Homework 9

just Q1 (10 points)
due Mon Dec 5 in class

Homework 10

due Tue Dec 13 at 11:59pm

(there will be no Homework 11)

download from Brightspace

`Psychopy.zip` (Python files)
(some files have been modified)

Homework 10

create an experiment using PsychoPy and other methods and techniques we have discussed in this class (including things we will cover this week)

make sure you carefully follow ALL the specifications

due during exam week (Tue Dec 13 at 11:59pm)

help sessions:
Mon Dec 5 @ 3:45pm (this room, after class)
Mon Dec 12 @ 3:45pm (Wilson 113, not this room)

Homework 10

the vision and memory lab

department of psychology, university of california, san diego

People
Research
Publications
Resources
Recent Talks
Join the Lab

Stimuli

Below are links to download the stimuli from the majority of our papers. These include thousands of categorized objects and scenes, objects matched into pairs that differ in subtle ways (e.g., color, state or pose changes, etc), as well as stimuli designed to be rotated in color space to allow psychophysical measures of memory precision.

Feel free to use these in academic settings. Of course we appreciate if you cite their source if you do use them.

Object stimuli from: Brady, T. F., Konkle, T., Alvarez, G. A. and Oliva, A. (2008). Visual long-term memory has a massive storage capacity for object details. *Proceedings of the National Academy of Sciences, USA*, 105 (38), 14325-14329.

- 2400 Unique Objects

- 100 State Pairs

- 100 Exemplar Pairs

Object stimuli that are color-manipulable as in: Brady, T. F., Konkle, T.F., Gill, J., Oliva, A. and Alvarez, G.A. (2013). Visual long-term memory has the same limit on fidelity as visual working memory. *Psychological Science*, 24(6), 981-990.

https://bradylab.ucsd.edu/stimuli.html

Homework 10

recognition memory experiment (**study**)

passive study (no response)

. . .

presentation time

number of study images

random order

Homework 10

recognition memory experiment (**test**)

respond "old" vs. "new"

. . .

record response
measure RT

1/2 old images
1/2 new images

random order

Timing

relative vs. absolute timing

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

`test8.py`

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

`test9.py`

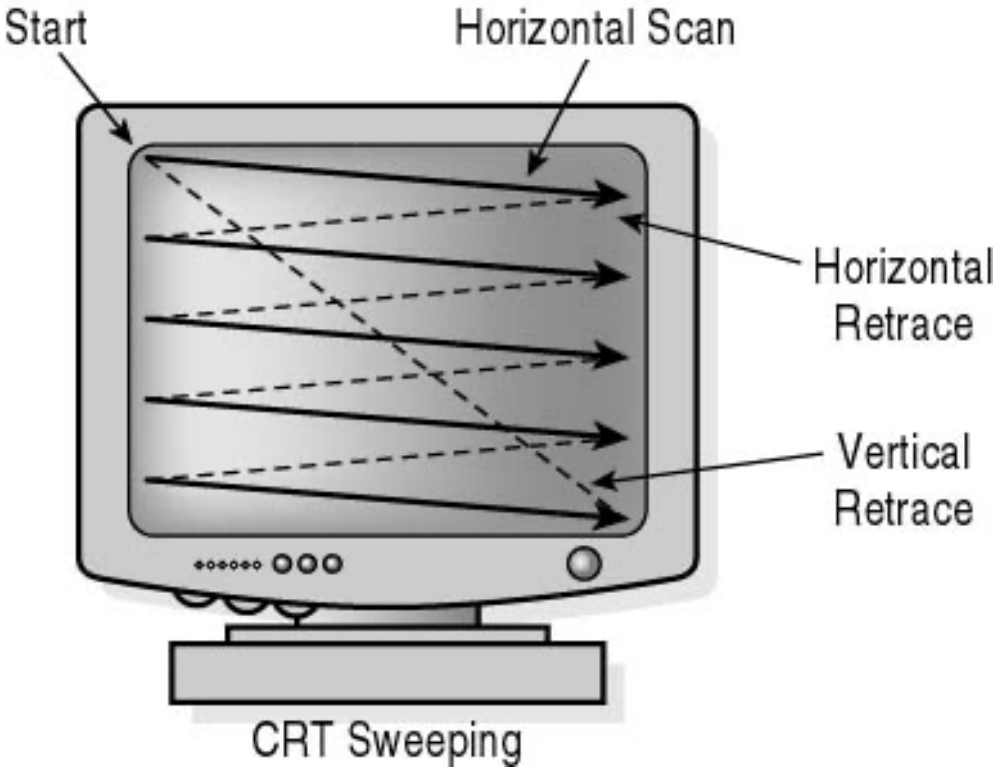
timing for `test8.py`

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

monitor refreshes



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

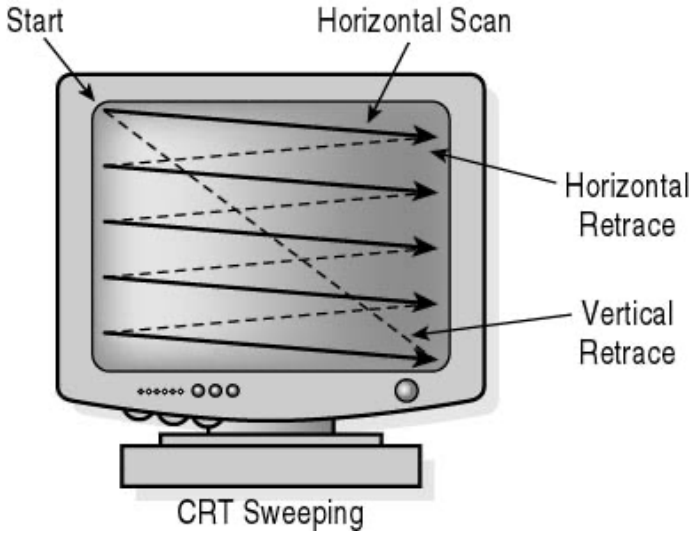
LCD, LED, OLED
changes of pixel elements per second

monitor refreshes

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

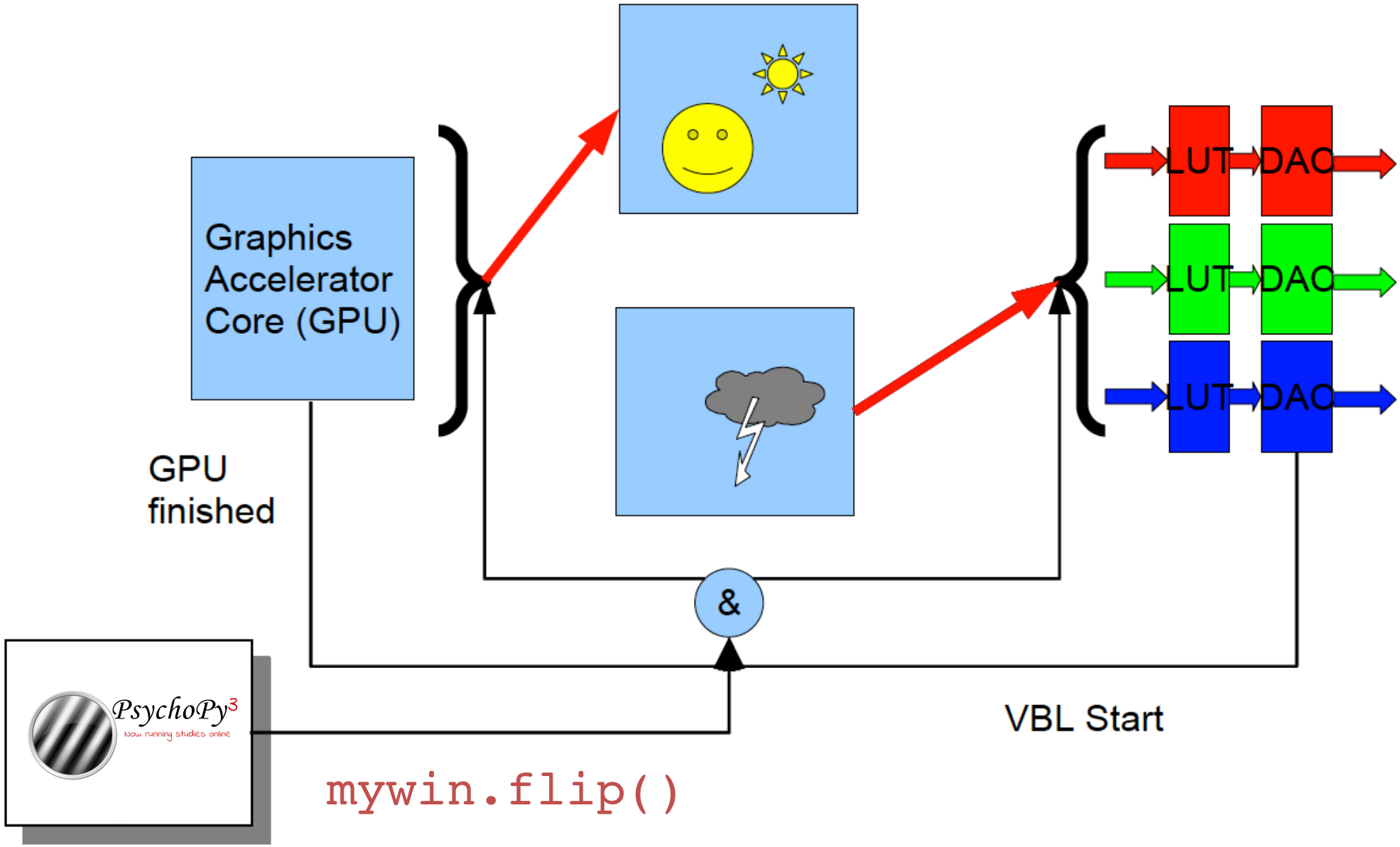
60Hz = 60 refreshes per second

1/60 second per refresh = 16.667ms per refresh

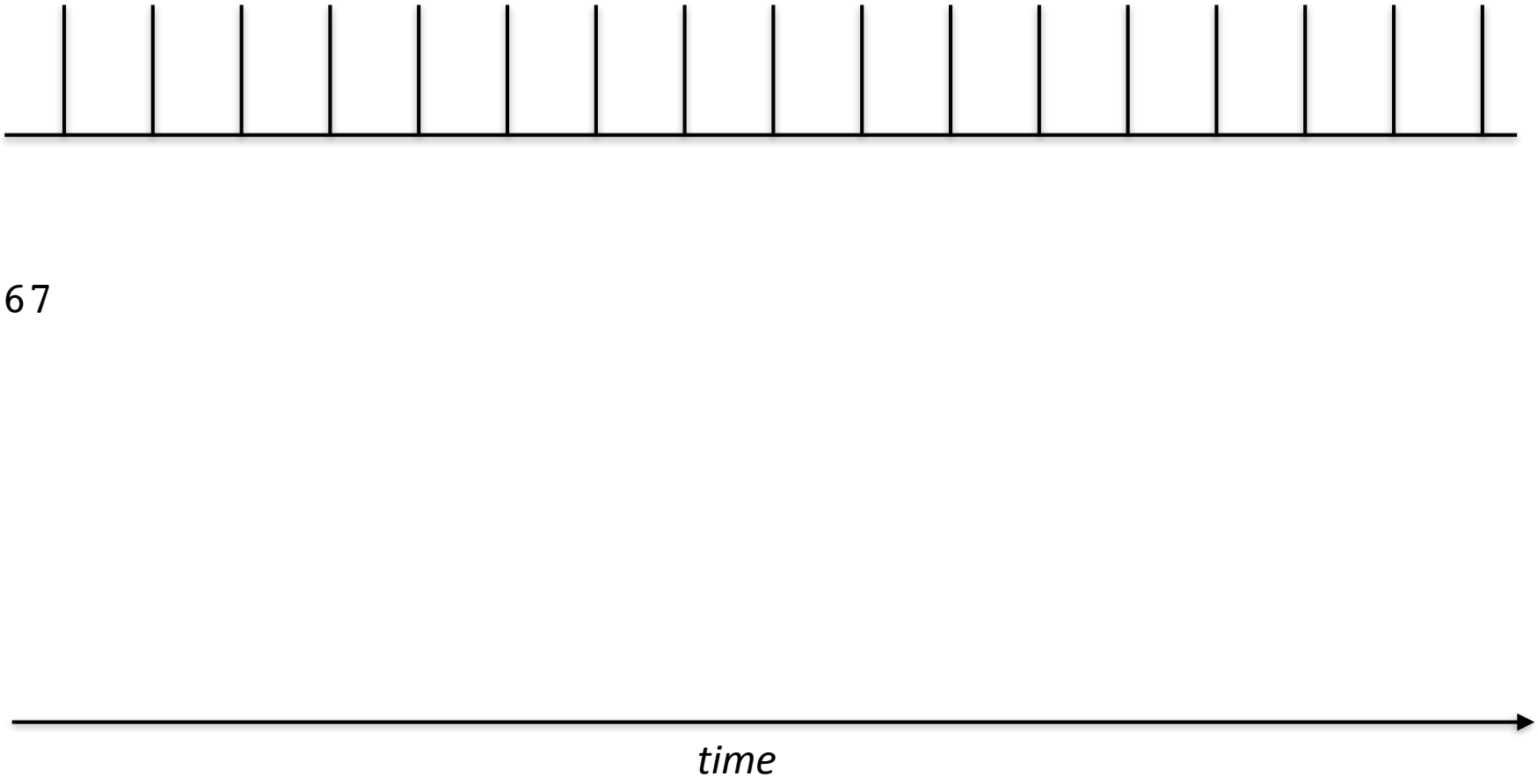


can only have timings that are integer multiples of the refresh times

recall

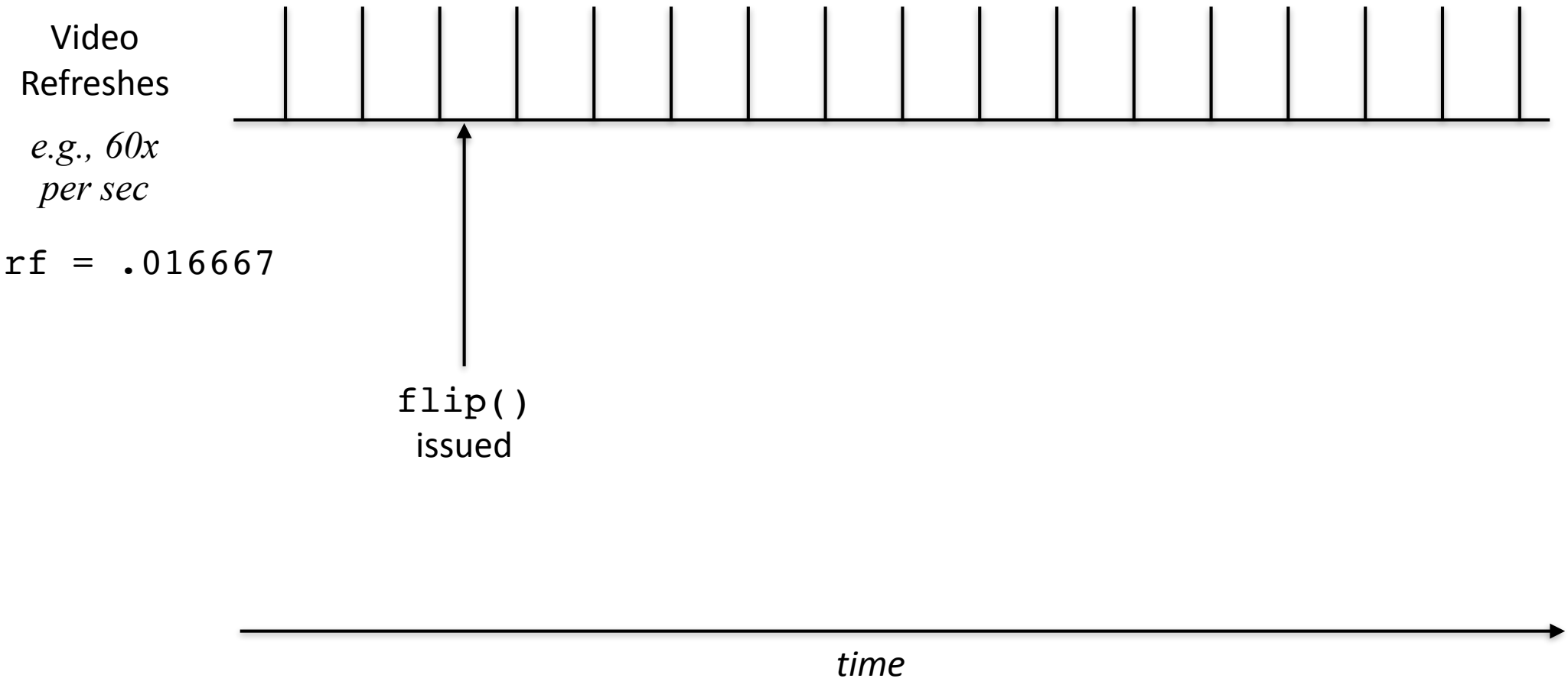

Graphics Accelerator Core (GPU)
LUT
DAC
LUT
DAC
LUT
DAC
GPU finished
&
PsychoPy3
Now running studies online
VBL Start
mywin.flip()

timing `win.flip()` with monitor refresh

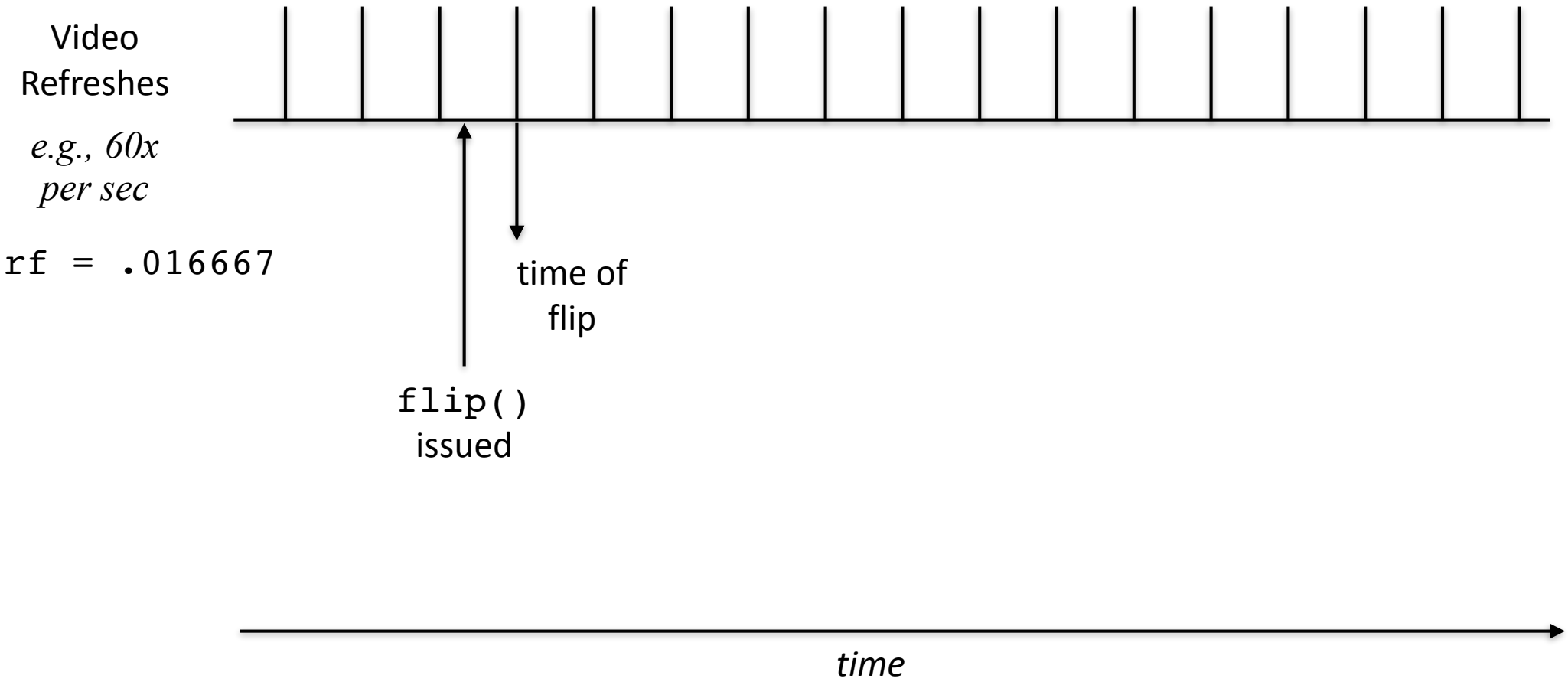



```
rf = .016667
```

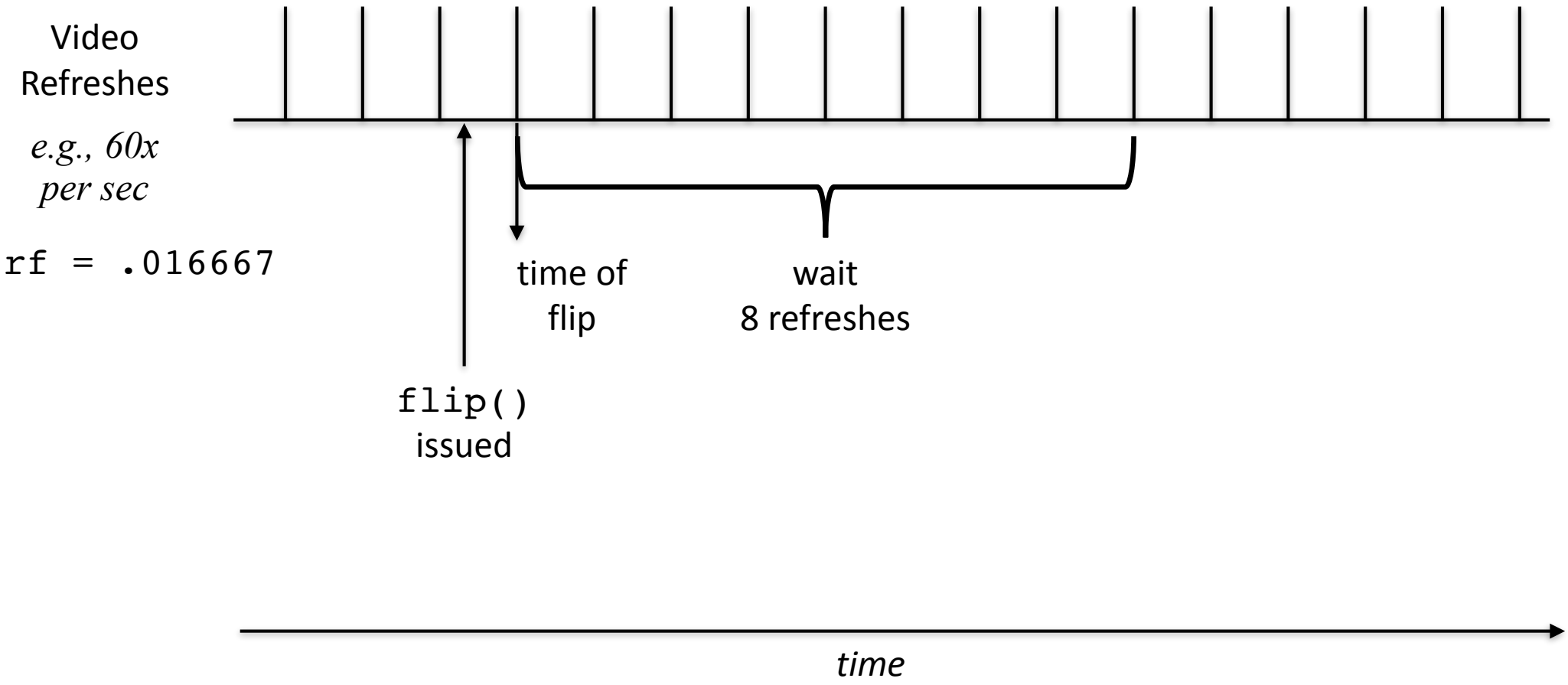
timing `win.flip()` with monitor refresh



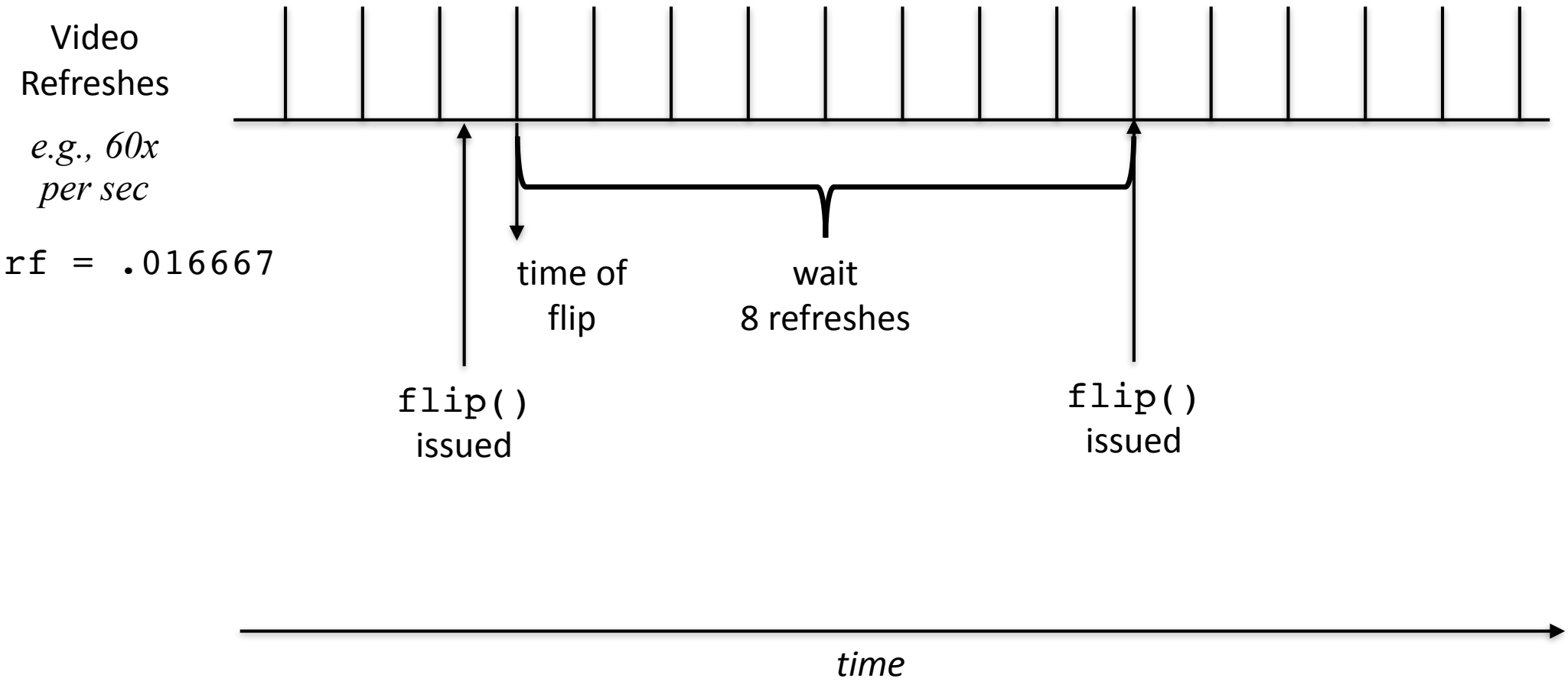
timing `win.flip()` with monitor refresh



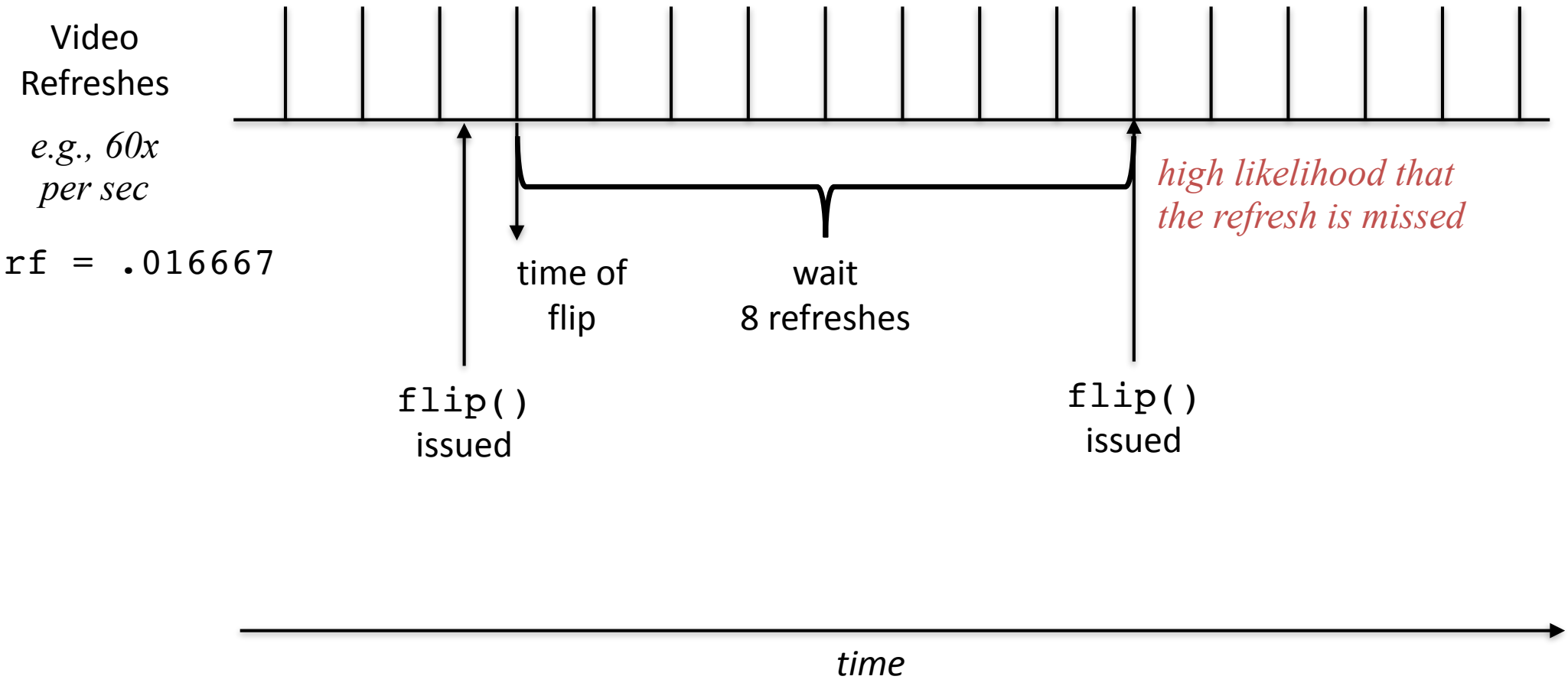
timing `win.flip()` with monitor refresh



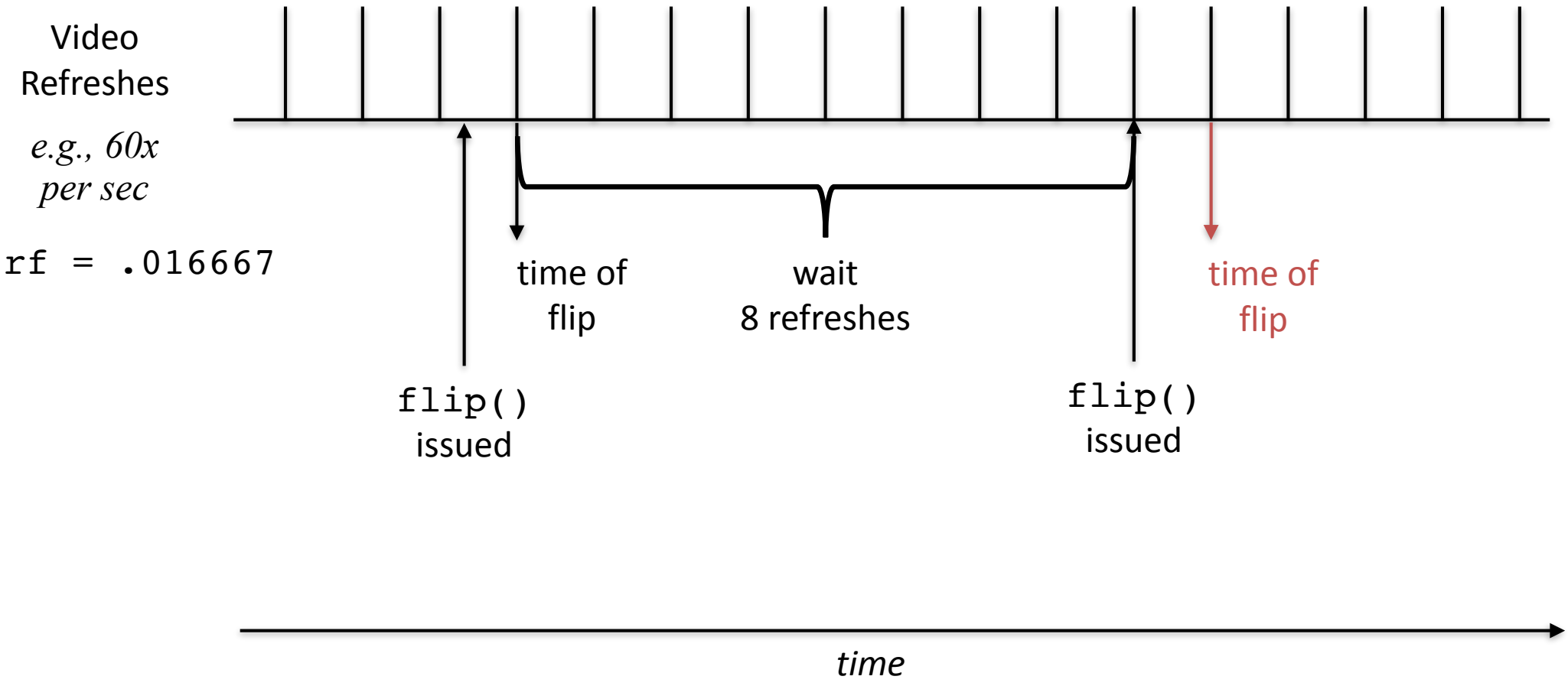
timing `win.flip()` with monitor refresh



timing `win.flip()` with monitor refresh



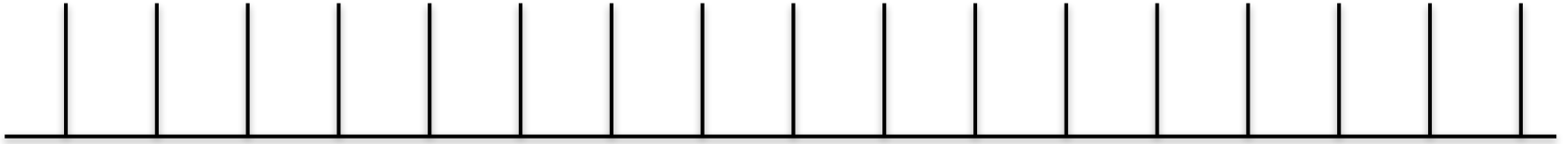
timing `win.flip()` with monitor refresh



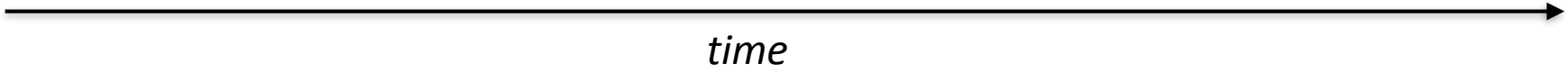
timing `win.flip()` using "slack"

Video Refreshes

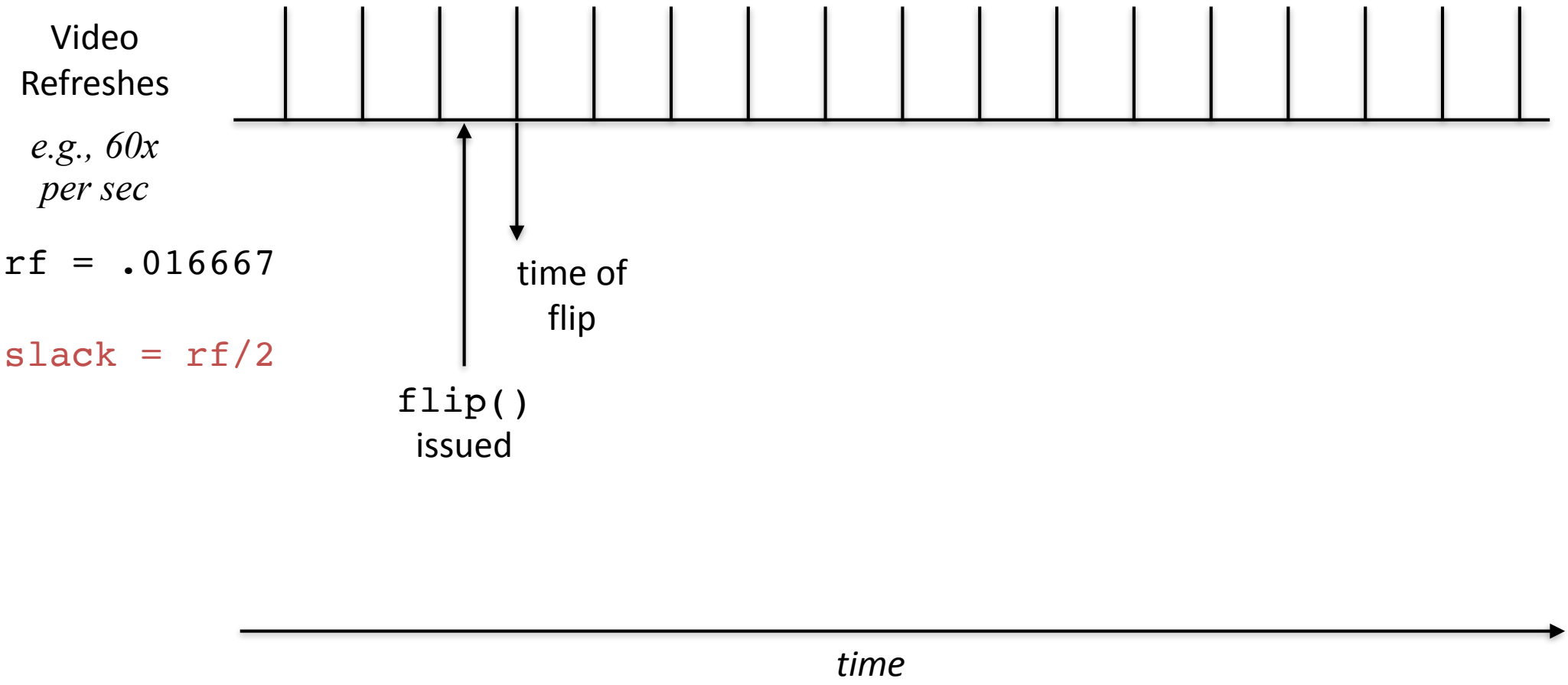
e.g., 60x per sec



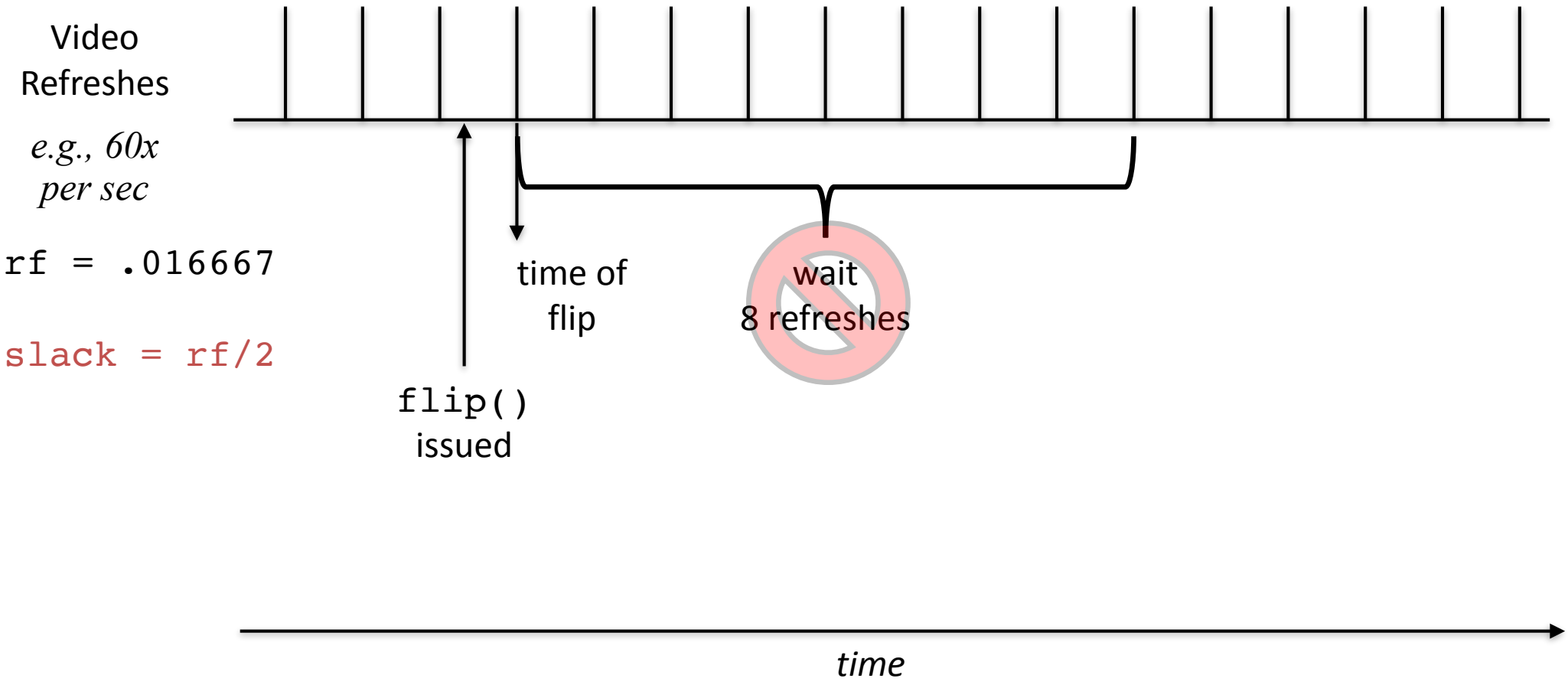
```
rf = .016667

slack = rf/2
```

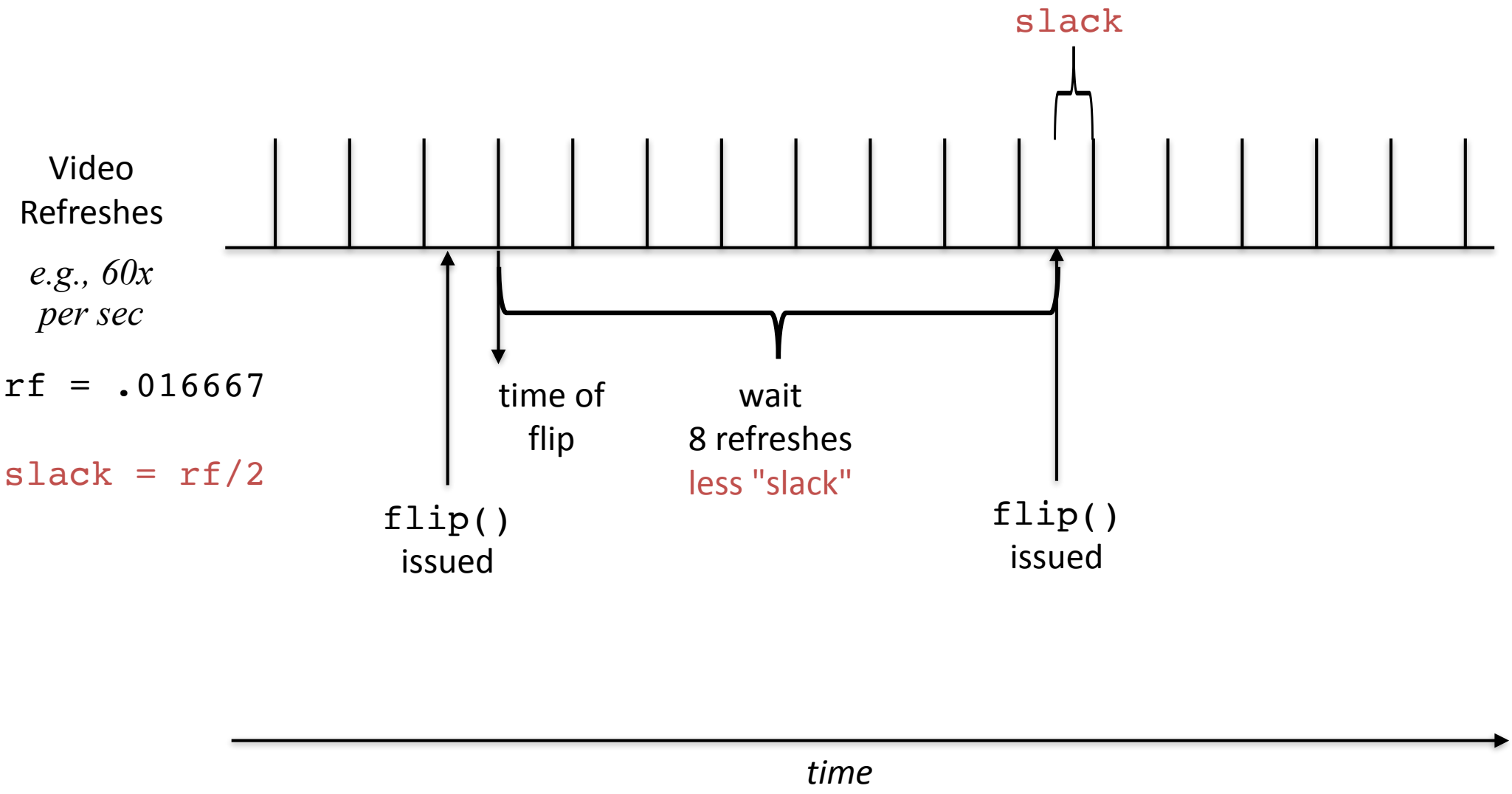


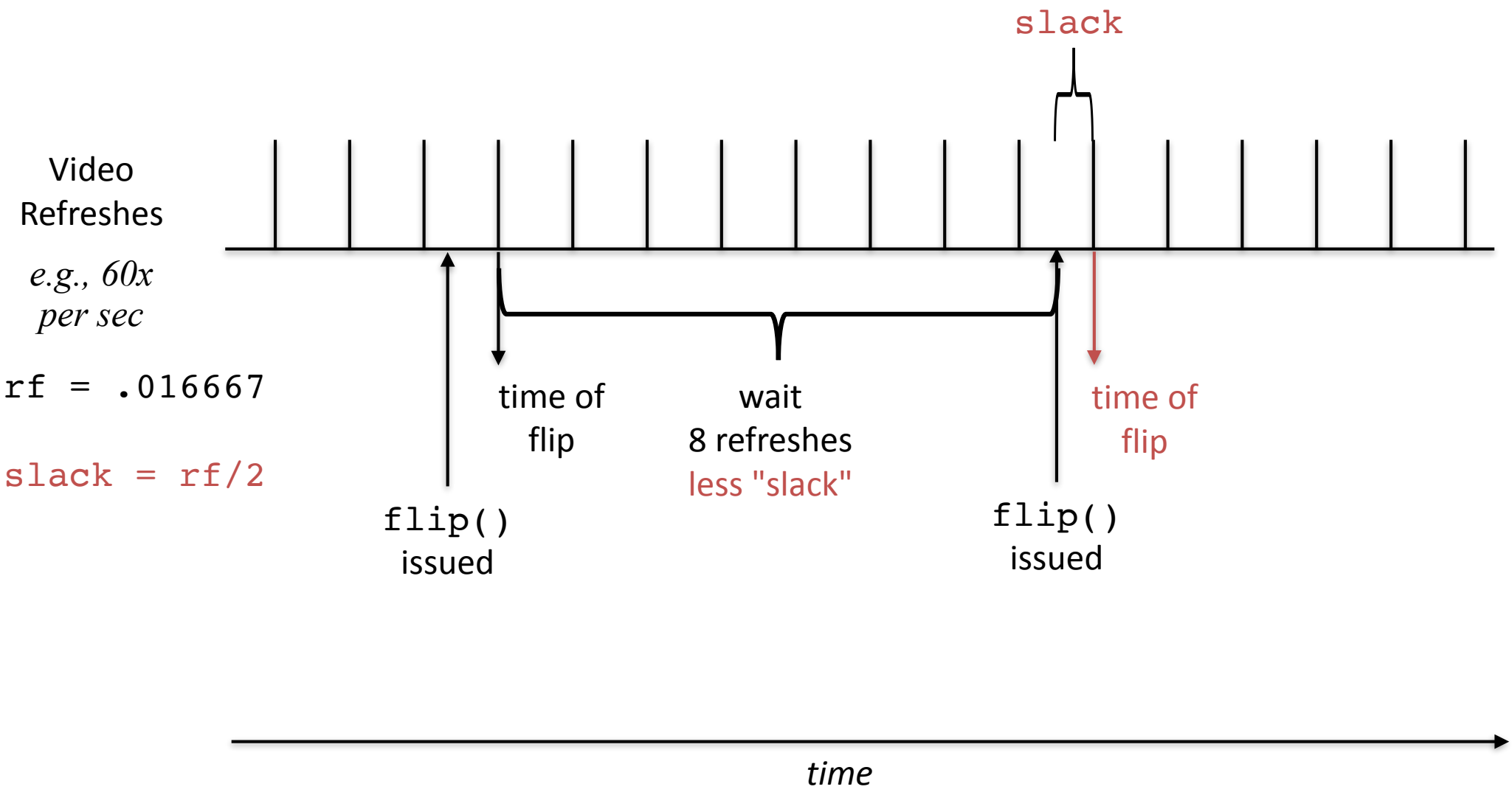
timing `win.flip()` using "slack"



timing `win.flip()` using "slack"



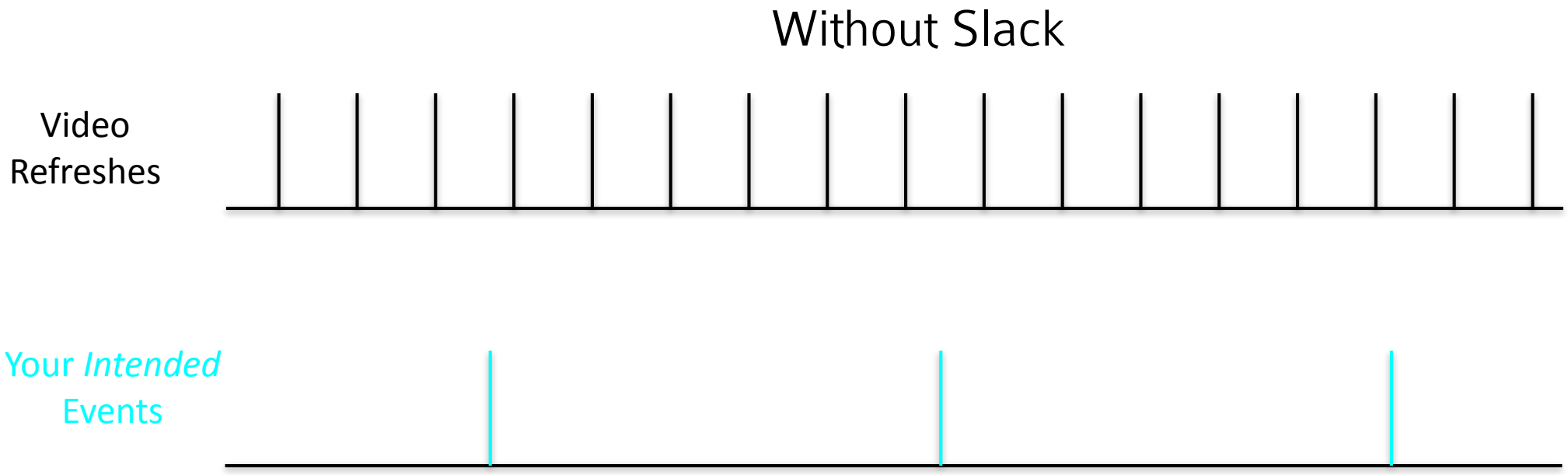

slack
Video
Refreshes
e.g., 60x
per sec
rf = .016667
slack = rf/2
time of
flip
wait
8 refreshes
less "slack"
flip()
issued
flip()
issued
time


slack
Video
Refreshes
e.g., 60x
per sec
rf = .016667
slack = rf/2
time of
flip
wait
8 refreshes
less "slack"
time of
flip
flip()
issued
flip()
issued
time

timing `win.flip()` using "slack"

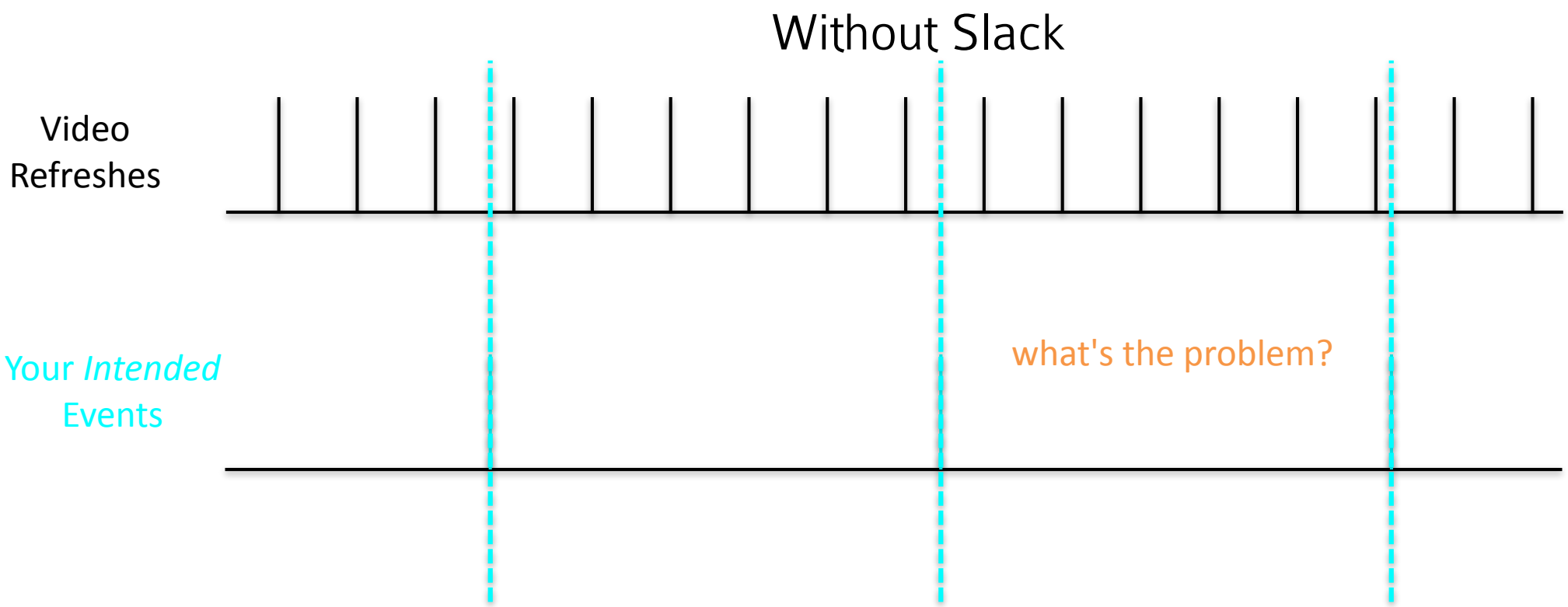
`test10.py`

using "slack" in general



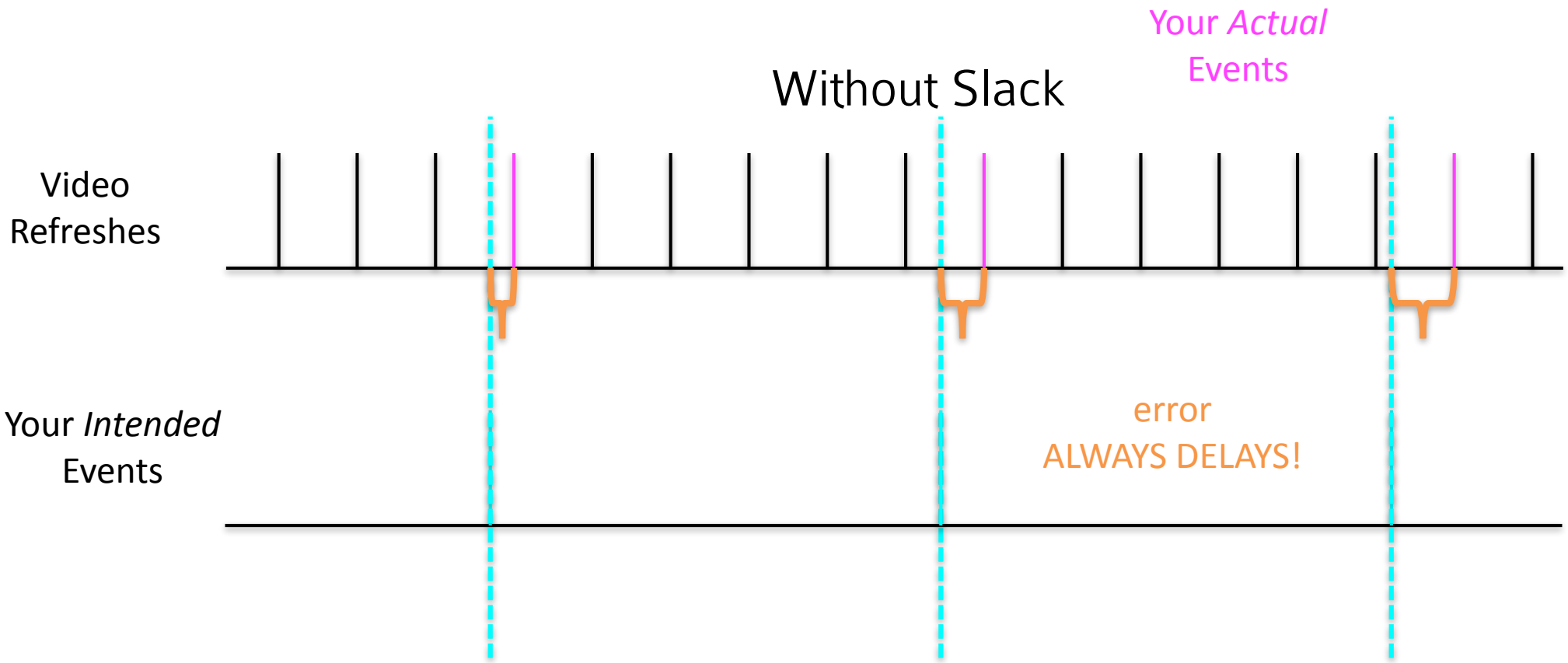
not necessarily trying to do precise stimulus timing

using "slack" in general



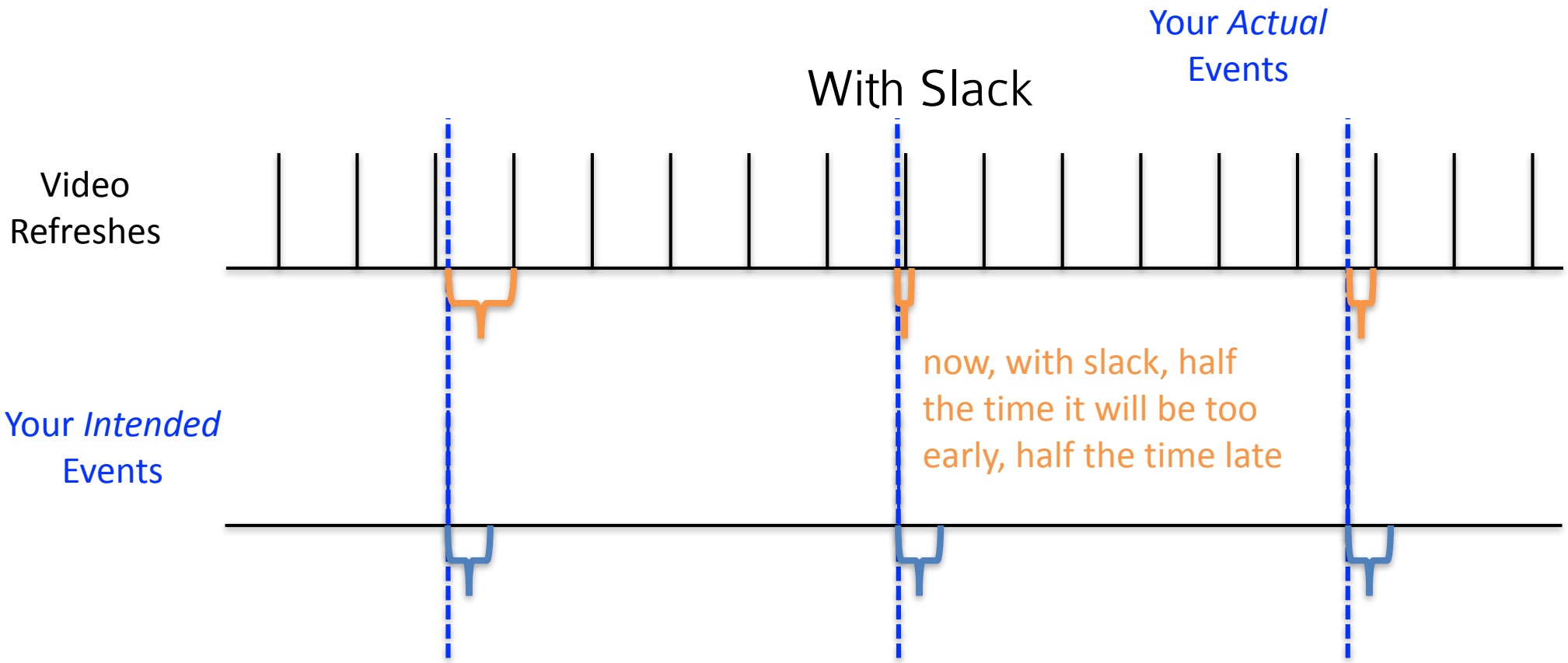
not necessarily trying to do precise stimulus timing

using "slack" in general



not necessarily trying to do precise stimulus timing

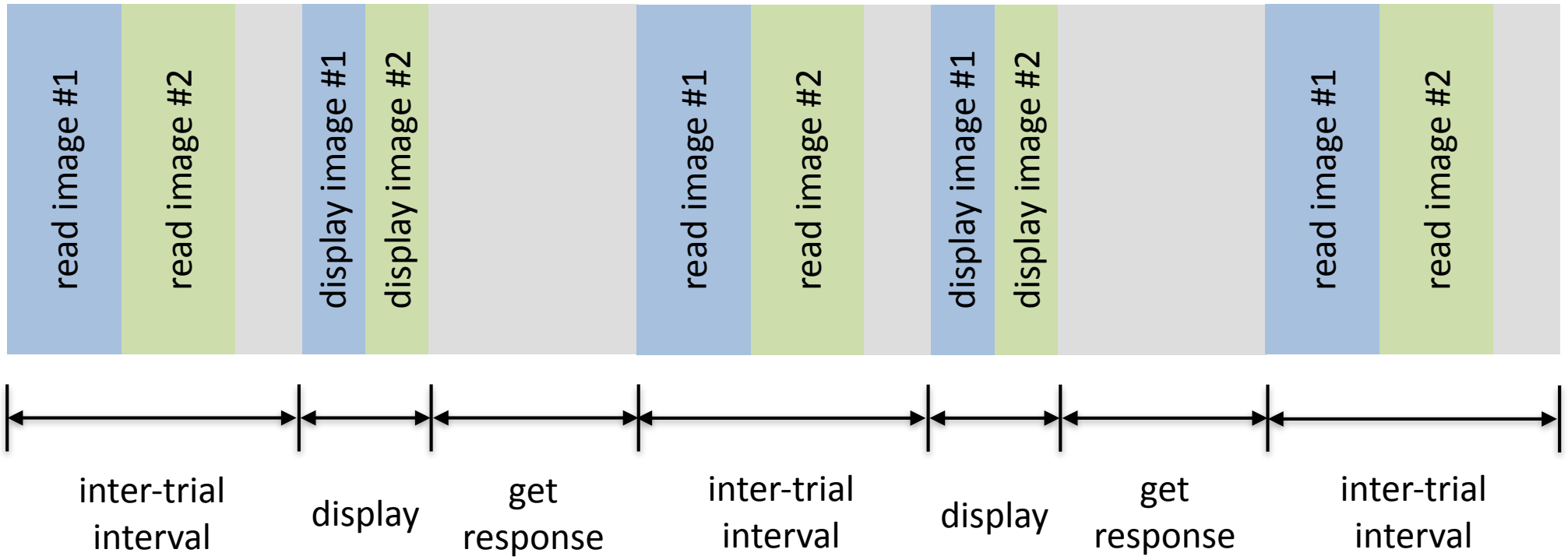
using "slack" in general



(and cannot be)

what can be done during short intervals

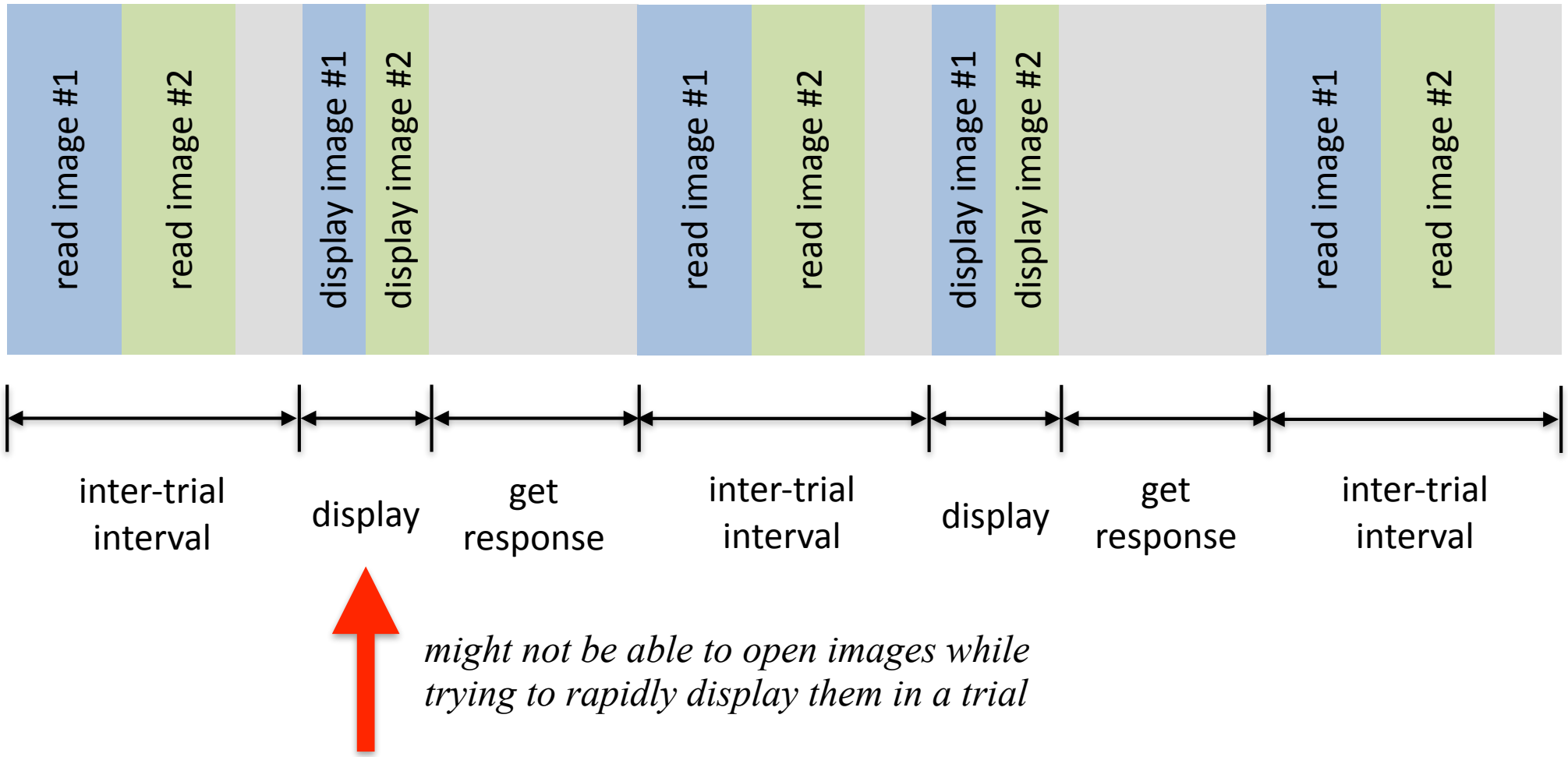
imagine a same-different experiment with rapid image presentation times



(and cannot be)

what can be done during short intervals

imagine a same-different experiment with rapid image presentation times



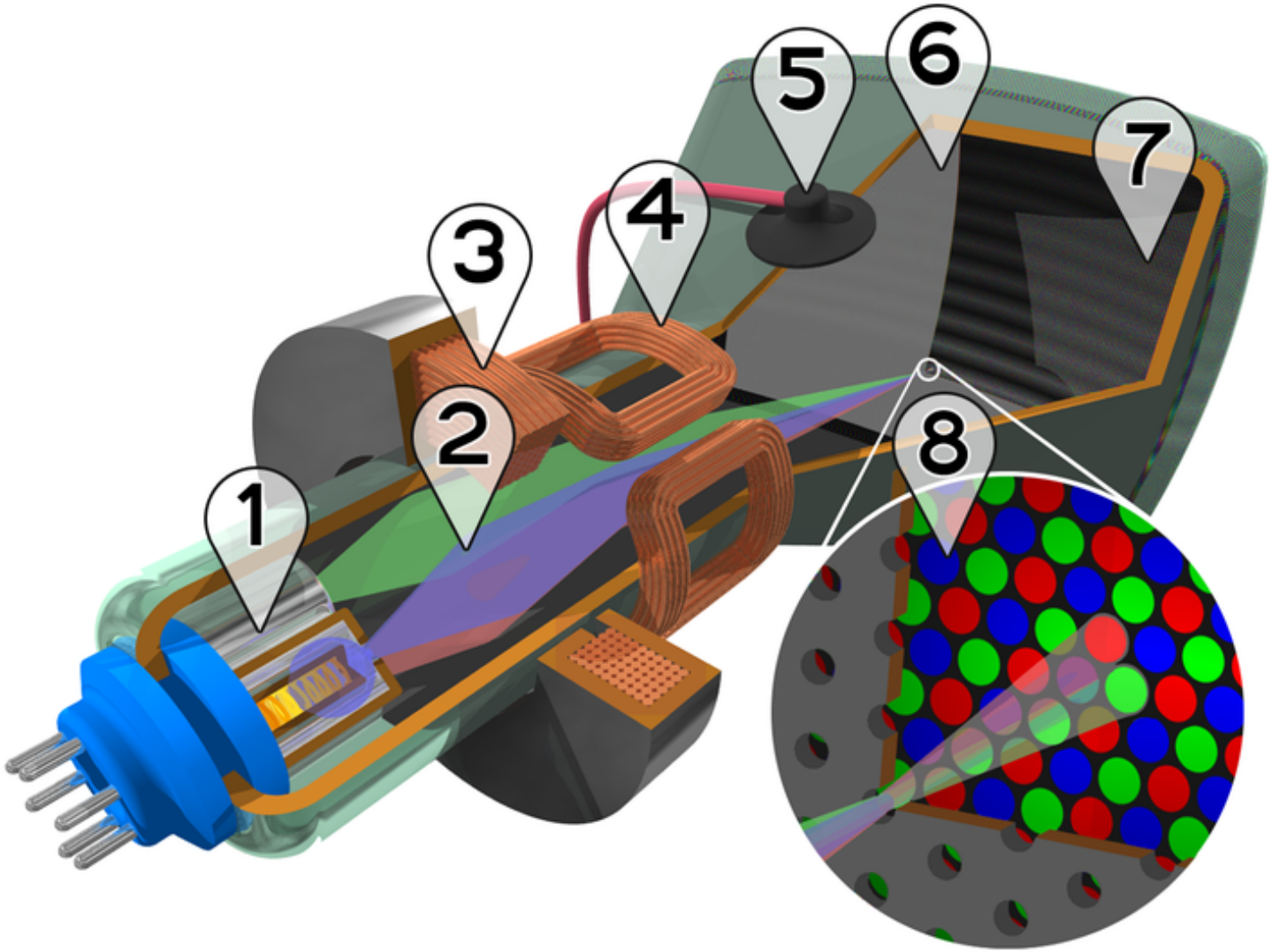
what can be done during short intervals

`test11.py`

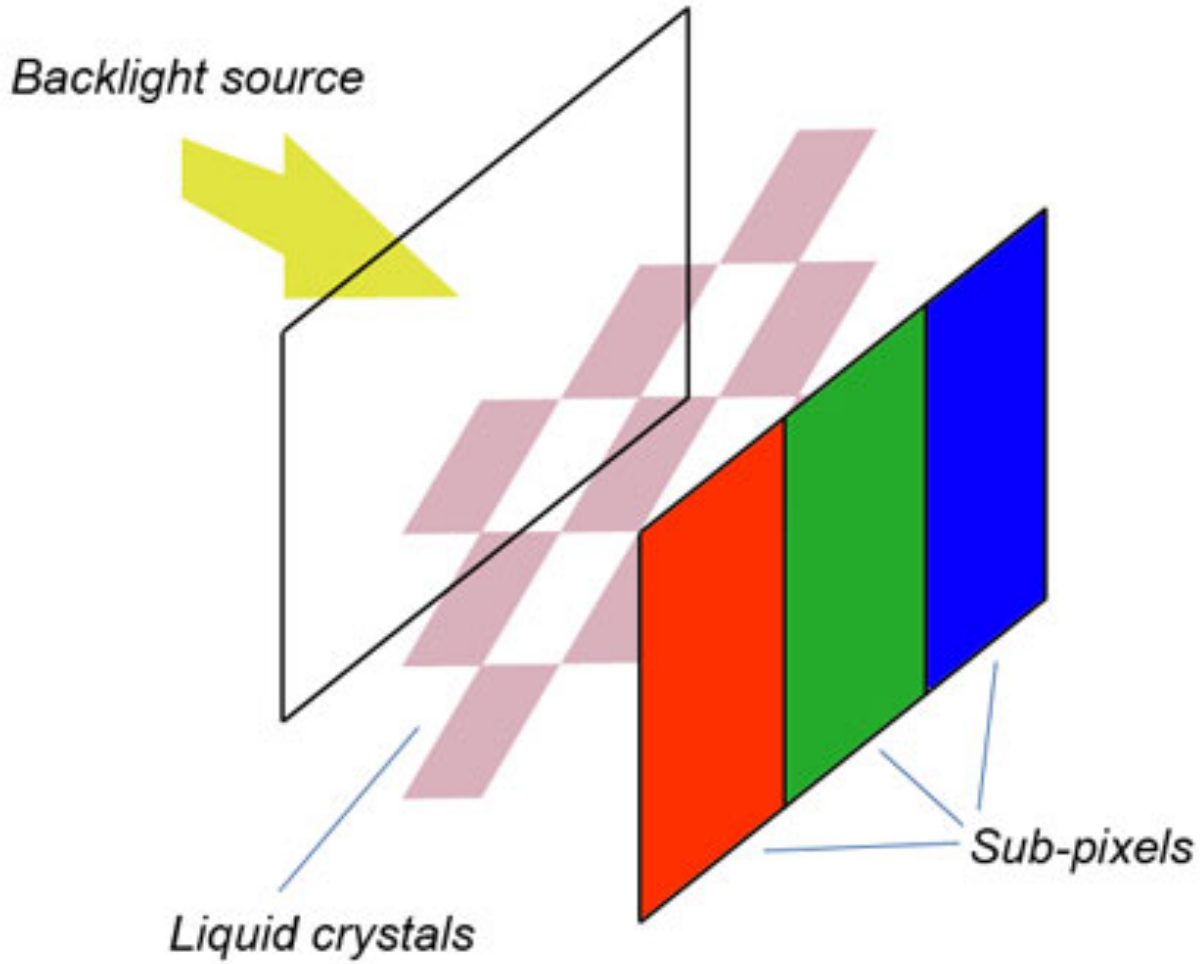
CRT versus LCD monitors

Timing of visual stimuli is not simply limited by proper updating of data on the graphics card. The physical monitor imposes limits as well, and CRTs and LCDs are different.

CRT monitor

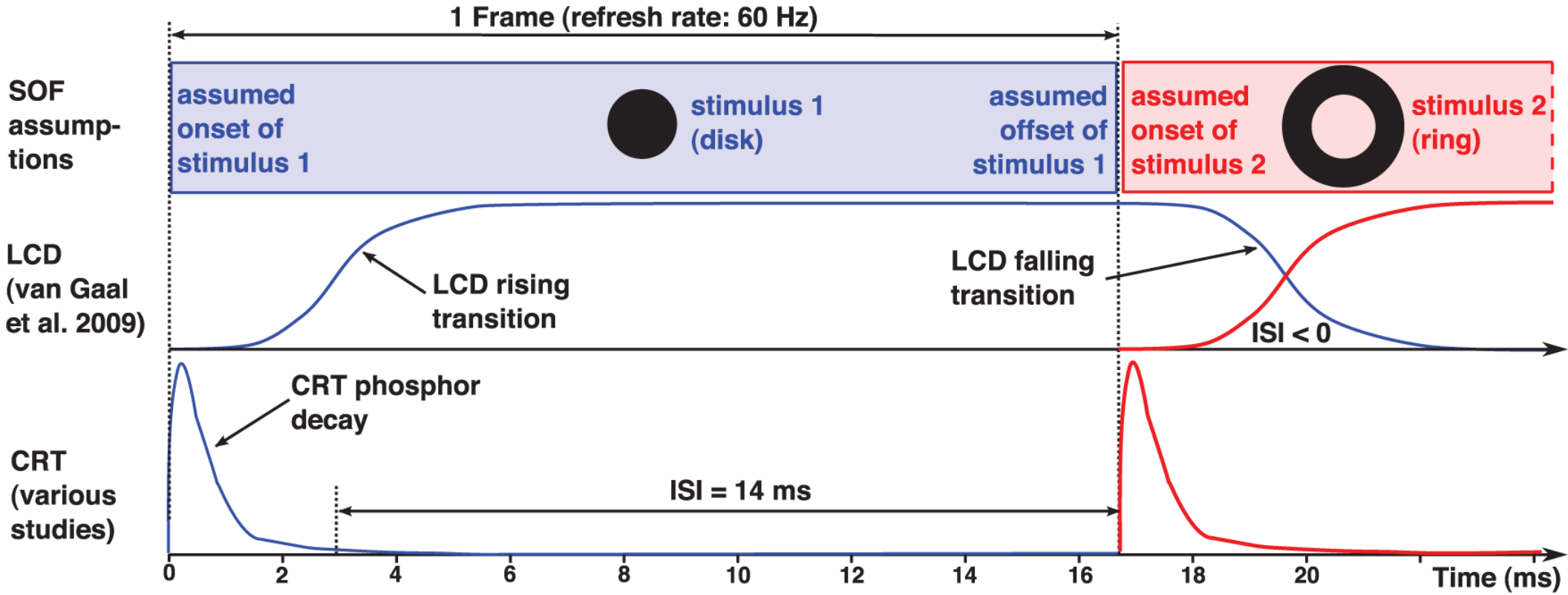


LCD monitor



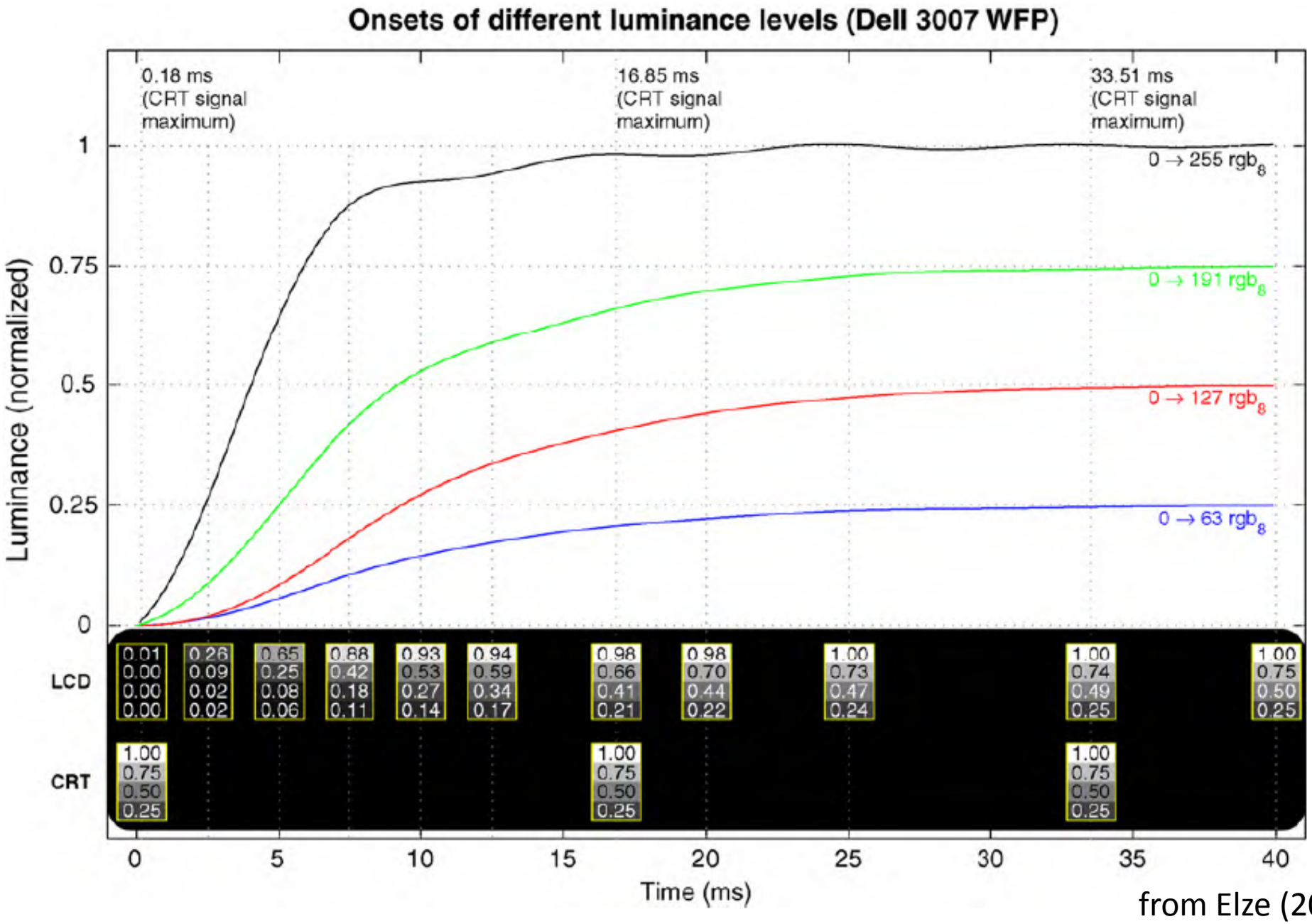
CRT versus LCD monitors

Make sure you understand the timing and how it differs between CRT and LCD monitors. Timing of stimulation is different from timing between events.



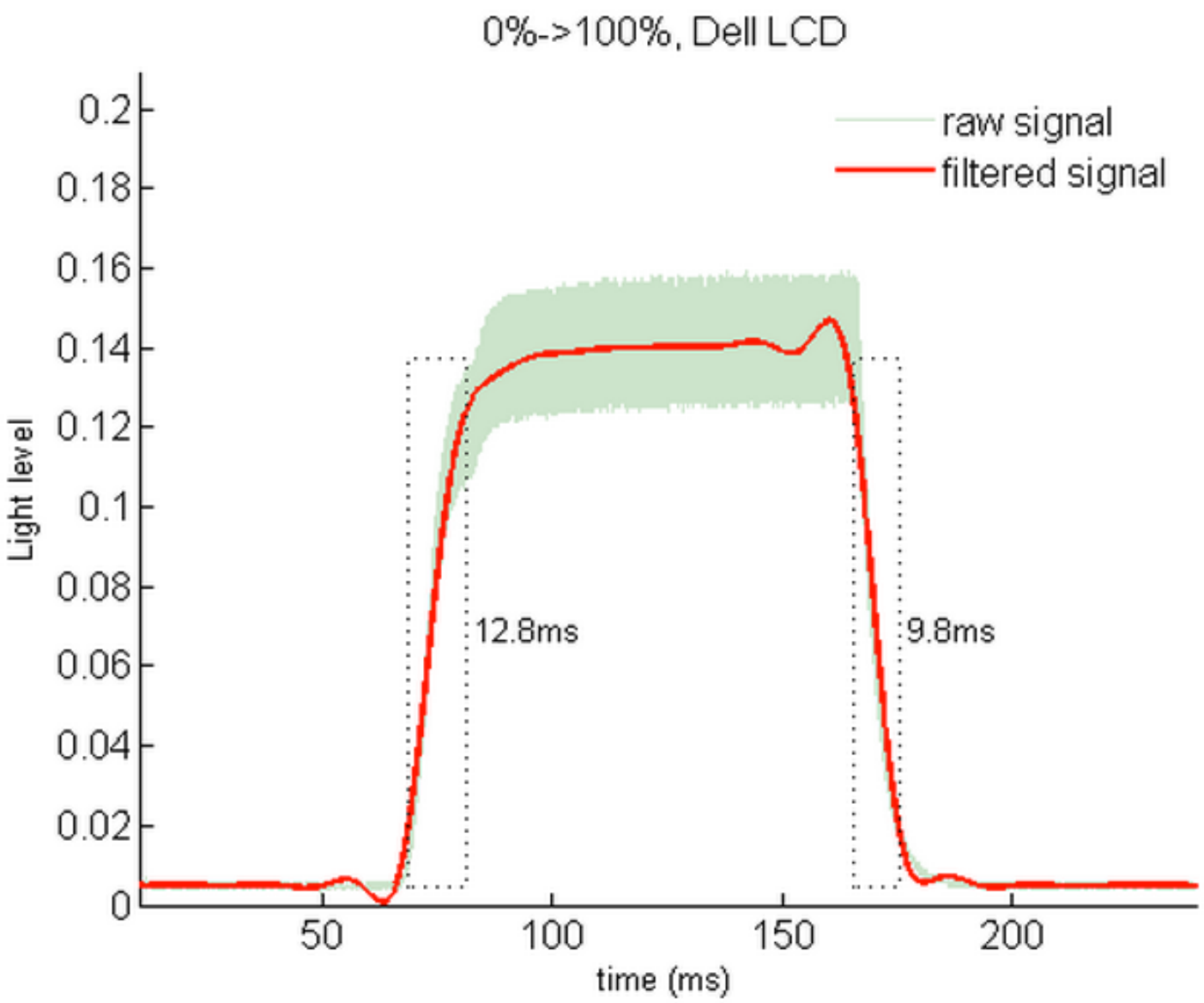
from Elze (2010)

a bad LCD monitor



from Elze (2010)

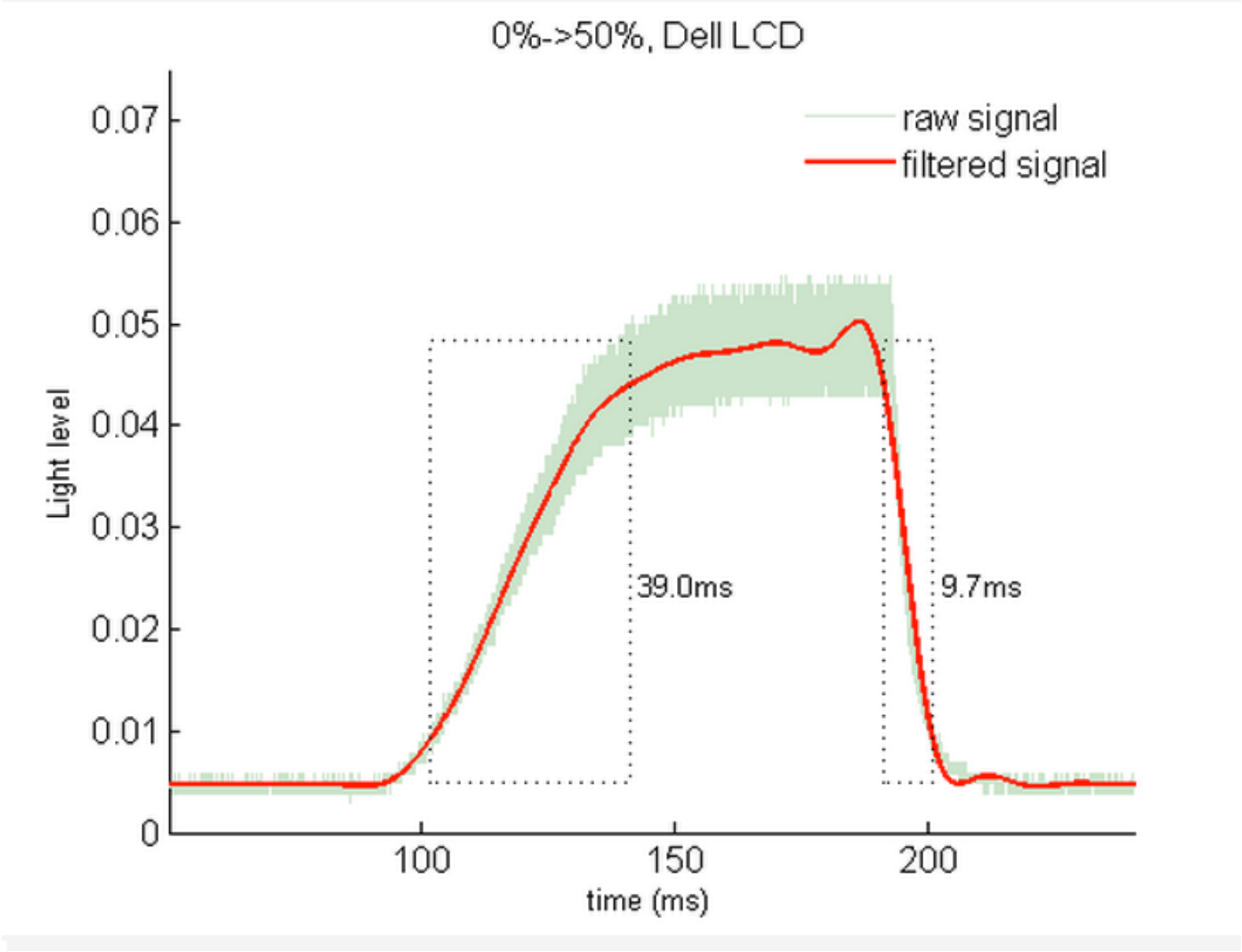
LCD in my lab several years ago



For comparison: CRT 1ms rise, 1.1ms fall

courtesy of Mike Mack

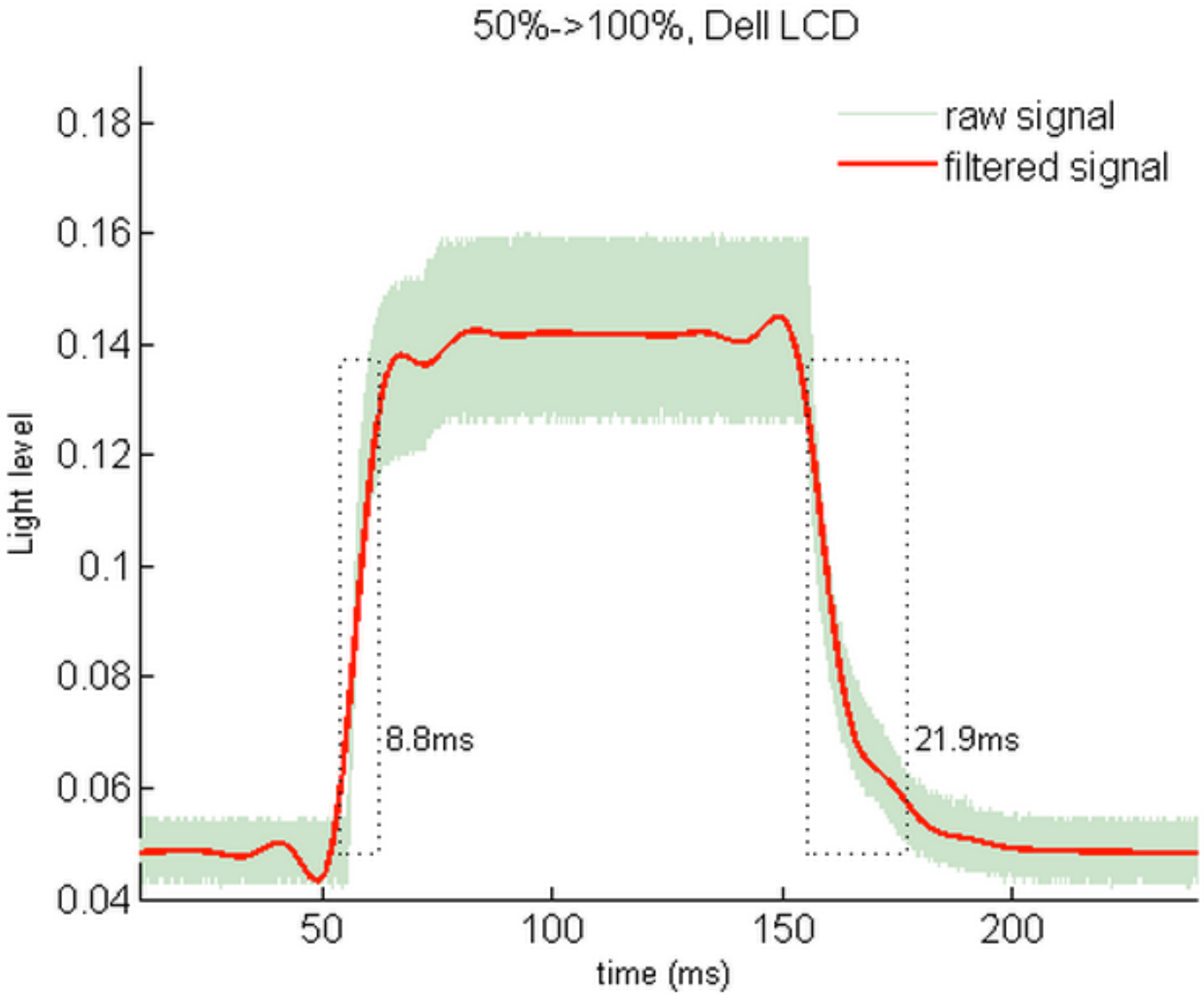
LCD in my lab several years ago



For comparison: CRT 1ms rise, 1.1ms fall

courtesy of Mike Mack

LCD in my lab several years ago



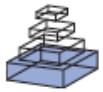
courtesy of Mike Mack

good LCD monitors are available

You need to do measurements on your own monitors.

frontiers in
HUMAN NEUROSCIENCE

METHODS ARTICLE
published: 18 August 2011
doi: 10.3389/fnhum.2011.00085



An LCD monitor with sufficiently precise timing for research in vision

Peng Wang*[1,2] *and Danko Nikolić*[1,2]*

[1] *Department of Neurophysiology, Max-Planck Institute for Brain Research, Frankfurt am Main, Germany*
[2] *Frankfurt Institute for Advanced Studies, Johann Wolfgang Goethe University, Frankfurt am Main, Germany*

Edited by:
Michael X. Cohen, University of Amsterdam, Netherlands

Reviewed by:
Clifford D. Saron, University of California at Davis, USA
Tomas Knapen, University of Amsterdam, Netherlands

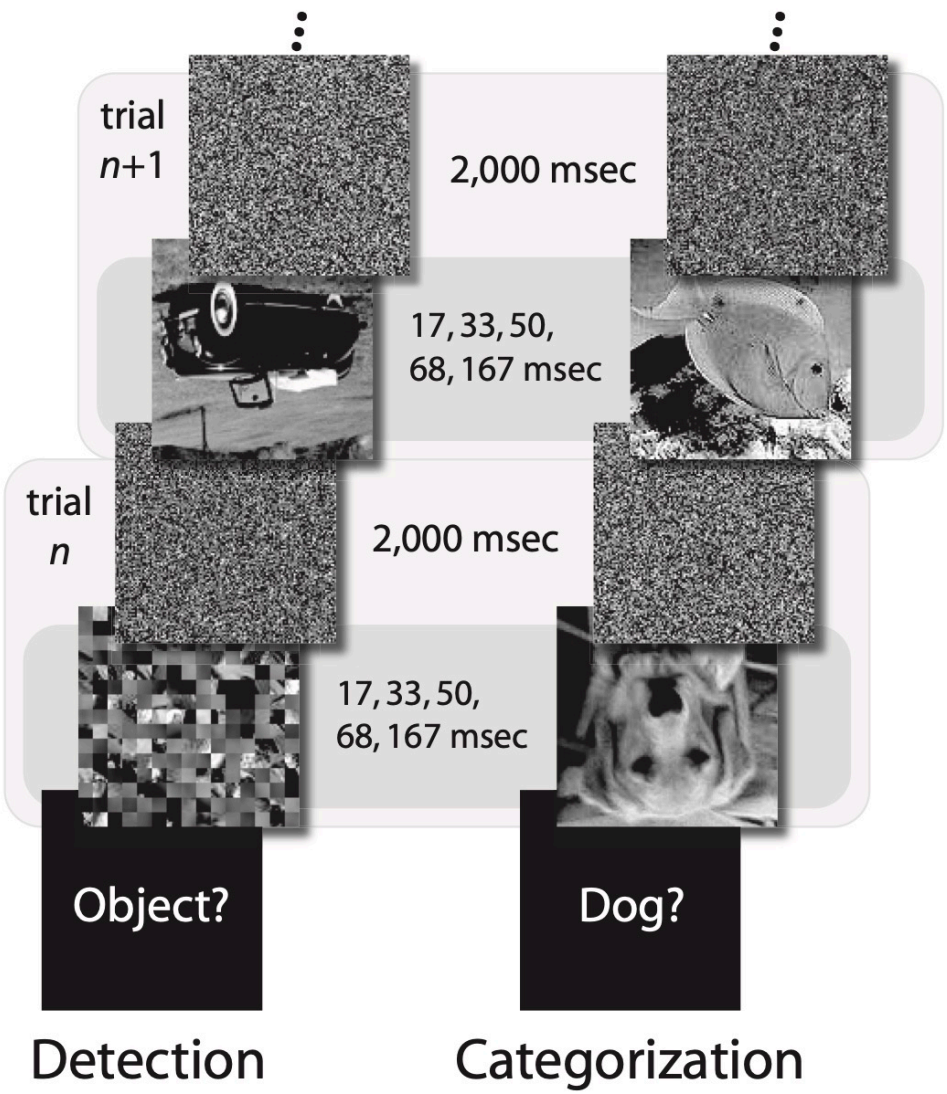
****Correspondence:***
Danko Nikolić, Department of Neurophysiology, Max-Planck Institute for Brain Research, Deutschordenstr. 46, 60528 Frankfurt am Main, Germany.
e-mail: danko.nikolic@gmail.com

Until now, liquid crystal display (LCD) monitors have not been used widely for research in vision. Despite their main advantages of continuous illumination and low electromagnetic emission, these monitors had problems with timing and reliability. Here we report that there is at least one new inexpensive 120 Hz model, whose timing and stability is on a par with a benchmark cathode-ray tube monitor, or even better. The onset time was stable across repetitions, 95% confidence interval (the error) of which was <0.01 ms. Brightness was also delivered reliably across repeated presentations (<0.04% error) and across blocks with different durations (<3% error). The LCD monitor seems suitable for many applications in vision research, including the studies that require combined accuracy of timing and intensity of visual stimulation.

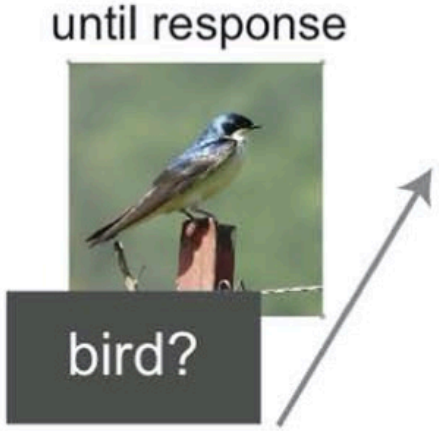
Keywords: visual stimulation, display, stimulus delivery, precise timing, LCD, CRT

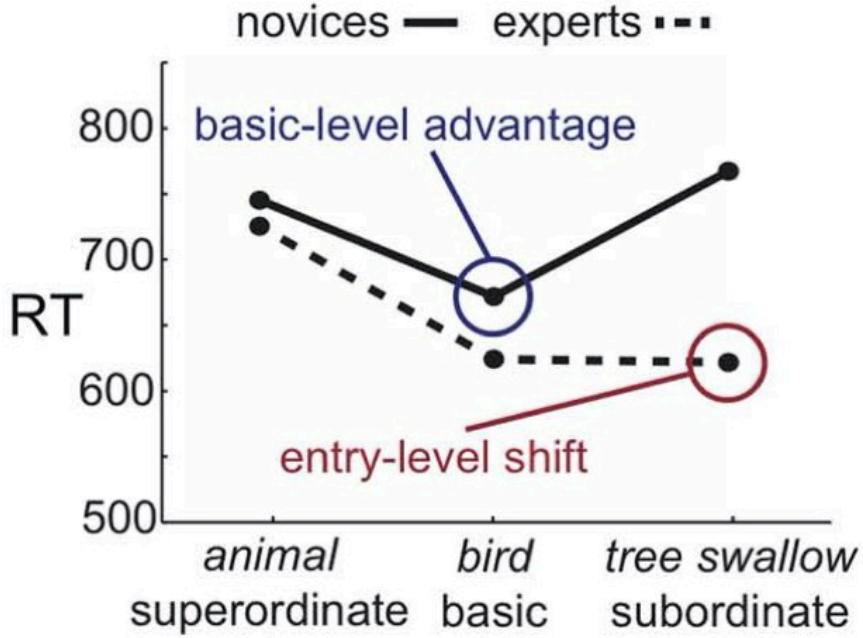
examples

1) manipulating time



2) measuring time

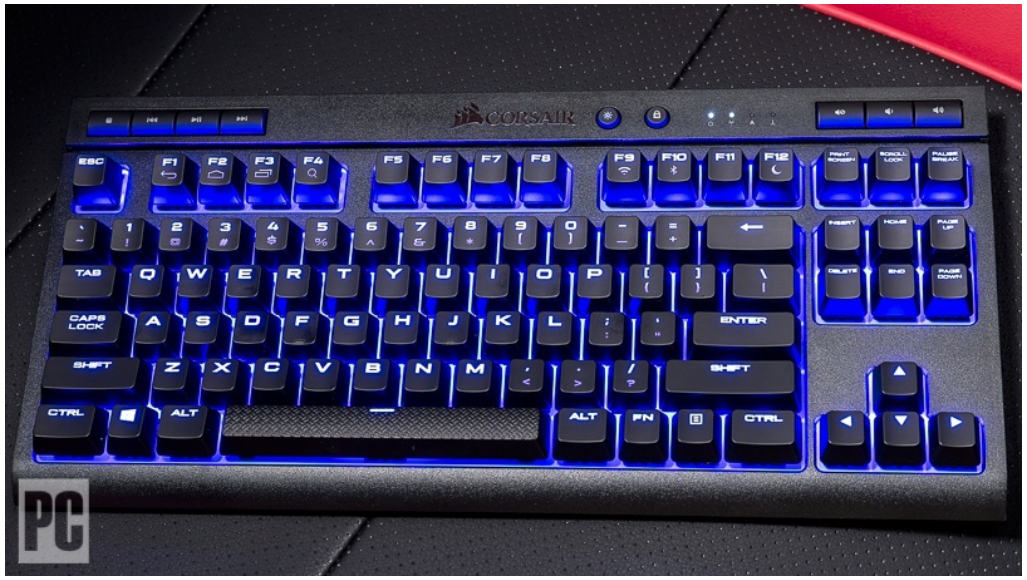


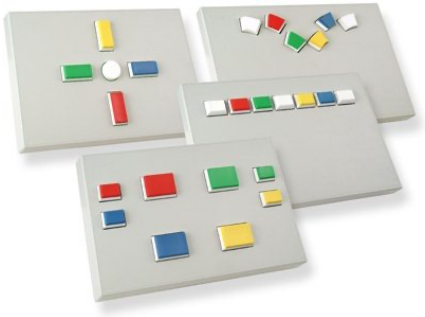


measuring time in PsychoPy

keyboard responses

other response interfaces





keyboard commands

`test12.py`

keyboard commands

```
from psychopy import events

event.clearEvents()

keys = event.waitKeys()
```

vs

```
keys = event.getKeys()
```

keyboard commands

```
from psychopy import events

event.clearEvents()
```

clear all mouse and keyboard events

```
keys = event.waitKeys()
```

vs

```
keys = event.getKeys()
```

keyboard commands

```
from psychopy import events

event.clearEvents()

keys = event.waitKeys()
```

returns a list of strings

vs

```
keys = event.getKeys()
```

returns a list of strings

keyboard commands

```
from psychopy import events

event.clearEvents()

keys = event.waitKeys()
```

waits for a key press

vs

```
keys = event.getKeys()
```

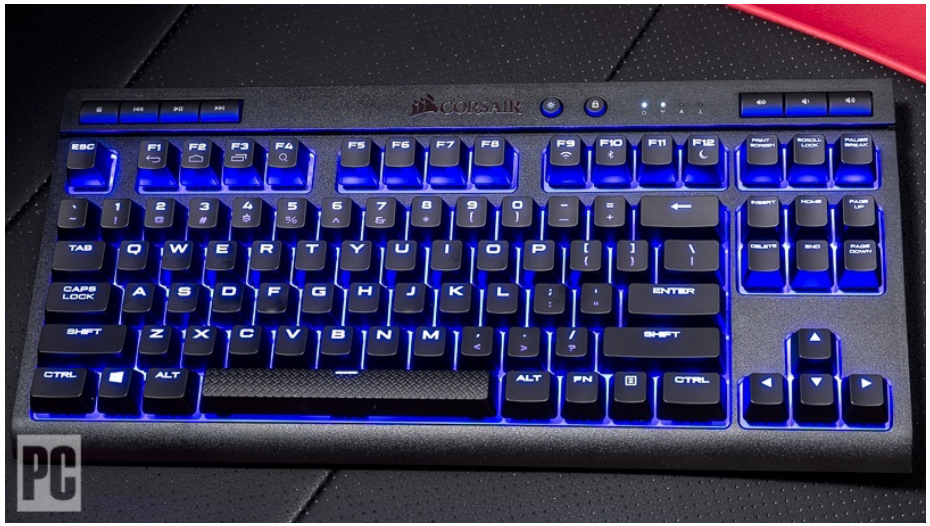
checks for a key press, otherwise returns [ ]

Note: in terms of how PsychoPy is coded, waitKeys() is just a wrapping of getKeys() within a while loop

keyboard commands

returns key that is hit:

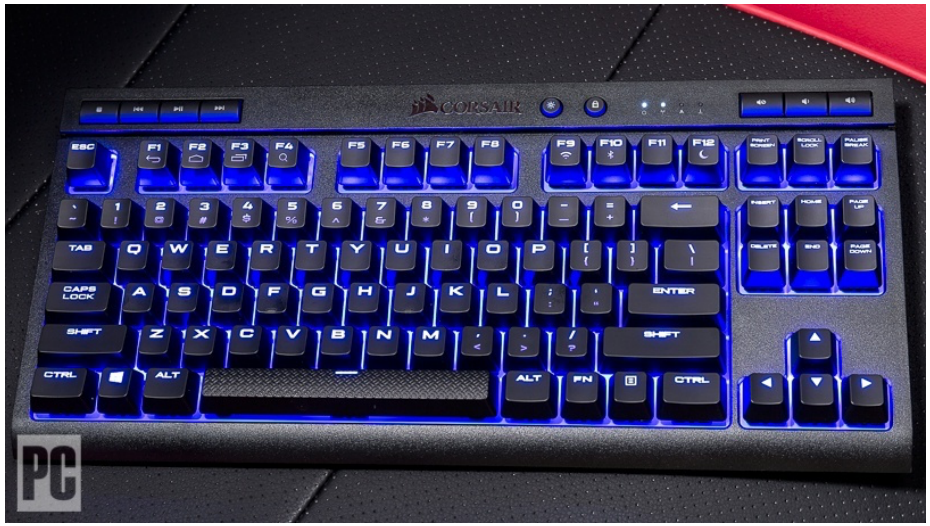
```
['1']
['lshift']
['f']
['quoteleft']
['rshift']
['lshift']
['escape']
['return']
['capslock']
['backspace']
['space']
['j']
['loption']
```



keyboard commands

can return multiple keys hit:

```
['d', 'f']
['s']
['i', 'j']
['k', 'o']
['k']
['g', 'd', 'h', 'j']
['s']
['n', '7', '8', '9', '0']
['d', 'h', 'g']
['k']
['j', 's']
['w', 'h']
['f', 'e']
['c', 'j', 'n', 's']
['q']
```



recording response time from key press

`test13.py`

recording response time from key press

```
myClock = core.Clock()
.
.
.
mywin.flip()
myClock.reset()

keys = event.waitKeys(keyList=['space', 'q'],
            timeStamped=myClock, maxWait=maxwait)
```

recording response time from key press

```
myClock = core.Clock()
.
.
.
mywin.flip()
myClock.reset()

keys = event.waitKeys(keyList=['space', 'q'],
            timeStamped=myClock, maxWait=maxwait)
```

create a clock early in the program

recording response time from key press

```
myClock = core.Clock()
.
.
.
mywin.flip()
myClock.reset()

keys = event.waitKeys(keyList=['space', 'q'],
            timeStamped=myClock, maxWait=maxwait)
```

flip() that displays the stimulus

recording response time from key press

```
myClock = core.Clock()
.
.
.
mywin.flip()
myClock.reset()
```

reset the clock to measure RT

```
keys = event.waitKeys(keyList=['space', 'q'],
            timeStamped=myClock, maxWait=maxwait)
```

recording response time from key press

```
myClock = core.Clock()
.
.
.
mywin.flip()
myClock.reset()
```

if you want to limit legal keys

```
keys = event.waitKeys(keyList=['space', 'q'],
            timeStamped=myClock, maxWait=maxwait)
```

recording response time from key press

```
myClock = core.Clock()
.
.
.
mywin.flip()
myClock.reset()

keys = event.waitKeys(keyList=['space', 'q'],
                      timeStamped=myClock, maxWait=maxwait)
```

what clock to use to measure RT

recording response time from key press

```
myClock = core.Clock()
.
.
.
mywin.flip()
myClock.reset()

keys = event.waitKeys(keyList=['space', 'q'],
            timeStamped=myClock, maxWait=maxwait)
```

when to stop waiting

recording response time while polling

`test14.py`

recording response time while polling

```
mywin.flip()
myClock.reset()

loop = True
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(keyList=['space', 'q'],
                         timeStamped=myClock)

    if len(keys):
        loop = False

    core.wait(0.0001)
```

recording response time while polling

```
mywin.flip()
myClock.reset()
```

start the clock

```
loop = True
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(keyList=['space', 'q'],
                         timeStamped=myClock)

    if len(keys):
        loop = False

    core.wait(0.0001)
```

recording response time while polling

```
mywin.flip()
myClock.reset()

loop = True
while loop:                    polling loop
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(keyList=['space', 'q'],
                         timeStamped=myClock)

    if len(keys):
        loop = False

    core.wait(0.0001)
```

recording response time while polling

```
mywin.flip()
myClock.reset()

loop = True
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(keyList=['space', 'q'],
                         timeStamped=myClock)

    if len(keys):
        loop = False

    core.wait(0.0001)
```

simple example of stimulus change while polling keyboard

recording response time while polling

```
mywin.flip()
myClock.reset()

loop = True
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(keyList=['space', 'q'],
                         timeStamped=myClock)
    if len(keys):
        loop = False

    core.wait(0.0001)
```

get key press and time stamp

recording response time while polling

```
mywin.flip()
myClock.reset()

loop = True
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(keyList=['space', 'q'],
                         timeStamped=myClock)

    if len(keys):
        loop = False

    core.wait(0.0001)
```

stopping looping when valid key hit

better way to check keyboard presses

`test15.py`

better way to check keyboard presses

```
mywin.flip(); myClock.reset()
loop = True; badkey = False
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(timeStamped=myClock)

    if keys:
        if keys[0][0] in ['d', 'k', 'q']:
            loop = False
            choice = keys[0][0]
            rt = keys[0][1]
        else:
            badkey = True

    core.wait(0.0001)
```

better way to check keyboard presses

```
mywin.flip(); myClock.reset()
loop = True; badkey = False
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()
```

allowing any key to be accepted

```
    keys = event.getKeys(timeStamped=myClock)

    if keys:
        if keys[0][0] in ['d', 'k', 'q']:
            loop = False
            choice = keys[0][0]
            rt = keys[0][1]
        else:
            badkey = True

    core.wait(0.0001)
```

better way to check keyboard presses

```
mywin.flip(); myClock.reset()
loop = True; badkey = False
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(timeStamped=myClock)

    if keys: has a key been pressed
        if keys[0][0] in ['d', 'k', 'q']:
            loop = False
            choice = keys[0][0]
            rt = keys[0][1]
        else:
            badkey = True

    core.wait(0.0001)
```

better way to check keyboard presses

```
mywin.flip(); myClock.reset()
loop = True; badkey = False
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(timeStamped=myClock)

    if keys:
        if keys[0][0] in ['d', 'k', 'q']:
            loop = False
            choice = keys[0][0]
            rt = keys[0][1]
        else:
            badkey = True

    core.wait(0.0001)
```

is it an acceptable keypress

better way to check keyboard presses

```
mywin.flip(); myClock.reset()
loop = True; badkey = False
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(timeStamped=myClock)

    if keys:
        if keys[0][0] in ['d', 'k', 'q']:
            loop = False
            choice = keys[0][0]
            rt = keys[0][1]
        else:
            badkey = True

    core.wait(0.0001)
```

stop looping and record choice and response time

better way to check keyboard presses

```
mywin.flip(); myClock.reset()
loop = True; badkey = False
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(timeStamped=myClock)

    if keys:
        if keys[0][0] in ['d', 'k', 'q']:
            loop = False
            choice = keys[0][0]
            rt = keys[0][1]
        else:
            badkey = True

    core.wait(0.0001)
```

note that a bad key initially pressed

better way to check keyboard presses

```
mywin.flip(); myClock.reset()
loop = True; badkey = False
while loop:
    if myClock.getTime() > maxstim:
        mywin.flip()

    keys = event.getKeys(timeStamped=myClock)

    if keys:
        if keys[0][0] in ['d', 'k', 'q']:
            loop = False
            choice = keys[0][0]
            rt = keys[0][1]
        else:
            badkey = True
```

could add a message that bad key pressed

```
    core.wait(0.0001)
```

more powerful keyboard routines, but may require a recompilation of PsychoPy

PsychoPy keyboard module

PsychoPy3

Forum **Download** Documentation

Keyboard

To handle input from keyboard (supercedes event.getKeys)

The Keyboard class was new in PsychoPy 3.1 and replaces the older *event.getKeys()* calls.

Psychtoolbox versus event.getKeys

On 64 bits Python3 installations it provides access to the Psychtoolbox kbQueue series of functions using the same compiled C code (available in python-psychtoolbox lib).

On 32 bit installations and Python2 it reverts to the older `psychopy.event.getKeys()` calls.

The new calls have several advantages:

- the polling is performed and timestamped asynchronously with the main thread so that times relate to when the key was pressed, not when the call was made
- the polling is direct to the USB HID library in C, which is faster than waiting for the operating system to poll and interpret those same packets
- we also detect the KeyUp events and therefore provide the option of returning keypress duration
- on Linux and Mac you can also distinguish between different keyboard devices (see `getKeyboards()`)

This library makes use, where possible of the same low-level asynchronous hardware polling as in Psychtoolbox

Example usage

```
from psychopy.hardware import keyboard
from psychopy import core

kb = keyboard.Keyboard()

# during your trial
kb.clock.reset()  # when you want to start the timer from
keys = kb.getKeys(['right', 'left', 'quit'], waitRelease=True)
if 'quit' in keys:
    core.quit()
for key in keys:
    print(key.name, key.rt, key.duration)
```

uses same routines developed for PsychToolbox

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