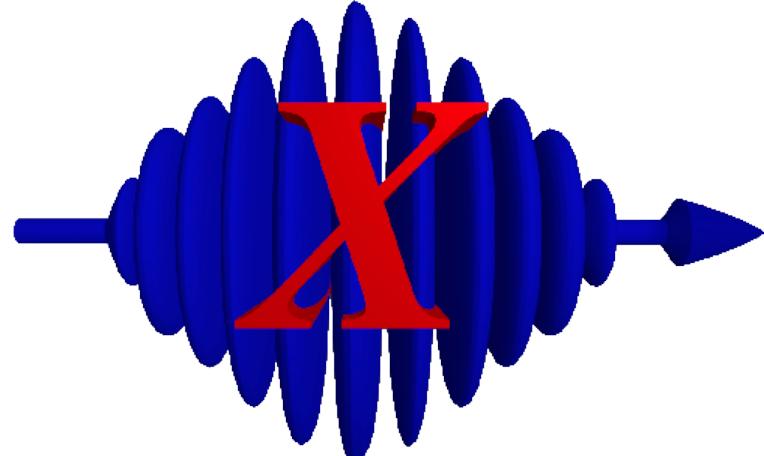
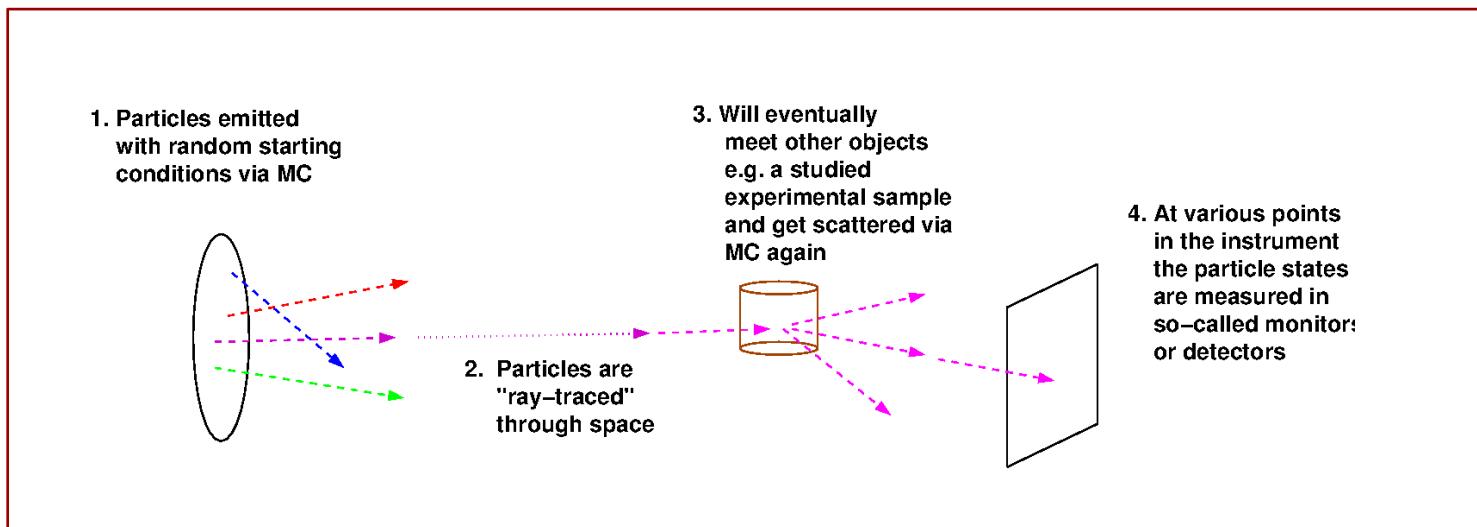


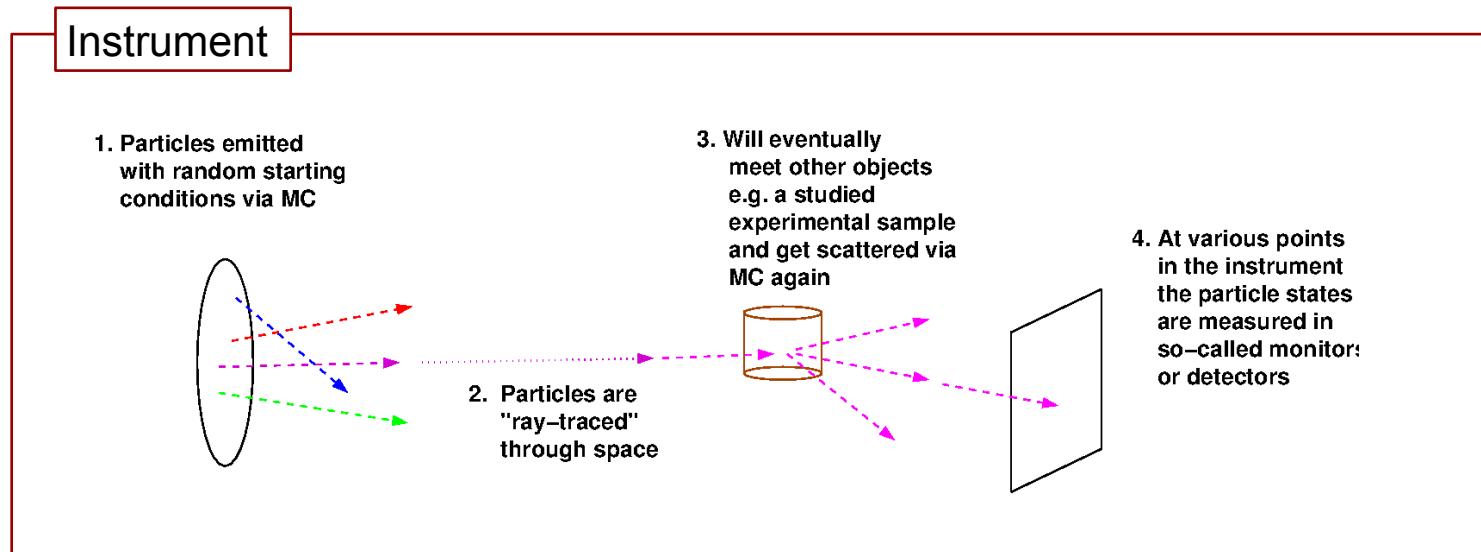
# McXtrace



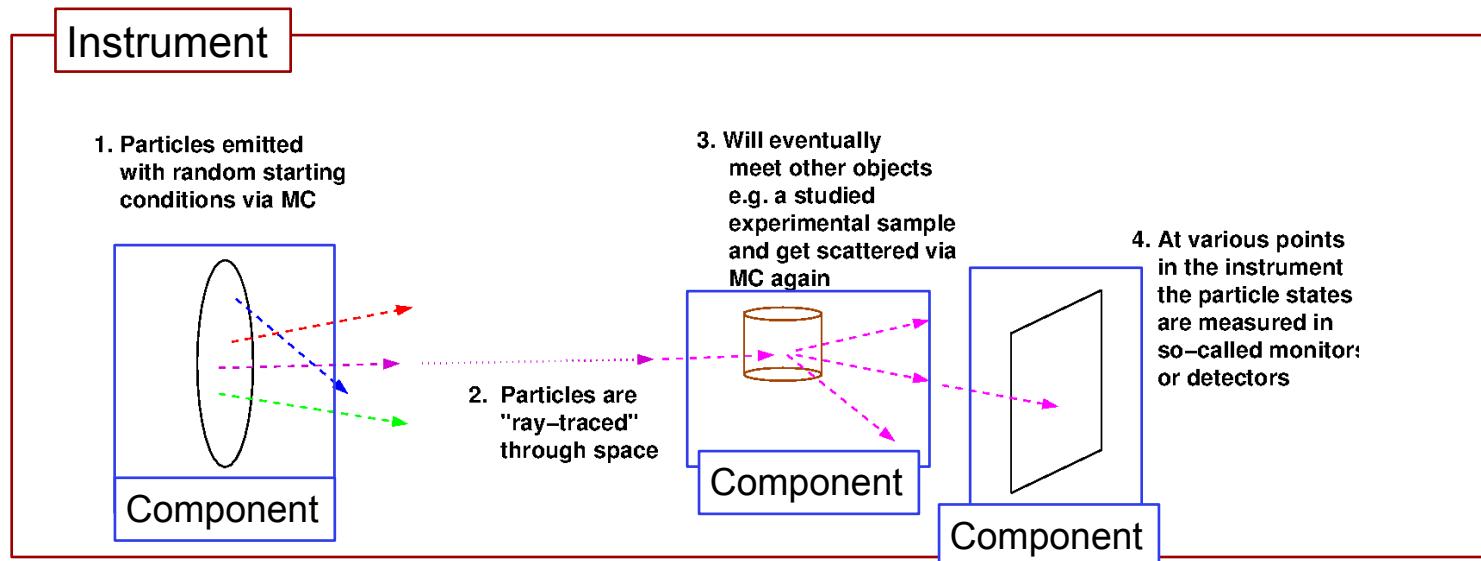
McXtrace components and instruments

Peter Willendrup ([pkwi@fysik.dtu.dk](mailto:pkwi@fysik.dtu.dk))



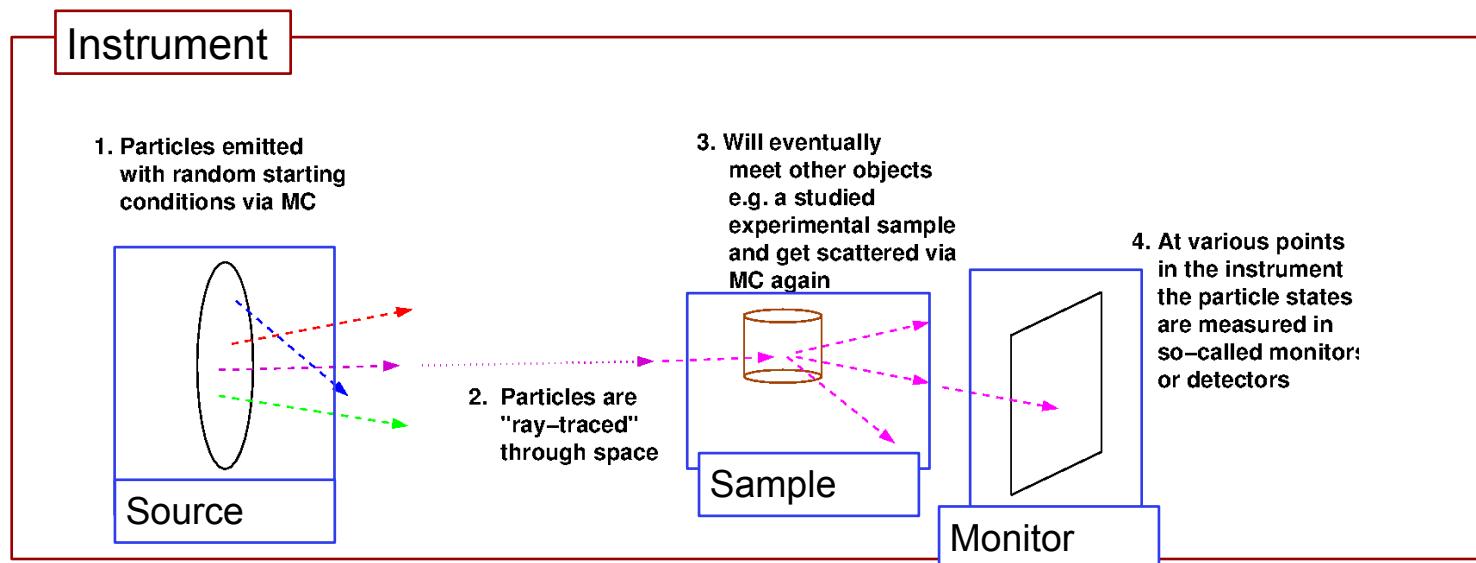


The instrument defines our “lab coordinate system”



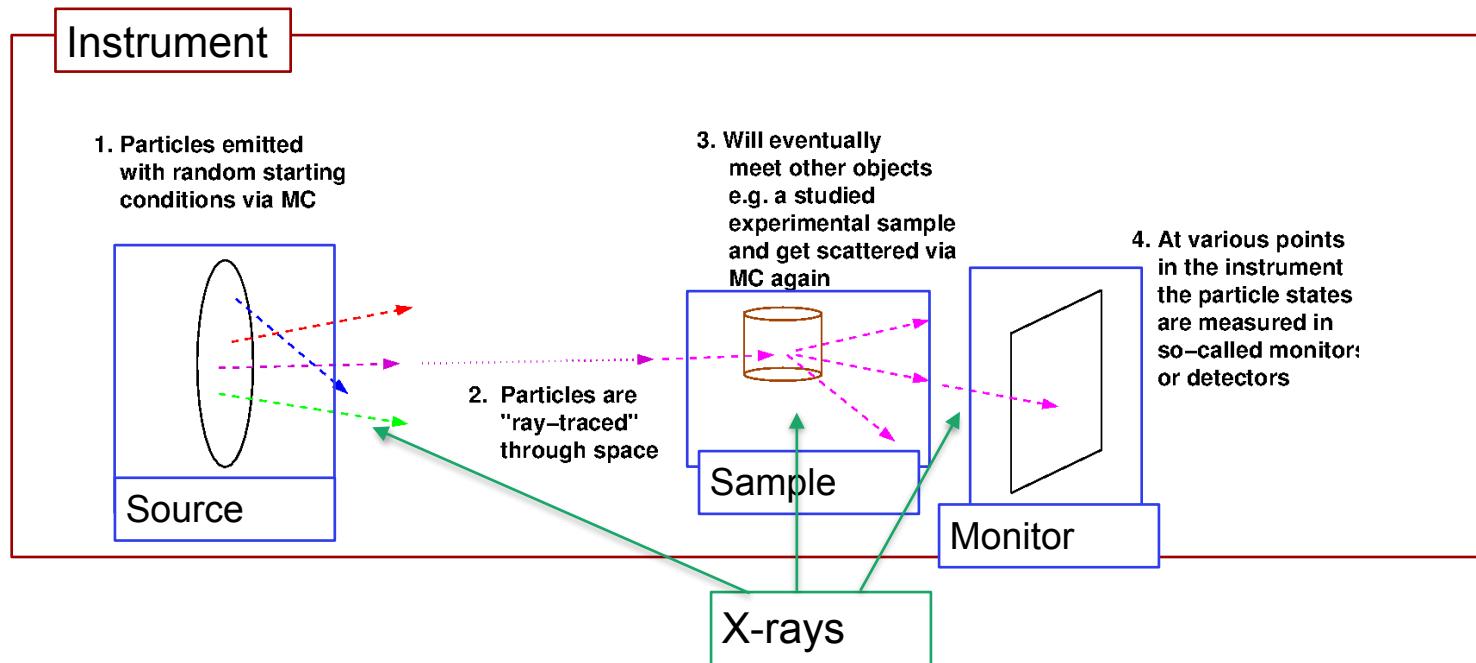
The instrument defines our “lab coordinate system”

The components define devices or features available in our instrument



The instrument defines our “lab coordinate system”

The components define devices or features available in our instrument  
- they have different function

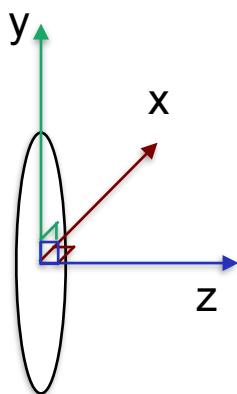


The instrument defines our “lab coordinate system”

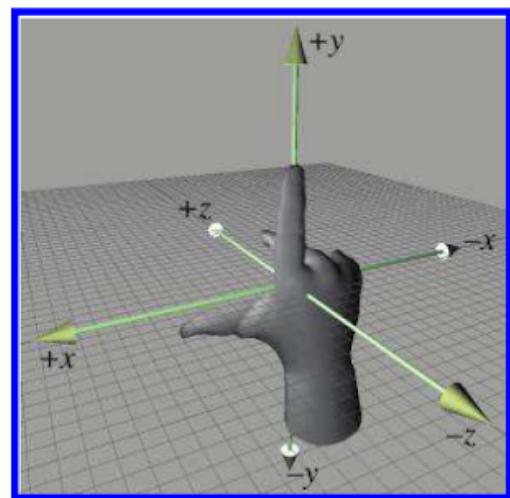
The components define devices or features available in our instrument  
- they have different function

X-ray photons particles are passed on from one component to the next,  
changing state under way

- One of the first components in your instrument is typically a source, which has a coordinate system like this....

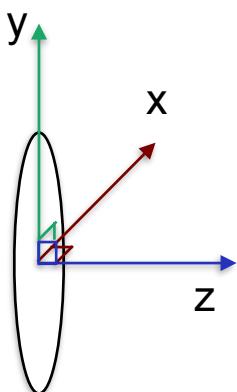


- z is along X-ray beam direction
- y is vertical
- x at an angle of 90° wrt. z,y



Right-handed  
coordinate system

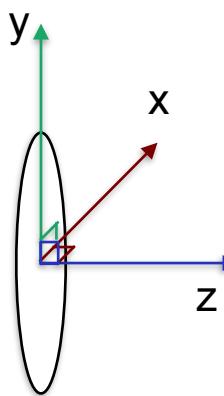
- Often the source coordinate system coincides with the “lab” coordinate system, denoted ABSOLUTE in McXtrace language, i.e.



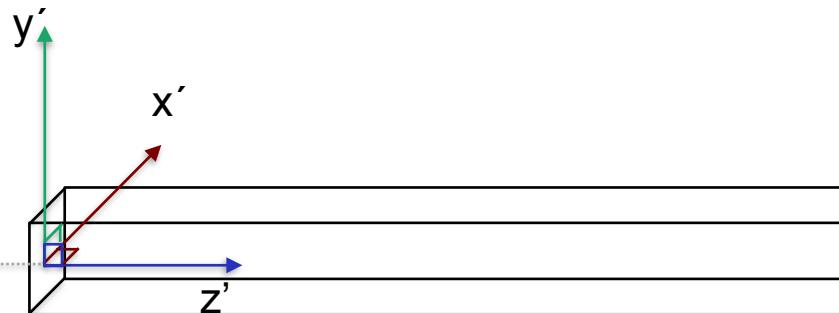
- COMPONENT Source = Source\_flat(...)  
AT (0,0,0) ABSOLUTE

Placing further components is done by order of

**1. Location, i.e**



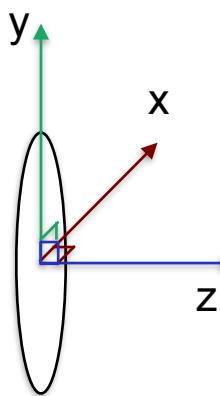
COMPONENT Source = Source\_flat(...)  
AT (0,0,0) ABSOLUTE



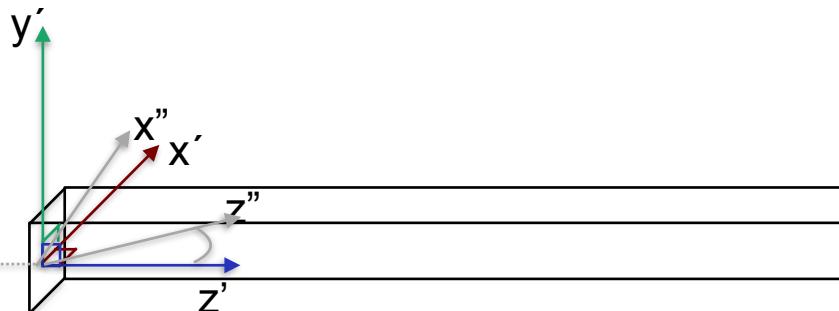
COMPONENT Capillary = Capillary(...)  
**AT (0,0,1) RELATIVE Source**

Placing further components is done by order of

## 2. Rotation, i.e



COMPONENT Source = Source\_flat(...)  
AT (0,0,0) ABSOLUTE



COMPONENT Capillary = Capillary(...)  
AT (0,0,1) RELATIVE Source  
**ROTATED (0,0,1,0) RELATIVE Source**

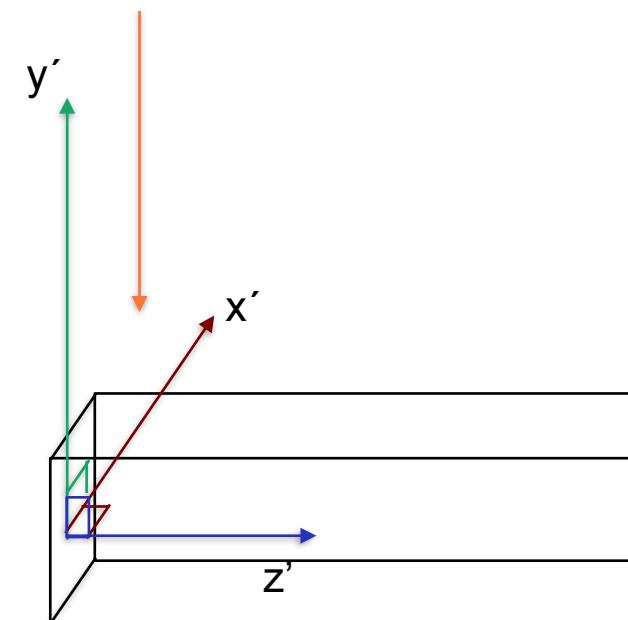
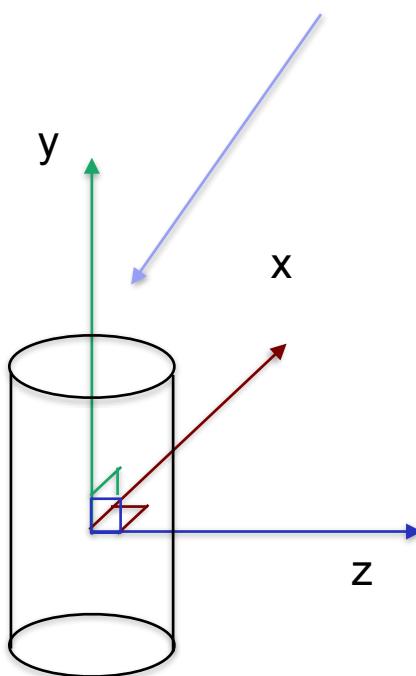
(Reference labels can also be PREVIOUS or PREVIOUS+1 etc.)

Note : AT and ROTATED can refer to different reference points  
and thus decoupled.

Components often have their origin at the centre of mass,  
i.e. for samples ... but not for e.g. Capillary

Placing further components is done by order of

## 2. Rotation, i.e.

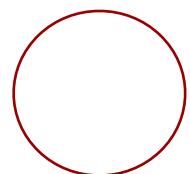


COMPONENT Sample = Some\_sample(...)  
AT (0,0,0) [RELATIVE] ABSOLUTE

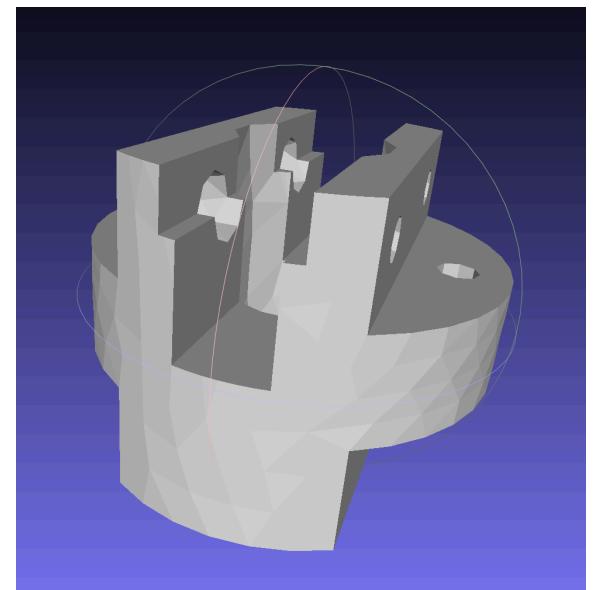
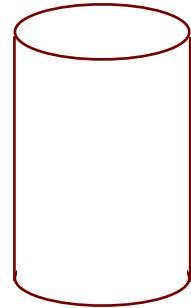
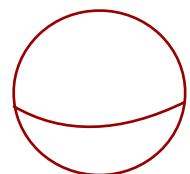
Generally speaking, the component author can choose  
**the meaningful coordinate system for the given problem!**  
- The McXtrace system takes care of the transformation between  
them....

Component geometries are typically simple objects... But  
some have polygon-description of the surface

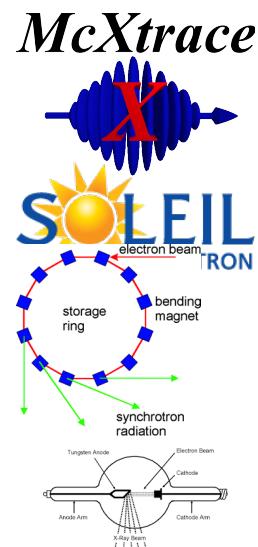
2D



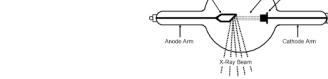
3D



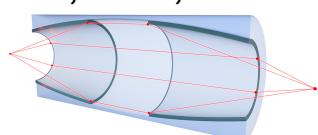
# Component classes



- Sources - these define MC starting conditions / “inject” rays to our simulation



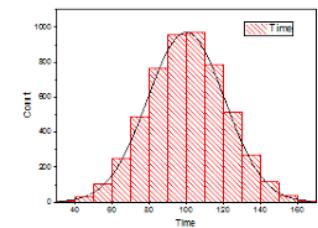
- Optics - used to tailor properties of the X-ray beam
  - Examples are mirrors, capillaries, lenses, zone-plates, choppers, collimators, slits, ...



- Samples - “matter” of some form
  - Powders, single crystals, liquids, micelles in solution, reflecting surfaces...



- Monitors - may probe the state of the X-ray beam and store histograms / event lists

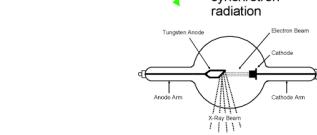


- Misc, obsolete
  - “Other stuff” and “Old stuff”

*Other  
Stuff*

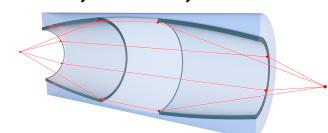


- Sources - these define MC starting conditions / “inject” X-rays to our simulation



- Optics - used to tailor properties of the X-ray beam

- Examples are mirrors, capillaries, lenses, zone-plates, choppers, collimators, slits, ...

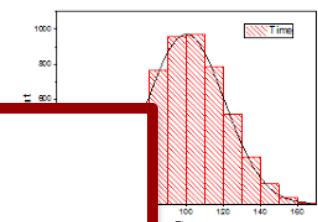


- Samples - “matter” of some form

- Powders, single crystals, liquids, micelles in solution, reflecting surfaces...



- Monitors - may probe the state of the X-ray beam and store histograms / event lists



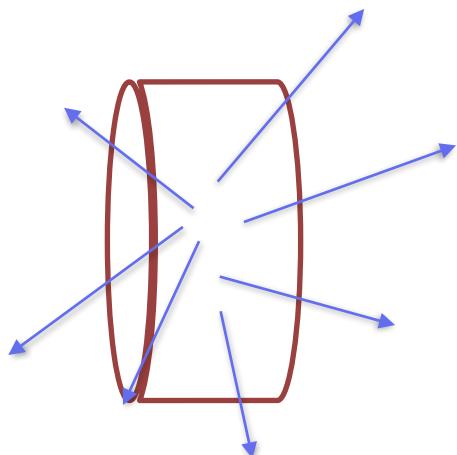
- Misc, obsolete

- “Other”

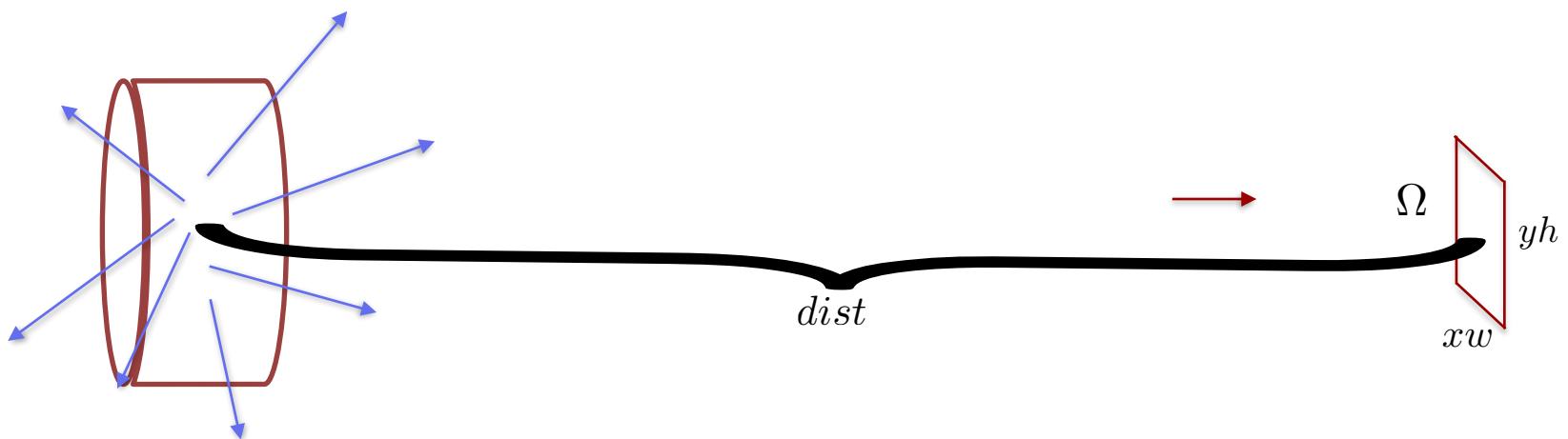
**Common to all components:**

 They set, manipulate/interact with  
or measure the **state of the X-ray (ray)**

- To first order emit uniformly into  $4\pi$  steradian

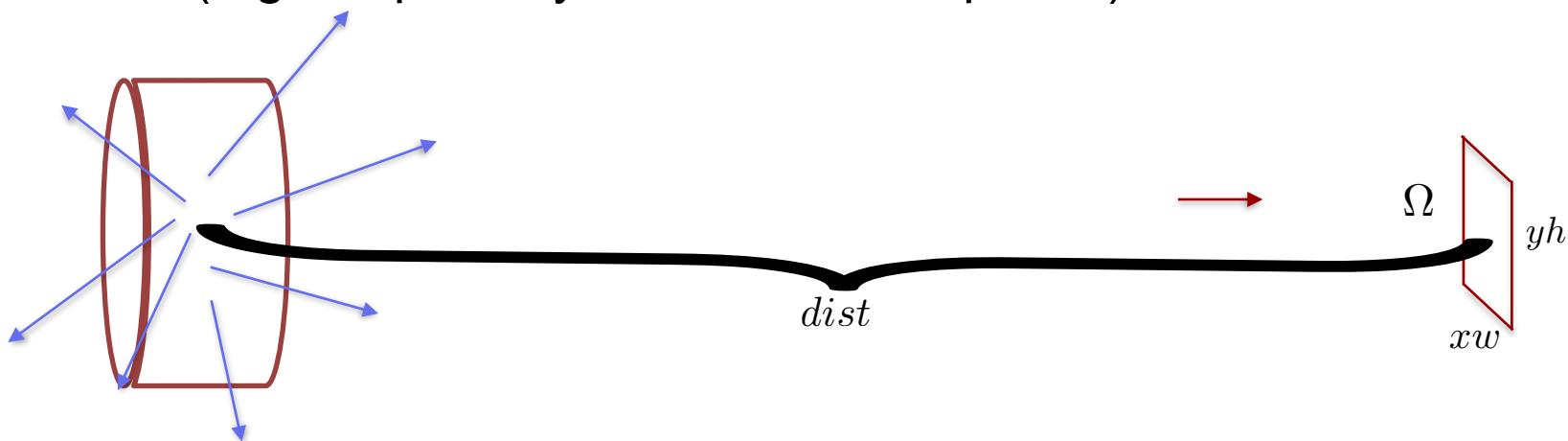


- To first order emit uniformly into  $4\pi$  steradian



- Generally we are interested in sending the input to an aperture, characterised by a certain solid angle  $\Omega$ , often corresponding to a rectangle  $xw \times yh$  at a distance  $dist$  from the source

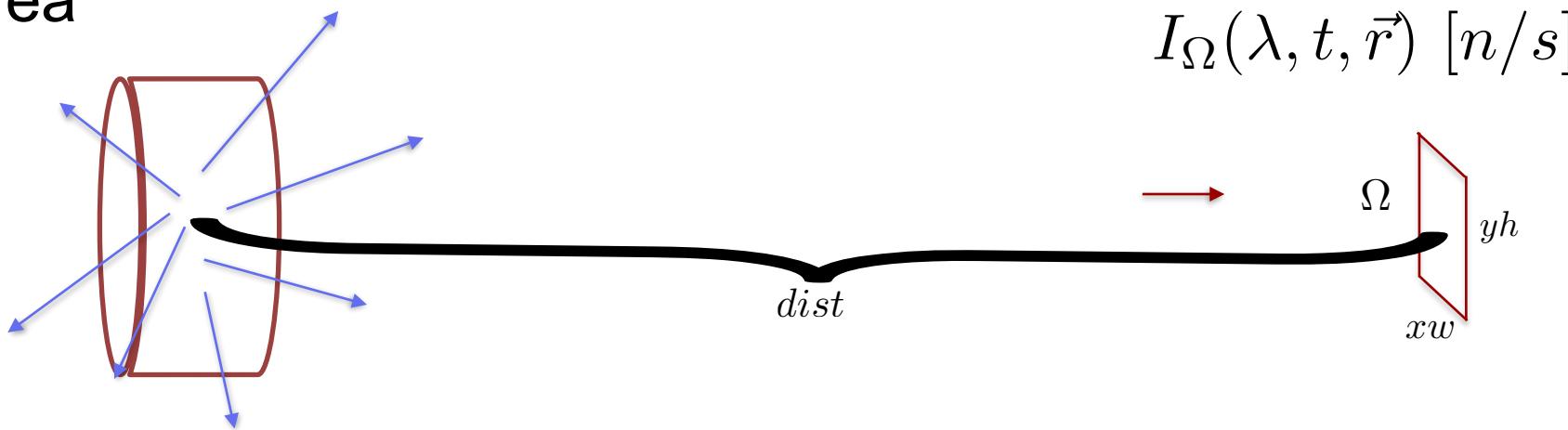
- The emission intensity into our chosen solid angle  $\Omega$  can be a function of wavelength, time (pulsed sources) and possibly point of origin on the source surface (e.g. for partially coherent descriptions).



- The emission of particles into the solid angle  $\Omega$  is in fact an integration and leads to a simulated “intensity” of  $I_\Omega$  [n/s].
- In McXtrace, that integrated intensity is partitioned over a given set of particle rays referred to as **ncount**, -n or --ncount
- The default **ncount** is 1e6 rays

$$\begin{aligned} I(\lambda) & [n/s/str] \\ I(\lambda, t) & [n/s/str] \\ I(\lambda, t, \vec{r}) & [n/s/str] \end{aligned}$$

- Our rays are emitted randomly, sampling  $\Omega$  and all variables of the source “spectrum”, i.e. wavelength, time and area



- assigning ray weights  $p$  such that

$$\sum_{j=1}^{\text{ncount}} p_j = \int_{d\lambda, dt, d\vec{r}} I_\Omega(\lambda, t, \vec{r})$$

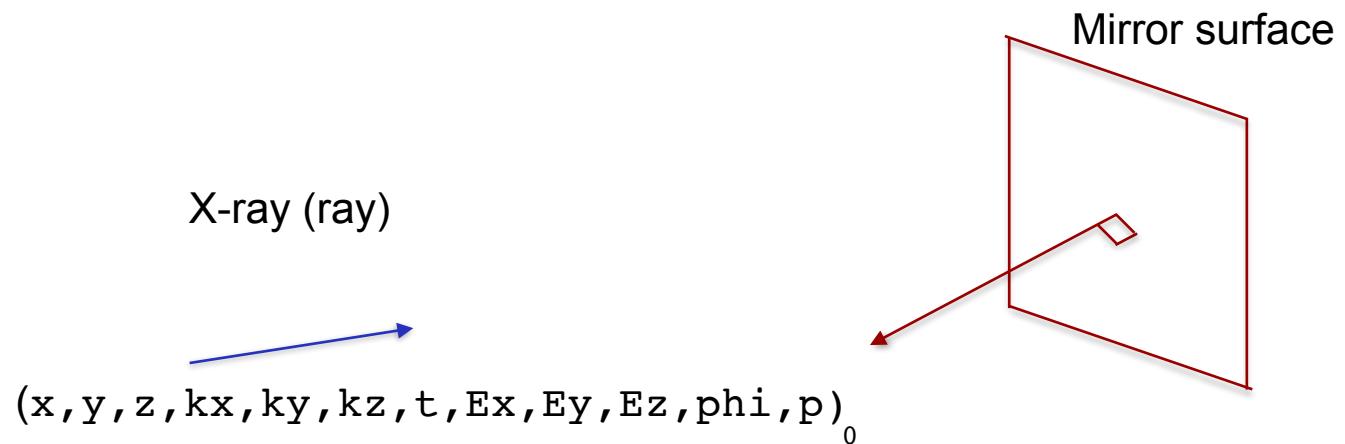
- Defining the ray starting conditions imply setting:

X-ray package:

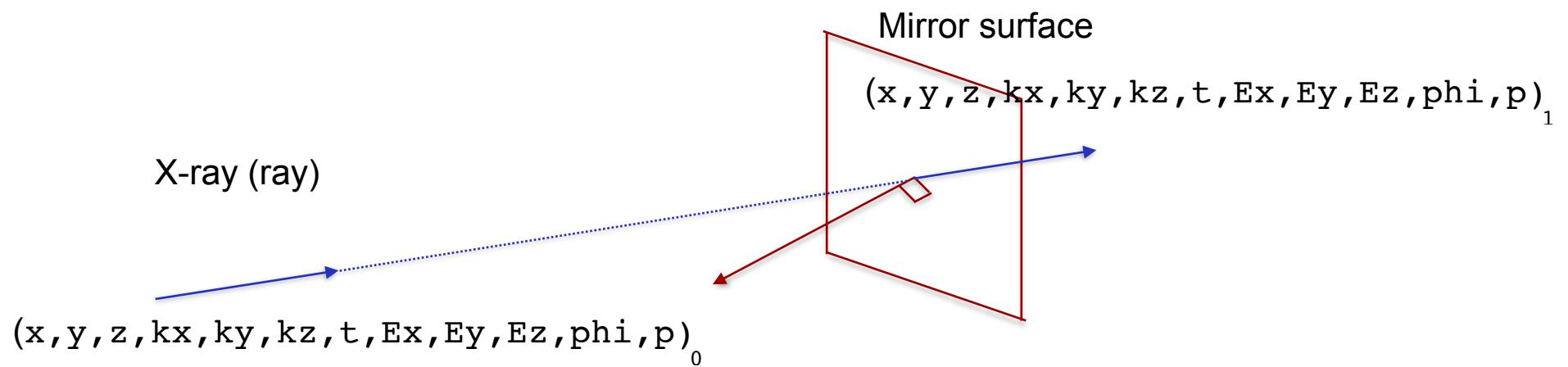
Weight ( $p$ ), # photons (left) in the package  
Coordinates ( $x, y, z$ )  
Wavevector ( $k_x, k_y, k_z$ )  
Polarization ( $E_x, E_y, E_z$ )  
Phase ( $\phi$ )  
Time( $t$ )

- The **starting point** on the surface, i.e.  $\vec{r}$  (in the code variables  $x, y, z$ )
- The **direction** into  $\Omega$  and our  $\lambda / k$  (in the code variables  $k_x, k_y, k_z$ )
- The **starting time** (in the code the variable  $t$ )
- The initial **intensity** / weight of the X-ray ray (in the code the variable  $p$ )
- If needed the initial
  - Electric field polarisation** (in the code the variables  $E_x, E_y, E_z$ )
  - X-ray phase (in the code the variable  $\phi$ )

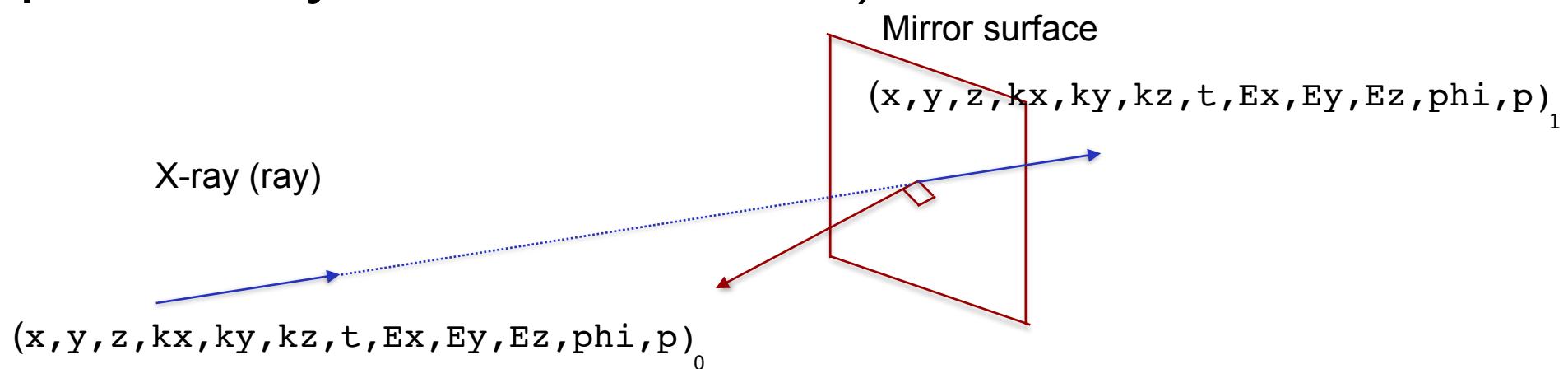
- 1 starting situation



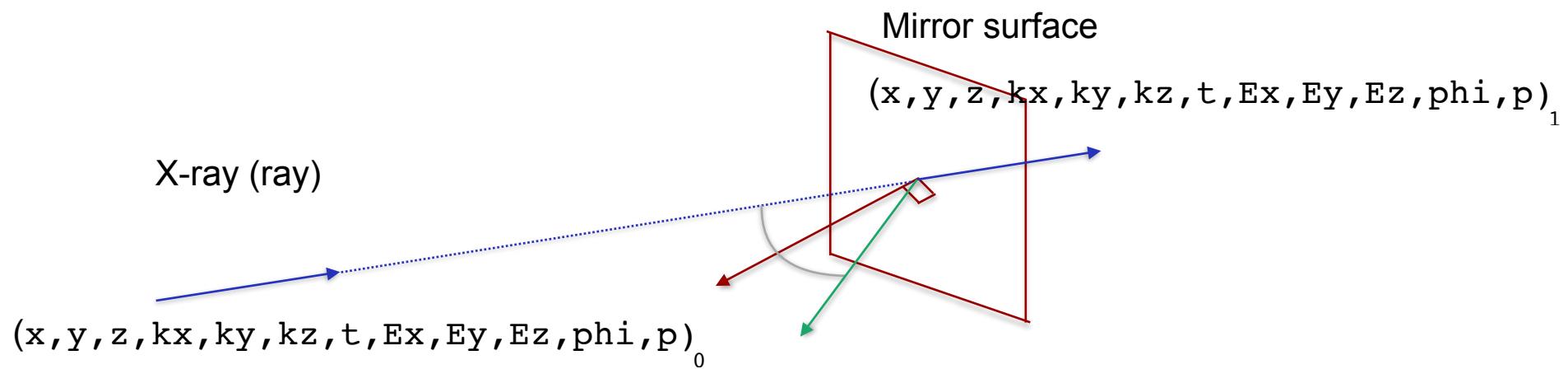
- 2. Propagate to the mirror surface



- 3. Checks (are we on surface, what is probability of reflection etc.)

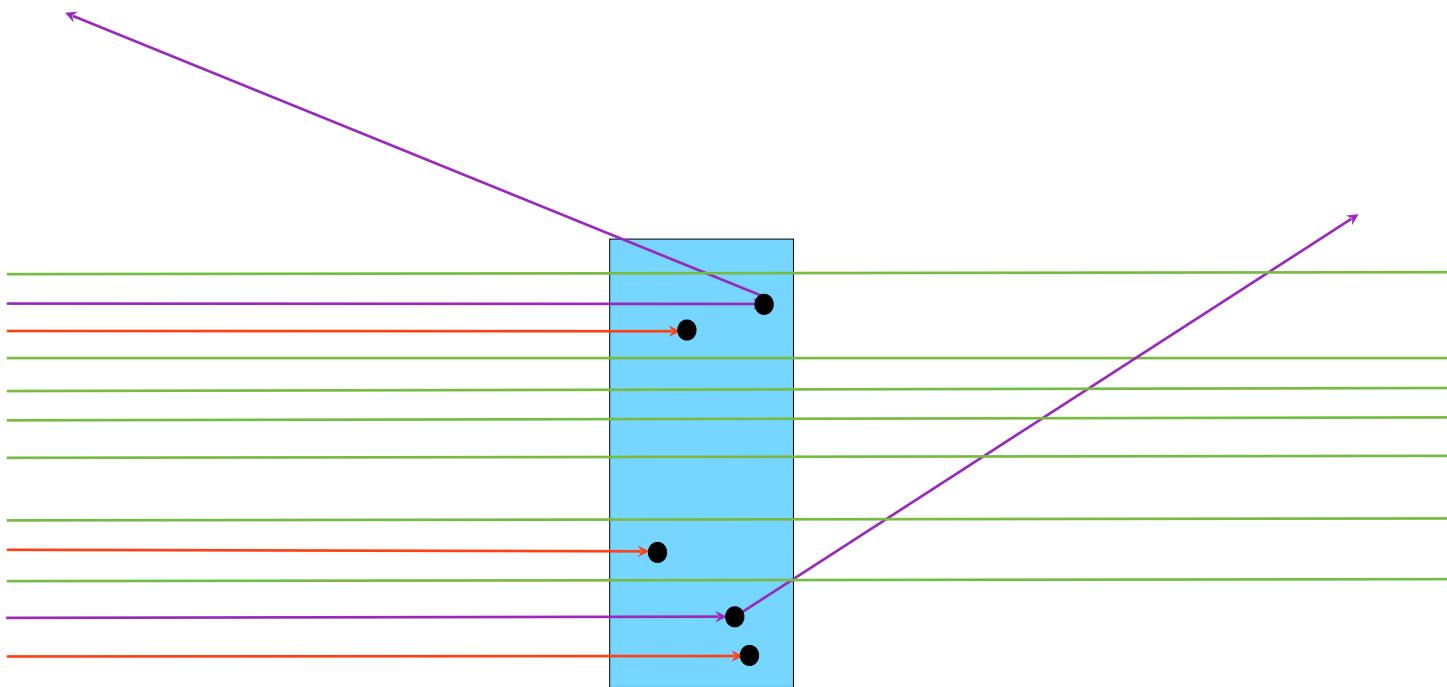


- 4. Reflect



**Weight of final ray is adjusted according to reflectivity, see next slide**

$(x, y, z, kx, ky, kz, t, Ex, Ey, Ez, \phi, p)_2$



A photon hitting a sample can be:  
**absorbed**, **transmitted**, or **scattered**

# Samples

For a **non-thin** sample the probabilities for **absorption**, **transmission** or **scattering** are given by

$$p_A = (1 - e^{-\Sigma_T t})(\Sigma_A/\Sigma_T)$$

$$p_S = (1 - e^{-\Sigma_T t})(\Sigma_S/\Sigma_T)$$

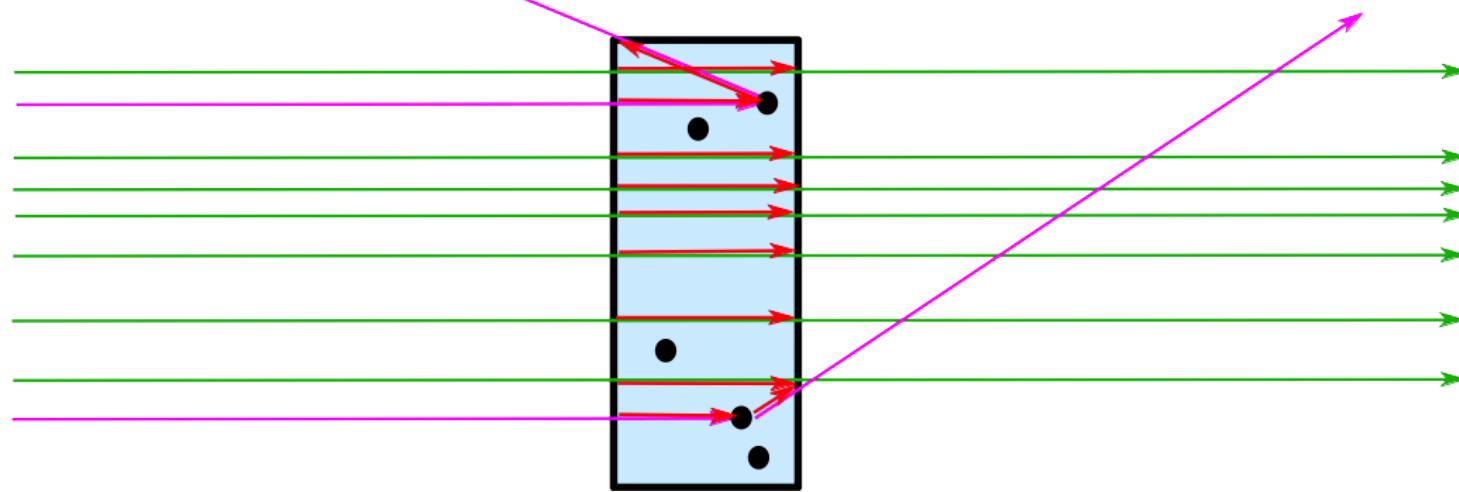
$$p_T = 1 - p_S - p_A = e^{-\Sigma_T t}$$

**t** = sample thickness

$$\Sigma_* = \rho \sigma_*$$

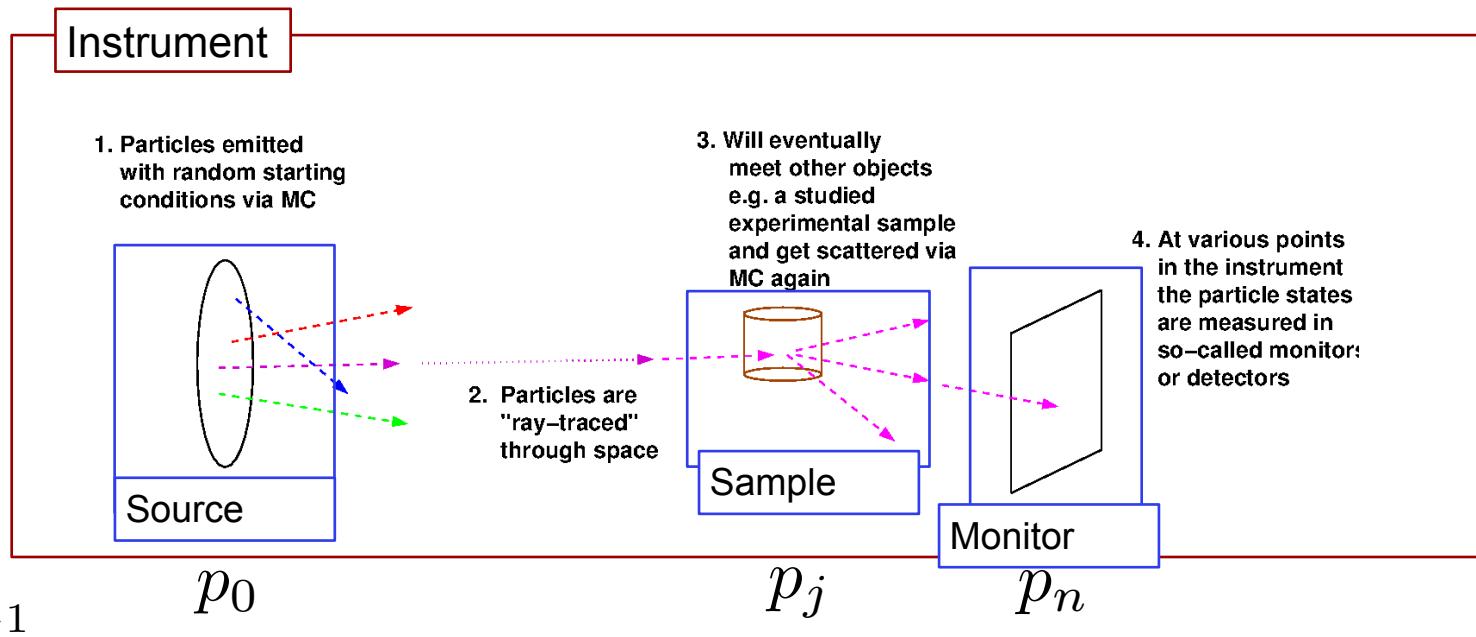
macroscopic cross section [cm<sup>-1</sup>]      microscopic cross section [barn/fm<sup>2</sup>]  
number density [atoms/cm<sup>3</sup>]

# Samples/Matter interaction in General in McXtrace



A photon ray hitting a sample can be:  
transmitted+absorption, or scattered+absorption

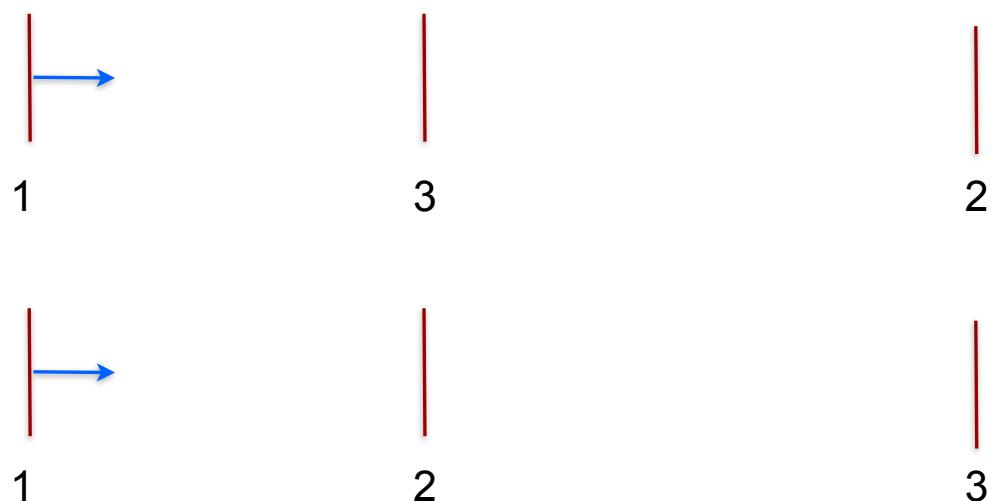
In a given component, the X-ray intensity is adjusted by a multiplicative factor (probability)



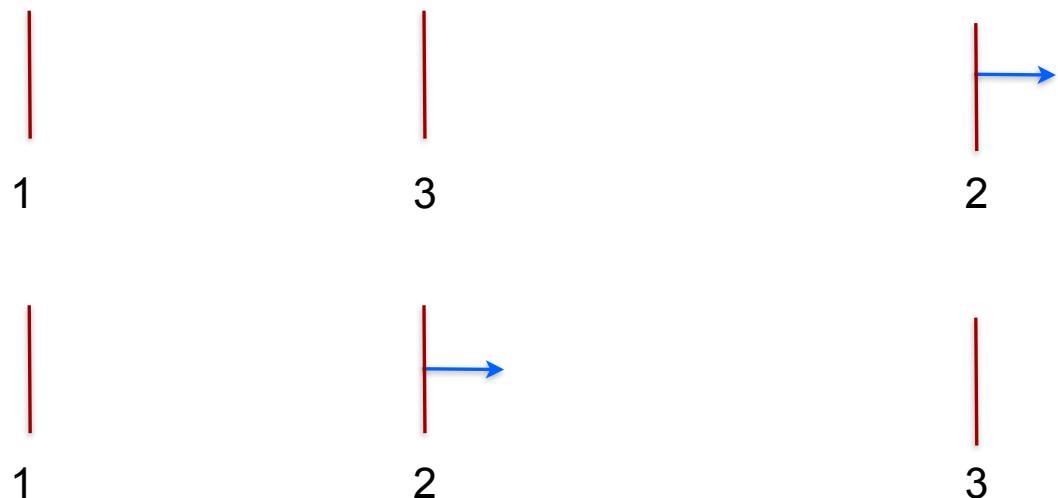
The weight multiplier of the  $j$ 'th component,  $w_j$ , is calculated by the probability rule  $f_{MC,b}w_j = P_b$  where  $P_b$  is the physical probability for the event "b", and  $f_{MC,b}$  is the probability that the Monte Carlo simulation selects this event.

In case of "branching", i.e. multiple outcomes, it is clear that  $\sum_b f_{MC,b} = 1$

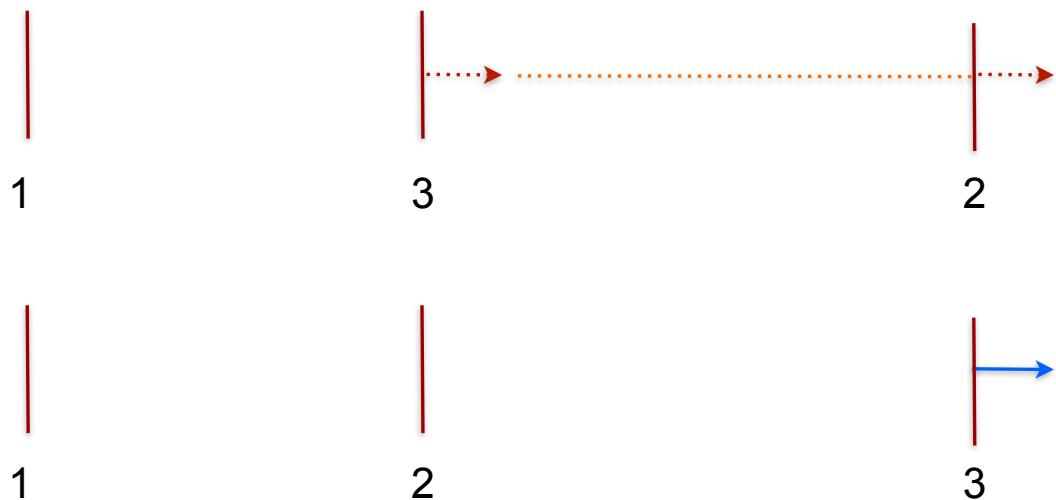
$$p_j = p_0 \prod_{k=1}^j w_k$$



Starting at the source



Moving to first comp in the list

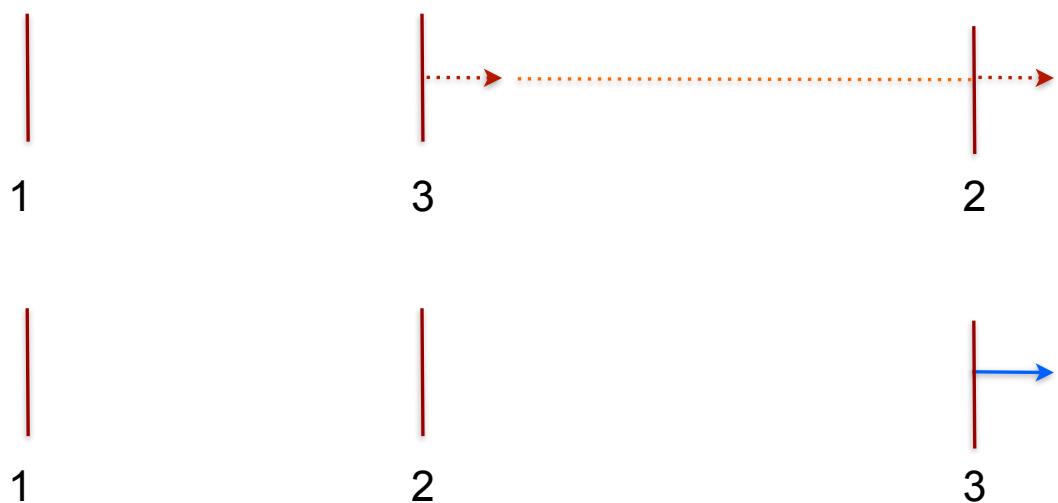


Moving to 3rd comp in list requires “moving back in time”.

Default behavior is to ABSORB this type of X-ray.

For monitors use `restore_xray=1` in this case.

For homegrown comps use `ALLOW_BACKPROP` macro.



Moving to 3rd comp in list requires “moving back in time”.

Default behavior is to ABSORB this type of X-ray.

For monitors use `restore_xray=1` in this case.

For homegrown comps use `ALLOW_BACKPROP` macro.

**The order of components is important, and in general overlaps should be avoided!**



## Monitors: in general

### REALITY:

#### Monitors:

- Intensity probe of the beam
- Semi-transparent to X-rays → Low Efficiency

#### Detectors:

- Should detect *all* photons → Efficiency as high as possible

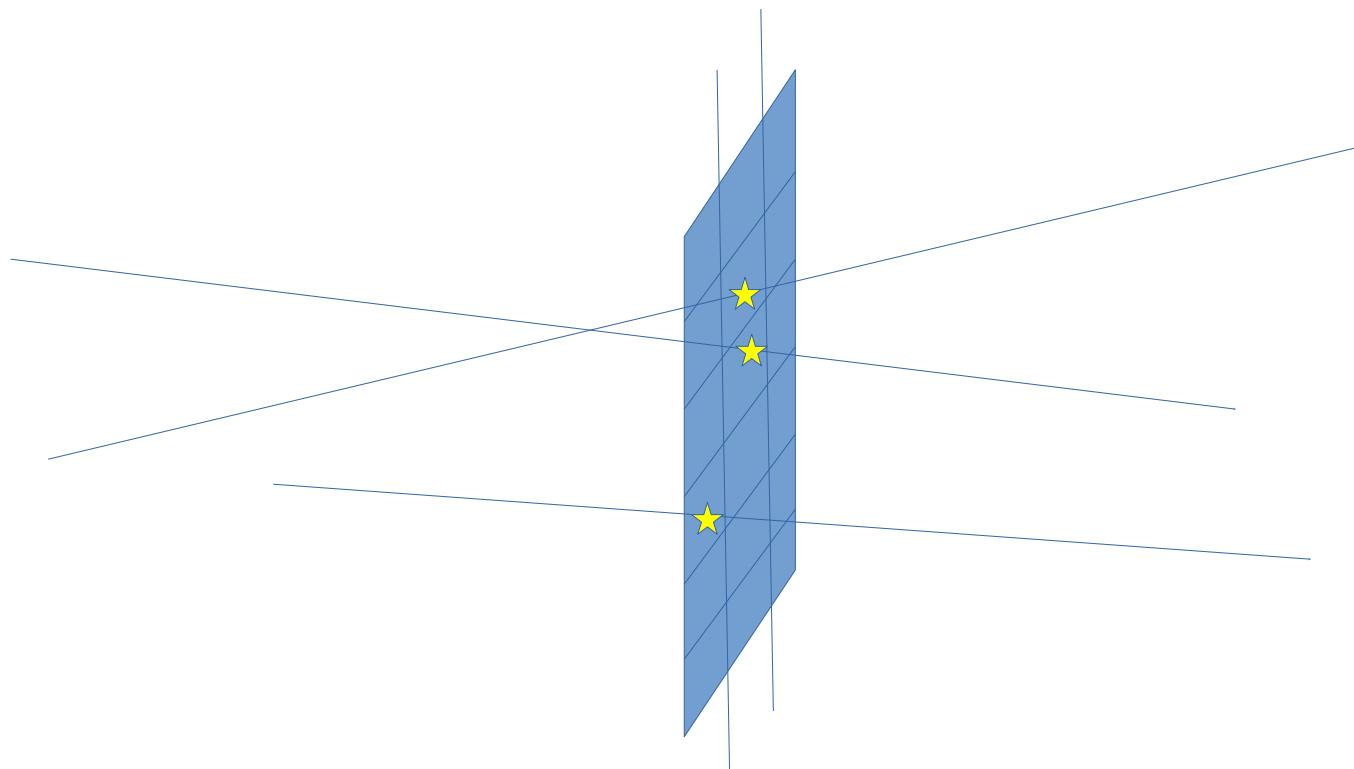
### SIMULATIONS (McXtrace):

#### In McXtrace:

- We can program monitors and detectors to behave any way we like. We refer to both of those indistinguishably as 'monitors'.
- E.g. monitor with Efficiency =100% and Transparency=100%

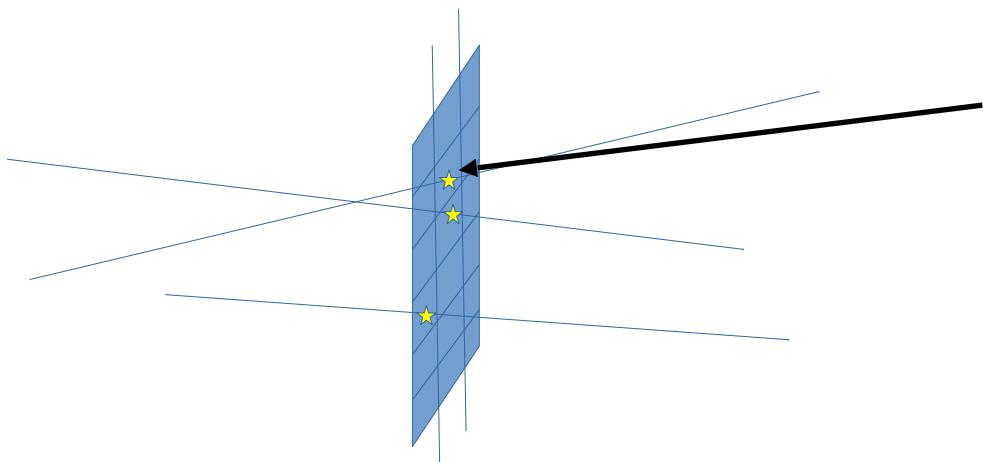


## Monitors: Example PSD\_monitor





## Monitors: Example PSD\_monitor



When the simulation has been completed, the detected intensity in pixel  $(i,j)$  is:

$$I(i,j) = \sum_{x_k, y_k \in pixel(i,j)} p_k; k = raynumber.$$

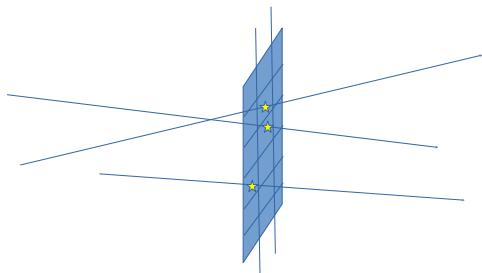
... during simulation, the pixels are maintained as running sums.



...

## Monitors:

Example PSD\_monitor and L\_monitor



### TRACE

```

COMPONENT origin = Progress_bar()
AT (0,0,0) ABSOLUTE

COMPONENT src = Source_flat(
    radius=0.05, lambda0=2.5, dlambda=1.5,
    focus_xw=0.1, focus_yh=0.1, dist=5)
AT (0,0,0) RELATIVE origin

COMPONENT psd = PSD_monitor(xwidth=0.2,
    yheight=0.2, filename="psd.dat")
AT (0,0,5) RELATIVE src

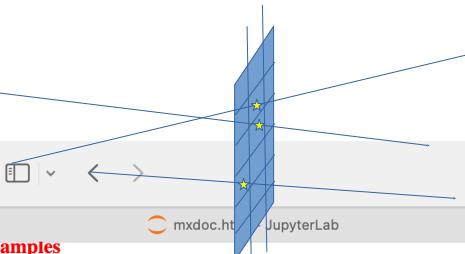
COMPONENT lm = L_monitor(xwidth=0.2, yheight=0.2,
    filename="lm.dat",
    Lmin=0, Lmax=8)
AT (0,0,5+0.01) RELATIVE src

```

# Monitors: what is available



mxdoc monitor from commandline



mxdoc monitor from commandline

127.0.0.1

mxdoc.html JupyterLab noVNC McXtrace : Components/Instruments Library Start Page

Samples	Name	Origin	Author(s)	Source code	Description
---------	------	--------	-----------	-------------	-------------

## Detectors and monitors

Name	Origin	Author(s)	Source code	Description
<a href="#">PreMonitor_nD</a>	<a href="#">ILL (France)</a>	<a href="#">Emmanuel Farhi</a>	<a href="#">comp</a>	Release: McXtrace 1.1 Xray parameters spatial cross-correlation monitor.
<a href="#">PSD_monitor_coh</a>	Risoe	Erik Knudsen	<a href="#">comp</a>	Release: McXtrace 0.1 Position-sensitive monitor with phase intergration.
<a href="#">E_monitor</a>	Risoe	Erik Knudsen	<a href="#">comp</a>	Release: McXtrace 0.1 Energy-sensitive monitor.
<a href="#">L_monitor</a>	Risoe	Kristian Nielsen and Kim Lefmann	<a href="#">comp</a>	Release: McXtrace 0.1 Wavelength-sensitive monitor.
<a href="#">Divergence_monitor</a>	DTU Physics	Erik B Knudsen	<a href="#">comp</a>	Horizontal+vertical divergence monitor.
<a href="#">TOF_monitor</a>	British Airways	Erik B Knudsen	<a href="#">comp</a>	Release: 1.2 Rectangular Time-of-flight monitor. %P INPUT PARAMETERS: xmin: Lower x bound of detector opening (m) xmax: Upper x bound of detector opening (m) ymin: Lower y bound of detector opening (m) ymax: Upper y bound of detector opening (m) xwidth: Width of detector. Overrides xmin,xmax. (m) yheight: Height of detector. Overrides ymin,ymax. (m) nt: Number of time bins (1) dt: Length of each time bin (mu-s) tmin: Lower time limit (mu-s) tmax: Upper time limit (mu-s) filename: Name of file in which to store the detector image (text) restore_xray: If set, the monitor does not influence the xray state (1) OUTPUT PARAMETERS: TOF_N: Array of xray counts TOF_p: Array of xray weight counts TOF_p2: Array of second moments
<a href="#">EPSD_monitor</a>		Erik B Knudsen	<a href="#">comp</a>	
<a href="#">Monitor</a>	Risoe	Erik Knudsen	<a href="#">comp</a>	Release: McXtrace 0.1 Energy-sensitive monitor.
<a href="#">DivPos_monitor</a>	DTU Physics	Erik B Knudsen	<a href="#">comp</a>	Release: McXtrace 1.3 Divergence/position monitor (acceptance diagram).
<a href="#">PSD_monitor</a>	Risoe	Erik B Knudsen	<a href="#">comp</a>	Position-sensitive monitor.
<a href="#">Monitor_nD</a>	<a href="#">ILL</a>	<a href="#">Emmanuel Farhi</a>	<a href="#">comp</a>	Release: McXtrace 1.6 This component is a general Monitor that can output 0/1/2D signals (Intensity or signal vs. [something] and vs. [something] ...)
<a href="#">PSD_monitor_4PI</a>	Risoe	Erik Knudsen	<a href="#">comp</a>	Release: McXtrace 0.1 Spherical position-sensitive detector.
<a href="#">W_psd_monitor</a>	Risoe	Erik Knudsen	<a href="#">comp</a>	Release: McXtrace 0.1 Position-sensitive wattage monitor.
<a href="#">DivE_monitor</a>	DTU Physics	Erik B Knudsen	<a href="#">comp</a>	Divergence/Energy monitor.

## Misc

Name	Origin	Author(s)	Source code	Description
------	--------	-----------	-------------	-------------

## Contributed components

Name	Origin	Author(s)	Source code	Description
<a href="#">SAXSQMonitor</a>	KU-Science	Martin Cramer Pedersen (mcpe@nbi.dk)	<a href="#">comp</a>	Release: McXtrace 1.0 A circular detector measuring the radial average of intensity as a function of the momentum transform in the sample.

## Obsolete (avoid usage whenever possible)

Name	Origin	Author(s)	Source code	Description
------	--------	-----------	-------------	-------------

## Instrument Examples

Name	Origin	Author(s)	Source code	Description
<a href="#">Focal_pt_monitor</a>	DTU Physics	Erik B Knudsen (erkn@fysik.dtu.dk)	<a href="#">instr</a>	Template for creating a focal point monitor.
<a href="#">Test_monitors</a>	DTU Physics	Erik B Knudsen (erkn@fysik.dtu.dk)	<a href="#">instr</a>	Unit test instrument for various monitors.

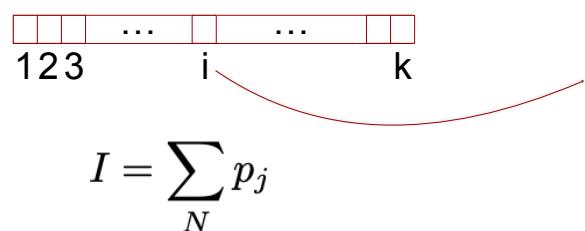
## Local components

Name	Origin	Author(s)	Source code	Description
------	--------	-----------	-------------	-------------



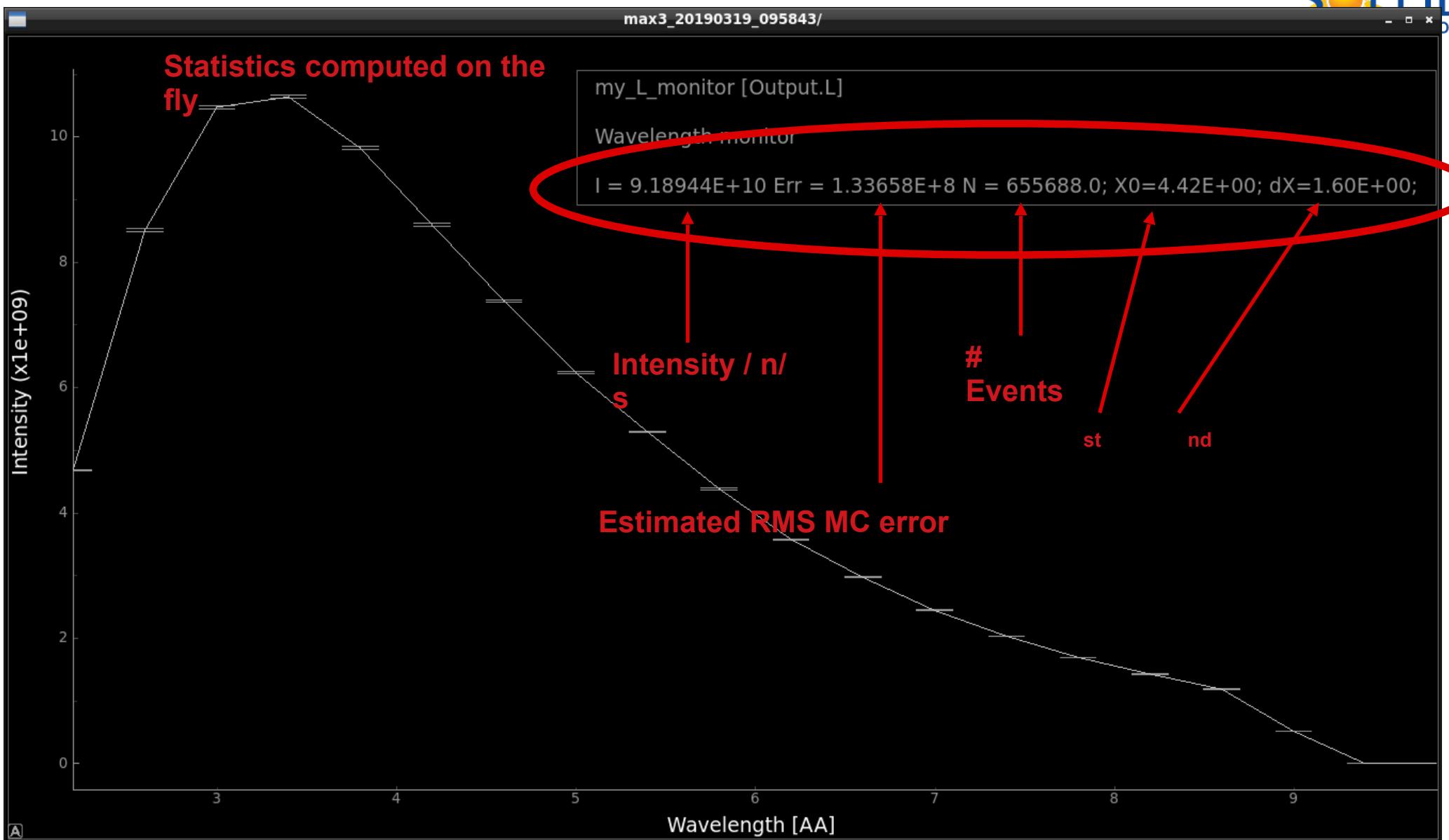
# In a histogram sense

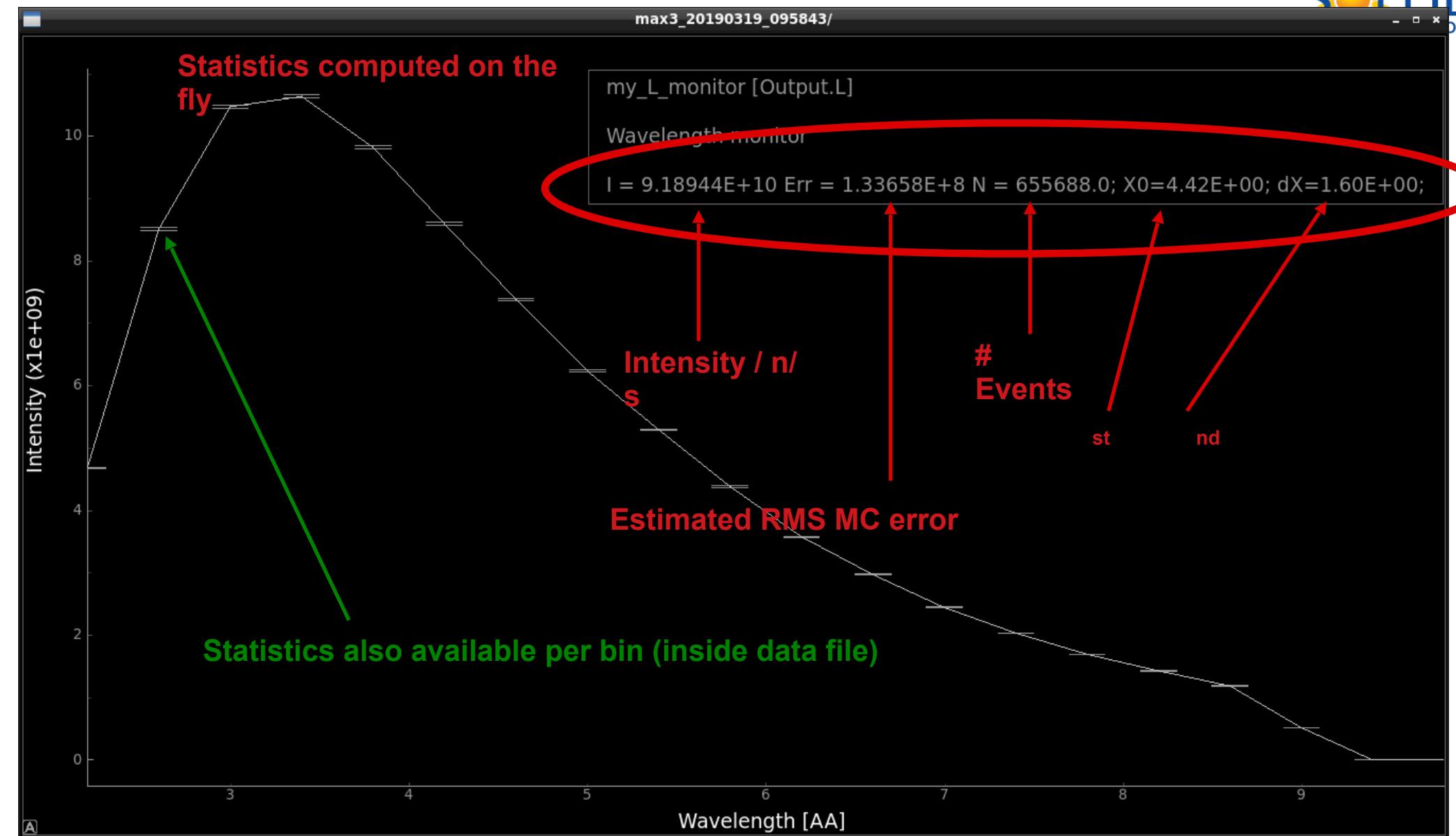
② Imagine a histogram, e.g.  $\mathbf{I}(\lambda)$



In bin  $i$ , **N** events each carrying a fractional intensity  $p_j$  so that

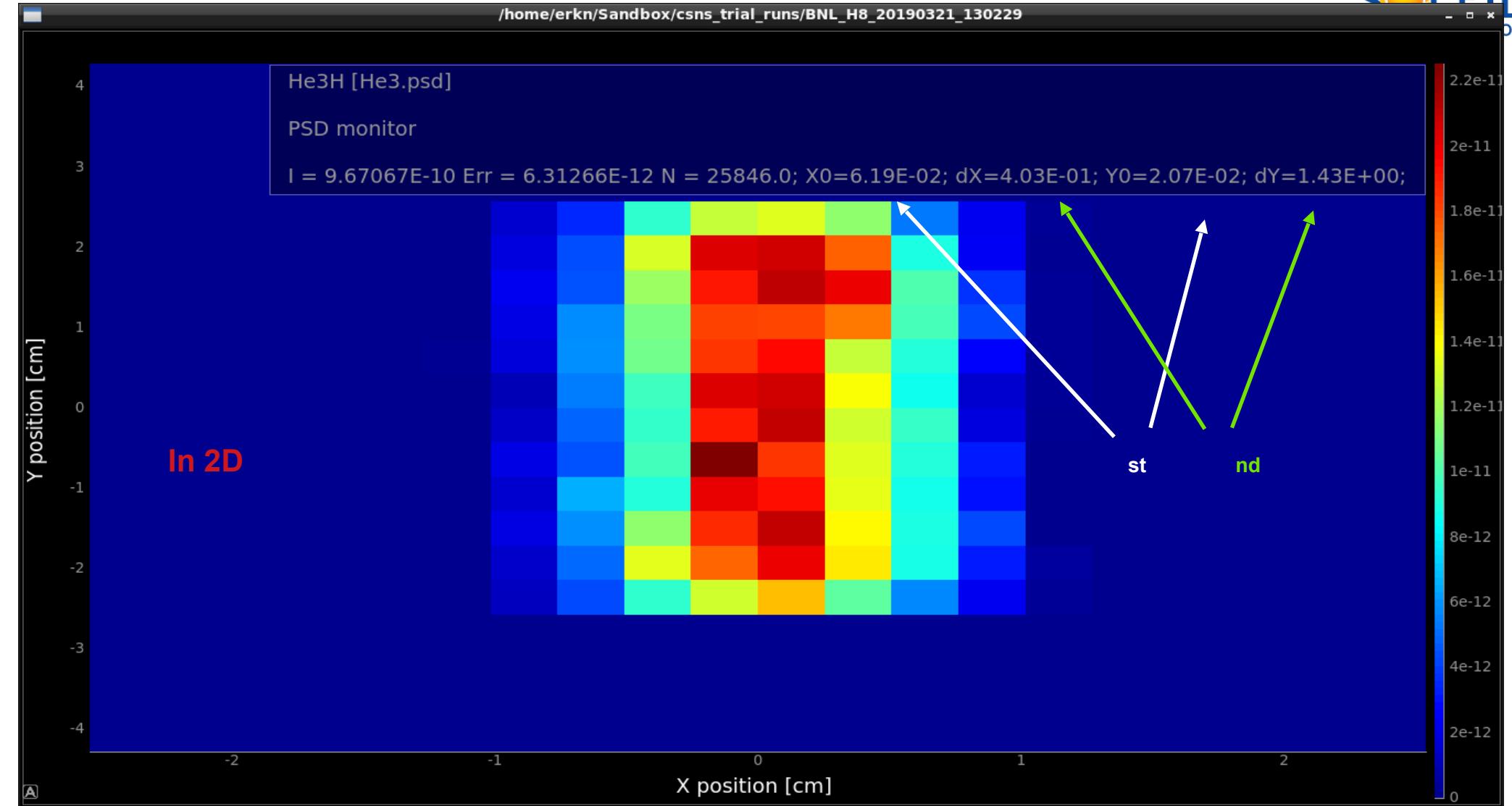
② The RMS variance over that set becomes our statistical error bar **E**







/home/erkn/Sandbox/csns\_trial\_runs/BNL\_H8\_20190321\_130229



## From "Virtual experiments - the ultimate aim of neutron ray-tracing simulations", K. Lefmann et al., Journal of Neutron F

Let  $n$  be the number of neutron rays reaching the detector, and let the rays have (different) weights,  $w_i$ . The simulated intensity is then given by

$$I = \sum_{i=1}^n w_i. \quad (1)$$

The estimate of the error on this number is calculated in the McStas manual [1], and the standard deviation is approximated by

$$\sigma^2(I) = \sum_{i=1}^n w_i^2. \quad (2)$$

In real experiments,  $w_i = 1$ , whence we reach  $I = n$  and  $\sigma(I) = \sqrt{I}$  as expected (for counts exceeding 10). Let the virtual time be denoted by  $t$ . The simulated counts during this time becomes

$$C = tI, \quad (3)$$

# From "Virtual experiments - the ultimate aim of neutron ray-tracing simulations". K. Lefmann et al.. Journal of Neutron Re

and its error bar estimate is

$$\sigma^2(C) = t^2 \sigma^2(I). \quad (4)$$

However, to simulate a realistic counting statistics, we must fulfill

$$\sigma_{\text{VE}}(C_{\text{VE}}) = \sqrt{C_{\text{VE}}}. \quad (5)$$

This is obtained by adding to (3) a Gaussian noise  $E(\Sigma)$  of mean value zero and standard deviation  $\Sigma$ :

$$C_{\text{VE}} = tI + E(\Sigma). \quad (6)$$

The standard deviation for the VE becomes

$$\sigma_{\text{VE}}^2(C) = t^2 \sigma^2(I) + \Sigma^2. \quad (7)$$

Now, the requirement (5) allows us to determine  $\Sigma$ :

$$\Sigma^2 = tI - t^2 \sigma^2(I). \quad (8)$$

Since  $\Sigma^2$  must remain positive, we reach an upper limit on  $t$

$$t_{\max} = \frac{I}{\sigma^2(I)}. \quad (9)$$

## Sketch of an algorithm...

1. On a given McXtrace histogram
2. For the non-zero bins, calculate

$$t_{\max} = \frac{I}{\sigma^2(I)}.$$

The *smallest*  $t_{\max}$  defines the “maximal counting time” allowed by your statistics

3. Preferably a “background” should be added - use a “known experimental value” or an estimate...



A general monitor for 0D/1D/2D records

## Monitor\_nD

*The all-in-one , swiss-army-knife of monitors*  
**Monitor\_nD** can have almost any shape, and record  
any requested standard quantities





A general monitor for 0D/1D/2D records

# Monitor\_nD

## Examples

```
COMPONENT MyMon = Monitor_nD( xwidth = 0.1, yheight = 0.1,  
zdepth = 0,  
options = "intensity per cm2 angle,limits=[-5 5],  
bins=10,with borders, file = mon1")
```

```
options = "multiple kx ky kz, auto abs log t, and list all  
neutrons"
```

```
options = "banana, theta limits=[10,130],  
bins=120, y"
```

# Monitor\_nD

... or monitor just about anything:

```
COMPONENT MyMon = Monitor_nD(xwidth = 0.1, yheight = 0.1,  
                               user1=age, username1="Age of the Captain [years]",  
                               options="user1, auto")
```