

PsychoPy basics

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()
```

if `AutoDraw` is `False`,
then need to call `draw()` method

PsychoPy basics

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()

core.wait(1.0)

mywin.flip()

core.wait(1.0)
```

write message to Window

this will be a blank Window

PsychoPy basics

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = False
msg.draw()
mywin.flip()

core.wait(1.0)

msg.draw()
mywin.flip()

core.wait(1.0)
```

write message to Window

this continues same message to Window

PsychoPy basics

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = True
mywin.flip()
```

write message to Window

```
core.wait(1.0)
```

```
mywin.flip()
```

this continues same message to Window

```
core.wait(1.0)
```

```
msg.text = 'world'
msg.color = 'black'
msg.height = .2
msg.pos = [+.2,-.3]
mywin.flip()
```

change properties of message and write

PsychoPy basics

```
msg = visual.TextStim(win=mywin, text='hello', color='red',
                      font='times', height=.1, pos=[-.2,+.2])
msg.autoDraw = True
mywin.flip()
```

write message to Window

```
core.wait(1.0)
```

```
mywin.flip()
```

this continue same message to Window

```
core.wait(1.0)

msg2 = visual.TextStim(win=mywin, text='world', color='black',
                       font='times', height=.2, pos=[+.2,-.3])
msg2.autoDraw = True

mywin.flip()
```

could instead create new message

images in PsychoPy

`test4.py`

images in PsychoPy

```
def main(mywin):
    dr = './images'
    pattern = r'.*.((jpg)|(JPG))'

    r = 600; c = 800; ar = r / c
    with os.scandir(dr) as entries:
        for entry in entries:
            if re.search(pattern, dr + '/' + entry.name):
                fullpath = dr + '/' + entry.name
                mypic = visual.ImageStim(win=mywin,
                                image=fullpath, size=[.6, .6 * ar],
                                contrast=.50, ori=180)

                mypic.draw()

                mywin.flip()

                core.wait(0.5)
```

we used this before

https://www.psychopy.org/api/visual/imagestim.html

images in PsychoPy

```
def main(mywin):
    dr = './images'
    pattern = r'.*.((jpg)|(JPG))'

    r = 600; c = 800; ar = r / c
    with os.scandir(dr) as entries:
        for entry in entries:
            if re.search(pattern, dr + '/' + entry.name):
                fullpath = dr + '/' + entry.name
                mypic = visual.ImageStim(win=mywin,
                                image=fullpath, size=[.6, .6 * ar],
                                contrast=.50, ori=180)

                mypic.draw()

                mywin.flip()

                core.wait(0.5)
```

load image to be displayed

https://www.psychopy.org/api/visual/imagestim.html

images in PsychoPy

```
def main(mywin):
    dr = './images'
    pattern = r'.*.((jpg)|(JPG))'

    r = 600; c = 800; ar = r / c
    with os.scandir(dr) as entries:
        for entry in entries:
            if re.search(pattern, dr + '/' + entry.name):
                fullpath = dr + '/' + entry.name
                mypic = visual.ImageStim(win=mywin,
                                image=fullpath, size=[.6, .6 * ar],
                                contrast=.50, ori=180)

                mypic.draw()

                mywin.flip()

                core.wait(0.5)
```

draw it

https://www.psychopy.org/api/visual/imagestim.html

images in PsychoPy

```
def main(mywin):
    dr = './images'
    pattern = r'.*.((jpg)|(JPG))'

    r = 600; c = 800; ar = r / c
    with os.scandir(dr) as entries:
        for entry in entries:
            if re.search(pattern, dr + '/' + entry.name):
                fullpath = dr + '/' + entry.name
                mypic = visual.ImageStim(win=mywin,
                                image=fullpath, size=[.6, .6 * ar],
                                contrast=.50, ori=180)

                mypic.draw()

                mywin.flip()

                core.wait(0.5)
```

flip to display

https://www.psychopy.org/api/visual/imagestim.html

draw shapes in PsychoPy

`test5.py`

draw shapes in PsychoPy

```
event.globalKeys.add(key='q', func=core.quit)
```

nothing to do with drawing shapes

this gives a way to quit a program during development

may not want to include in a deployed experiment

draw shapes in PsychoPy

```
el1 = visual.Line(win=mywin, start=[+.3,-.2],
                  end=[-.4,-.3], lineWidth=20,
                  lineColor='blue')
el1.autoDraw = True
el1.draw()

mywin.flip()
core.wait(0.5)
```

draw a `Line`

`start` **and** `end` **point**

https://www.psychopy.org/api/visual/line.html#psychopy.visual.Line

draw shapes in PsychoPy

```
el1 = visual.Line(win=mywin, start=[+.3,-.2],
                  end=[-.4,-.3], lineWidth=20,
                  lineColor='blue')
el1.autoDraw = True
el1.draw()

mywin.flip()
core.wait(0.5)
```

https://www.psychopy.org/api/visual/line.html#psychopy.visual.Line

draw shapes in PsychoPy

```
el2 = visual.Circle(win=mywin, radius=.1,
                    pos=[-.2,+.1], edges=128,
                    fillColor='green')
el2.autoDraw = True
el2.draw()

mywin.flip()
core.wait(0.5)
```

draw a `Circle`
`radius` **and** `pos` **center**

https://www.psychopy.org/api/visual/circle.html#psychopy.visual.Circle

draw shapes in PsychoPy

```
el2 = visual.Circle(win=mywin, radius=.1,
                    pos=[-.2,+.1], edges=128,
                    fillColor='green')
el2.autoDraw = True
el2.draw()

mywin.flip()
core.wait(0.5)
```

draw a `Circle`

`edges` **specifies number of edges of circle (approximated by polygon)**

https://www.psychopy.org/api/visual/circle.html#psychopy.visual.Circle

draw shapes in PsychoPy

```
el3 = visual.Rect(win=mywin, size=[.2,.1],
                  pos=[+.3,+.2], lineWidth=10,
                  lineColor='#F016A3')
el3.autoDraw = True
el3.draw()

mywin.flip()
core.wait(0.5)
```

draw a `Rect`

`size` specifies width and height

`pos` specifies center

https://www.psychopy.org/api/visual/rect.html#psychopy.visual.Rect

draw shapes in PsychoPy

```
el3 = visual.Rect(win=mywin, size=[.2,.1],
                  pos=[+.3,+.2], lineWidth=10,
                  lineColor='#F016A3')
el3.autoDraw = True
el3.draw()

mywin.flip()
core.wait(0.5)
```

can specify color in hex (RGB)

https://www.psychopy.org/api/visual/rect.html#psychopy.visual.Rect

draw shapes in PsychoPy

```
vert = np.array([[-.1,+.2], [+.1,+.1], [+.2,-.1],
                 [-.1,-.3], [-.1,+.2]])
el4 = visual.ShapeStim(win=mywin, vertices=vert,
                       fillColor=[+.3,-.7,+.8],
                       lineWidth=20,
                       lineColor=[-.6,+.4,+.6])

el4.autoDraw = True
el4.draw()

mywin.flip()
core.wait(0.5)
```

draw a `ShapeStim` specified by `vertices` (x,y) points

https://www.psychopy.org/api/visual/shapestim.html#psychopy.visual.ShapeStim

draw shapes in PsychoPy

```
vert = np.array([[-.1,+.2], [+.1,+.1], [+.2,-.1],
                 [-.1,-.3], [-.1,+.2]])
el4 = visual.ShapeStim(win=mywin, vertices=vert,
                       fillColor=[+.3,-.7,+.8],
                       lineWidth=20,
                       lineColor=[-.6,+.4,+.6])
```

another way to specify colors

[0, 0, 0] middle gray

[-1, -1, -1] black

[+1, +1, +1] white

```
el4.autoDraw = True
el4.draw()

mywin.flip()
core.wait(0.5)
```

https://www.psychopy.org/api/visual/shapestim.html#psychopy.visual.ShapeStim

Homework 9 (due Dec 5)

Homework 9
Due December 5 in class
40 points

PSY4219/6219
Fall 2022

Note that this is only Q1 of the assignment. Q2 will be posted later this week (and will be worth 30 points).

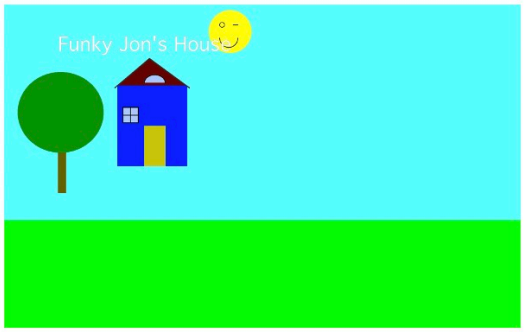
Q1 (10 points). This first part requires you to demonstrate your ability to use some basic PsychoPy functions and methods.

To make clear the grading rubric, I have put in *italics* the points (e.g., *2 points*) for each components required to achieve full credit.

Model your code after the examples we went over in class. For example, use the `try-except` structures so that your program "dies gracefully". If you're not using the mouse, turn it off. Just remember to turn it back on again when you're done.

Use proper Python style (comments, functions, not hard-coding, etc.). (*2 points*)

Use PsychoPy to draw a picture of your choosing on the computer monitor. This is a screen shot of an example (don't draw this, this is just an example of what I mean):



I don't really care what you draw – let your inner 6 year old run wild. I just want you to show me that you can (a) set the background to a particular color that isn't white or black (*1 point*), (b) purposefully place some drawing elements (rectangles, lines, circles, arcs) in places on the screen that make a drawing (not just a random collection of randomly placed shapes) (*2 points*), (c) place some text on the drawing (*1 points*), and (d) have elements of the drawing appear (or disappear) at different times *(2 points*). It doesn't need to be complicated or artistic, but it should use at least 10 elements; they should not be placed

using the mouse in PsychoPy

`test6.py`

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

using the mouse in PsychoPy

```
def main(mywin):

    mouse = event.Mouse(win=mywin, visible=True)

    mouse.setPos([0,0])
    mouse.clickReset(buttons=(0, 1, 2))

    cpos = [0, 0]
    center = visual.Circle(win=mywin, radius=.025, pos=cpos, edges=128, fillColor='green')
    center.draw()
    mywin.flip()

    prevpos = mouse.getPos()
    mouse.setVisible(True)

    while not event.getKeys():
        currpos = mouse.getPos()

        if not(np.array_equal(currpos, prevpos)):
            print(currpos)
            prevpos = currpos

        but = mouse.getPressed()
        if but[0]:
            print('mouse clicked')
            mouse.setVisible(False)
        else:
            mouse.setVisible(True)

        if mouse.isPressedIn(shape=center):
            print('mouse clicked in center')
```

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

using the mouse in PsychoPy

```
mouse = event.Mouse(win=mywin, visible=True)
```

create a mouse object in `mywin`

```
mouse.setPos([0,0])
mouse.clickReset(buttons=(0, 1, 2))

cpos = [0, 0]
center = visual.Circle(win=mywin, radius=.025,
                       pos=cpos, edges=128,
                       fillColor='green')
center.draw()
mywin.flip()
```

using the mouse in PsychoPy

```
mouse = event.Mouse(win=mywin, visible=True)

mouse.setPos([0,0])
mouse.clickReset(buttons=(0, 1, 2))
```

set position and clear mouse buttons

```
cpos = [0, 0]
center = visual.Circle(win=mywin, radius=.025,
                       pos=cpos, edges=128,
                       fillColor='green')
center.draw()
mywin.flip()
```

using the mouse in PsychoPy

```
mouse = event.Mouse(win=mywin, visible=True)

mouse.setPos([0,0])
mouse.clickReset(buttons=(0, 1, 2))

cpos = [0, 0]
center = visual.Circle(win=mywin, radius=.025,
                       pos=cpos, edges=128,
                       fillColor='green')
center.draw()
mywin.flip()
```

drawing a circular center point

using the mouse in PsychoPy

```
prevpos = mouse.getPos(); mouse.setVisible(True)
```

get current mouse position

```
while not event.getKeys():
    currpos = mouse.getPos()

    if not(np.array_equal(currpos, prevpos)):
        print(currpos)
        prevpos = currpos

    but = mouse.getPressed()
    if but[0]:
        print('mouse clicked')
        mouse.setVisible(False)
    else:
        mouse.setVisible(True)

    if mouse.isPressedIn(shape=center):
        print('mouse clicked in center')
```

using the mouse in PsychoPy

```
prevpos = mouse.getPos(); mouse.setVisible(True)

while not event.getKeys():
    currpos = mouse.getPos()

    if not(np.array_equal(currpos, prevpos)):
        print(currpos)
        prevpos = currpos

    but = mouse.getPressed()
    if but[0]:
        print('mouse clicked')
        mouse.setVisible(False)
    else:
        mouse.setVisible(True)

    if mouse.isPressedIn(shape=center):
        print('mouse clicked in center')
```

loop until keyboard key pressed

more generally, this is an example of **event polling**, infinitely looping and responding to particular events

using the mouse in PsychoPy

```
prevpos = mouse.getPos(); mouse.setVisible(True)

while not event.getKeys():
    currpos = mouse.getPos()

    if not(np.array_equal(currpos, prevpos)):
        print(currpos)
        prevpos = currpos

    but = mouse.getPressed()
    if but[0]:
        print('mouse clicked')
        mouse.setVisible(False)
    else:
        mouse.setVisible(True)

    if mouse.isPressedIn(shape=center):
        print('mouse clicked in center')
```

get current mouse position

using the mouse in PsychoPy

```
prevpos = mouse.getPos(); mouse.setVisible(True)

while not event.getKeys():
    currpos = mouse.getPos()

    if not(np.array_equal(currpos, prevpos)):
        print(currpos)
        prevpos = currpos

    but = mouse.getPressed()
    if but[0]:
        print('mouse clicked')
        mouse.setVisible(False)
    else:
        mouse.setVisible(True)

    if mouse.isPressedIn(shape=center):
        print('mouse clicked in center')
```

check if mouse position has changed

using the mouse in PsychoPy

```
prevpos = mouse.getPos(); mouse.setVisible(True)

while not event.getKeys():
    currpos = mouse.getPos()

    if not(np.array_equal(currpos, prevpos)):
        print(currpos)
        prevpos = currpos

    but = mouse.getPressed()
    if but[0]:
        print('mouse clicked')
        mouse.setVisible(False)
    else:
        mouse.setVisible(True)

    if mouse.isPressedIn(shape=center):
        print('mouse clicked in center')
```

get mouse button presses

using the mouse in PsychoPy

```
prevpos = mouse.getPos(); mouse.setVisible(True)

while not event.getKeys():
    currpos = mouse.getPos()

    if not(np.array_equal(currpos, prevpos)):
        print(currpos)
        prevpos = currpos

    but = mouse.getPressed()
    if but[0]:
        print('mouse clicked')
        mouse.setVisible(False)
    else:
        mouse.setVisible(True)

    if mouse.isPressedIn(shape=center):
        print('mouse clicked in center')
```

here, if mouse clicked then make invisible

otherwise, make visible

using the mouse in PsychoPy

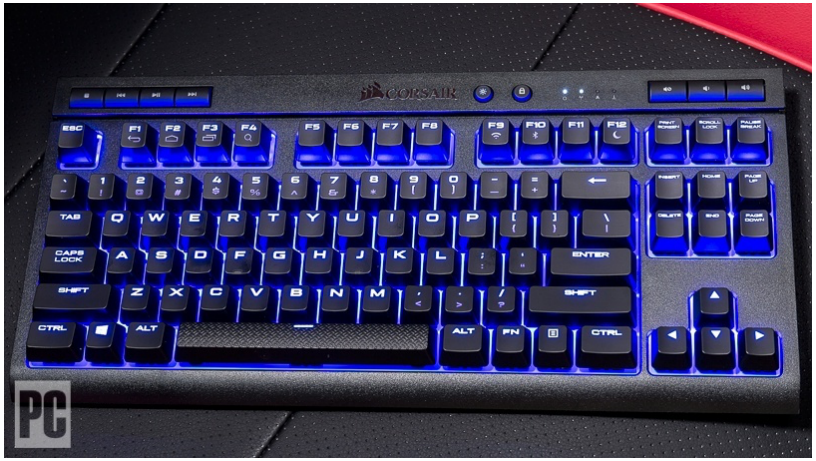
```
prevpos = mouse.getPos(); mouse.setVisible(True)

while not event.getKeys():
    currpos = mouse.getPos()

    if not(np.array_equal(currpos, prevpos)):
        print(currpos)
        prevpos = currpos

    but = mouse.getPressed()
    if but[0]:
        print('mouse clicked')
        mouse.setVisible(False)
    else:
        mouse.setVisible(True)
    if mouse.isPressedIn(shape=center):
        print('mouse clicked in center')
```

check if mouse click is in a particular position

basic interfaces with PsychoPy

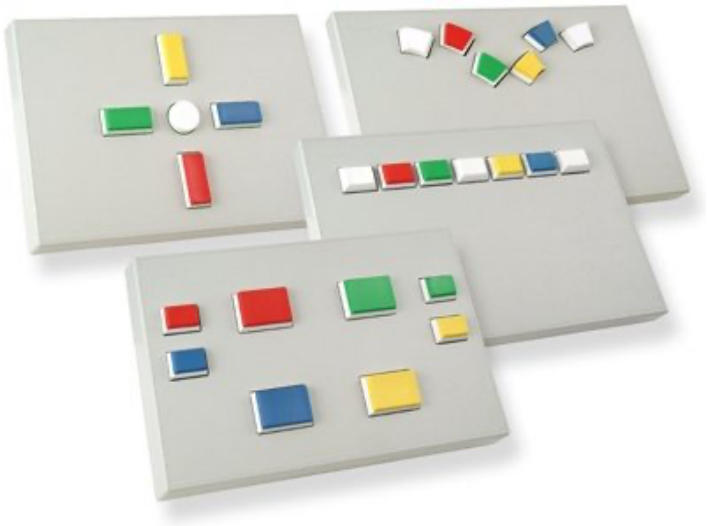
mouse

keyboard



other interfaces with PsychoPy

response buttons



fMRI

game pads

eye tracker

EEG

voice trigger

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

gui in PsychoPy

`test7.py`

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

gui in PsychoPy

```
from psychopy import gui

subgui = gui.Dlg()
subgui.addField('Subject Number:')
subgui.addField('Session Number:')
subgui.addField('Condition:', choices=['Exp', 'Ctrl'])
subgui.addField('Check Box:', initial=False)

while True:
    subgui.show()
    if (subgui.data[0].isdigit() and
        subgui.data[1].isdigit()):
        subj = int(subgui.data[0])
        sess = int(subgui.data[1])
        cond = subgui.data[2]
        chck = subgui.data[3]
        break
```

gui in PsychoPy

```
from psychopy import gui
```

import gui module

```
subgui = gui.Dlg()
subgui.addField('Subject Number:')
subgui.addField('Session Number:')
subgui.addField('Condition:', choices=['Exp', 'Ctrl'])
subgui.addField('Check Box:', initial=False)

while True:
    subgui.show()
    if (subgui.data[0].isdigit() and
        subgui.data[1].isdigit()):
        subj = int(subgui.data[0])
        sess = int(subgui.data[1])
        cond = subgui.data[2]
        chck = subgui.data[3]
        break
```

gui in PsychoPy

```
from psychopy import gui

subgui = gui.Dlg()
subgui.addField('Subject Number:')
subgui.addField('Session Number:')
subgui.addField('Condition:', choices=['Exp', 'Ctrl'])
subgui.addField('Check Box:', initial=False)

while True:
    subgui.show()
    if (subgui.data[0].isdigit() and
        subgui.data[1].isdigit()):
        subj = int(subgui.data[0])
        sess = int(subgui.data[1])
        cond = subgui.data[2]
        chck = subgui.data[3]
        break
```

create empty `gui` object

gui in PsychoPy

```
from psychopy import gui

subgui = gui.Dlg()
subgui.addField('Subject Number:')
subgui.addField('Session Number:')
subgui.addField('Condition:', choices=['Exp', 'Ctrl'])
subgui.addField('Check Box:', initial=False)

while True:
    subgui.show()
    if (subgui.data[0].isdigit() and
        subgui.data[1].isdigit()):
        subj = int(subgui.data[0])
        sess = int(subgui.data[1])
        cond = subgui.data[2]
        chck = subgui.data[3]
        break
```

add different fields

gui in PsychoPy

```
from psychopy import gui

subgui = gui.Dlg()
subgui.addField('Subject Number:')
subgui.addField('Session Number:')
subgui.addField('Condition:', choices=['Exp', 'Ctrl'])
subgui.addField('Check Box:', initial=False)

while True:
    subgui.show()
    if (subgui.data[0].isdigit() and
        subgui.data[1].isdigit()):
        subj = int(subgui.data[0])
        sess = int(subgui.data[1])
        cond = subgui.data[2]
        chck = subgui.data[3]
        break
```

keep showing the gui until ...

gui in PsychoPy

```
from psychopy import gui

subgui = gui.Dlg()
subgui.addField('Subject Number:')
subgui.addField('Session Number:')
subgui.addField('Condition:', choices=['Exp', 'Ctrl'])
subgui.addField('Check Box:', initial=False)

while True:
    subgui.show()
    if (subgui.data[0].isdigit() and
        subgui.data[1].isdigit()):
        subj = int(subgui.data[0])
        sess = int(subgui.data[1])
        cond = subgui.data[2]
        chck = subgui.data[3]
        break
```

... satisfied with info entered