

xkcd.com

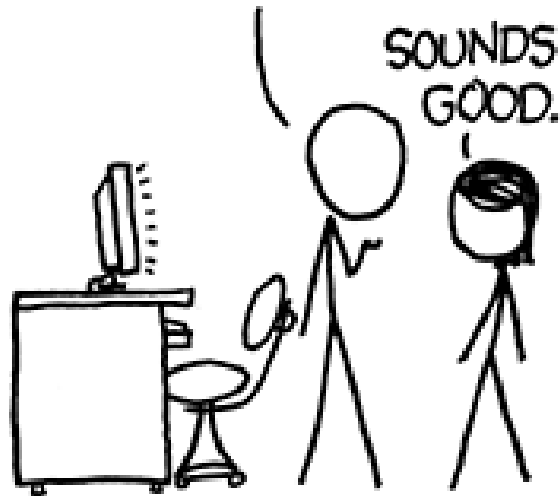
YOU KNOW THIS METAL  
RECTANGLE FULL OF  
LITTLE LIGHTS?

YEAH.



I SPEND MOST OF MY LIFE  
PRESSING BUTTONS TO MAKE  
THE PATTERN OF LIGHTS  
CHANGE HOWEVER I WANT.

SOUNDS  
GOOD.



BUT TODAY, THE PATTERN  
OF LIGHTS IS *ALL WRONG!*

OH GOD! TRY  
PRESSING MORE  
BUTTONS!  
*IT'S NOT  
HELPING!*



# Stimulus Precision using Psychopy

Jonas Kristoffer Lindeløv

Email: [jonas@cnru.dk](mailto:jonas@cnru.dk)

Twitter: [@jonaslindeloev](https://twitter.com/jonaslindeloev)

Blog for general public: ["hjernebloggen" at videnskab.dk](http://hjernebloggen.videnskab.dk)

Blog for neuroscientists: [lindeloev.net](http://lindeloev.net)

# Appearance

Visual angle  
Luminance



# Size and location

- Use cm or visual angle as general unit for size!

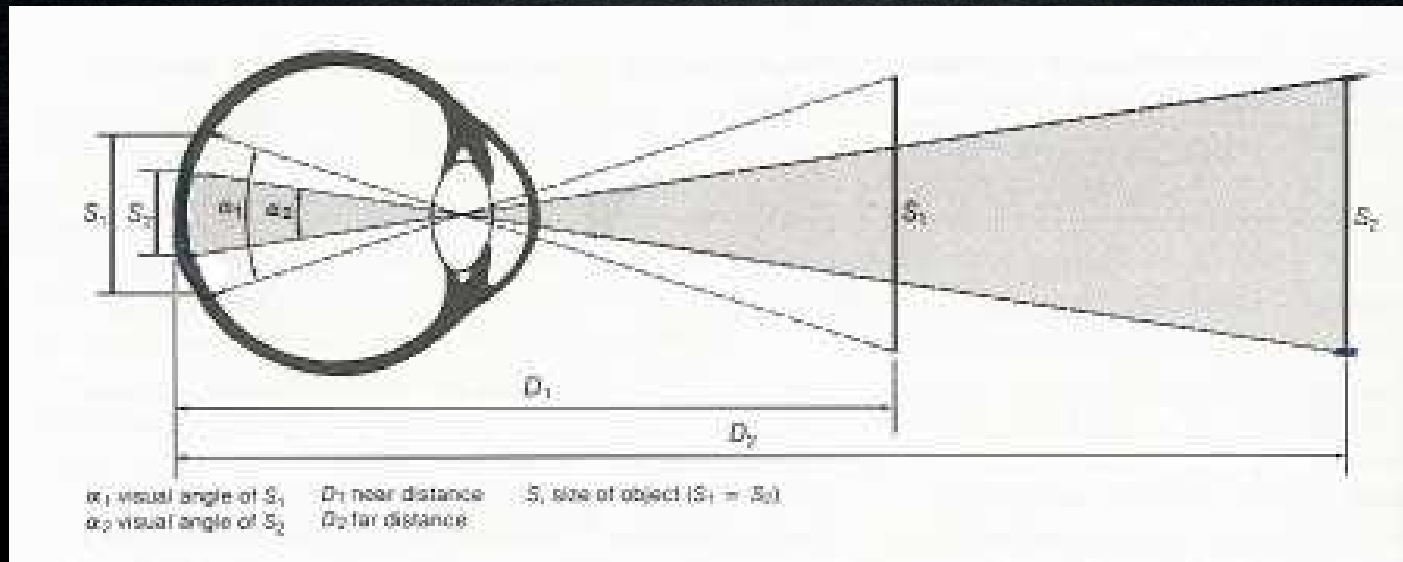
```
win = visual.Window(units='deg')
```

```
stim = visual.GratingStim(units='degFlat')
```

- Check the actual size and adjust as needed.

```
print ppc.deg2cm(angle=2, distance=60)
```

```
Text = visual.TextStim(win, height=2 * 1.6)
```



<http://www.psychopy.org/general/units.html>

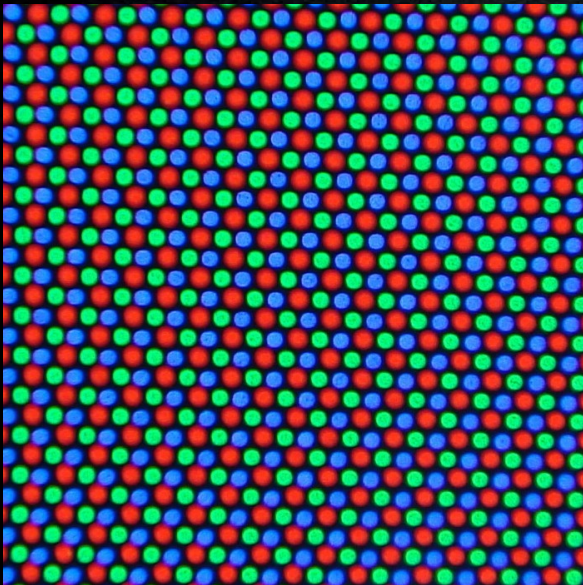
# Isoluminant color

- Get isoluminant colors using DKL colorspace.

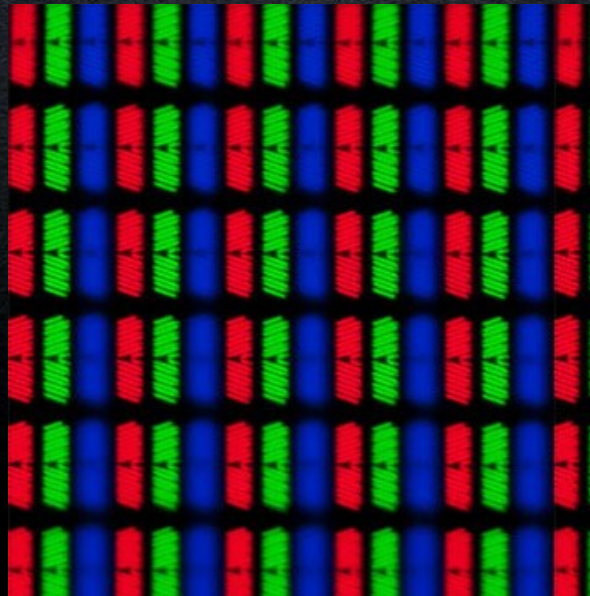
```
visual.ShapeStim(win, colorSpace='dkl',  
                  fillColor=[0, 0, 1])  
ppc.dkl2rgb([0, 0, 1])
```

- Adjust monitor using photometer.

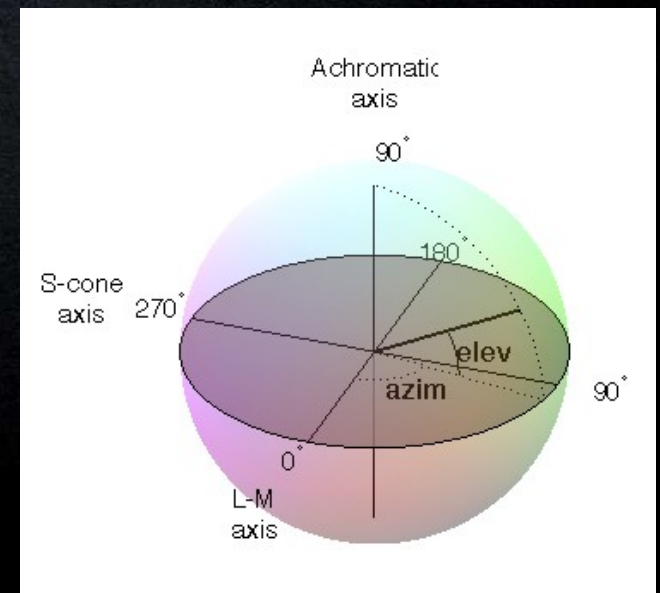
CRT



LCD



DKL colorspace



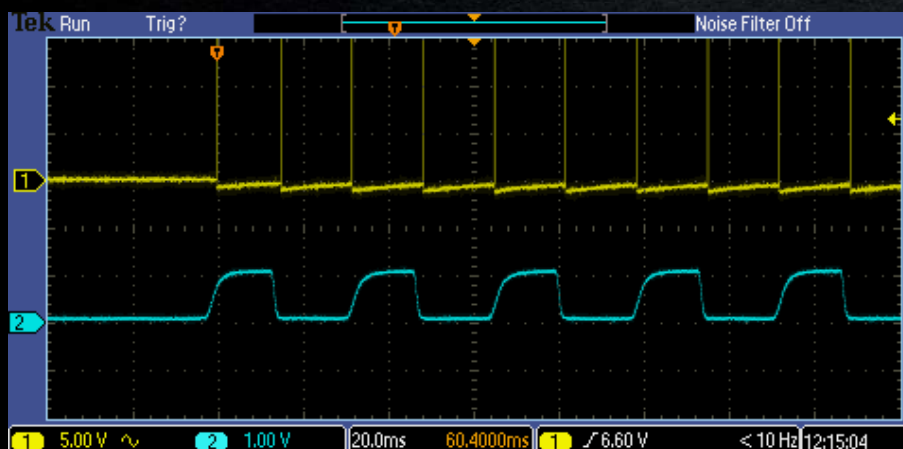
<http://www.psychopy.org/general/colours.html>



# Timing stimulation

# Digital sound is pretty continuous. Digital video is not.

- All about resolution!
- Video (ms)
  - 60 Hz: **16.7**, 33.3, 50.0, 66.7, 83.3, 100 ms, ...
  - 100 Hz: **10.0**, 20.0, 30.0, 40.0, 50.0, 60.0 ms, ...
- Audio (ms)
  - 44100 Hz: **0.02**, 0.05, 0.07, 0.09, 0.11 ms, ...
  - 11025 Hz: **0.09**, 0.18, 0.27, 0.36, 0.45 ms, ...



Slow motion videos  
of minitors:

Visual frames on LCD + CRT + TFT  
One frame on big LCD  
Visual frames on CRT



# Digital sound is pretty continuous. Digital video is not.

- Modern monitors are just fast slide projectors!
- ... that draw on the slides just before they are shown and discard them afterwards.





# Visual timing to frames

- Use frames for visual timing, NOT `core.wait()`!  
For frame in range(3):  
    `stim.draw()`  
    `win.flip()`
- Check your actual frame rate. It's not 60 Hz!  
    `ppc.getActualFrameRate()`
- Triggers and logging AFTER stimulus presentation!

## Yes:

```
win.flip()  
duration = clock.getTime()  
port.setData(15)
```

## No:

```
duration = clock.getTime()  
port.setData(15)  
win.flip()
```

# Visual timing to frames

- Test timing of everything in the `win.flip()` loop except `win.flip()` and make sure that processing duration is way below frame interval.

```
Script = """
    stimA.setOri(1.1, '*')
    stimA.setOpacity(0.01, '-')
    stimB.setPos([0.1, 0.1], '+')
    stimA.draw()
    stimB.draw()
    # win.flip()
    duration = clock.getTime()
    clock.reset()
    """
```

```
ppc.timer(script, setup='stimA, stimB, clock')
```



# Timing sound

- **Sound:** Use `winsound` on Windows and discard first play. PsychoPy is not precise enough yet.

```
beep = ppc.Sound('myBeep.wav')  
beep.play()
```

# Timing input



# Timing keyboard and mouse input

- `Psychopy.event` was the only keyboard/mouse input option until recently. It dominates the online docs and materials.
- While `psychopy.event` runs in the same python process as the stimulus presentation, `psychopy.iohub` runs in a parallel process. Iohub has the following advantages:
  - **Timing** input while the main python session is busy.  
`io_event.time` # relative to `core.monotonicClock.getTime()`
  - Also records key **releases** and **currently pressed** keys.  
`keyboard.state` # list of currently pressed keys  
`io_event.type` # compare to `iohub.EventConstants`.
  - Easily captures **characters** produced by combinations of keys.  
`keyboard.getEvents(char=['@', '!'])`
- OBS: to make iohub it work in Mac OS, drag-drop it into System Preferences → Security & Privacy → Privacy → Accessibility. See <http://kb.parallels.com/en/116418>

# event versus iohub

## iohub

```
# Set up
from psychopy import iohub
io = iohub.launchHubServer()
keyboard = io.devices.keyboard

# Keep main process busy
flip_times = []
io.clearEvents('all')
for frame in range(60):
    stim.draw()
    flip_times += [win.flip()]

# Get response and RT
response = keyboard.getPresses()

print response[0].time - flip_times[0]

# Or use one of these:
keyboard.getReleases()
response = keyboard.getKeys()
```

## event

```
# Set up
from psychopy import event

# Keep main process busy
flip_times = []
event.clearEvents()
for frame in range(60):
    stim.draw()
    flip_times += [win.flip()]

# Get response and RT
response =
    event.getKeys(timeStamped=True)
print response[0][0] - flip_times[0]
```

<http://www.psychopy.org/api/iohub/device/keyboard.html>



**General advice**

# General advice on precision

- A log is suggestive.  
Physical measurement is proof.
- Timing: run the code once before the critical presentation. Slow first-run functions include:

```
core.wait()  
ppc.sound.play()  
stim.draw()
```



You're the python god and python functions are your slaves.



(Monty Python's impression)

# You're the python god and python functions are your slaves.

- You should be in **total control**. They should do everything that you tell them to and nothing else. Complex modules are threads!
- Test them frequently, **don't trust them** until they've prooved that they are true to your wishes for all eternity!
- Good functions should be able to serve you properly. Give them **proper hardware**!