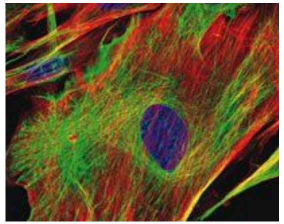
A DSL for Fluorescence Microscopy

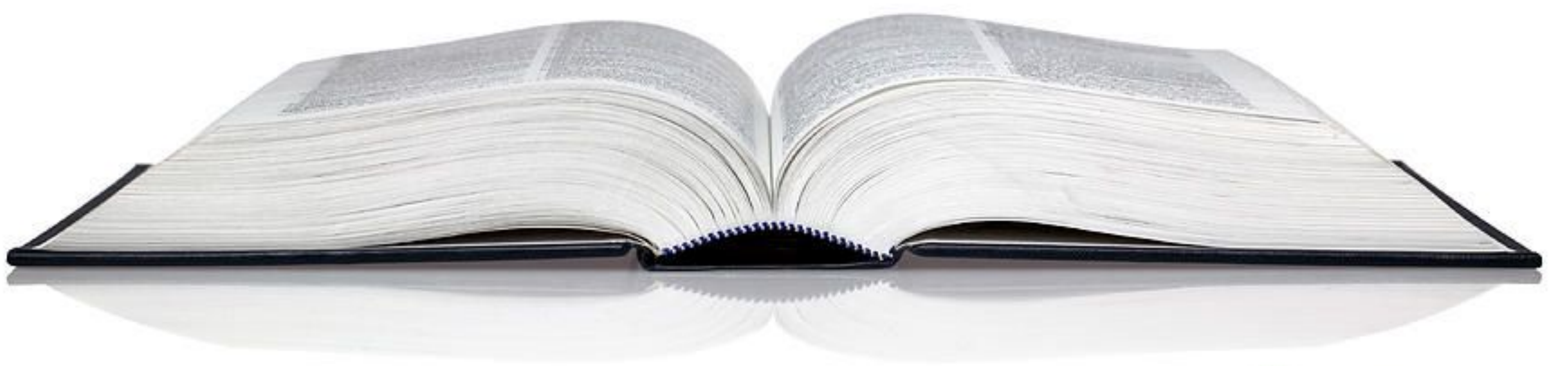
Birthe van den Berg
Tom Schrijvers
Peter Dedecker
KU LEUVEN

Introduction

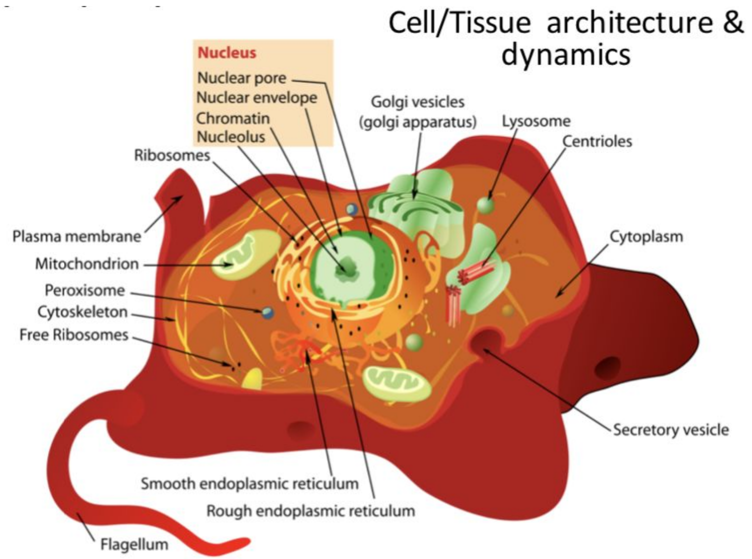
NanoBiology

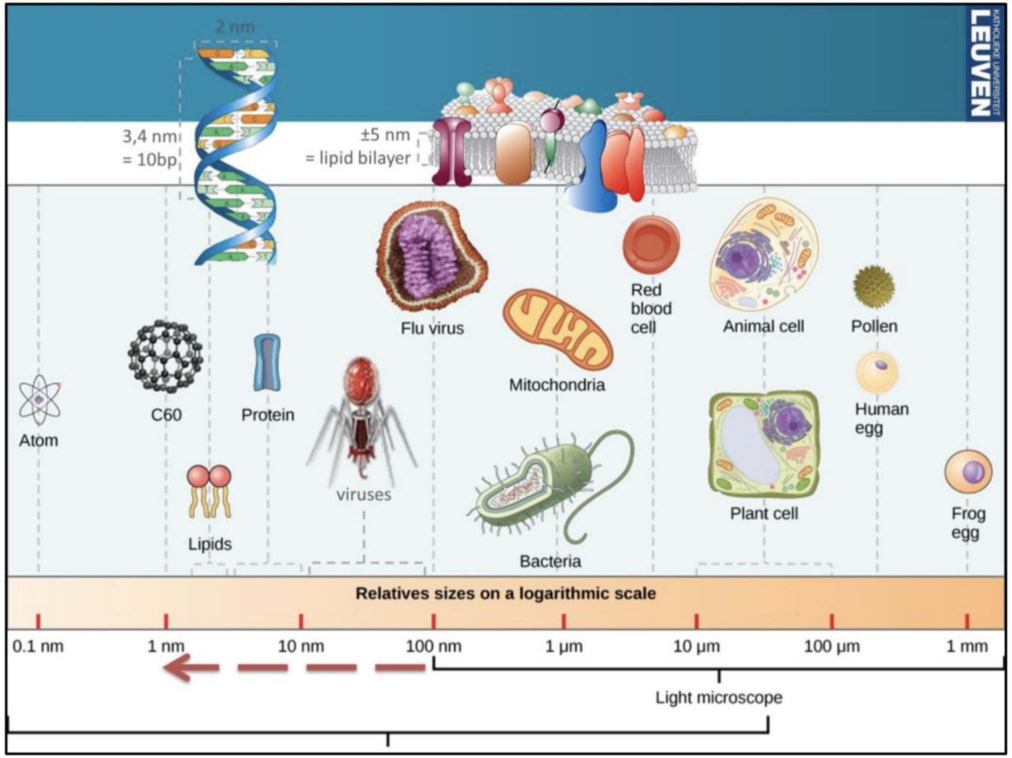


DSL



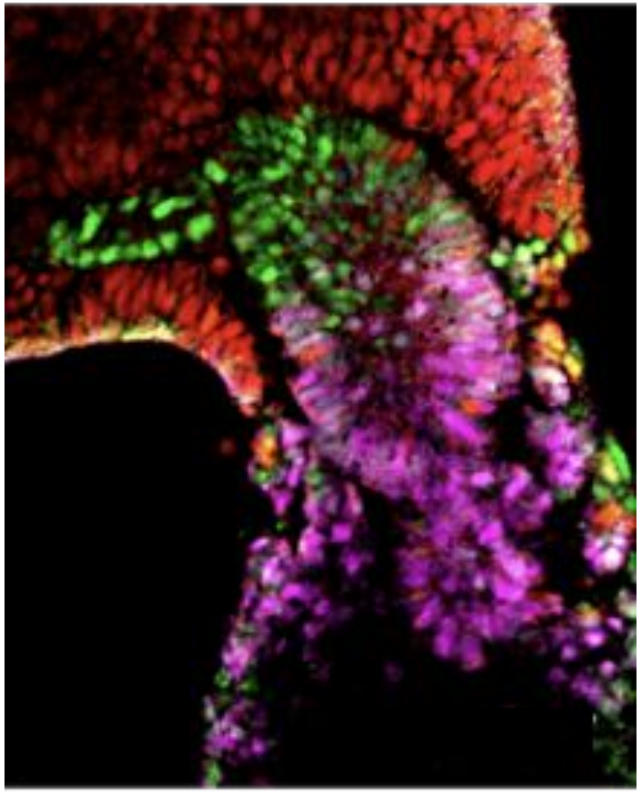
Why fluorescence?





Why fluorescence?

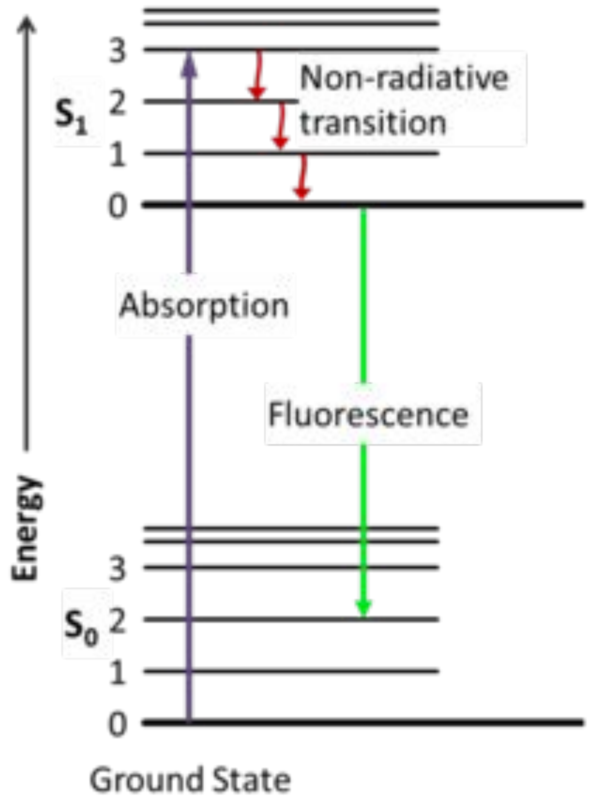
- **non-invasive**
  - in situ
  - in vivo
- **selective**
- sample preparation
  - **simple**
  - **wide range** of fluorescent probes



What is fluorescence?

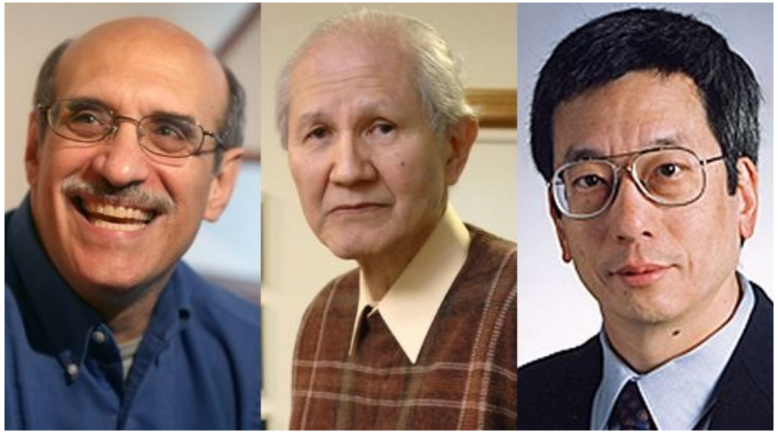
Fluorescence

Jablonski diagram

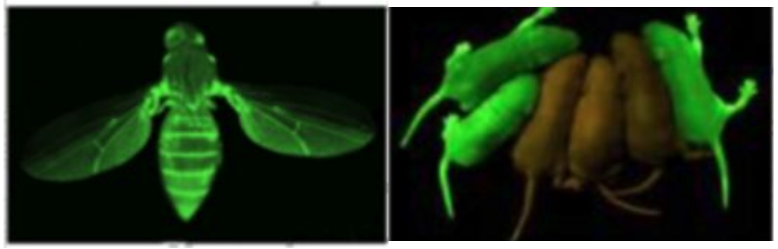


Fluorescence

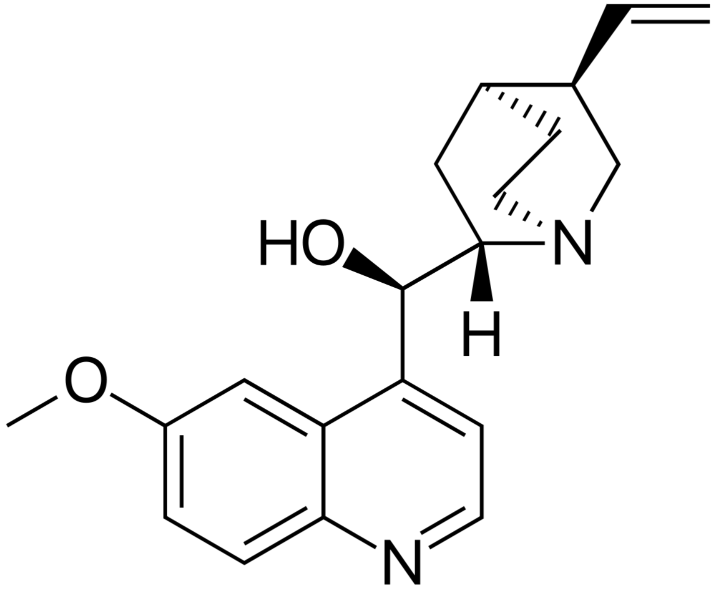
Green
Fluorescent
Protein



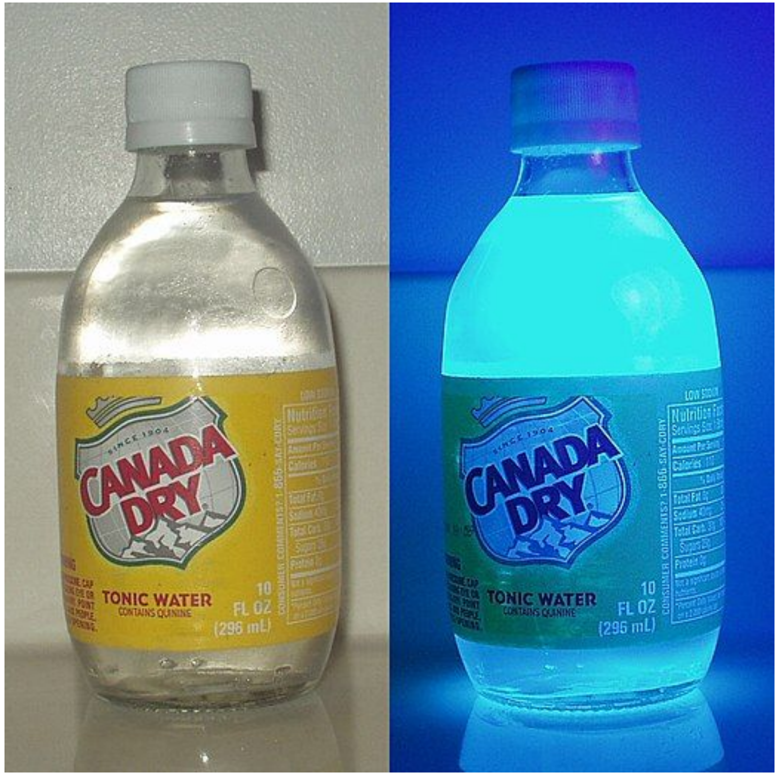
M.Chalfie - O.Shimomura - R. Tsien
Nobel Prize in Chemistry 2008



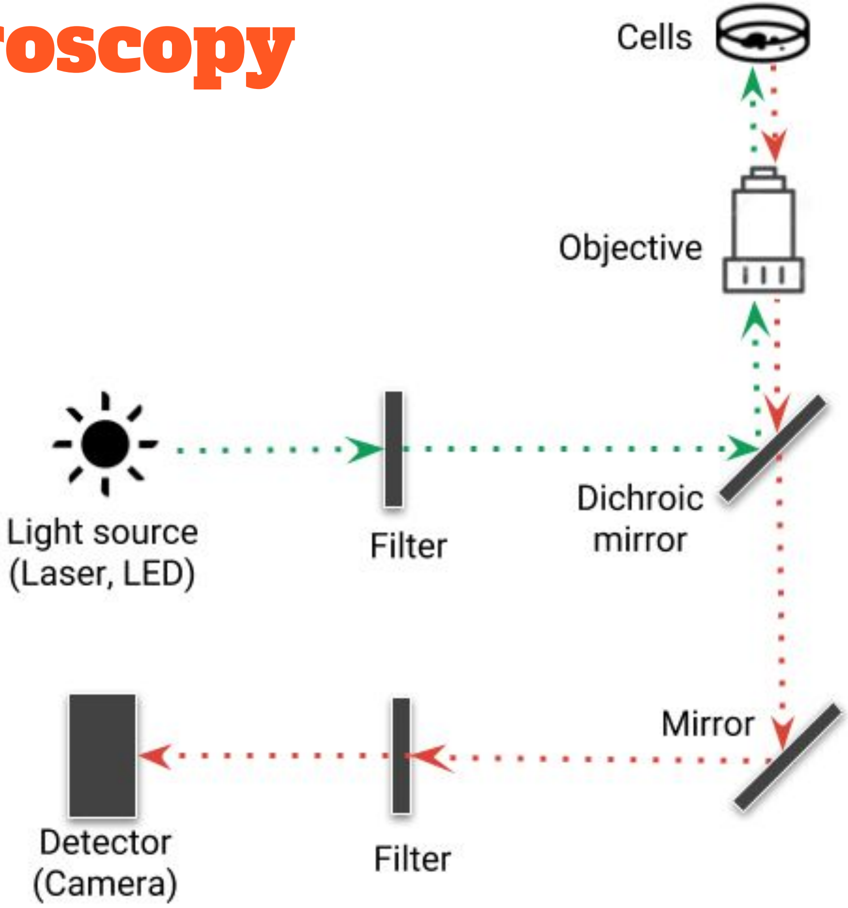
Fluorescence



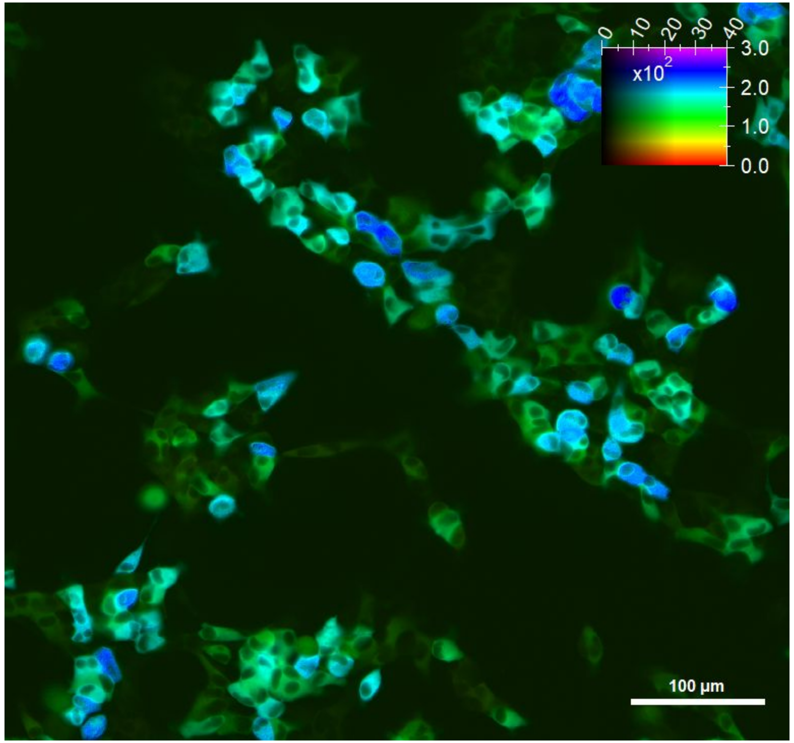
Quinine



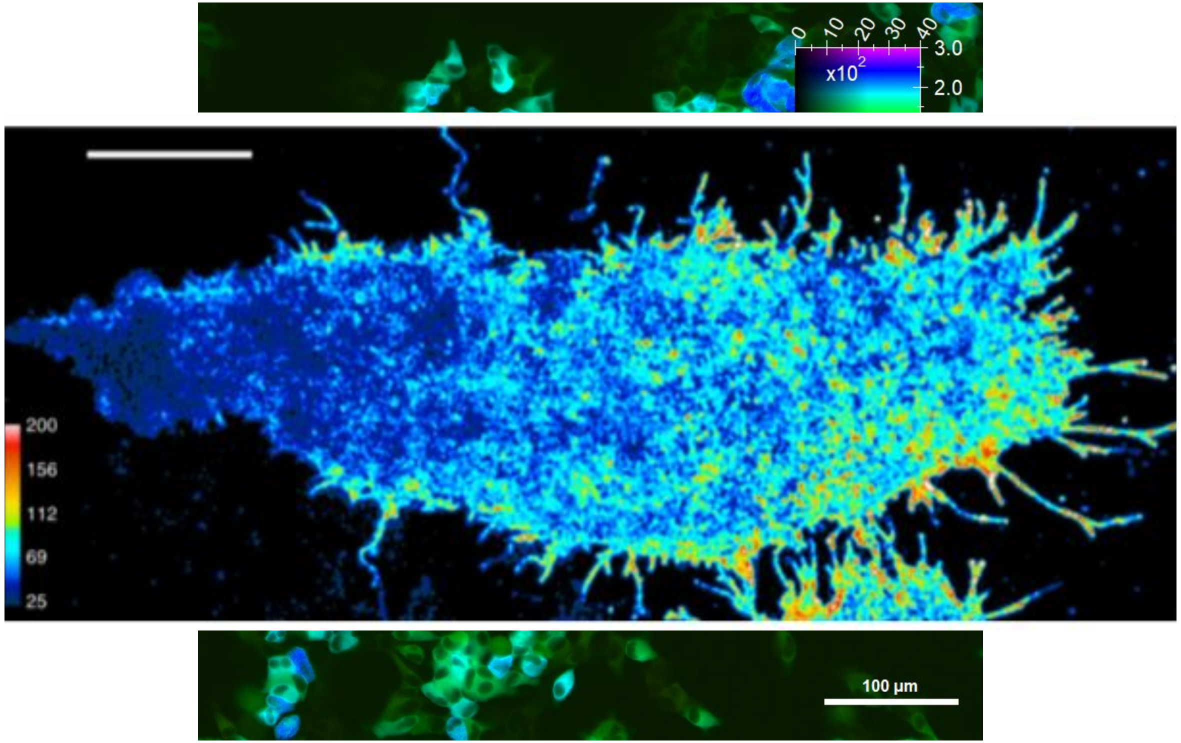
Fluorescence Microscopy



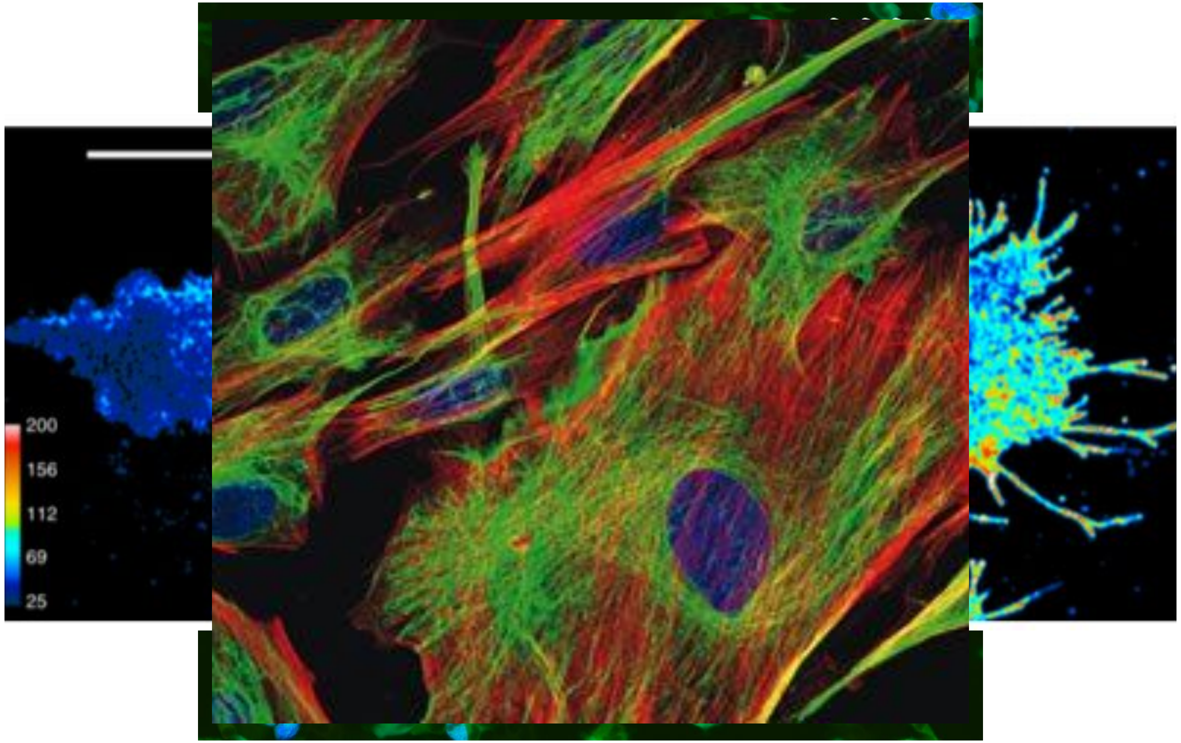
Fluorescence Microscopy



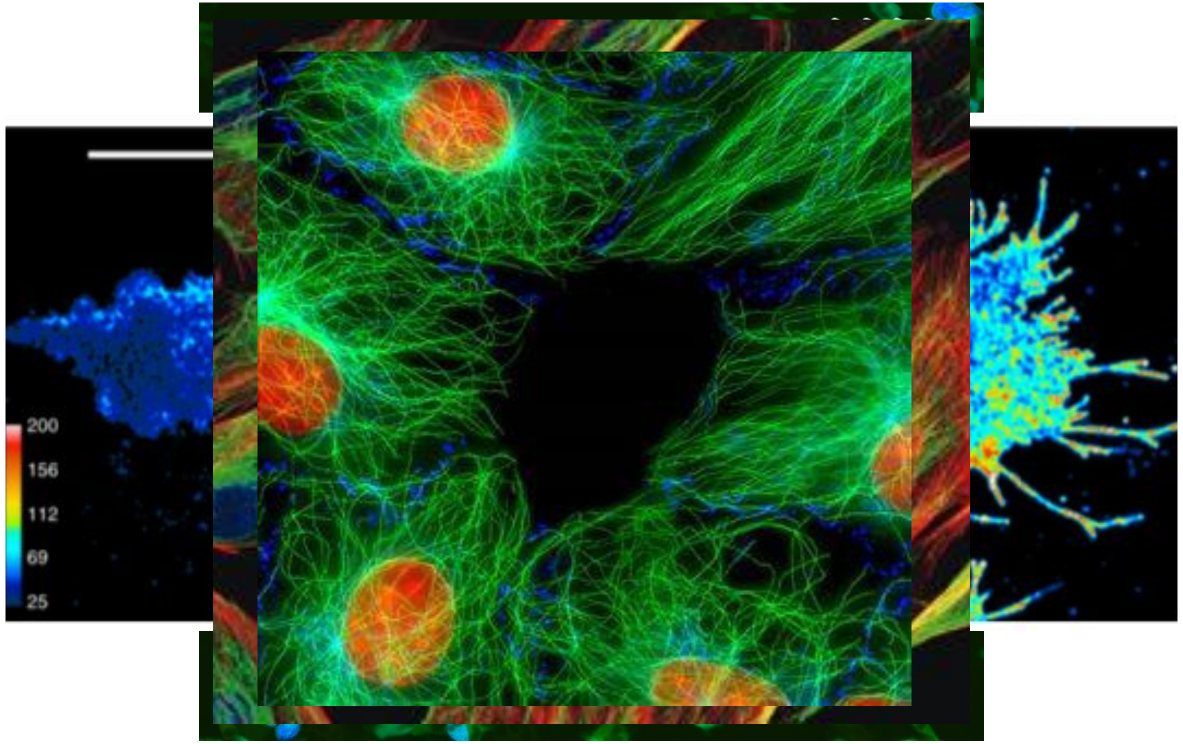
Fluorescence Microscopy



Fluorescence Microscopy



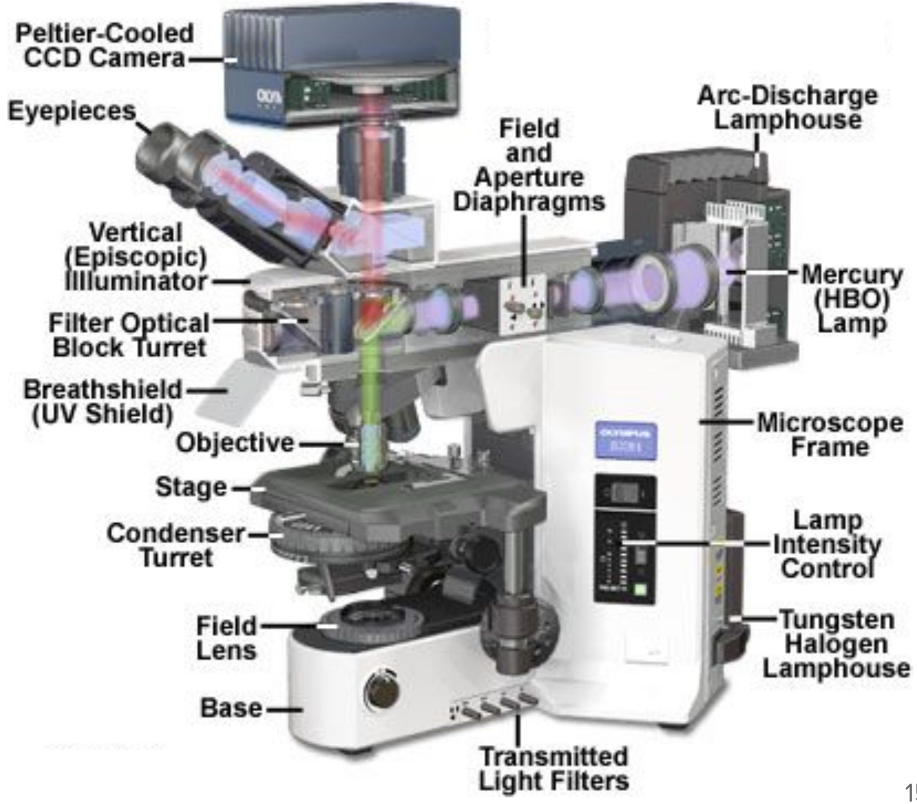
Fluorescence Microscopy



So, what's the problem?

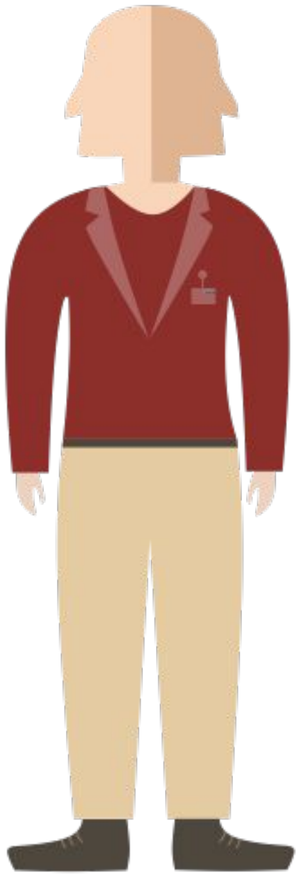
So, what's the problem?

- Growing **technical complexity**
- **Expensive** instrumentation


Peltier-Cooled CCD Camera
Eyepieces
Field and Aperture Diaphragms
Arc-Discharge Lamphouse
Vertical (Episcopic) Illluminator
Filter Optical Block Turret
Mercury (HBO) Lamp
Breathshield (UV Shield)
Objective
Microscope Frame
Stage
Condenser Turret
Lamp Intensity Control
Field Lens
Tungsten Halogen Lamphouse
Base
Transmitted Light Filters

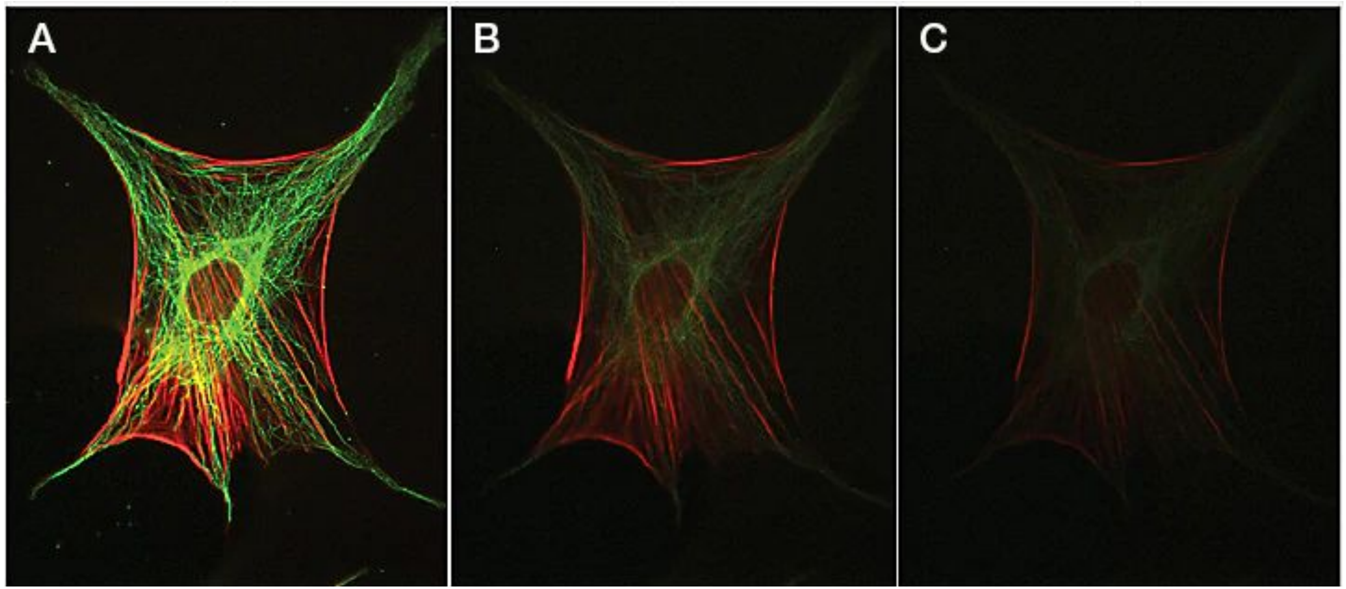
So, what's the problem?

- Growing **technical complexity**
- **Expensive** instrumentation
- **Dedicated staff members** (researchers + technicians)

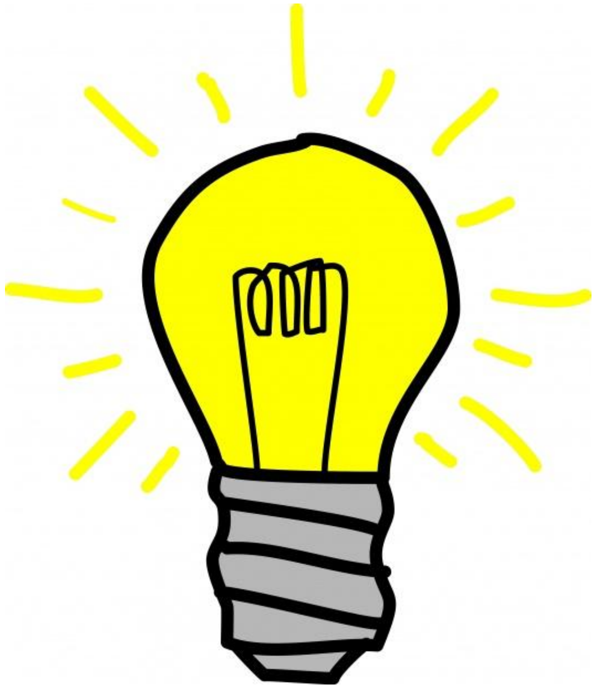


So, what's the problem?

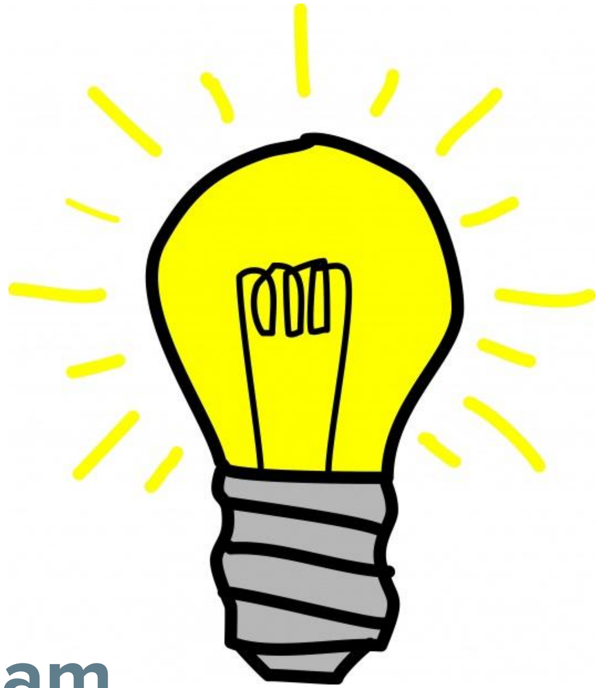
- Adapting the **experimental strategy**, setting **many parameters**
- No or little **safety** controls



So, what's the clue?



So, what's the clue?



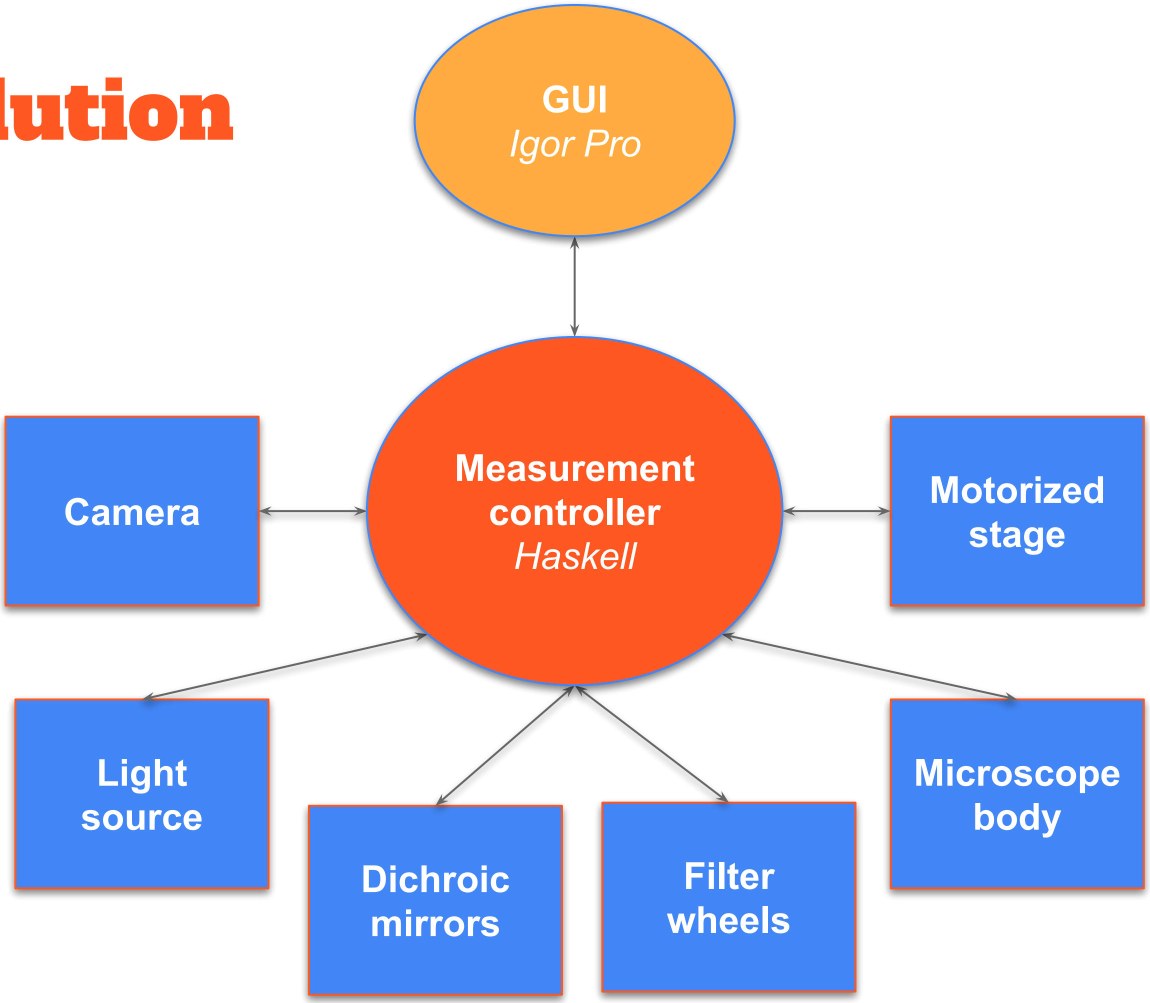
Setting up an experiment

=

Constructing a computer program

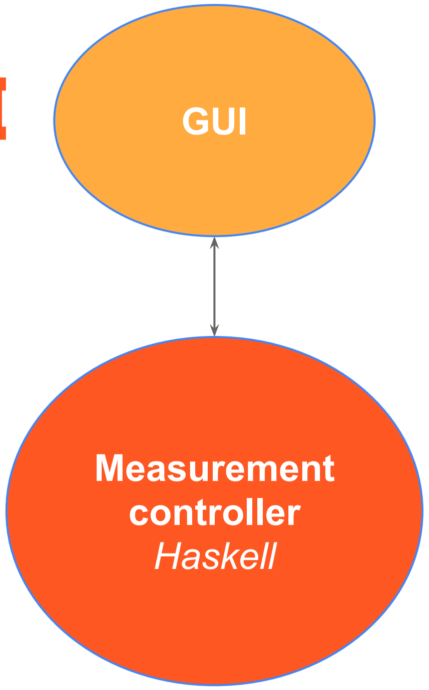
Our Solution

Our Solution



Solution: Controller vs GUI

- Communication with **GUI**

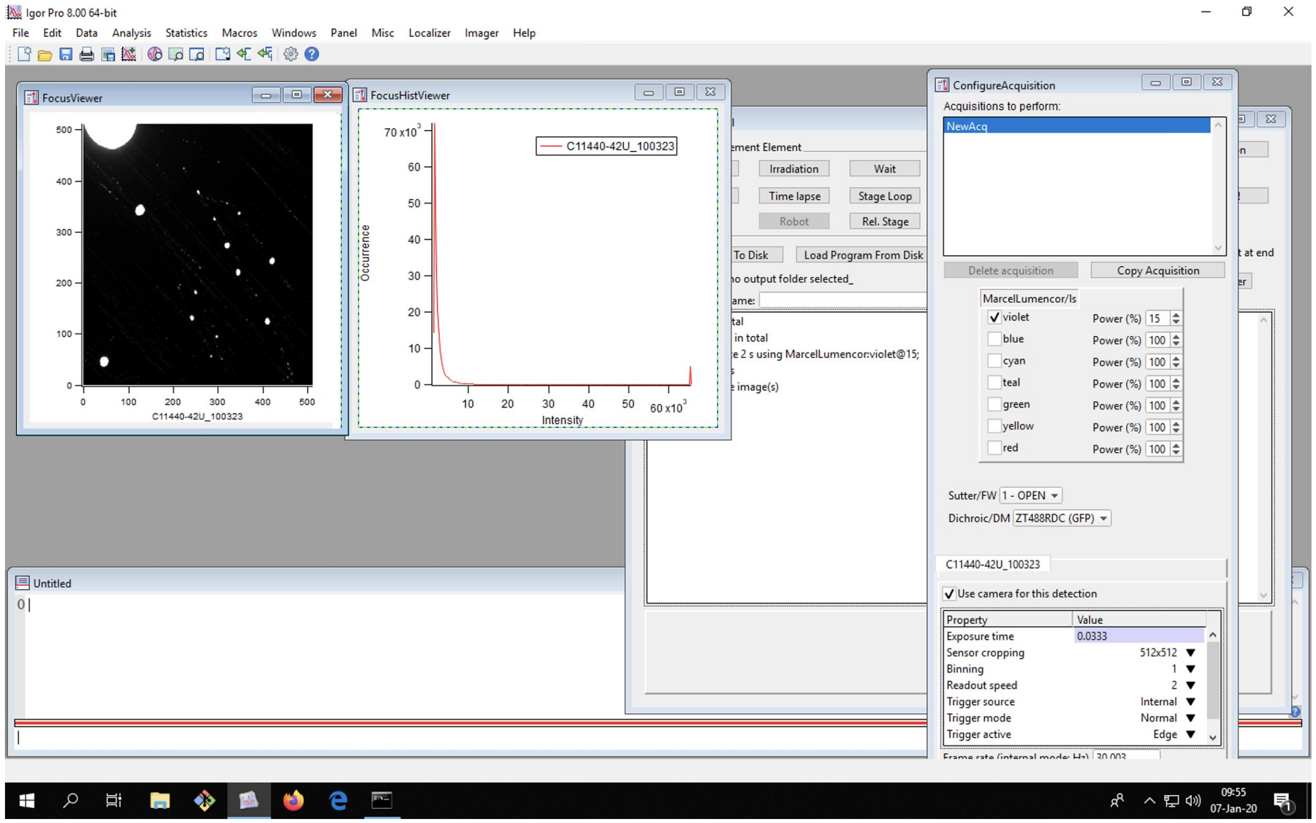


GUI

Igor Pro

Technical Computing for Scientists and Engineers

WaveMetrics, Inc.



o output folder selected_

ame:

tal
in total
e 2 s using MarcelLumencor:violet@15;
image(s)

Acquisitions to perform:
NewAcq

MarcelLumencor/ls

✓ violet	Power (%)	15
blue	Power (%)	100
cyan	Power (%)	100
teal	Power (%)	100
green	Power (%)	100
yellow	Power (%)	100
red	Power (%)	100

Sutter/FW 1 - OPEN

Dichroic/DM ZT488RDC (GFP)

C11440-42U_100323

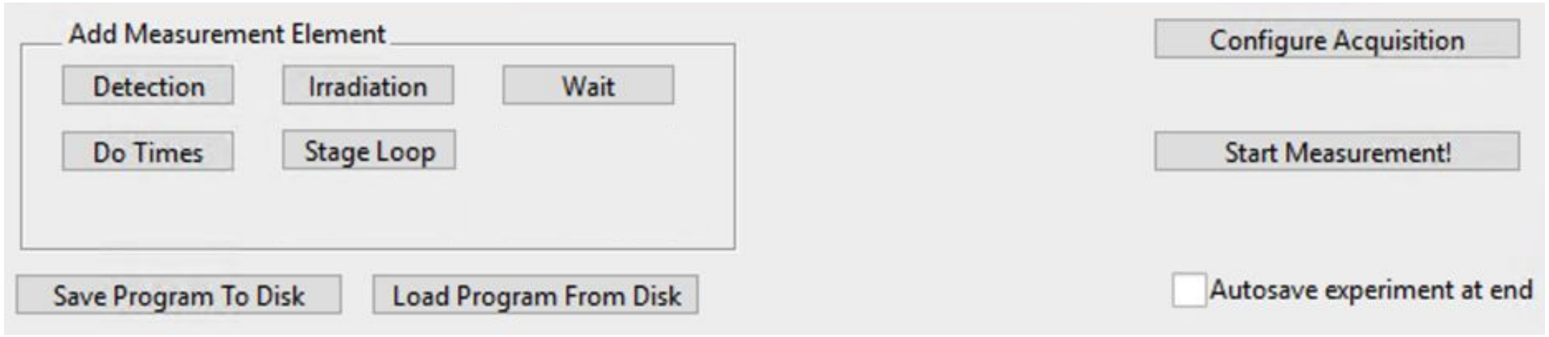
✓ Use camera for this detection

Property	Value
Exposure time	0.0333
Sensor cropping	512x512
Binning	1
Readout speed	2
Trigger source	Internal
Trigger mode	Normal
Trigger active	Edge

Frame rate (internal mode; Hz) 30.003

09:55 07-Jan-20

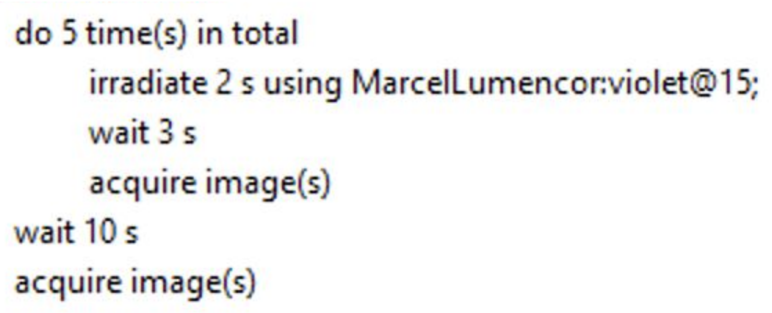
GUI: Example



Igor Pro

Technical Computing for Scientists and Engineers

WaveMetrics, Inc.



```
do 5 time(s) in total
    irradiate 2 s using MarcelLumencor:violet@15;
    wait 3 s
    acquire image(s)
wait 10 s
acquire image(s)
```

Solution: Controller DSL

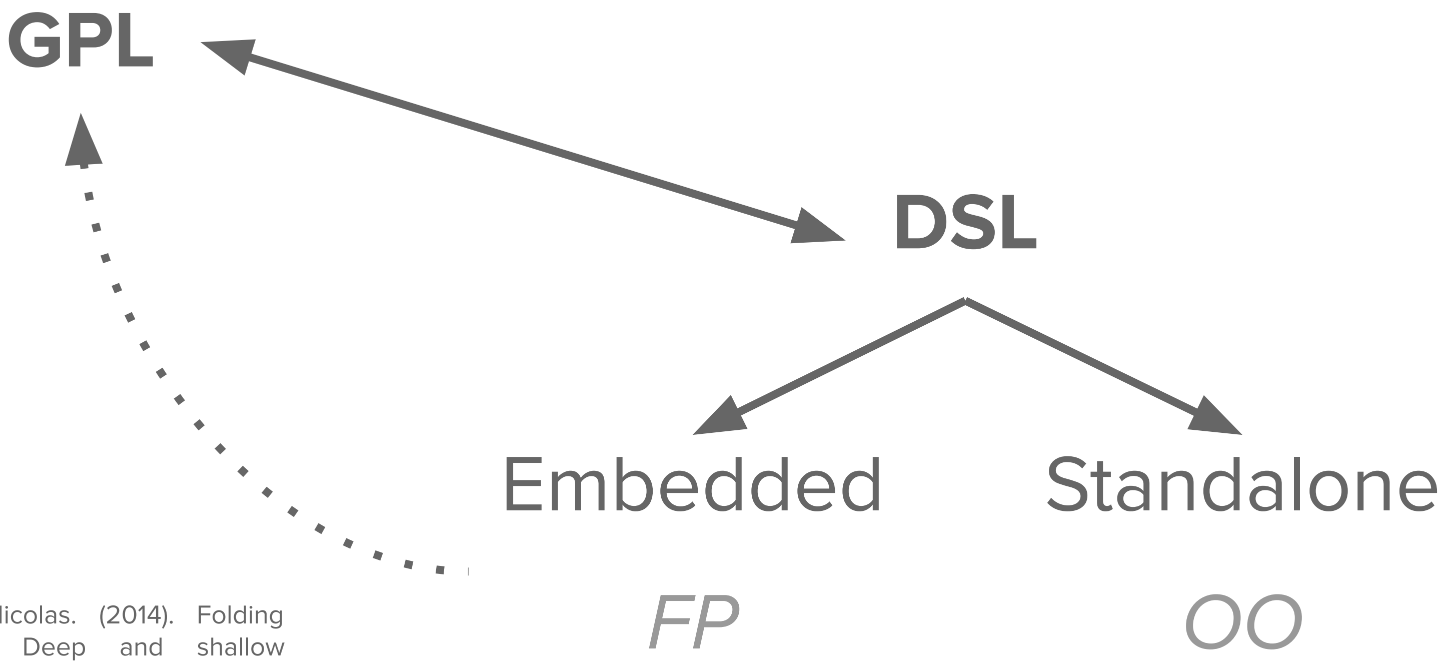
- Representing and reasoning over **domain-specific knowledge**

DSL

Haskell

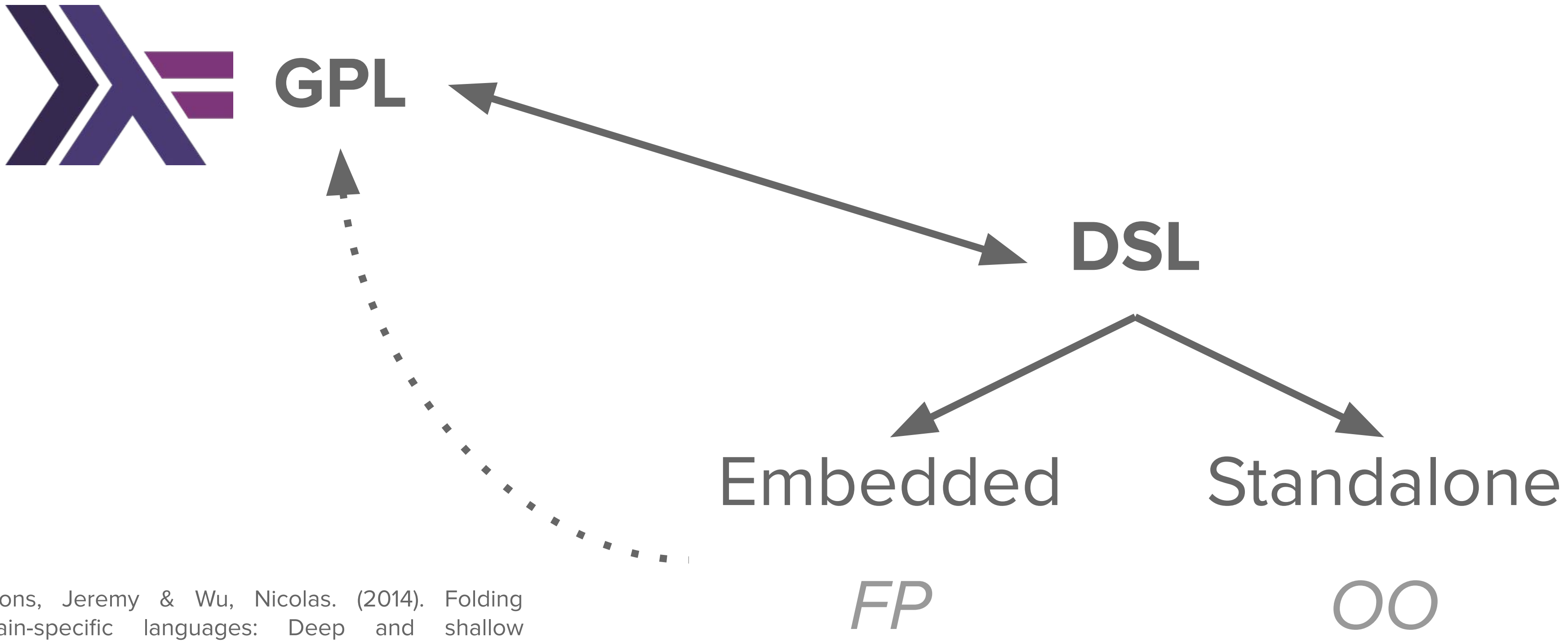
Measurement controller

Why DSL? Why Haskell?



Gibbons, Jeremy & Wu, Nicolas. (2014). Folding domain-specific languages: Deep and shallow embeddings (functional Pearl). Proceedings of the ACM SIGPLAN International Conference on Functional Programming, ICFP. 49. 10.1145/2628136.2628138.

Why DSL? Why Haskell?



Gibbons, Jeremy & Wu, Nicolas. (2014). Folding domain-specific languages: Deep and shallow embeddings (functional Pearl). Proceedings of the ACM SIGPLAN International Conference on Functional Programming, ICFP. 49. 10.1145/2628136.2628138.

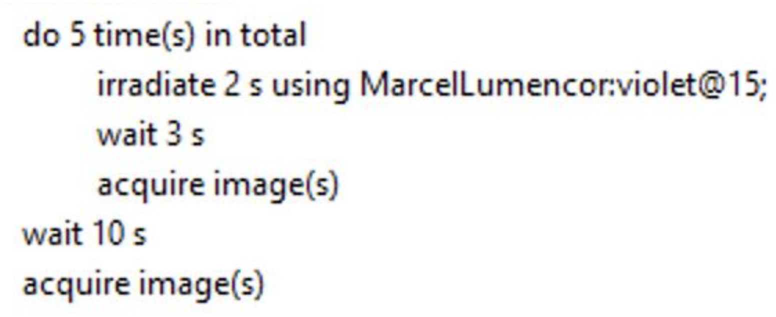
DSL Fragment

```
data StagePosition          =  StagePosition {  x :: Double
                                               , y :: Double
                                               , z :: Double  }

data MeasurementElement  = MEDetect
                          | MEWait  Double
                          | MEIrradiate  Double ( String,  Double )
                                    -- duration ( light source , power )
                          | MEDoTimes  Int  Prog
                          | MEStageLoop  [ StagePosition ]  Prog
                          | ...

type Prog                   =  [ MeasurementElement ]
```

DSL: Example



```
do 5 time(s) in total
        irradiate 2 s using MarcelLumencor:violet@15;
        wait 3 s
        acquire image(s)
wait 10 s
acquire image(s)
```

```
[ MEDoTimes 5 [ MEIrradiate 2 ( MarcelLumencor:violet , 15 )
              , MEWait 3
              , MEDetect ]
, MEWait 10
, MEDetect ]
```

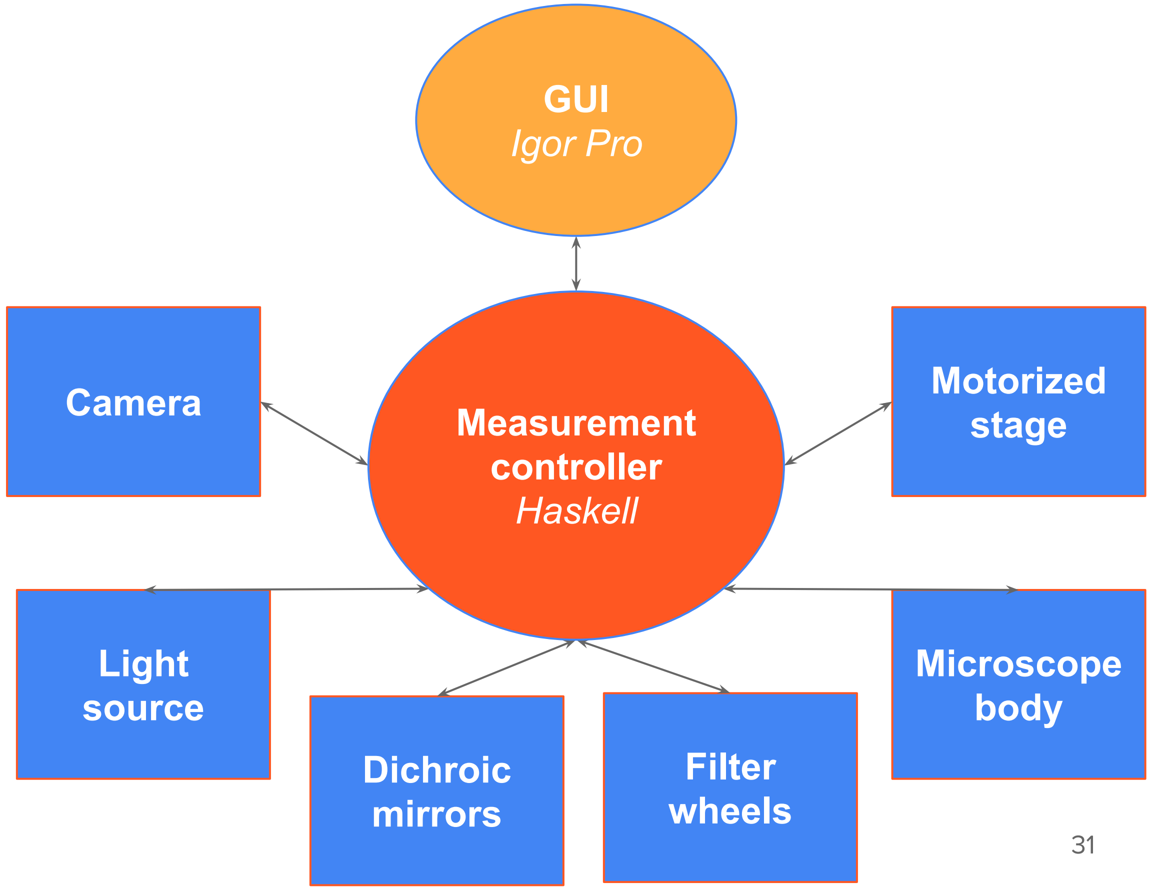
Deep embedding

```
executeProg :: Prog → IO ()
executeProg prog = foldMap executeME prog

executeME :: MeasurementElement → IO ()
executeME  MEDetect                   = executeDetection >> putStrLn (“detecting…”)
executeME (MEWait dur)                = threadDelay (round $ dur * 1e6)
                                      >> putStrLn (“waiting…”)
executeME (MEIrradiate dur params) = executeIrradiation dur params
                                      >> putStrLn (“irradiating…”)
executeME (MEDoTimes n pr)            =
          mapM_ (\prs → executeProg prs) (take n . repeat $ pr)
                                       >> putStrLn (“times…”)
executeME (MEStageLoop poss pr)     =
          mapM_ (\pos → setStagePosition pos >> executeProg pr) poss
                                       >> putStrLn (“stage looping…”)
```

Solution: Controller vs Hardware

- Controlling **hardware**



Hardware

Camera

Light source

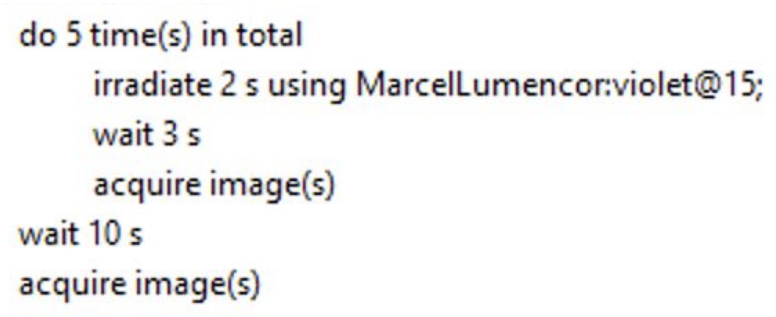
Dichroic mirrors

Motorized stage

Filter wheels

Microscope body

Hardware: Example

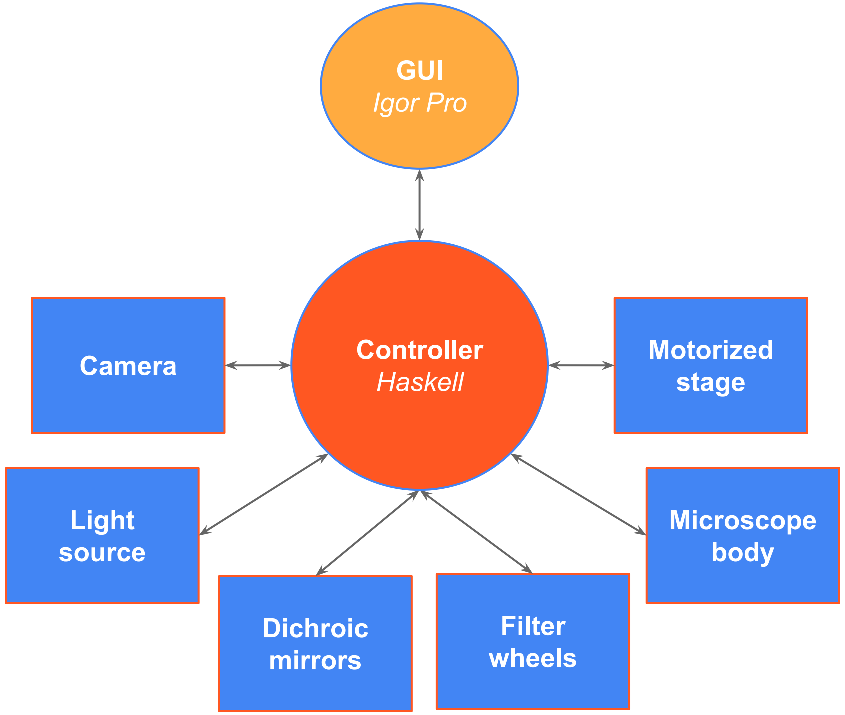


```
do 5 time(s) in total
        irradiate 2 s using MarcelLumencor:violet@15;
        wait 3 s
        acquire image(s)
wait 10 s
acquire image(s)
```

```
[ MEDoTimes 5 [ MEIrradiate 2
                    ( MarcelLumencor:violet , 15 )
              , MEWait 3
              , MEDetect ]
, MEWait 10
, MEDetect ]
```

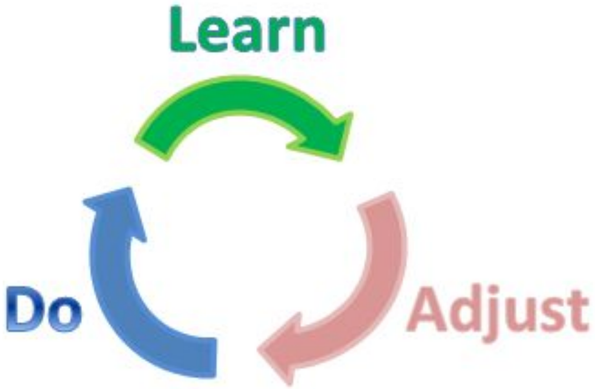
Results

Results



usable software

feedback loop

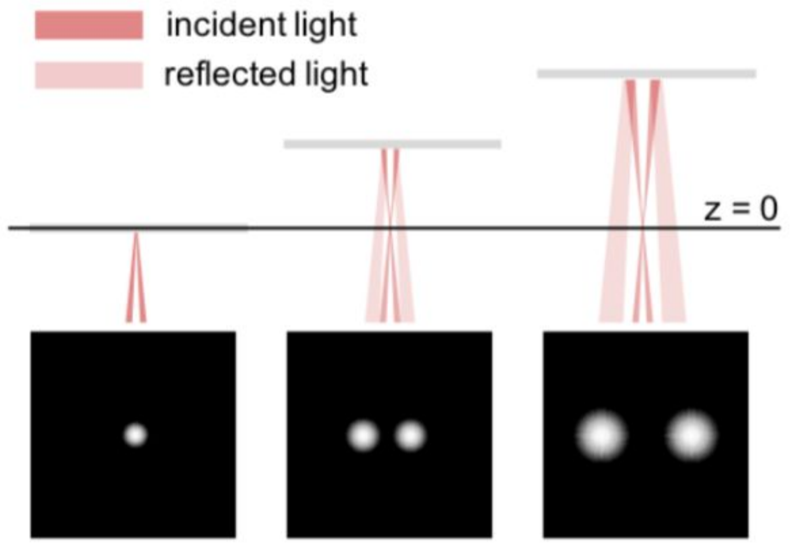


Compared to already existing SW

- **Modularity**
  - easily extendible for new hardware
  - works with several hardware setups
- Arbitrary long, **complex** programs

Compared to already existing SW

- **Modularity**
  - easily extendible for new hardware
  - works with several hardware setups
- Arbitrary long, **complex** programs
- Also non-trivial, more complex tasks
  - E.g. **autofocus** system



Future Work

- GUI
- DSL for knowledge representation
- Operational DSL for controlling HW/SW
- Fluorescence Microscopy Applications

Future Work

- GUI ✓
- DSL for knowledge representation
- Operational DSL for controlling HW/SW
- Fluorescence Microscopy Applications

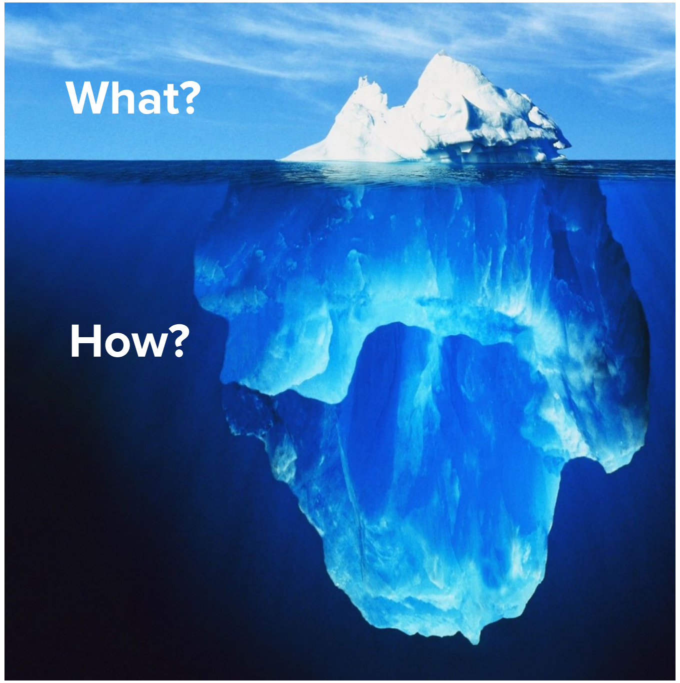
Future Work

GUI

DSL for knowledge representation

Operational DSL for controlling HW/SW

Fluorescence Microscopy Applications



Future Work

GUI

DSL for knowledge representation

Operational DSL for controlling HW/SW ✓

Fluorescence Microscopy Applications

- Increase **performance**
  - optimal rescheduling
  - parallelizing
- **Safety** and **sanity checks**

Future Work

GUI

DSL for knowledge representation

Operational DSL for controlling HW/SW

Fluorescence Microscopy Applications

Future Work

“A system that can learn from scientists and operators, and vice versa.”

Thanks!