

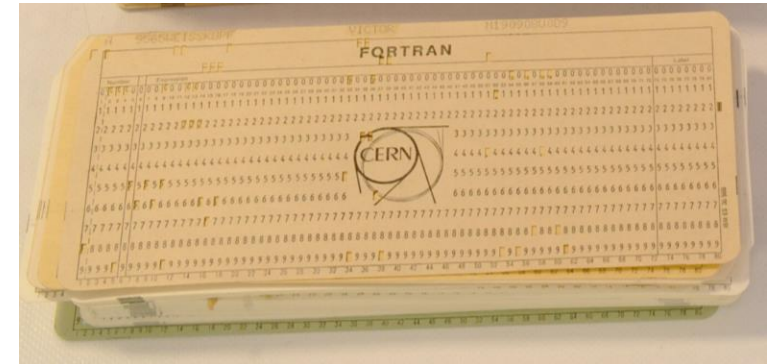


Introduction to Flair and basic input

A very basic introduction to perform your first simulation

A very short introduction

- Fluka's story began a long time ago (1970s)...
 - ...no graphical interfaces, input and output via text file
- Inputfile can be very long > 50k lines
- Inputfile based on "cards": `.inp` file
- Each card has 1 name, 6 values (called WHATs), 1 string (called SDUM)
- Two examples of cards (the actual meaning is not relevant here):

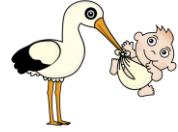


BEAMPOS	4750.5	130.0	4866.5				NEGATIVE
BEAM	-0.4	0.2	5.0	1.E-4	1.E-4		ELECTRON

↑ Card name ↑ WHAT(1) ↑ WHAT(2) ↑ WHAT(3) ↑ WHAT(4) ↑ WHAT(5) ↑ WHAT(6) ↑ SDUM

A very short introduction

- In 2006, Flair was born!



Fluka advanced interface

Input file creation

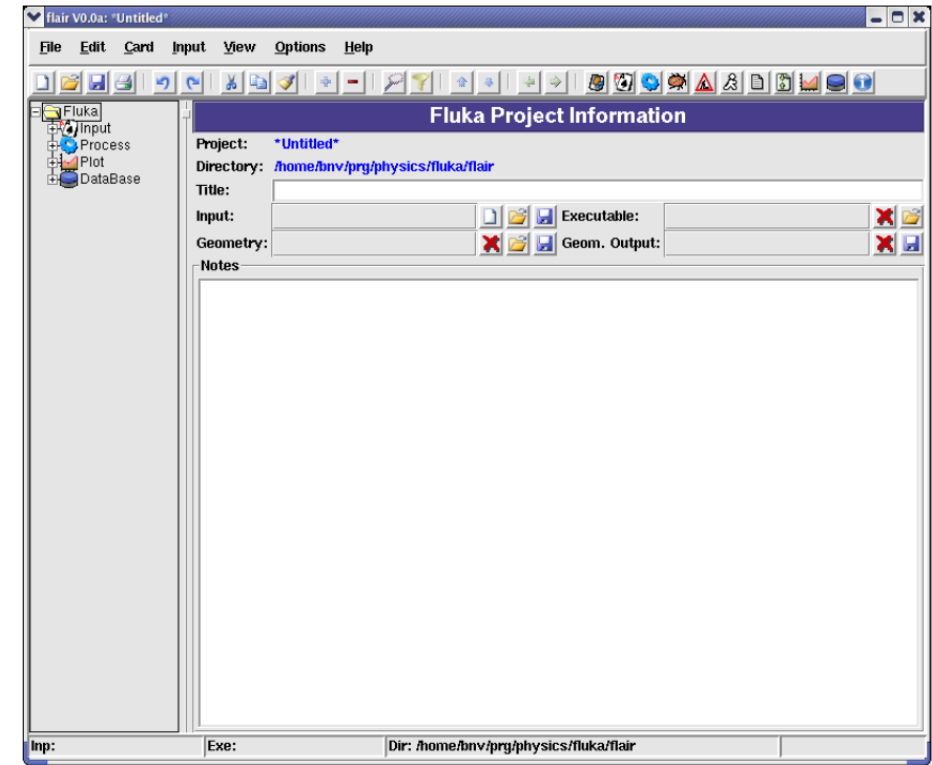
Geometry visualisation and construction

Simulation execution

Results visualisation

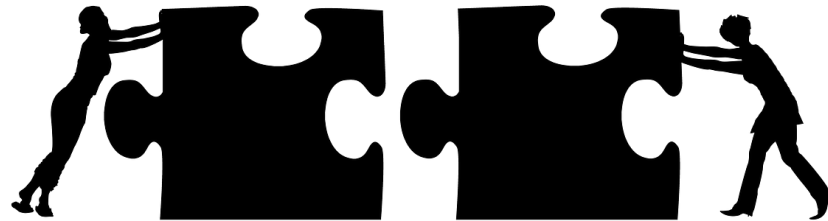
- Flair acts as an intermediate layer between the user and the inputfile
- It allows a user friendly editing of the Fluka input
- Based on a `.flair` file and generates the `.inp` file that is run by Fluka

Flair ≠ Fluka



Fluka & Flair

- Although strongly linked, they are two different things (`.inp` \neq `.flair`)
- Fluka is a Monte Carlo transport code based on text files
- Flair is a graphical interface to Fluka
- They work together but are different



- It is possible to work with Fluka only using text editors (for expert or old users)
- Flair is not just a graphical interface for text editing
- Flair has a lot of features very useful for expert users
- This entire course will be based on Flair

Starting Flair and basic nomenclature

- Flair can be started from command line, e.g.:

```
$ flair my_input.flair
```

- Flair can also be used to open .inp files, e.g:

```
$ flair my_old_file.inp
```

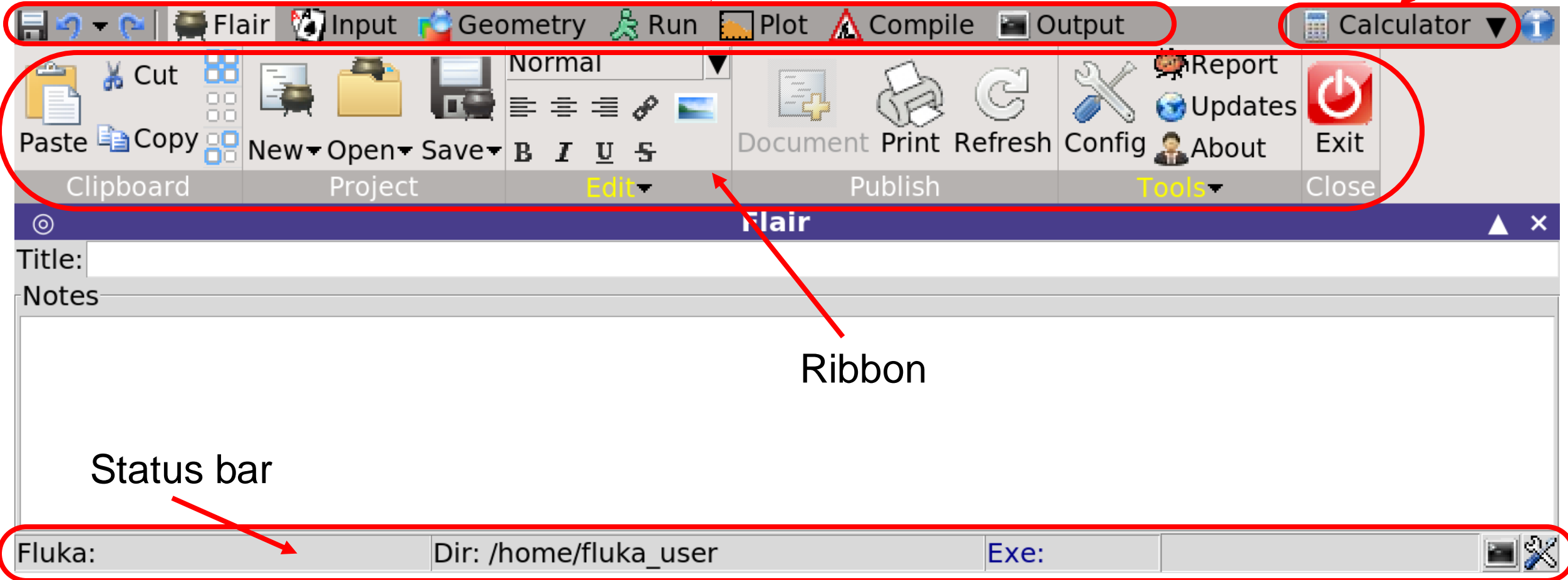
- Linux reminder about copying files:

```
$ cp exercise1.flair my_dir/.
```

Starting Flair and basic nomenclature

Configurable “Ribbon tab” or “Program tab”

Dynamic tab



What's each tab for?

The screenshot shows the Flair software interface with several tabs highlighted by red circles. Red arrows point from these tabs to descriptive text boxes:

- Input**: Build input and geometry
- Geometry**: Build geometry and plot results
- Run**: Run and merge results
- Plot**: Plot results
- Compile**: Compile own executable
- Output**: Visualize output files and messages

The interface also includes a menu bar with options like Cut, Copy, Paste, New, Open, Save, Document, Print, Refresh, Config, Report, Updates, About, and Exit. The status bar at the bottom shows 'Fluka: Dir: /home/fluka_user Exe:'.

The input as a text file

- Mentioned here just for completeness

.flair

```
TITLE
basic template
* Set the defaults for precision simulations
DEFAULTS                                PRECISIO
* Define the beam characteristics
BEAM
* Define the beam position
BEAMPOS
GEOBEGIN                                COMBNAME
0 0
* Black body
SPH blkbody 0.0 0.0 0.0 100000.0
* Void sphere
SPH void 0.0 0.0 0.0 10000.0
* Cylindrical target
RCC target 0.0 0.0 0.0 0.0 0.0 10.0 5.0
END
* Black hole
BLKBODY 5 +blkbody -void
* Void around
VOID 5 +void -target
* Target
TARGET 5 +target
END
GEOEND
* ..+...1...+...2...+...3...+...4...+...5...+...6...+...7...
ASSIGNMA BLCKHOLE BLKBODY
ASSIGNMA VACUUM VOID
ASSIGNMA COPPER TARGET
* Set the random number seed
RANDOMIZ 1.0
* Set the number of primary histories to be simulated in the run
START
STOP
-:--- basic.inp All (26,69) (Fluka)
```

.inp

.flair file includes
info & instructions
for the flair project

This course is based
on the use of flair,
no further mention
of these text files

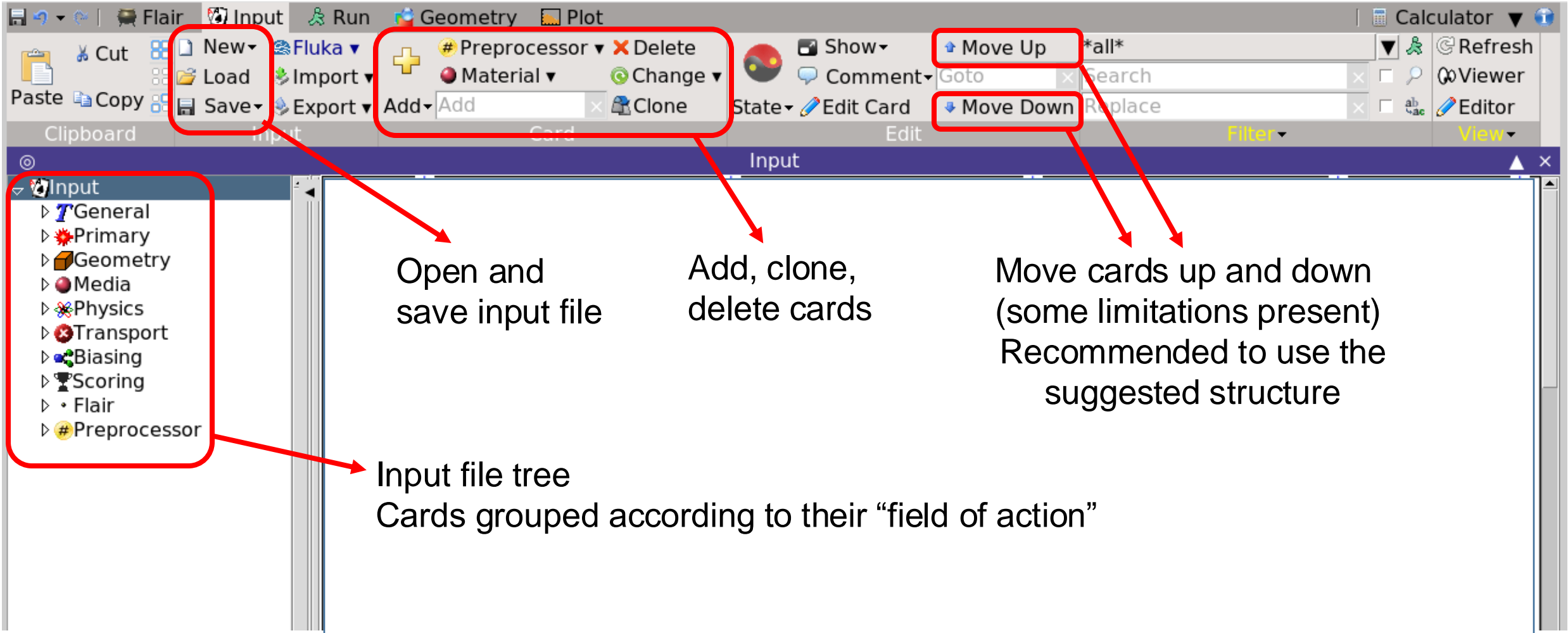
```
# flair project file
Version: 300
Mode: fluka
md5: c8e26fe184526e9282e8555b8fab2455
Input:
TITLE
fully-working template
#define pointless_define_1 10
#define pointless_define_2
*Set the defaults for precision simulations
DEFAULTS PRECISIO
*Define the beam characteristics
BEAM PROTON 0.8
*Define the beam position
BEAMPOS , 0. 0. -1.
GEOBEGIN COMBNAME
*Black body
SPH blkbody 0.0 0.0 0.0 100000.0
*Void sphere
SPH void 0.0 0.0 0.0 10000.0
*Cylindrical target
RCC target 0.0 0.0 0.0 0.0 0.0 10.0 5.0
END
*Black hole
REGION BLKBODY 5
+blkbody -void
*Void around
REGION VOID 5
+void -target
*Target
REGION TARGET 5
+target
END
GEOEND
*..+...1...+...2...+...3...+...4...+...5...+...6...+...7...
ASSIGNMA , BLCKHOLE BLKBODY
ASSIGNMA , VACUUM VOID
ASSIGNMA , COPPER TARGET
USRBIN allpart 10 ALL-PART -21 6. 6. 11. -6. -6. -2. 120. 120. 130.
USRBIN edep 10 ENERGY -22 6. 6. 11. -6. -6. -2. 120. 120. 130.
*Set the random number seed
RANDOMIZ , 1.0
*Set the number of primary histories to be simulated in the run
START , 10000.
STOP
EndInput

Page: Plot

# Run information
Run: <default>
End
Run: test/test
Define: pointless_define_2=10
Start: 1000
StartRun: 1598620157
End
Run: small_prod/small
Define: pointless_define_2=10
Start: 1000
Last: 1
```


Input tab – 1: general info

- Standard looking “Windows” tab

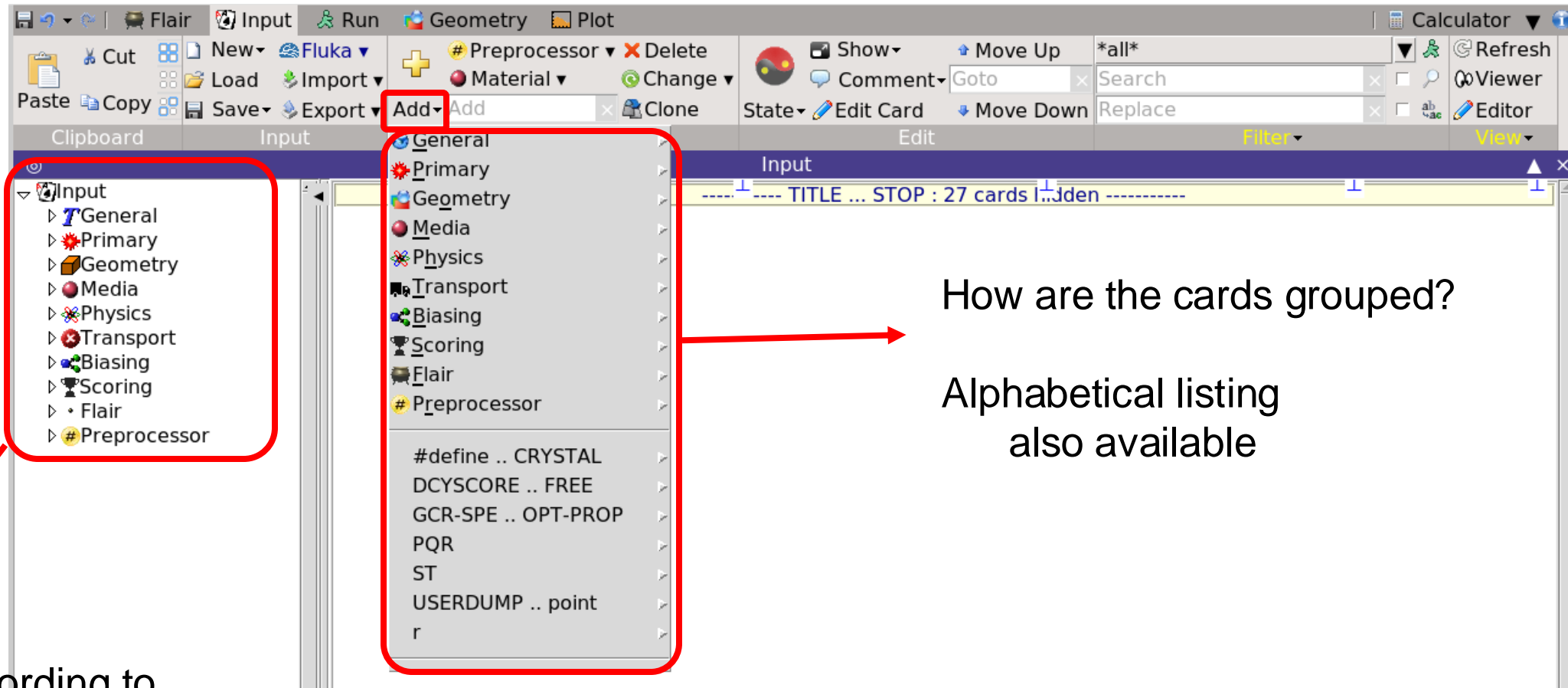


The screenshot displays the FLUKA software interface with the 'Input' tab selected. The interface is divided into several sections:

- Top Panel:** Contains various toolbars and buttons. A red box highlights the 'New', 'Load', and 'Save' buttons, with an arrow pointing to the text 'Open and save input file'. Another red box highlights the 'Add', 'Clone', 'Delete', and 'Change' buttons, with an arrow pointing to the text 'Add, clone, delete cards'. A third red box highlights the 'Move Up' and 'Move Down' buttons, with an arrow pointing to the text 'Move cards up and down (some limitations present) Recommended to use the suggested structure'.
- Left Panel:** Shows the 'Input file tree' with a red box around it and an arrow pointing to the text 'Input file tree Cards grouped according to their “field of action”'. The tree includes categories like General, Primary, Geometry, Media, Physics, Transport, Biasing, Scoring, Flair, and Preprocessor.
- Main Area:** Displays the 'Input' card editor with a search bar and a list of cards.

Input tab – 2: input file tree and card grouping

- Input file tree and card grouping


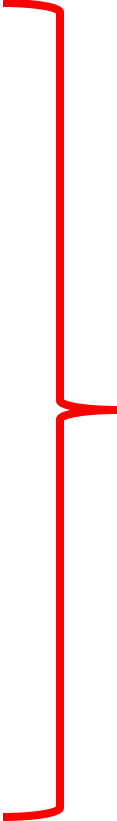


How are the cards grouped?

Alphabetical listing
also available

Input file tree
Cards grouped according to
their "field of action"

Input tab – 3: input file tree and card grouping

- General: defaults selection  this lecture
 - Primary: definition of the particle source
 - Geometry: definition of the geometry
 - Media: definition and assignment of “materials”
 - Physics: control specific physics processes
 - Transport: control specific transport details
 - Biasing: definition of biasing
 - Scoring: definition of estimators
 - Preprocessor: definition of preprocessor instructions
 - Flair: definition of flair add-ons for visualization
- 
- dedicated lectures

Input tab – 4: General cards

TITLE

START

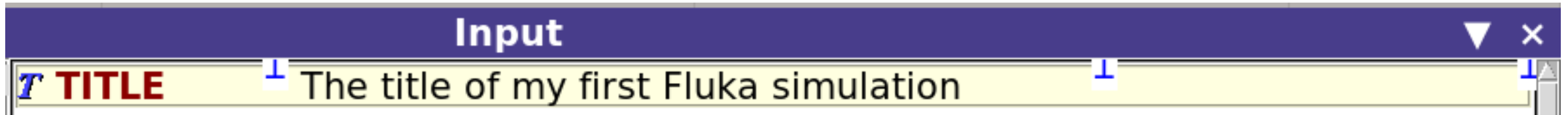
STOP

RANDOMIZe

DEFAULTS

TITLE

- Not a mandatory card
- Allows to assign a title to the simulations
- The title is printed in the output files



Input tab – 5: Mandatory cards

TITLE

START

STOP

RANDOMIZe

DEFAULTS

START

- Listed among the “Primary” cards
- It is a **mandatory** card (lack of it will result in an error)
- Allows to set the number of primary particles to be simulated
- Allows to set other parameters for advanced use

Set the number of primary histories to be simulated in the run

 **START**

No.: 10000.

Time:

Core: ▼

Report: default ▼

Input tab – 6: General cards

TITLE

START

STOP

RANDOMIZe

DEFAULTS

STOP

- Stop the execution of the program
- Not really mandatory (program stops at the end of the input)
- Can become handy for debugging purposes



STOP

Input tab – 7: General cards

TITLE

START

STOP

RANDOMIZ_e

DEFAULTS

RANDOMIZ

- Allows to initialize different random sequences
- For debugging purposes, the “random seed” must be the same
- Different “random seeds” are required in order to differentiate histories
- Flair takes care of the “random seeds” when spawning runs (see later)

Set the random number seed

 **RANDOMIZ**

Unit: 01 ▼

Seed: 123

Input tab – 8: General cards

TITLE

START

STOP

RANDOMIZe

DEFAULTS

DEFAULTS

- Allows to select the physics defaults (list of predefined defaults available)
- Physics defaults can be overridden with specific cards
- Can be preceded only by the **TITLE** and **GLOBAL** cards
- Given the progress over time in computer power, it is a reasonable approach to:
 - always select the most detailed physics defaults: **PRECISIO**
 - depending on the needs of the problem, override specific defaults

Set the defaults for precision simulations



DEFAULTS

: PRECISIO ▼

Input tab – 9: Expressions

- It is possible to specify values using expressions
- Possible to make parametric runs
- Fields starting with “=” will be evaluated by flair, e.g.:

```
BEAMPOS      x: =2+10*length
```

- Expressions are stored in the `.flair` file
- Expressions are also stored in the `.inp` file as comments, e.g.:

```
!@what.1=2+10*length
```

- The cards in the `.inp` file contain the evaluated values

Do not change by hand, they will be overwritten by flair!!!

Input tab – 10: Expressions

- See manual for details (see next slide for the manual)
- Useful predefined quantities
 - Units, e.g.: *MeV, mm, ms...* (warning: only treated as conversion factors)
 - Constants: *fwhm, c, qe...*
 - Particle masses: *Mp, Me...*
- All common mathematical functions: *sin(x), cos(x), exp(x)...*
- Some physics functions
- Card reference functions
 - *what(n)*
 - *body(name, what)*
 - *card(tag, sdum/id, what)*

Input tab – 11: “Reg:”, “to Reg:”, “Step:”

- Recurring feature in Fluka

- Not just regions:

- Regions
- Materials
- Detectors
- Lattices
- Particles
- ...

✂ EMFCUT

✂ EMFCUT

Fudgem:
🎯 ASSIGNMAT

🎯 BIASING

Opt: ▼
🎯 AUXSCORE

Delta Ray: ▼

🎯 LATTICE ▼

🔑 LOW-PWXS
db: ▼

🔑 OPT-PROD

🎯 PHOTONUC
E>0.7GeV: off ▼

🔑 EMF-BIAS
Old brems.: off ▼
Compton: off ▼

🔑 LAM-BIAS
Mat: ▼

Type: transport ▼			
e-e+ Threshold: Total ▼		e-e+ E:	γ:
Reg: ▼	_____	to Reg: ▼	Step:
Type: PROD-CUT ▼			
e-e+ Threshold: Total ▼		e-e+ E:	γ:
Mat: ▼	_____	to Mat: ▼	Step:
Mat: ▼		Reg: ▼	to Reg: ▼
Mat(Decay): ▼		Step: _____	Field: ▼
Type: ▼		RR:	Imp:
Reg: ▼	_____	to Reg: ▼	Step:
Type: ▼		Part: ▼	Set: ▼
Z: 0		A: 0	Isomer: 0
Det: ▼	_____	to Det: ▼	Step:
Reg: ▼	_____	to Reg: ▼	Step:
Lat: _____		to Lat: _____	Step:
Mat: ▼	_____	to Mat: ▼	Step:
IAZ:		S(α,β): ▼	T:
Type: ▼			
Mat: ▼	_____	to Mat: ▼	Step:
Type: ▼			All E: off ▼
Δ resonance: off ▼		Quasi D: off ▼	Giant Dipole: off ▼
Mat: ▼	_____	to Mat: ▼	Step:
Type: ▼		Ethr e-e+:	Ethr γ:
Bremsstrahlung: off ▼		Pair Prod.: off ▼	e+ ann @rest: off ▼
Bhabha&Moller: off ▼		Photo-electric: off ▼	e+ ann @flight: off ▼
Reg: ▼	_____	to Reg: ▼	Step:
Type: ▼		x mean life:	x λ inelastic:
Part: ▼	_____	to Part: ▼	Step:

Input tab – 12: “Reg:”, “to Reg:”, “Step:”

- Allows to assign a property to multiple “regions” (or whatever) in one single card
- Example 1: “CARBON” is assigned to all regions from “region_1” to “region_4”

REGION region_1 expr: +reg1	Neigh:		
REGION region_2 expr: +reg2	Neigh:		
REGION region_3 expr: +reg3	Neigh:		
REGION region_4 expr: +reg4	Neigh:		
ASSIGNMAT	Mat: CARBON ▼ Mat(Decay): ▼	Reg: region_1 ▼ Step:	to Reg: region_4 ▼ Field: ▼

REGION region_1 expr: +reg1	Neigh:		
REGION region_2 expr: +reg2	Neigh:		
REGION region_3 expr: +reg3	Neigh:		
REGION region_4 expr: +reg4	Neigh:		
ASSIGNMAT	Mat: CARBON ▼ Mat(Decay): ▼	Reg: region_1 ▼ Step: 1	to Reg: region_4 ▼ Field: ▼

Input tab – 13: “Reg:”, “to Reg:”, “Step:”

- Allows to assign a property to multiple “regions” (or whatever) in one single card
- Example 2: “CARBON” is assigned to “region_1” and “region_3”

REGION region_1	Neigh:		
expr: +reg1			
REGION region_2	Neigh:		
expr: +reg2			
REGION region_3	Neigh:		
expr: +reg3			
REGION region_4	Neigh:		
expr: +reg4			
ASSIGNMAT	Mat: CARBON ▼	Reg: region_1 ▼	to Reg: region_3 ▼
	Mat(Decay): ▼	Step: 2	Field: ▼

REGION region_1	Neigh:		
expr: +reg1			
REGION region_2	Neigh:		
expr: +reg2			
REGION region_3	Neigh:		
expr: +reg3			
REGION region_4	Neigh:		
expr: +reg4			
ASSIGNMAT	Mat: CARBON ▼	Reg: region_1 ▼	to Reg: region_4 ▼
	Mat(Decay): ▼	Step: 2	Field: ▼

Input tab – 14: “Reg:”, “to Reg:”, “Step:”

- Allows to assign a property to multiple “regions” (or whatever) in one single card
- Example 3: activate “PHOTONUC” (exact meaning not relevant here) for “CARBON”, “OXYGEN”, “ALUMINUM”, “COPPER”, etc.

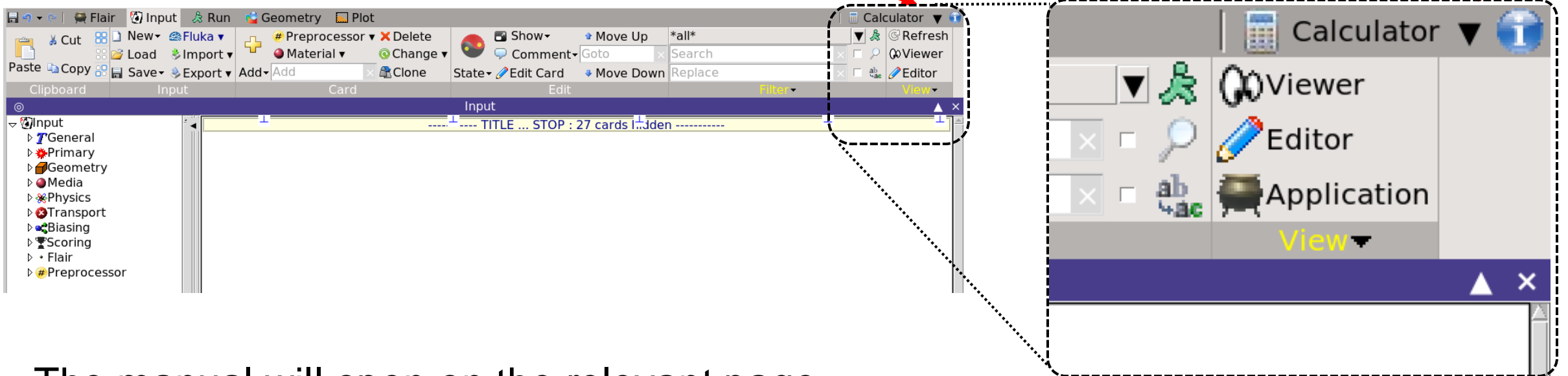
PHOTONUC E>0.7GeV: off ▼	Type: ▼ Δ resonance: off ▼ Mat: CARBON ▼	Quasi D: off ▼ to Mat: ▼	All E: off ▼ Giant Dipole: off ▼ Step: 2
REGION region_1 expr: +reg1		Neigh: CARBON	
REGION region_2 expr: +reg2		Neigh: NITROGEN	
REGION region_3 expr: +reg3		Neigh: OXYGEN	
REGION region_4 expr: +reg4		Neigh: MAGNESIU	
ASSIGNMAT	Mat: CARBON ▼	Neigh: ALUMINUM	
ASSIGNMAT	Mat(Decay): ▼ Mat: CARBON ▼	Neigh: IRON	
		Neigh: COPPER	
		Reg: SILVER	Reg: ▼
		Step: SILICON	Field: ▼
		Reg: GOLD	Reg: ▼

Input tab – 15: “Reg:”, “to Reg:”, “Step:”

- Allows to assign a property to multiple “regions” (or whatever) in one single card
 - The same concept applies to all other cases:
materials, particles, lattices, etc.
 - Special variables:
 - @LASTEREG i.e. the last defined region
 - @LASTMAT i.e. the last defined material
 - @LASTPART i.e. the last pre-defined particle
- as of today: AOMEGAC0 ($\overline{\Omega_c^0}$)

The manual

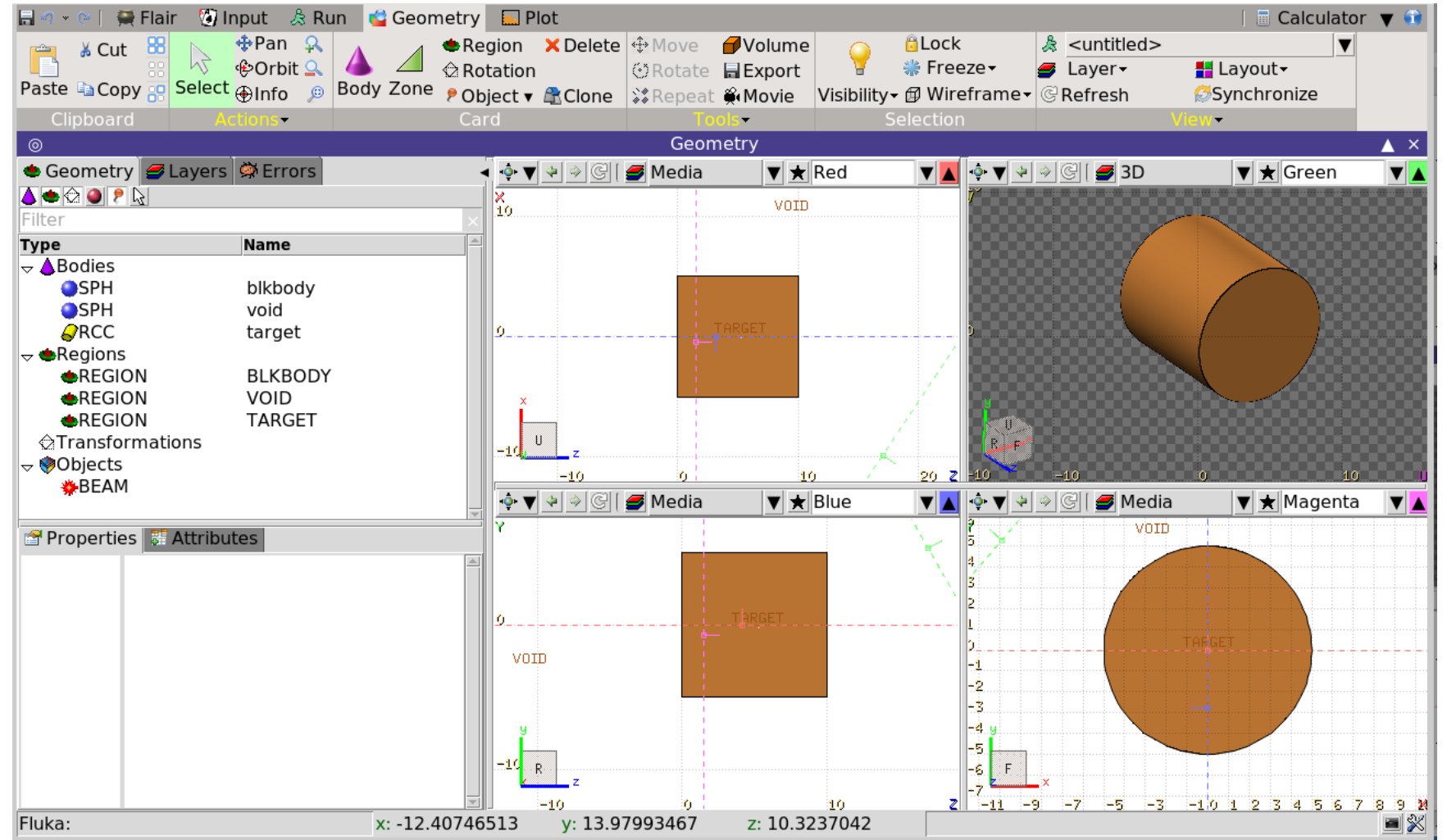
- Can be accessed using F1 button
- Can be accessed clicking on the “info” button



- The manual will open on the relevant page
- The manual is also available on the Fluka web page www.fluka.cern

Geometry tab – 1

- Visualize and edit geometry
- Plot results
- Dedicated lectures



Geometry tab – 2

- Viewports automatically refreshed when input is changed

Layout management

The screenshot shows the FLUKA Geometry tab interface. On the left, there are three red arrows pointing to specific areas: 'Filters' points to the 'Geometry', 'Layers', and 'Errors' tabs; 'Objects Listbox' points to a tree view of objects; and 'Properties & Attributes Listbox' points to a table of properties for the selected 'target' object. The main area contains three viewports: a 3D perspective view (top right), a top-down view (top left), and a side view (bottom left). A 'Layout' dropdown menu is highlighted in the top right corner of the software window.

Filters

Objects Listbox

Type	Name
SPH	blkbody
SPH	void
RCC	target
REGION	BLKBODY
REGION	VOID
REGION	TARGET
BEAM	

Properties	Attributes
name	target
comment	Cylindrical target
type	RCC
x	0.0
y	0.0
z	0.0
Hx	0.0
Hy	0.0
Hz	10.0
R	5.0
@xmid	0.0
@ymid	0.0

Geometry tab – 3

The screenshot displays the FLUKA Geometry tab interface. The top toolbar contains various tools such as Cut, Paste, Select, Pan, Orbit, Info, Body Zone, Region, Delete, Move, Volume, Rotate, Export, Lock, Freeze, Wireframe, Refresh, Synchronize, and Trans. The left sidebar shows a tree view of the geometry with 'RCC target' selected. The bottom left shows the properties of the selected object.

name	target
comment	Cylindrical target
type	RCC
x	0.0
y	0.0
z	0.0
Hx	0.0
Hy	0.0
H _z	10.0
R	5.0
@xmid	0.0
@ymid	0.0

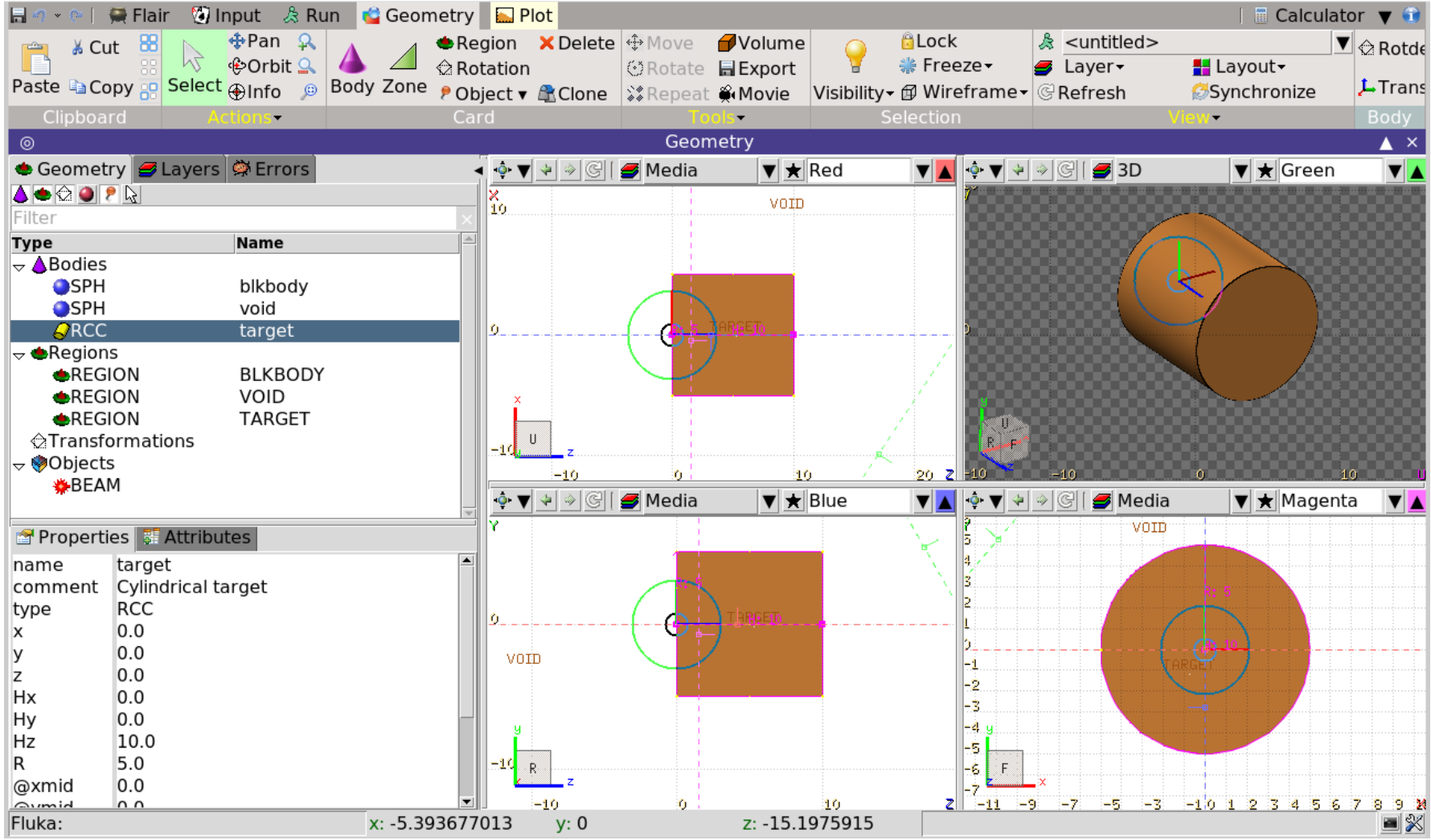
The main area is divided into four viewports, each with its own toolbar and color selection dropdown (highlighted with red boxes):

- Red viewport:** Shows a top-down view of the cylindrical target in a red background.
- Green viewport:** Shows a 3D perspective view of the cylindrical target in a green background.
- Blue viewport:** Shows a top-down view of the cylindrical target in a blue background.
- Magenta viewport:** Shows a side view of the cylindrical target in a magenta background.

At the bottom of the interface, the coordinates are displayed: $x: -5.393677013$, $y: 0$, $z: -15.1975915$.

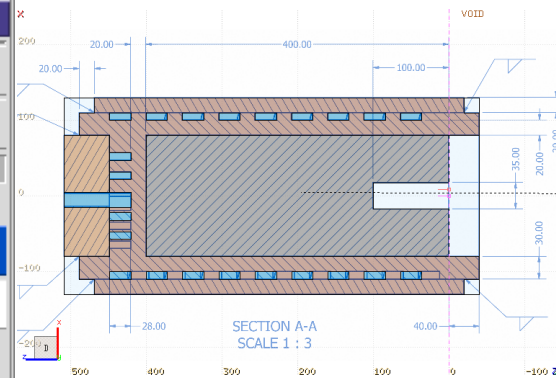
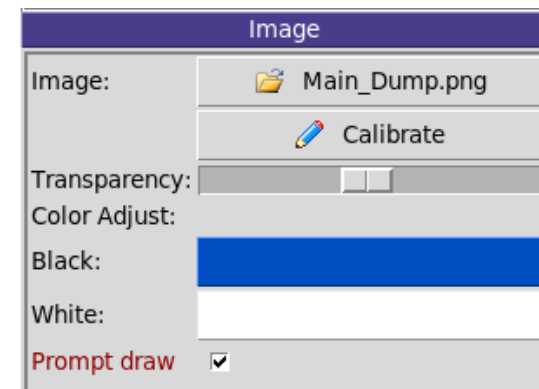
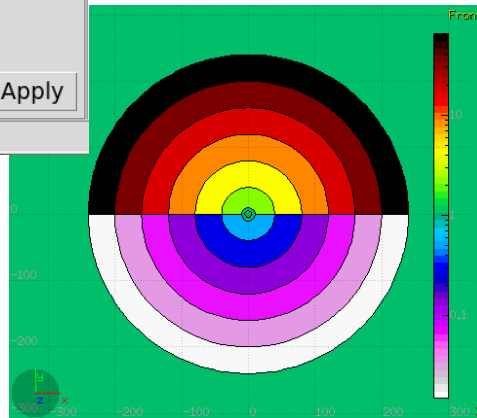
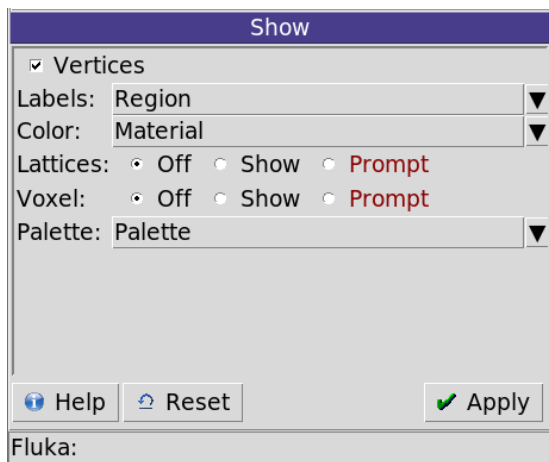
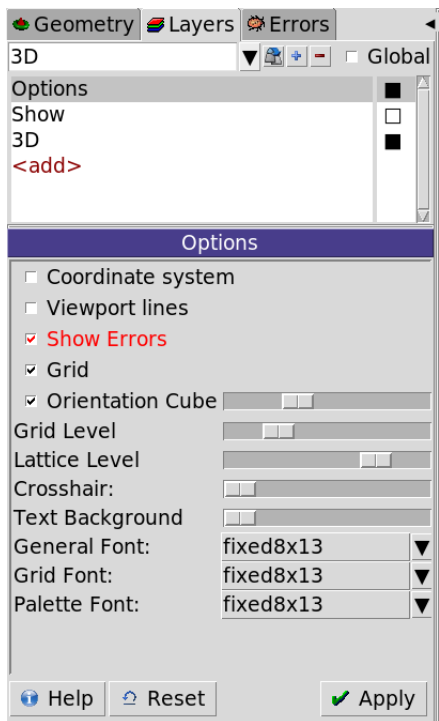
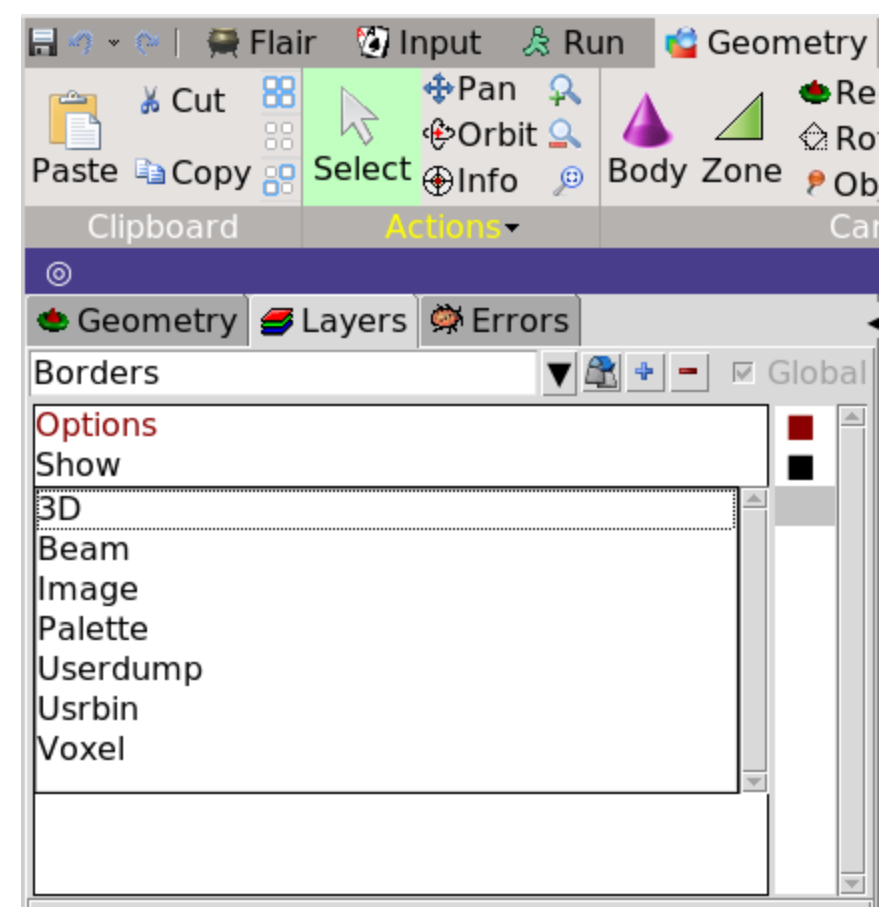
Geometry tab – 4

- Possible to navigate with mouse and keyboard (see dedicated lecture)



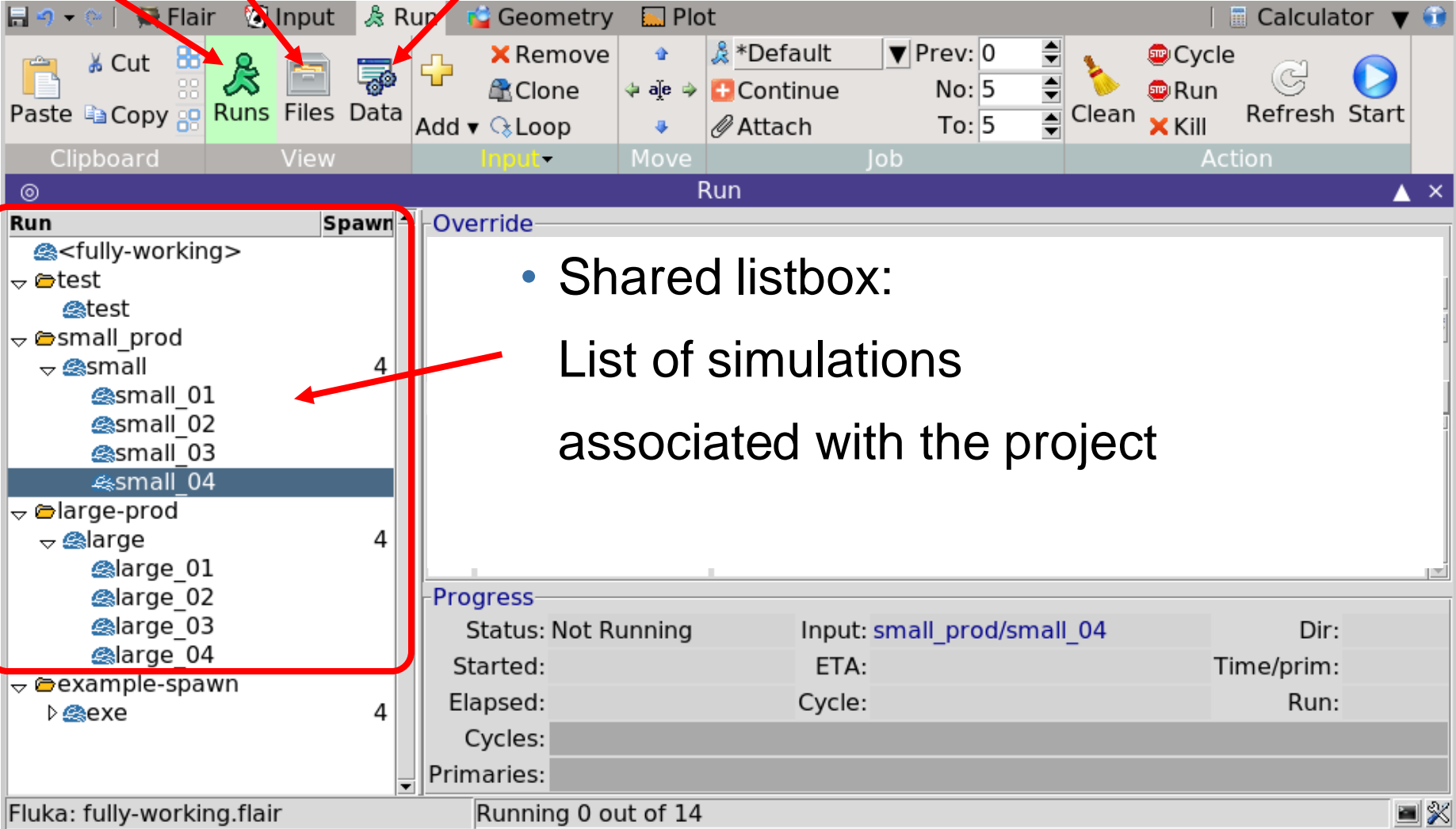
Geometry tab – 5

- Possible to add layers for better visualization:
 - Appearance (fonts, etc.)
 - Scoring (see Scoring-1 lecture)
 - Special quantities (e.g. region importance)
 - Background images (to help building geometry)



Run tab

- 3 views: “Runs”, “Files”, and “Data”

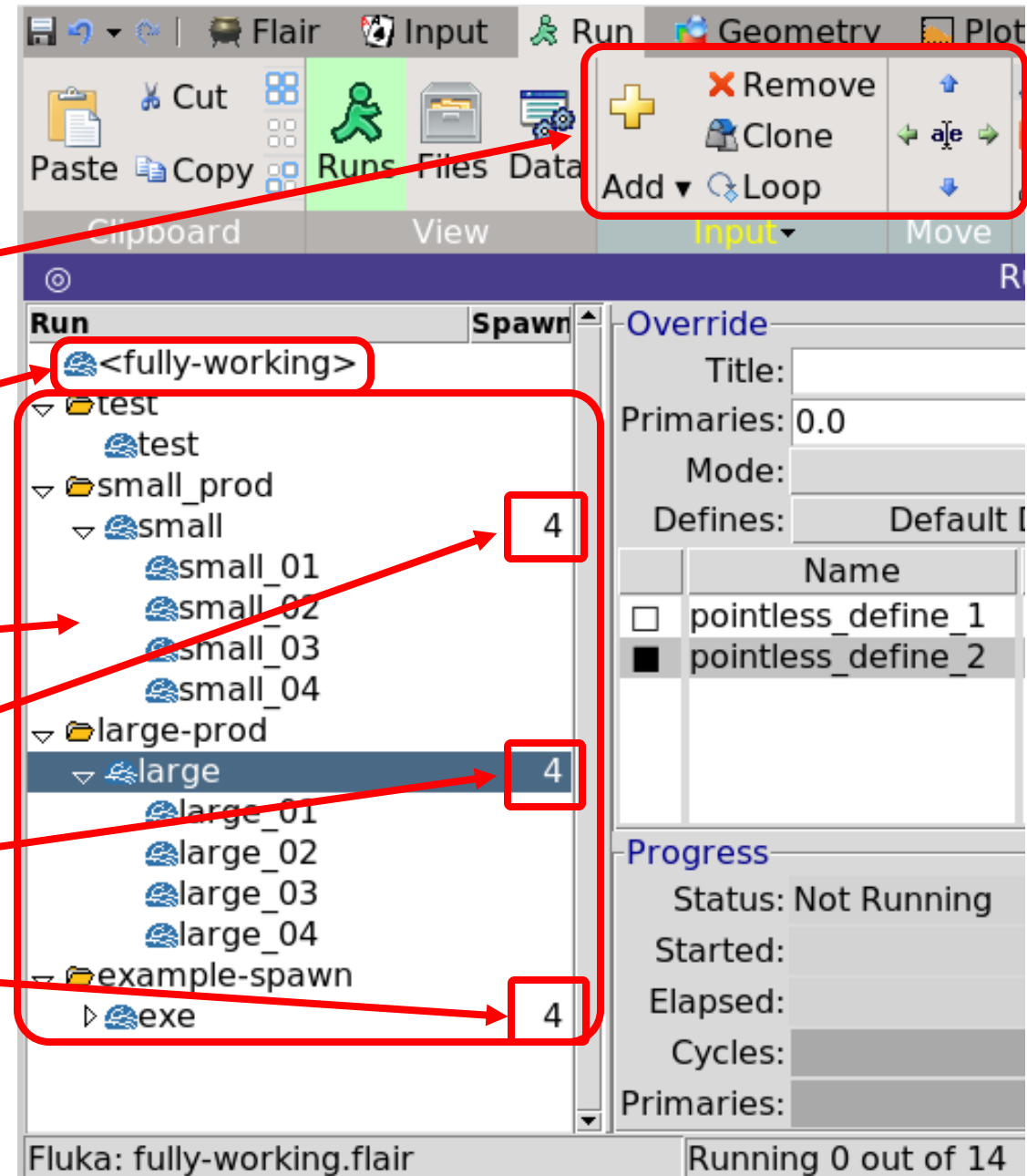


The screenshot shows the Fluka Run tab interface. The top toolbar includes icons for 'Runs', 'Files', and 'Data', which are highlighted with red arrows. Below the toolbar is a tree view of simulation runs, also highlighted with a red box. The tree view shows a hierarchy of folders and simulation instances, with a 'Spawn' column on the right. A red arrow points from the text 'Shared listbox: List of simulations associated with the project' to the tree view. Below the tree view is a 'Progress' table with columns for Status, Input, Dir, Started, Elapsed, Cycles, Primaries, ETA, Cycle, Time/prim, and Run.

Status	Input	Dir	Started	Elapsed	Cycles	Primaries	ETA	Cycle	Time/prim	Run
Not Running	small_prod/small_04									

Run tab – Runs view – 1

- Management of the various simulations
- Basic inputfile of the Flair project
- Different simulations associated with the Flair project
- Number of spawns



Run tab – Runs view – 2

- Override of inputs

- Number of primaries

- Executable

- #define

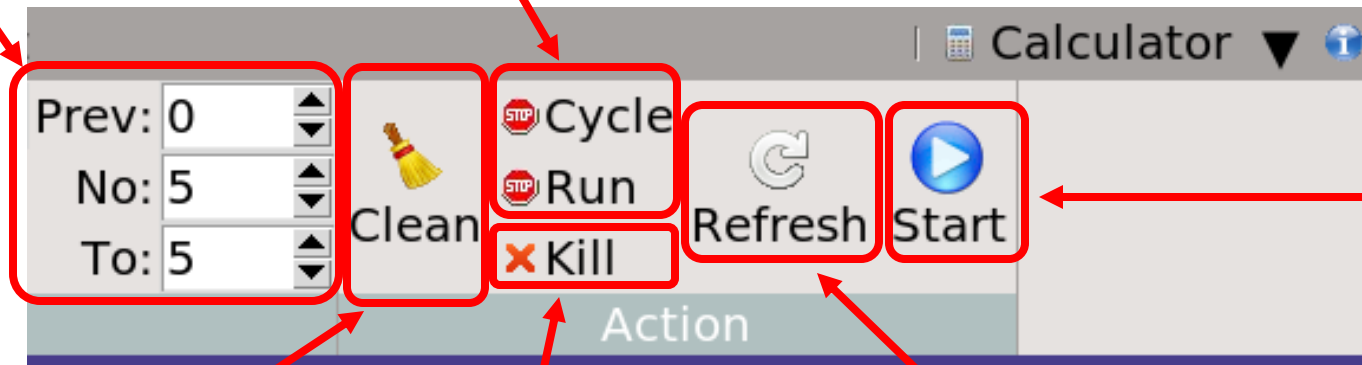
Dedicated lecture

Defines:		Default Defines
	Name	Value
<input type="checkbox"/>	pointless_define_1	
<input checked="" type="checkbox"/>	pointless_define_2	10

Run tab – Runs view – 3

Cycles control: how many cycles to run, starting from which cycle

Cleanly stop the cycles/runs currently running



Start a simulation

Remove files from a previous simulation

Refresh the progress field

Kill the current simulations

Run tab – Runs view – 4

The screenshot shows the 'Run' window in FLUKA. On the left is a tree view with folders like 'test', 'small_prod', 'large-prod', and 'example-spawn'. The 'small' folder under 'small_prod' is expanded, showing 'small_01' through 'small_04'. The 'small_01' job is selected and highlighted. The main area shows the 'Progress' for this job. It includes a table of statistics and two progress bars. Red arrows point from text labels to specific elements in the interface.

Status: Running	Input: small_prod/small_01	Dir: fluka_31164
Started: 2023.10.25 16:00:22	ETA: 2023.10.25 16:13:37	Time/prim: 197 ms
Elapsed: 15.8 s	Cycle: 3m 1s	Run: 12m 56s
Cycles:	Current: 2 [5] Completed: 20%	
Primaries:	Current: 81 [1000] Completed: 8%	

• Overall job progress bar

• Cycle progress bar

• Status

• Time since the start of the cycle

• Input file actually running

• Estimated end of the simulation

• Time until the end of the cycle

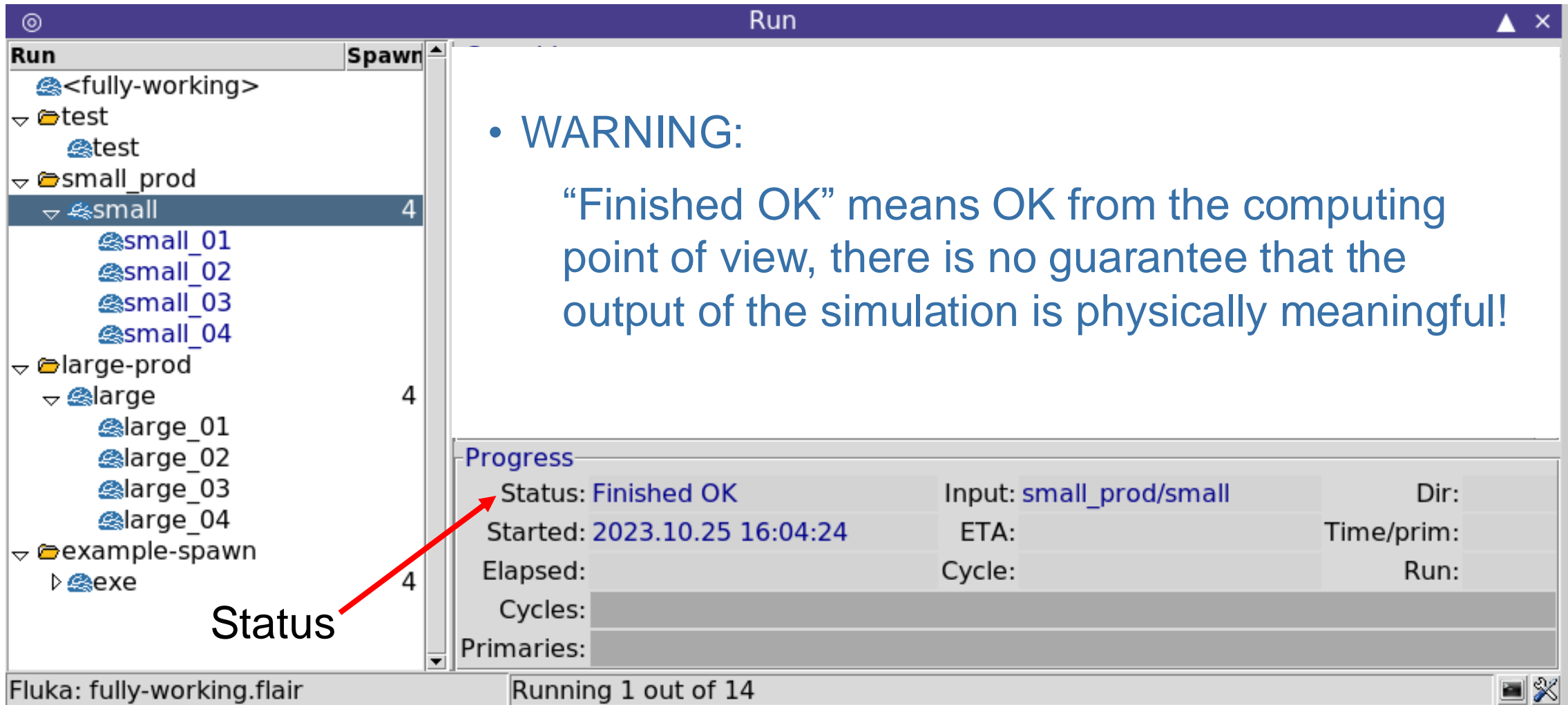
• Time per primary

• Time until the end of the simulation

• Temporary directory

Run tab – Runs view – 5

- At the end of the simulations...



The screenshot shows the 'Run' tab in the Flair software interface. On the left, a tree view displays the hierarchy of simulation runs. The 'small' directory under 'small_prod' is selected, showing four sub-runs: 'small_01', 'small_02', 'small_03', and 'small_04'. The 'large-prod' directory also shows four sub-runs: 'large_01', 'large_02', 'large_03', and 'large_04'. The 'example-spawn' directory contains an 'exe' sub-run. The 'Status' label is positioned below the tree view, with a red arrow pointing to the 'Status: Finished OK' entry in the progress table.

• **WARNING:**

“Finished OK” means OK from the computing point of view, there is no guarantee that the output of the simulation is physically meaningful!

Progress			
Status: Finished OK	Input: small_prod/small	Dir:	
Started: 2023.10.25 16:04:24	ETA:	Time/prim:	
Elapsed:	Cycle:	Run:	
Cycles:			
Primaries:			

Fluka: fully-working.flair Running 1 out of 14

After running – 1

- Content of the working directory

```
fluka_user:/home/fluka_user$  
fluka_user:/home/fluka_user$ ls  
fully-working.flair  fully-working.inp  my.exe  small_prod  tutorial.flair  
fluka_user:/home/fluka_user$  
fluka_user:/home/fluka_user$ _
```

- Content of the working sub-directory

```
fluka_user:/home/fluka_user$ cd small_prod/  
fluka_user:/home/fluka_user/small_prod$ ls  
ransmall_01001  ransmall_04005      small_01004_fort.21  small_02003_fort.21  small_03002_fort.21  small_04001_fort.21  
ransmall_01002  ransmall_04006      small_01004_fort.22  small_02003_fort.22  small_03002_fort.22  small_04001_fort.22  
ransmall_01003  small_01.inp        small_01005.err      small_02004.err      small_03003.err      small_04002.err  
ransmall_01004  small_01.out        small_01005.log      small_02004.log      small_03003.log      small_04002.log  
ransmall_01005  small_01001.err     small_01005.out      small_02004.out      small_03003.out      small_04002.out  
ransmall_01006  small_01001.log     small_01005_fort.21  small_02004_fort.21  small_03003_fort.21  small_04002_fort.21  
ransmall_02001  small_01001.out     small_01005_fort.22  small_02004_fort.22  small_03003_fort.22  small_04002_fort.22  
ransmall_02002  small_01001_fort.21  small_02.inp         small_02005.err      small_03004.err      small_04003.err  
ransmall_02003  small_01001_fort.22  small_02.out         small_02005.log      small_03004.log      small_04003.log  
ransmall_02004  small_01002.err     small_02001.err      small_02005.out      small_03004.out      small_04003.out  
ransmall_02005  small_01002.log     small_02001.log      small_02005_fort.21  small_03004_fort.21  small_04003_fort.21  
ransmall_02006  small_01002.out     small_02001.out      small_02005_fort.22  small_03004_fort.22  small_04003_fort.22  
ransmall_03001  small_01002_fort.21  small_02001_fort.21  small_03.inp         small_03005.err      small_04004.err  
ransmall_03002  small_01002_fort.22  small_02001_fort.22  small_03.out         small_03005.log      small_04004.log  
ransmall_03003  small_01003.err     small_02002.err      small_03001.err      small_03005.out      small_04004.out  
ransmall_03004  small_01003.log     small_02002.log      small_03001.log      small_03005_fort.21  small_04004_fort.21  
ransmall_03005  small_01003.out     small_02002.out      small_03001.out      small_03005_fort.22  small_04004_fort.22  
ransmall_03006  small_01003_fort.21  small_02002_fort.21  small_03001_fort.21  small_04.inp         small_04005.err  
ransmall_04001  small_01003_fort.22  small_02002_fort.22  small_03001_fort.22  small_04.out         small_04005.log  
ransmall_04002  small_01004.err     small_02003.err      small_03002.err      small_04001.err      small_04005.out  
ransmall_04003  small_01004.log     small_02003.log      small_03002.log      small_04001.log      small_04005_fort.21  
ransmall_04004  small_01004.out     small_02003.out      small_03002.out      small_04001.out      small_04005_fort.22  
fluka_user:/home/fluka_user/small_prod$ _
```

After running – 2

- Content of the working directory

```
fluka_user:/home/fluka_user$  
fluka_user:/home/fluka_user$ ls  
fully-working.flair  fully-working.inp  my.exe  small_prod  tutorial.flair  
fluka_user:/home/fluka_user$  
fluka_user:/home/fluka_user$ .
```

- Content of the working sub-directory

```
fluka_user:/home/fluka_user$ cd small_prod/  
fluka_user:/home/fluka_user/small_prod$ ls  
ransmall_01001  ransmall_04005  small_01004_fort.21  small_02003_fort.21  small_03002_fort.21  small_04001_fort.21  
ransmall_01002  ransmall_04006  small_01004_fort.22  small_02003_fort.22  small_03002_fort.22  small_04001_fort.22  
ransmall_01003  small_01.inp      small_01005.err      small_02004.      err  
ransmall_01004  small_01.out     small_01005.log      small_02004.      log  
ransmall_01005  small_01001.err  small_01005.out      small_02004.      out  
ransmall_01006  small_01001.log  small_01005_fort.21  small_02004.      fort.21  
ransmall_02001  small_01001.out  small_01005_fort.22  small_02004.      fort.22  
ransmall_02002  small_01001_fort.21  small_02005.err      small_03004.err      small_04003.err  
ransmall_02003  small_01001_fort.22  small_02005.log      small_03004.log      small_04003.log  
ransmall_02004  small_01002.err  small_02001.err      small_02005.out      small_03004.out      small_04003.out  
ransmall_02005  small_01002.log  small_02001.log      small_02005_fort.21  small_03004_fort.21  small_04003_fort.21  
ransmall_02006  small_01002.out  small_02001.out      small_02005_fort.22  small_03004_fort.22  small_04003_fort.22  
ransmall_03001  small_01002_fort.21  small_02001_fort.21  small_03005.err      small_04004.err  
ransmall_03002  small_01002_fort.22  small_02001_fort.22  small_03005.log      small_04004.log  
ransmall_03003  small_01003.err  small_02002.err      small_03005.out      small_04004.out  
ransmall_03004  small_01003.log  small_02002.log      small_03005_fort.21  small_04004_fort.21  
ransmall_03005  small_01003.out  small_02002.out      small_03005_fort.22  small_04004_fort.22  
ransmall_03006  small_01003_fort.21  small_02002_fort.21  small_03001_fort.21  small_04005.err  
ransmall_04001  small_01003_fort.22  small_02002_fort.22  small_03001_fort.22  small_04005.log  
ransmall_04002  small_01004.err  small_02003.err      small_03002.err      small_04005.out  
ransmall_04003  small_01004.log  small_02003.log      small_03002.log      small_04005_fort.21  
ransmall_04004  small_01004.out  small_02003.out      small_03002.out      small_04005_fort.22  
fluka_user:/home/fluka_user/small_prod$ .
```

.inp and .out files specific of each spawn

Run tab – Files view – 1

- Generated files accessible via the Files view

Cycles	File	Type	Size	Date
001	small_prod/ransmall_01001	-file-	1651	2023.10.25 16:13:54
002	small_prod/small_01001_fort.21	21	242	2023.10.25 16:13:57
003	small_prod/small_01001_fort.22	22	242	2023.10.25 16:13:57
004	small_prod/small_01001.err	Error	714	2023.10.25 16:13:57
005	small_prod/small_01001.log	Log	0	2023.10.25 16:13:53
006	small_prod/small_01.out	Output	2193	2023.10.25 16:14:13
compile	small_prod/small_01001.out	Output	56913	2023.10.25 16:13:57
data				
input				
plot				
temporary				

Fluka: fully-working.flair Files: 0 Total Size: 0

Run tab – Files view – 2

- File per each cycle:
 - one (1) fluka .out file & one (1) flair .out file
 - one (1) .log file
 - one (1) .err file
 - one (1) random seed file
 - one (1) scoring file per each logical unit scoring used

Cycles	File	Type▲	Size	Date
001	small_prod/ransmall_01001	-file-	1651	2023.10.25 16:13:54
002	small_prod/small_01001_fort.21	21	242	2023.10.25 16:13:57
003	small_prod/small_01001_fort.22	22	242	2023.10.25 16:13:57
004	small_prod/small_01001.err	Error	714	2023.10.25 16:13:57
005	small_prod/small_01001.log	Log	0	2023.10.25 16:13:53
006	small_prod/small_01.out	Output	2193	2023.10.25 16:14:13
compile	small_prod/small_01001.out	Output	56913	2023.10.25 16:13:57
data				
input				
plot				
temporary				

Run tab – Files view – 3

- Naming convention for file names; the filename contains:
 - the name of the run, e.g.: `small`
 - The spawn identifier, e.g.: `01`
 - The cycle identifier, e.g.: `001`
 - The file type identifier, e.g.: `.err` , `fort.21` , `ran`

Cycles	File	Type▲	Size	Date
001	small_prod/ransmall_01001	-file-	1651	2023.10.25 16:13:54
002	small_prod/small_01001_fort.21	21	242	2023.10.25 16:13:57
003	small_prod/small_01001_fort.22	22	242	2023.10.25 16:13:57
004	small_prod/small_01001.err	Error	714	2023.10.25 16:13:57
005	small_prod/small_01001.log	Log	0	2023.10.25 16:13:53
006	small_prod/small_01.out	Output	2193	2023.10.25 16:14:13
compile	small_prod/small_01001.out	Output	56913	2023.10.25 16:13:57
data				
input				
plot				
temporary				

Run tab – Files view – 4

- Naming convention for file names; the filename contains:
 - the name of the run, e.g.: `small`
 - The spawn identifier, e.g.: `01`
 - The cycle identifier, e.g.: `001`
 - The file type identifier, e.g.: `.err` , `fort.21` , `ran`
- In this example 7 files were generated:

`small_01001.err`

`small_01001.log`

`small_01001.out`

`ransmall_01001`

`small_01001_fort.21`

`small_01001_fort.22`

`small_01.out`

Run tab – Files view – 4

- Naming convention for file names; the filename contains:
 - the name of the run, e.g.: `small`
 - The spawn identifier, e.g.: `01`
 - The cycle identifier, e.g.: `001`
 - The file type identifier, e.g.: `.err` , `fort.21` , `ran`

- In this example 7 files were generated:

`small_01001.err`

`small_01001.log`

`small_01001.out`

`ransmall_01001`

`small_01001_fort.21`

`small_01001_fort.22`

`small_01.out`

← renaming is planned

Run tab – Files view – 5

- Spawn 1 Cycle 1

Cycles	File	Type▲	Size	
001	small_prod/ransmall_01001	-file-	1651	2
002	small_prod/small_01001_fort.21	21	242	2
003	small_prod/small_01001_fort.22	22	242	2
004	small_prod/small_01001.err	Error	714	2
005	small_prod/small_01001.log	Log	0	2
006	small_prod/small_01.out	Output	2193	2
compile	small_prod/small_01001.out	Output	56913	2
data				
input				
plot				
temporary				

- Spawn 1 Cycle 5

Cycles	File	Type▲	Size	
001	small_prod/ransmall_01005	-file-	1651	2023.1
002	small_prod/small_01005_fort.21	21	242	2023.1
003	small_prod/small_01005_fort.22	22	242	2023.1
004	small_prod/small_01005.err	Error	714	2023.1
005	small_prod/small_01005.log	Log	0	2023.1
006	small_prod/small_01005.out	Output	81778	2023.1
compile	small_prod/small_01.out	Output	2193	2023.1
data				
input				
plot				
temporary				

Run tab – Files view – 6

- Spawn 1 Cycle 1

Cycles	File	Type▲	Size	
001	small_prod/ransmall_01001	-file-	1651	2
002	small_prod/small_01001_fort.21	21	242	2
003	small_prod/small_01001_fort.22	22	242	2
004	small_prod/small_01001.err	Error	714	2
005	small_prod/small_01001.log	Log	0	2
006	small_prod/small_01.out	Output	2193	2
compile	small_prod/small_01001.out	Output	56913	2
data				
input				
plot				
temporary				

- Spawn 2 Cycle 1

Cycles	File	Type▲	Size	
001	small_prod/ransmall_02001	-file-	1651	2
002	small_prod/small_02001_fort.21	21	242	2
003	small_prod/small_02001_fort.22	22	242	2
004	small_prod/small_02001.err	Error	714	2
005	small_prod/small_02001.log	Log	0	2
006	small_prod/small_02001.out	Output	81901	2
compile	small_prod/small_02.out	Output	2193	2
data				
input				
plot				
temporary				

Run tab – Files view – 7

- Spawn 1 Cycle 1

Cycles	File	Type▲	Size	
001	small_prod/ransmall_01001	-file-	1651	2
002	small_prod/small_01001_fort.21	21	242	2
003	small_prod/small_01001_fort.22	22	242	2
004	small_prod/small_01001.err	Error	714	2
005	small_prod/small_01001.log	Log	0	2
006	small_prod/small_01.out	Output	2193	2
compile	small_prod/small_01001.out	Output	56913	2
data				
input				
plot				
temporary				

- Spawn 1 Cycle 6

Cycles	File	Type▲	Size	
001	small_prod/ransmall_02006	-file-	1651	202
002	small_prod/small_02.out	Output	2193	202
003				
004				
005				
006				
compile				
data				
input				
plot				
temporary				

- Random file for the next cycle is generated

Run tab – Data view – 1

- All the generated files need to be merged to be analyzed

```
small_01001_fort.21  
small_01002_fort.21  
small_01003_fort.21  
small_01004_fort.21`  
small_01005_fort.21
```

```
small_02001_fort.21  
small_02002_fort.21  
small_02003_fort.21  
small_02004_fort.21  
small_02005_fort.21
```

```
small_03001_fort.21  
small_03002_fort.21  
small_03003_fort.21  
small_03004_fort.21  
small_03005_fort.21
```

```
small_04001_fort.21  
small_04002_fort.21  
small_04003_fort.21  
small_04004_fort.21  
small_04005_fort.21
```

Run tab – Data view – 2

- Flair automatically identifies the logical units used from the input file

Run	Type	Output	Name/Unit
small_prod/small	usrbin	small_prod/small_21.bnn	21
small_prod/small	usrbin	small_prod/small_22.bnn	22

File	Type	Size	Date

Run tab – Data view – 3

- Flair finds all the corresponding file (per spawn and per cycle)

The screenshot shows the Flair software interface. The top menu bar includes 'Flair', 'Input', 'Run', 'Geometry', and 'Plot'. Below the menu is a toolbar with icons for 'Cut', 'Paste', 'Copy', 'Runs', 'Files', 'Data', 'Scan', 'Clone', 'Remove', 'Refresh', 'Add', 'Remove', 'Filter', 'Clean Process', and 'Action'. The main window is titled 'Run' and contains a tree view on the left and a data table on the right.

Tree View (Left):

- <fully-working>
 - test
 - test
 - small_prod
 - small (4)
 - small_01
 - small_02
 - small_03
 - small_04
 - large-prod (4)
 - large (4)
 - large_01
 - large_02
 - large_03
 - large_04
 - example-spawn (4)
 - exe

Data Table (Right):

Run	Type	Output	Name/Unit
small_prod/small	usrbin	small_prod/small_21.bnn	21
small_prod/small	usrbin	small_prod/small_22.bnn	22

Files Table (Bottom Right):

File	Type	Size	Date
small_prod/small_01001_fort.21	21	242	2023.10.25 16:13:57
small_prod/small_01001_fort.22	22	242	2023.10.25 16:13:57
small_prod/small_01002_fort.21	21	242	2023.10.25 16:14:01
small_prod/small_01002_fort.22	22	242	2023.10.25 16:14:01
small_prod/small_01003_fort.21	21	242	2023.10.25 16:14:05

Fluka: fully-working.flair | Files: 20

Run tab – Data view – 4

- Process can be forced by hand:

- 1-Select the run
- 2-Refresh
- 3-Scan
- 4-Process (merge)

- Processed binary results files are generated (specific extensions: **.bnn**, **.bnx**, **.rnc**, etc. more in other lectures)

The screenshot shows the Flair software interface with the Run tab selected. The interface is divided into several sections:

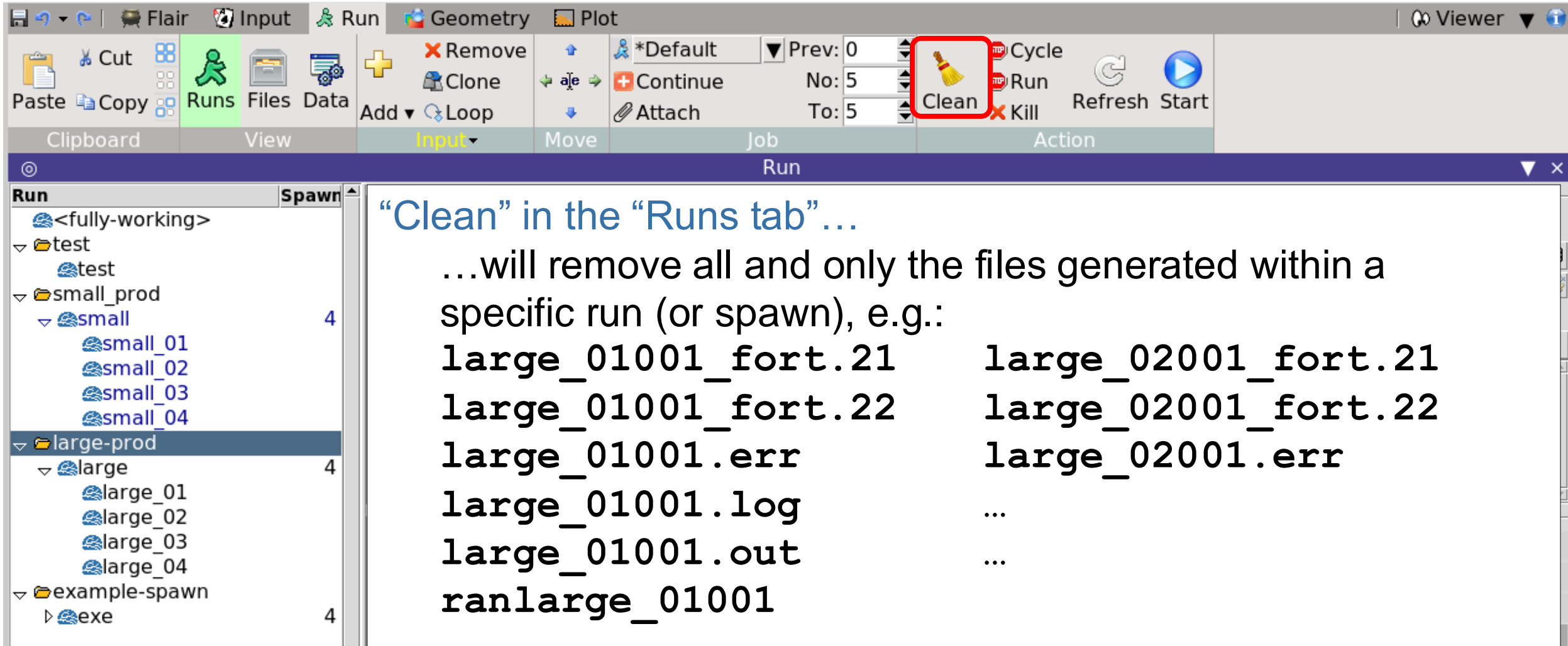
- Toolbar:** Contains buttons for 'Scan' (3), 'Refresh' (2), and 'Process' (4). The 'Data' tab is also visible.
- Run Tree:** A hierarchical tree view on the left. The 'small' run under 'small_prod' is selected and highlighted with a red box and the number 1.
- Detectors Table:** A table showing detector information for the selected run.
- Files Table:** A table showing the generated files for the selected run.

Run	Type	Output
small_prod/small	usrbin	small_prod/small_21.bnn
small_prod/small	usrbin	small_prod/small_22.bnn

File	Type	
small_prod/small_01001_fort.21	21	242
small_prod/small_01001_fort.22	22	242
small_prod/small_01002_fort.21	21	242
small_prod/small_01002_fort.22	22	242
small_prod/small_01003_fort.21	21	242

Run tab – Cleaning – 1

- Removing files generated for the cycles and merge files are different actions!



The screenshot shows the FLUKA software interface. The top toolbar includes buttons for 'Clean' (a broom icon), 'Cycle', 'Run', 'Kill', 'Refresh', and 'Start'. The 'Clean' button is highlighted with a red box. Below the toolbar, the 'Run' window is open, showing a tree view of runs and a list of files to be cleaned.

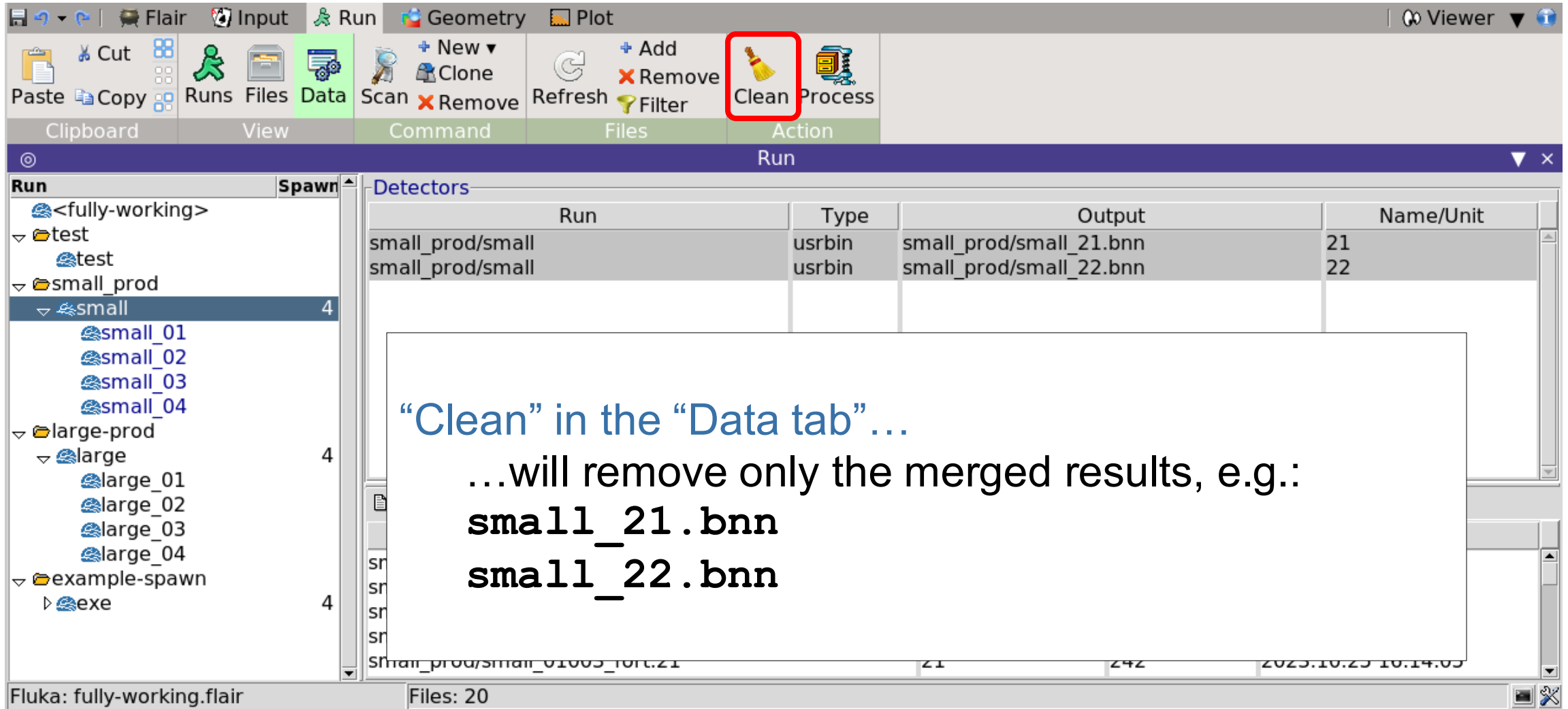
“Clean” in the “Runs tab” ...

...will remove all and only the files generated within a specific run (or spawn), e.g.:

large_01001_fort.21	large_02001_fort.21
large_01001_fort.22	large_02001_fort.22
large_01001.err	large_02001.err
large_01001.log	...
large_01001.out	...
ranlarge_01001	

Run tab – Cleaning – 2

- Removing files generated for the cycles and merge files are different actions!



The screenshot shows the Flair software interface. The 'Run' tab is active, and the 'Clean' button in the 'Data' section is highlighted with a red box. The 'Run' panel on the left shows a tree view of runs, with 'small' selected under 'small_prod'. The 'Detectors' table on the right shows the following data:

Run	Type	Output	Name/Unit
small_prod/small	usrbin	small_prod/small_21.bnn	21
small_prod/small	usrbin	small_prod/small_22.bnn	22

“Clean” in the “Data tab” ...
...will remove only the merged results, e.g.:
small_21.bnn
small_22.bnn

Compile tab

- Only very basic information is given here

- Routine to be compiled

- Add routines

- Select linker

- Build executable

The screenshot shows the FLUKA software interface with the 'Compile' tab active. The interface includes a menu bar with options like 'Flair', 'Input', 'Geometry', 'Run', 'Plot', 'Compile', and 'Output'. Below the menu bar is a toolbar with various icons and buttons. A red box highlights the 'Add Database' button in the toolbar. Another red box highlights the linker dropdown menu, which is currently set to 'lfluka'. A third red box highlights the 'Build' button. A fourth red box highlights the file list in the 'Compile' window, which contains the following data:

File	Type	Size	Date
source.f	Fortran	9203	2020.08.31 15:16:21
mgdraw.f	Fortran	14783	2020.08.31 15:16:23
own_routine.cc	C++	24064	2020.08.31 15:17:07

- User routines are discussed in a dedicated lecture...

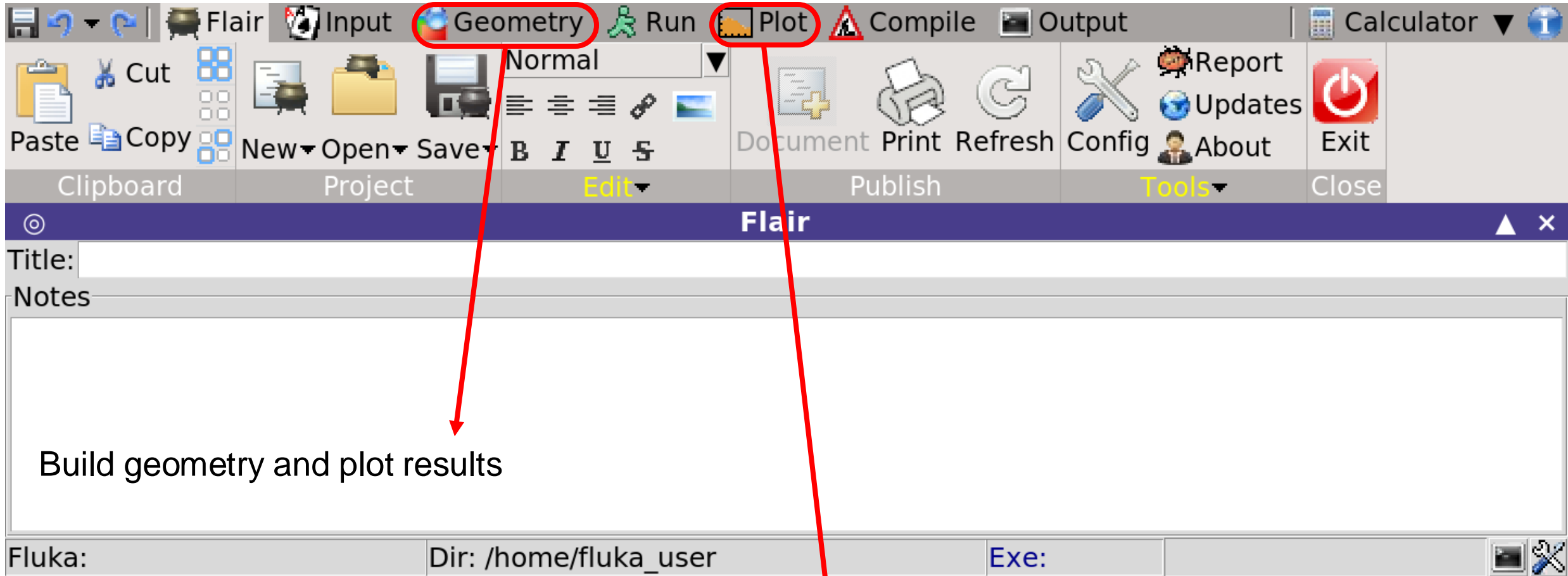
Do you remember slide 6?

The screenshot shows the Flair software interface. At the top, several menu items are circled in red: **Input**, **Geometry**, **Run**, **Plot**, **Compile**, and **Output**. Red arrows point from these menu items to descriptive text in the main window:

- Input** points to "Build input and geometry".
- Geometry** points to "Build geometry and plot results".
- Run** points to "Run and merge results".
- Plot** points to "Plot results".
- Compile** points to "Compile own executable".
- Output** points to "Visualize output files and messages".

The interface also includes a menu bar with options like Cut, Copy, Paste, New, Open, Save, Document, Print, Refresh, Config, Report, Updates, About, and Exit. The status bar at the bottom shows "Fluka: Dir: /home/fluka_user Exe:".

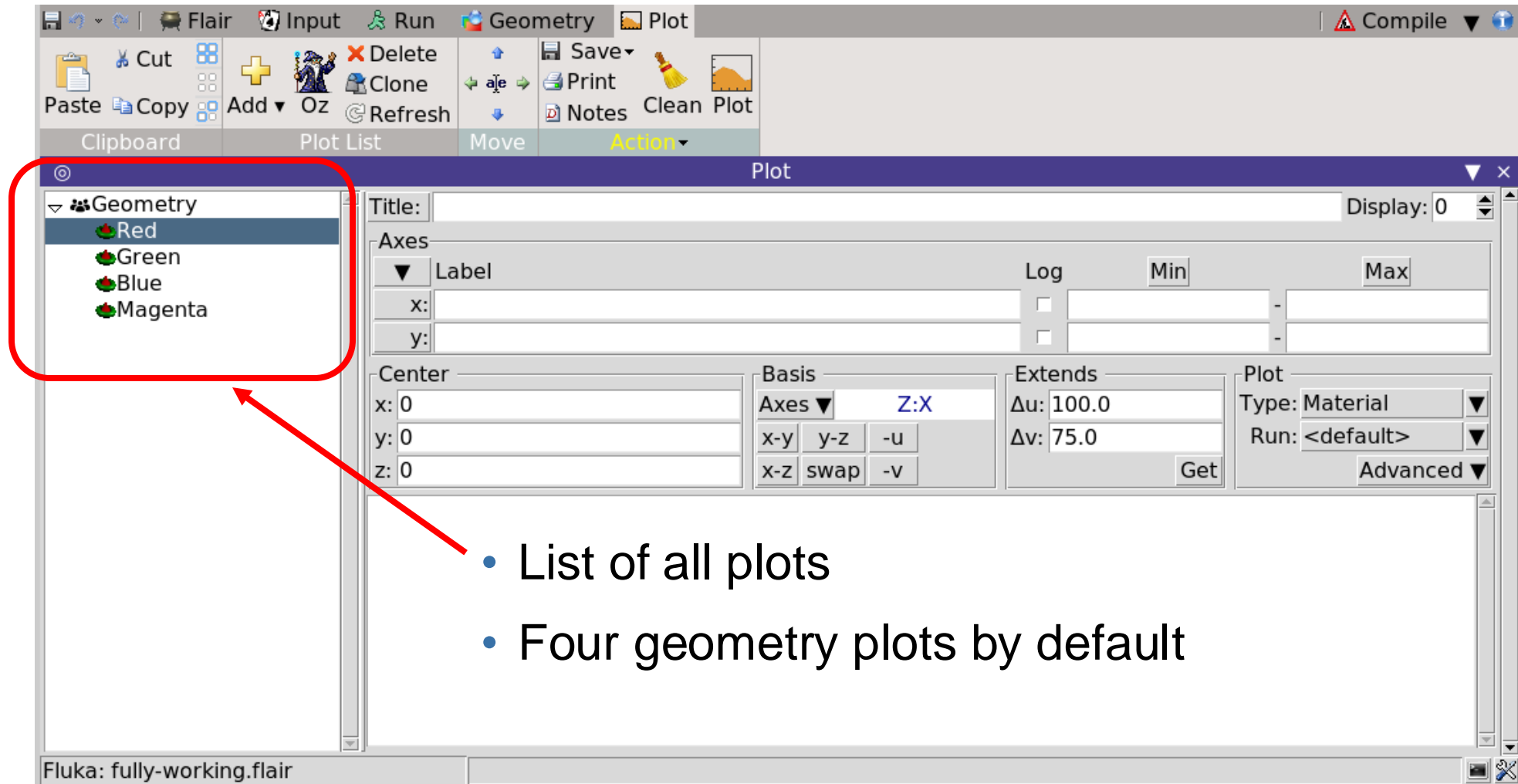
Do you remember slide 6?



Plot results

Plotting results in the Plot tab – 1

- Possible to plot geometry and all built-in scorings results



The screenshot shows the Flair software interface with the Plot tab active. The left sidebar displays a tree view under 'Geometry' with four sub-items: Red, Green, Blue, and Magenta. A red box highlights this list, and a red arrow points from it to the following list:

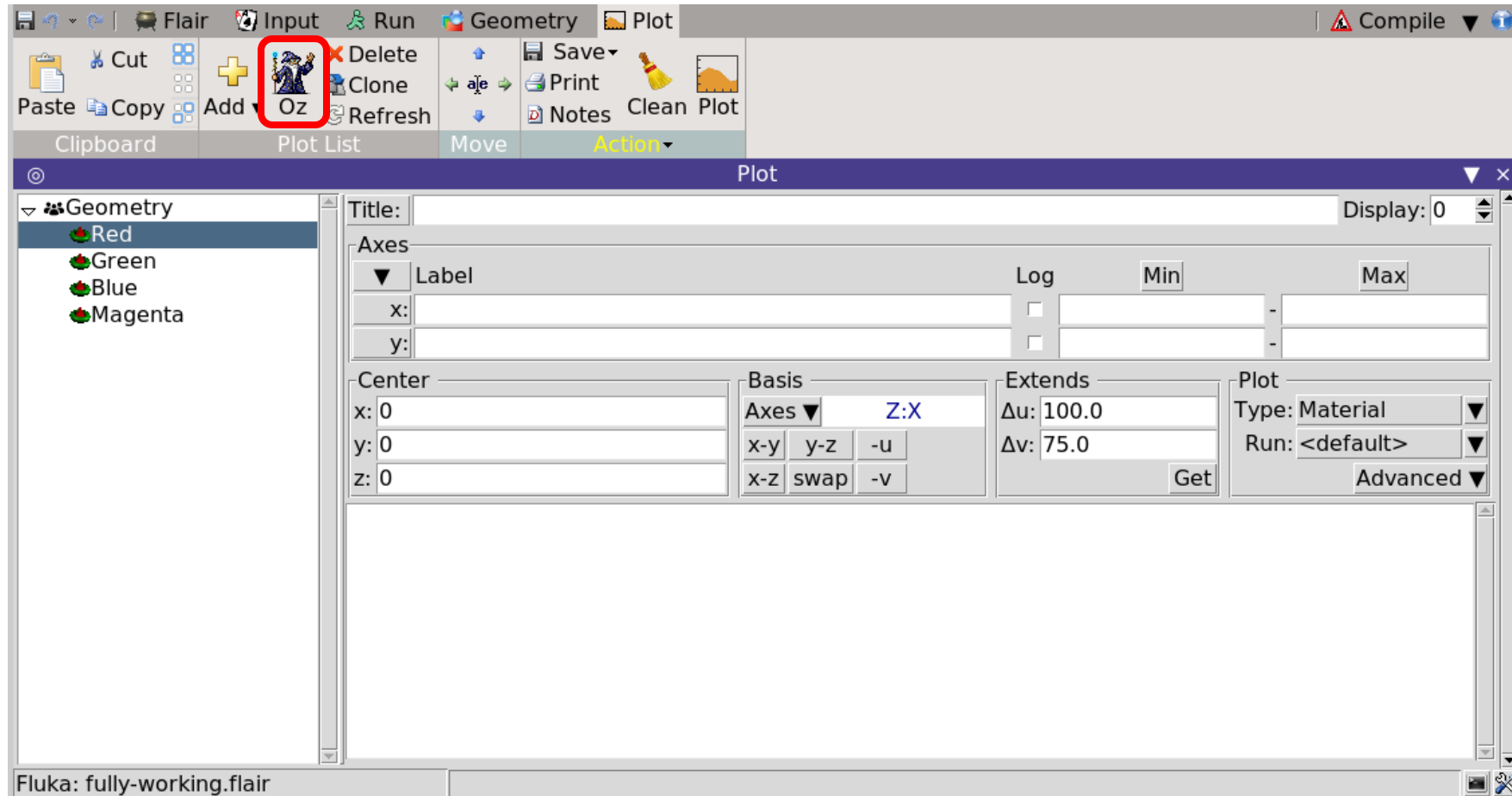
- List of all plots
- Four geometry plots by default

The main plot area shows a configuration panel with the following settings:

- Title: [empty]
- Display: 0
- Axes: x, y
- Center: x: 0, y: 0, z: 0
- Basis: Axes: Z:X, x-y, y-z, -u, x-z, swap, -v
- Extends: Δu : 100.0, Δv : 75.0
- Plot: Type: Material, Run: <default>

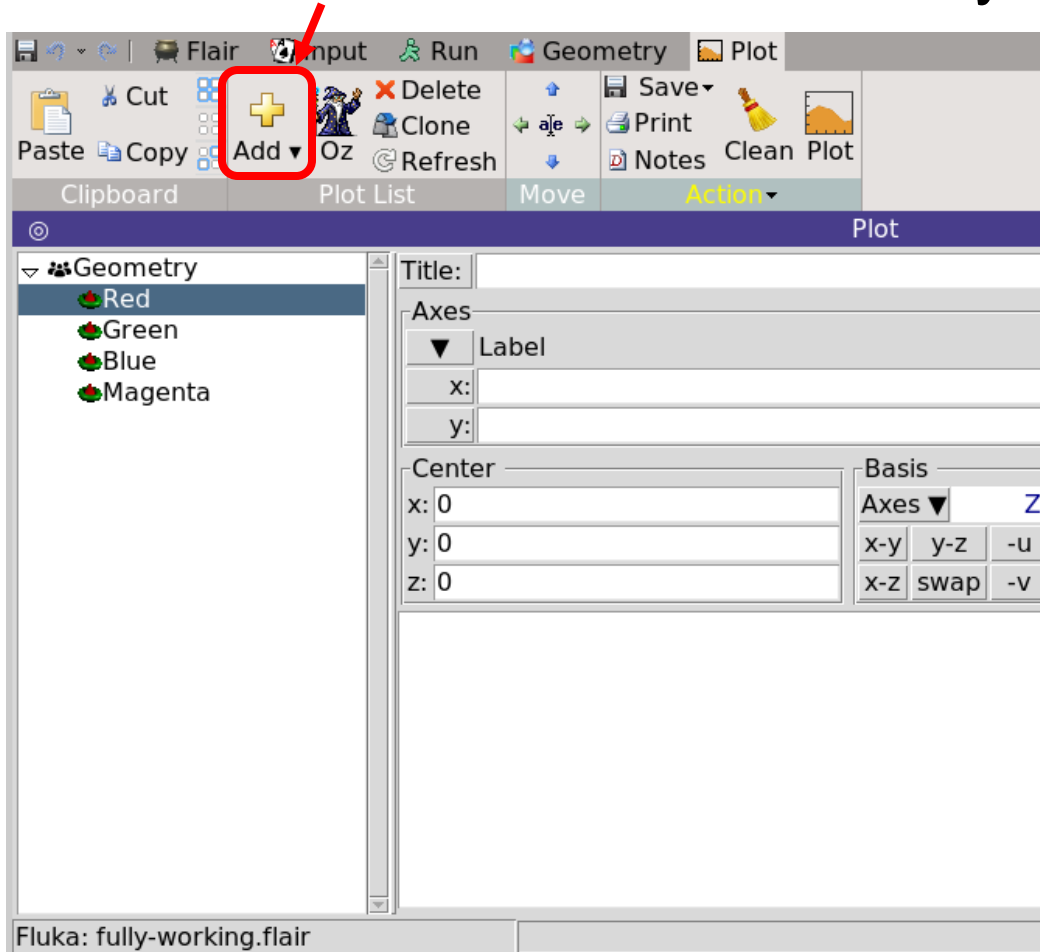
Plotting results in the Plot tab – 2

- It is possible to automatically generate the plots for all scorings in the input
- The program scans the input when “Oz” is invoked

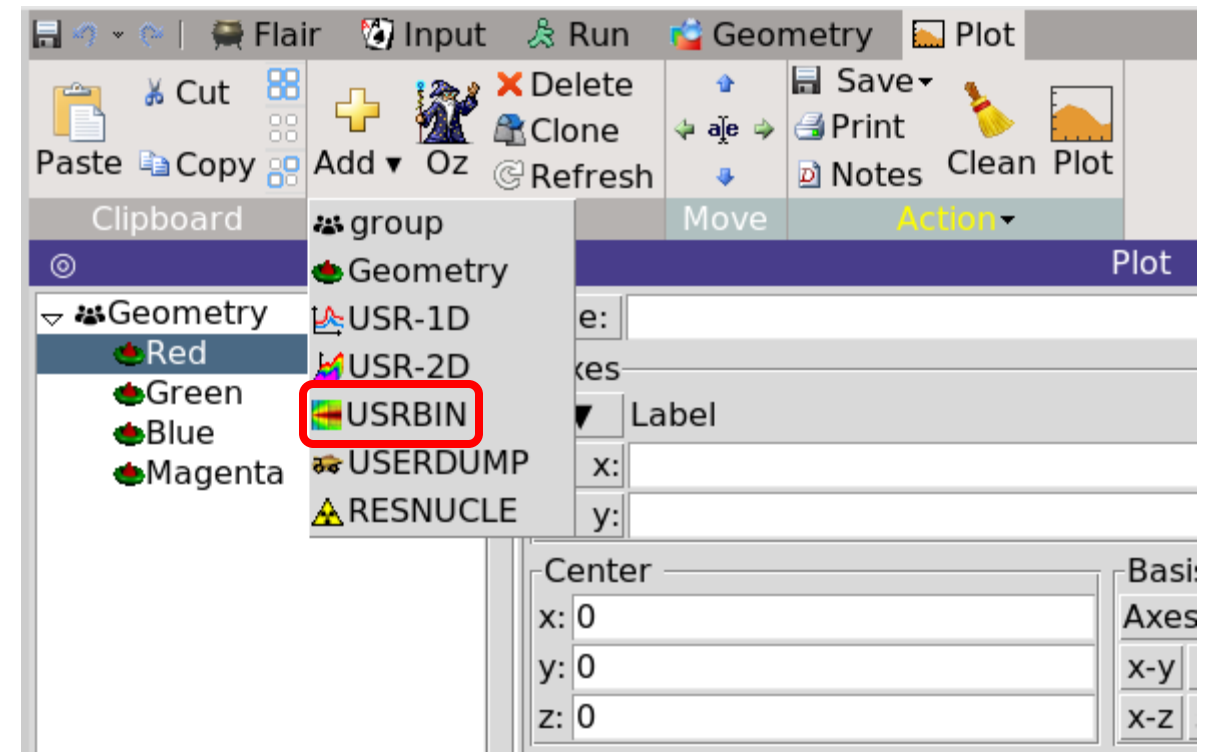


Plotting results in the Plot tab – 3

- It is possible to add plots by hand, one by one
- Click on “Add” and select the one you like from the pull down menu



(here, we'll see only USRBIN)

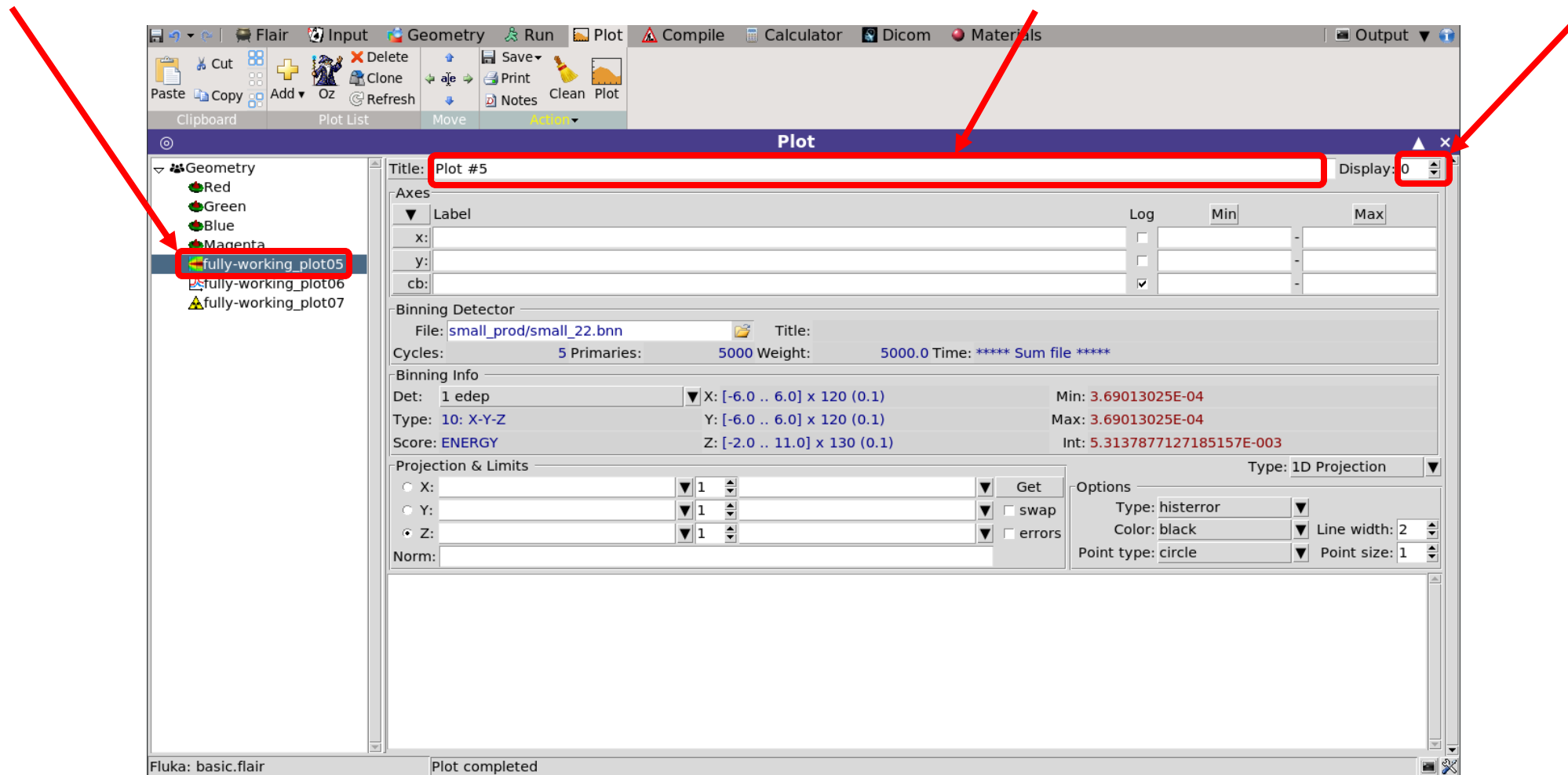


Plotting results in the Plot tab – 4

- Name of the file that will be saved

- Title of the plot

- Display ID



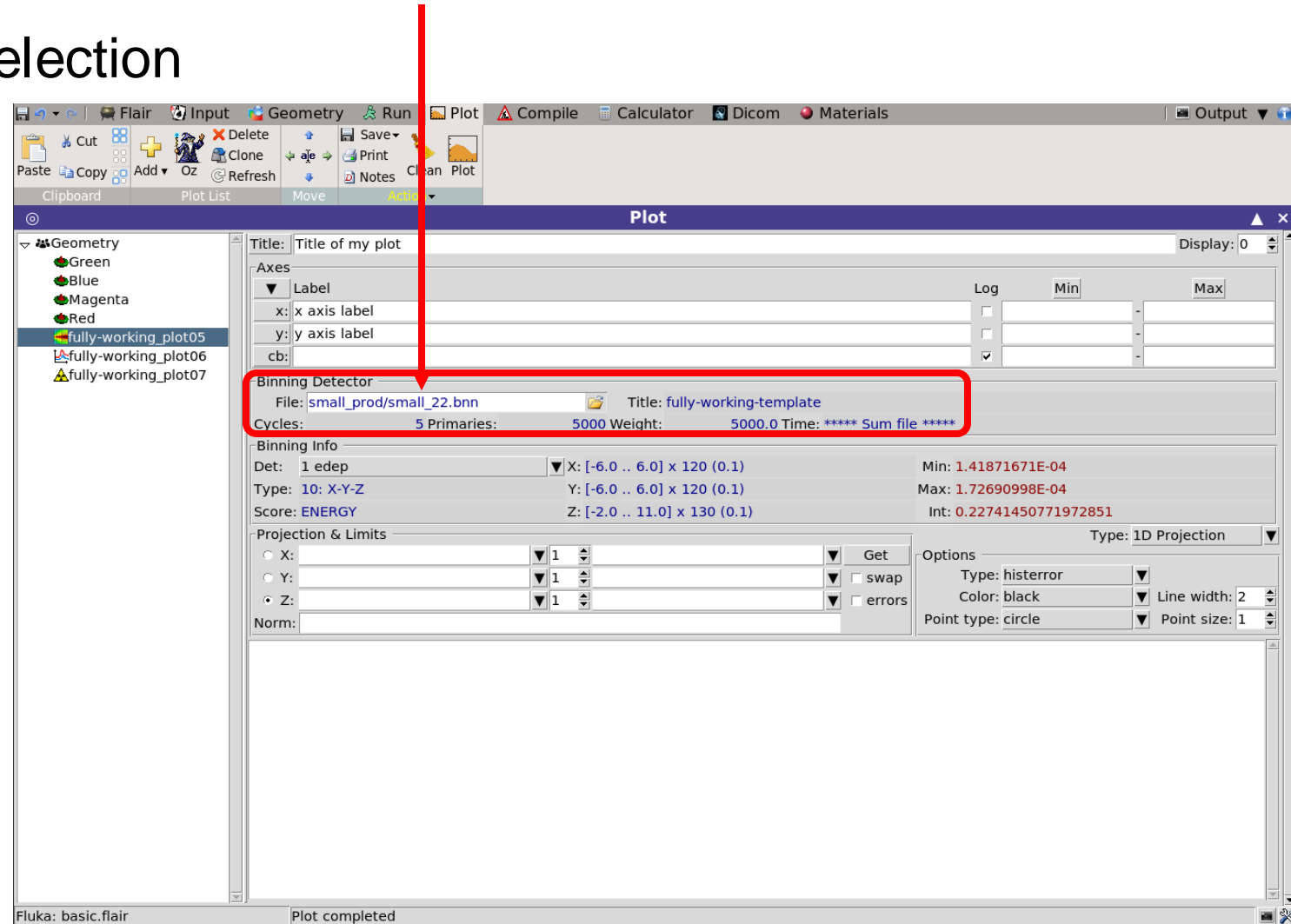
Plotting results in the Plot tab – 5

The screenshot shows the Flair software interface with the Plot tab active. The 'Options' section is highlighted with a red box, and the 'Display' dropdown is highlighted with a blue box. The 'Options' section includes settings for font, color, grid, aspect, and lines. The 'Axes' section includes settings for x and y axis labels, including font, color, and tics. The 'Binning Detector' section includes settings for file, title, and primaries. The 'Display' dropdown is set to 0. Two inset windows show the resulting plots: a rectangular heatmap on the left and a circular heatmap on the right.

- Plenty of options for plot customization

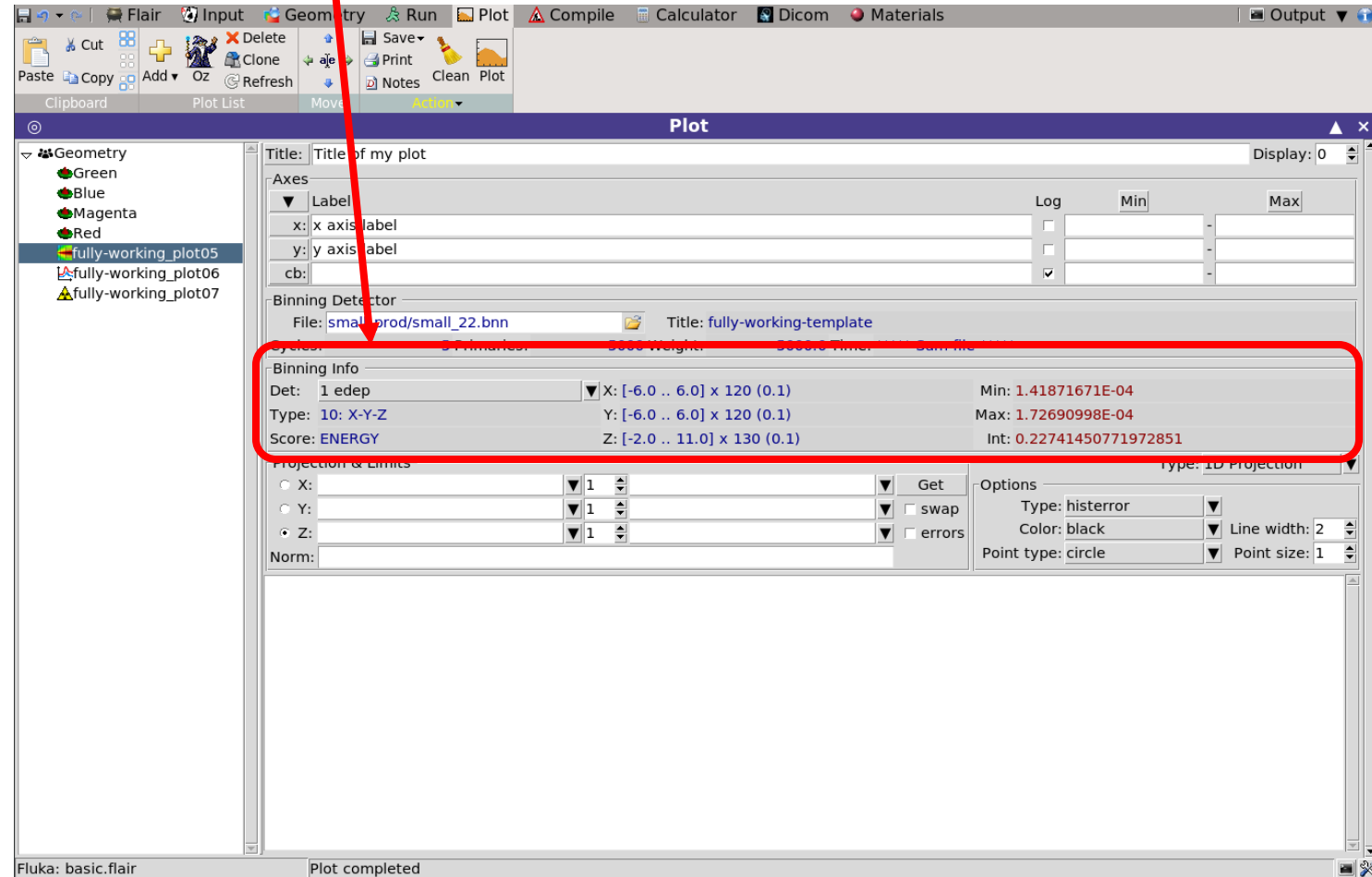
Plotting results in the Plot tab – 6

- Selection of the file containing the results of the simulations
- Opens standard pop-up for file selection
- Extra info available
 - #primaries
 - #cycles



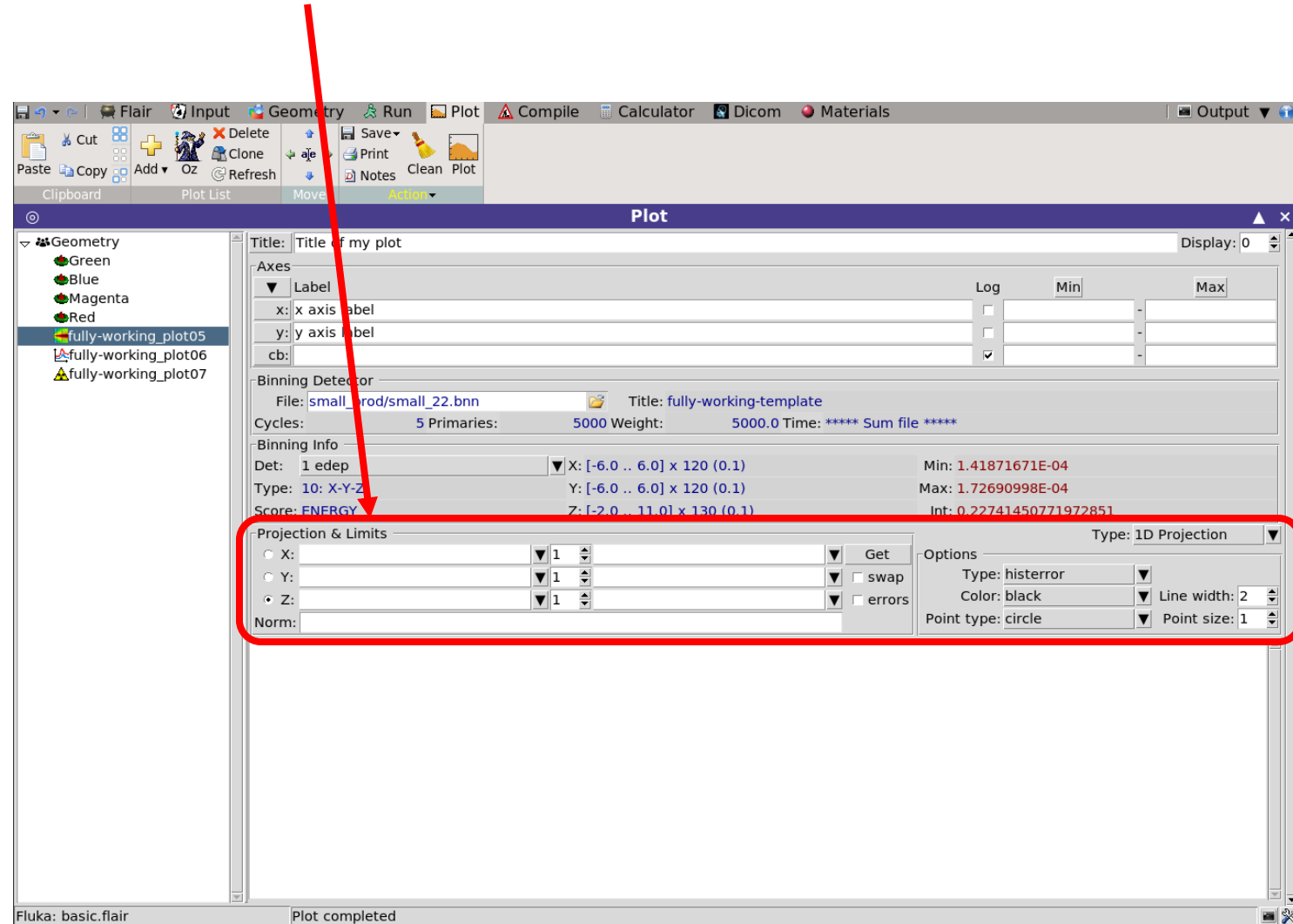
Plotting results in the Plot tab – 7

- Selection of the scoring within the chosen file (see scoring lecture)
- Standard pull-down menu
- Extra info available
 - Quantity scored
 - Type of mesh
 - Mesh details
 - Min & max values



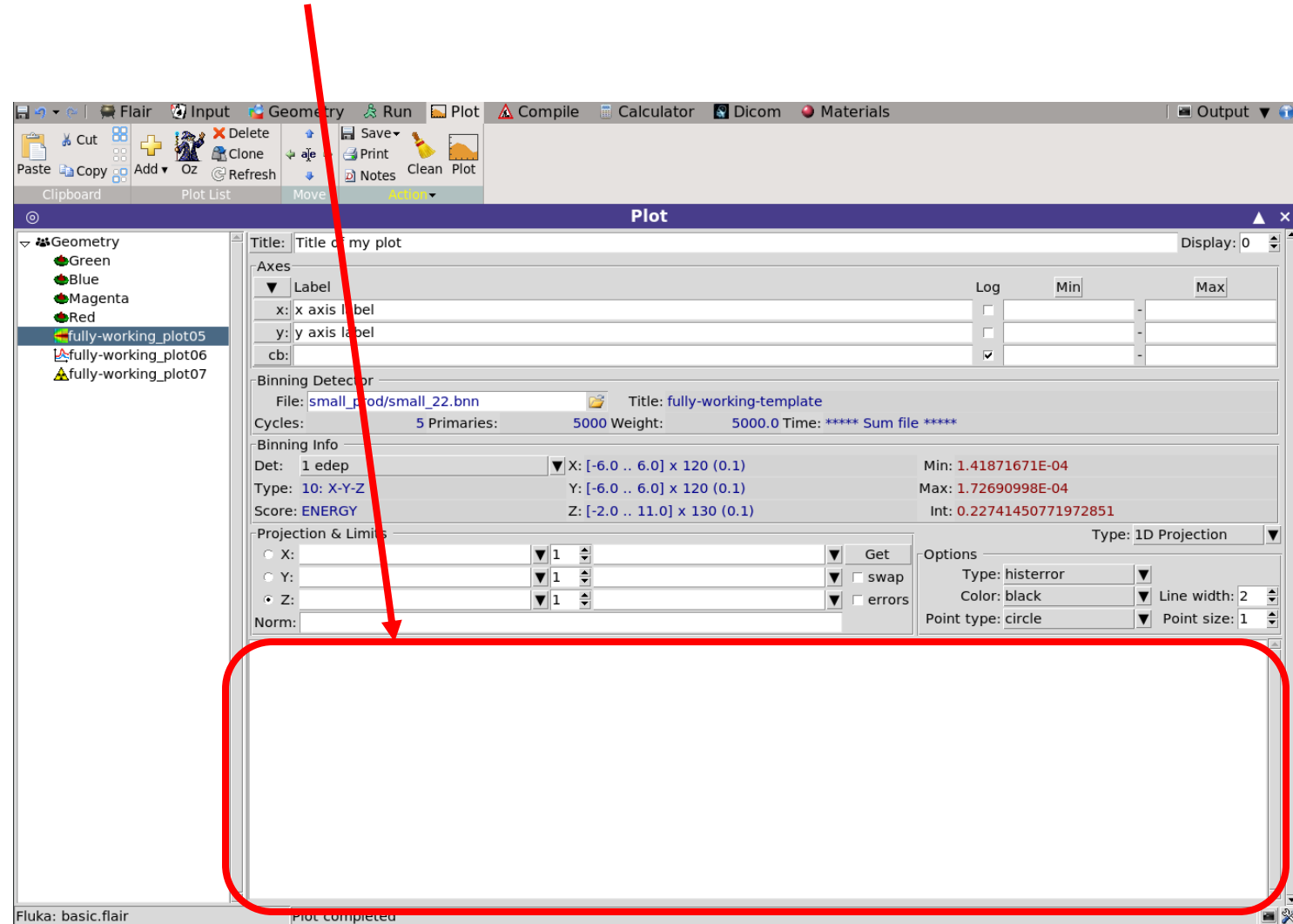
Plotting results in the Plot tab – 8

- Selection of plot type and options
 - 2D vs 1D projections
 - Plot extension
 - Uncertainty
 - Graphical options
 - Normalisation



Plotting results in the Plot tab – 9

- Additional plot customization
 - Gnuplot commands
 - Plot extents
 - Axis location
 - Label offsets
 - And much more....



Plotting results in the Plot tab – 10

Projection & Limits

X: 1 Get

Y: 1 swap

Z: 1 errors

Norm:

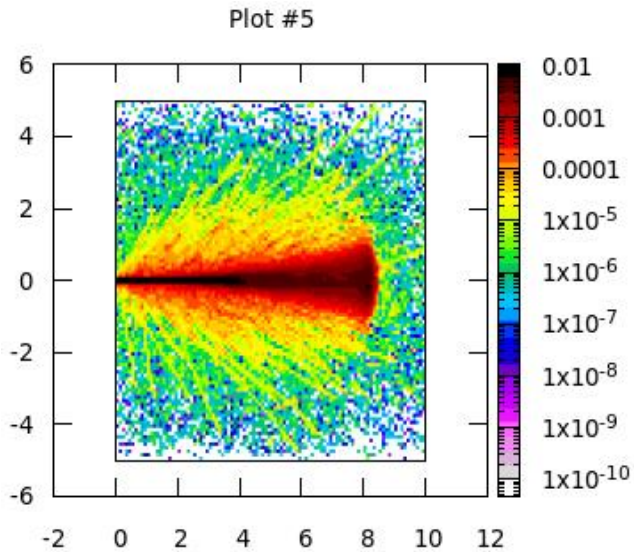
Type: 2D Projection

Geometry

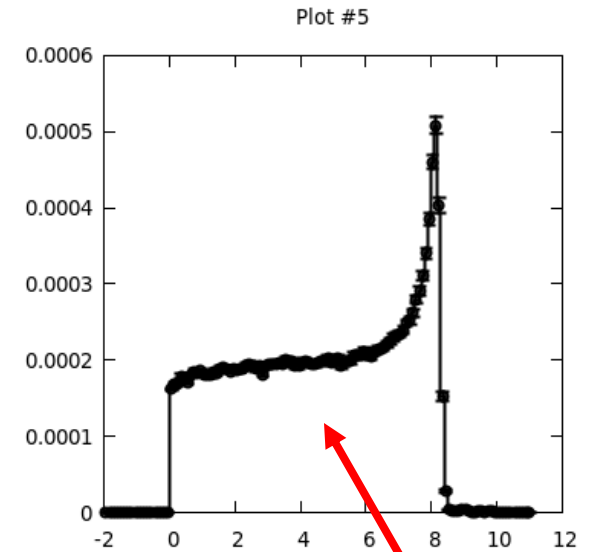
Use: -Auto-

Pos:

Axes: Auto



- 2D vs 1D projections



Projection & Limits

X: 1 Get

Y: 1 swap

Z: 1 errors

Norm:

Type: 1D Projection

Options

Type: histerror

Color: black Line width: 2

Point type: circle Point size: 1

Plotting results in the Plot tab – 11

Projection & Limits

X: ▼ 1 ▼ Get

Y: ▼ 1 ▼ swap

Z: ▼ 1 ▼ errors

Norm:

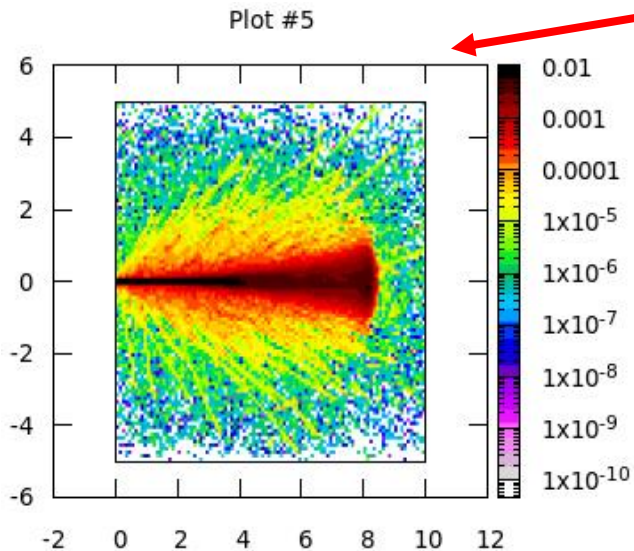
Type: 2D Projection ▼

Geometry

Use: -Auto- ▼

Pos:

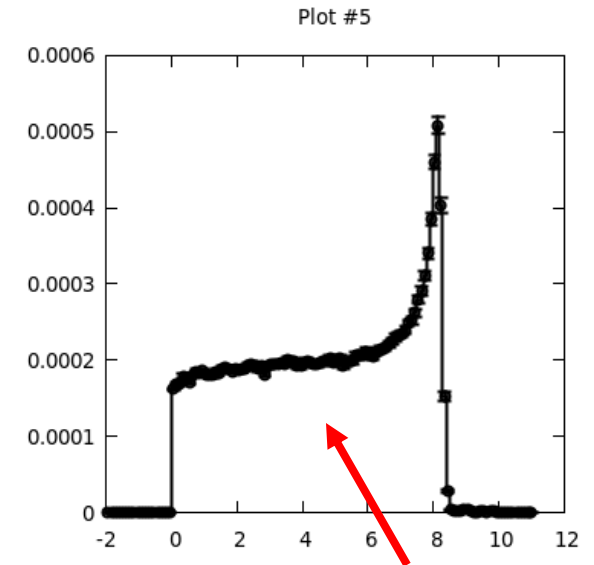
Axes: Auto ▼



- The result is averaged over the selected coordinate i.e. a X-Z plot averaged over the Y coordinate

Plotting results in the Plot tab – 12

- The result is projected along the selected coordinate and averaged over the non-selected coordinates i.e. a projection along the Z axis



Projection & Limits				Type: 1D Projection	
<input type="radio"/> X:	▼ 1 ▲	▼	Get	Options	
<input type="radio"/> Y:	▼ 1 ▲	▼	<input type="checkbox"/> swap	Type: histerror	▼
<input checked="" type="radio"/> Z:	▼ 1 ▲	▼	<input type="checkbox"/> errors	Color: black	▼ Line width: 2
Norm:				Point type: circle	▼ Point size: 1

Plotting results in the Plot tab – 13

Projection & Limits

X: 1

Y: -0.2

Z: 1

Norm:

Type: 2D Projection

Geometry

Use: -Auto-

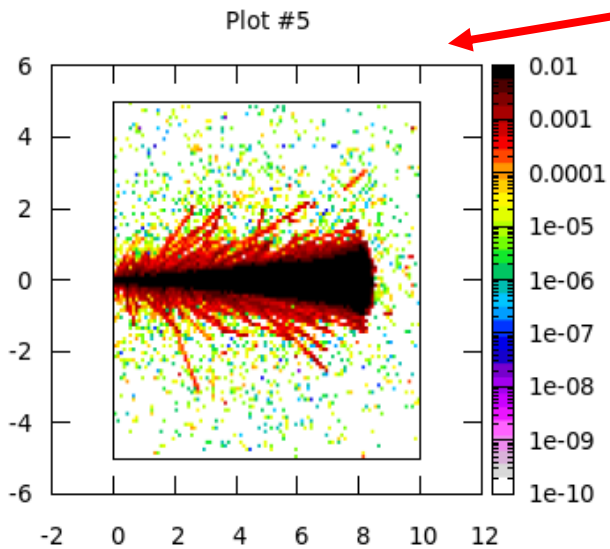
Pos:

Axes: Auto

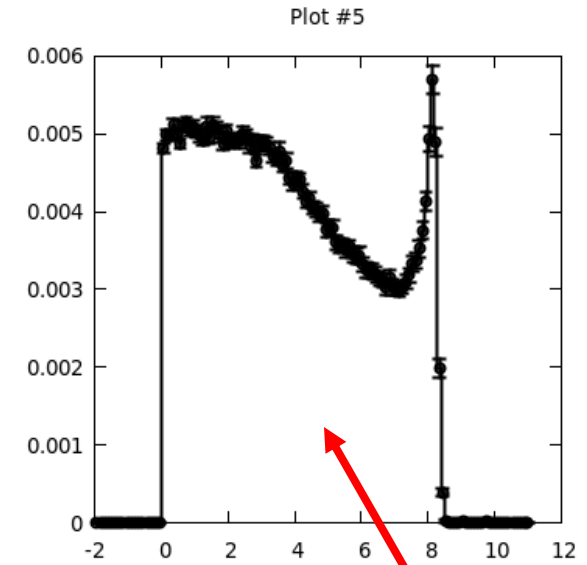
Get

swap

errors



- Plot extension
- The results is averaged only within the specified limits



Projection & Limits

X: 1

Y: -0.2

Z: 1

Norm:

Type: 1D Projection

Options

Type: histerror

Color: black

Point type: circle

Line width: 2

Point size: 1

Get

swap

errors

Plotting results in the Plot tab – 14

Projection & Limits

X: 1

Y: 1

Z: 1

Norm:

Type: 2D Projection

Geometry

Use: -Auto-

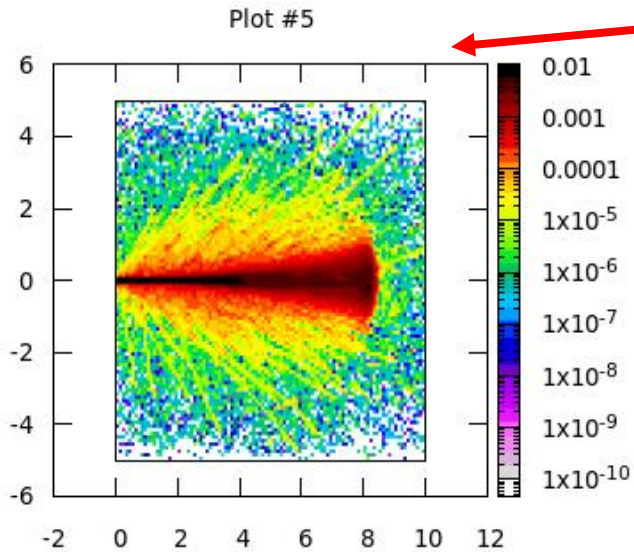
Pos:

Axes: Auto

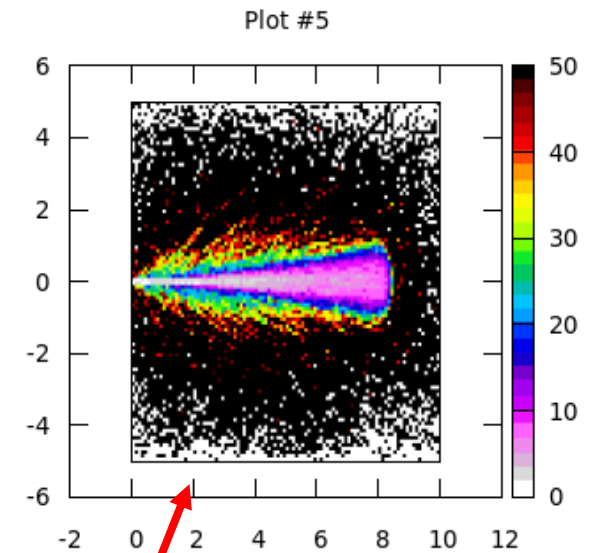
Get

swap

errors



- Uncertainty
- By ticking the “errors” box, it is possible to plot the statistical uncertainty in percent



Projection & Limits

X: 1

Y: 1

Z: 1

Norm:

Type: 2D Projection

Geometry

Use: -Auto-

Pos:

Axes: Auto

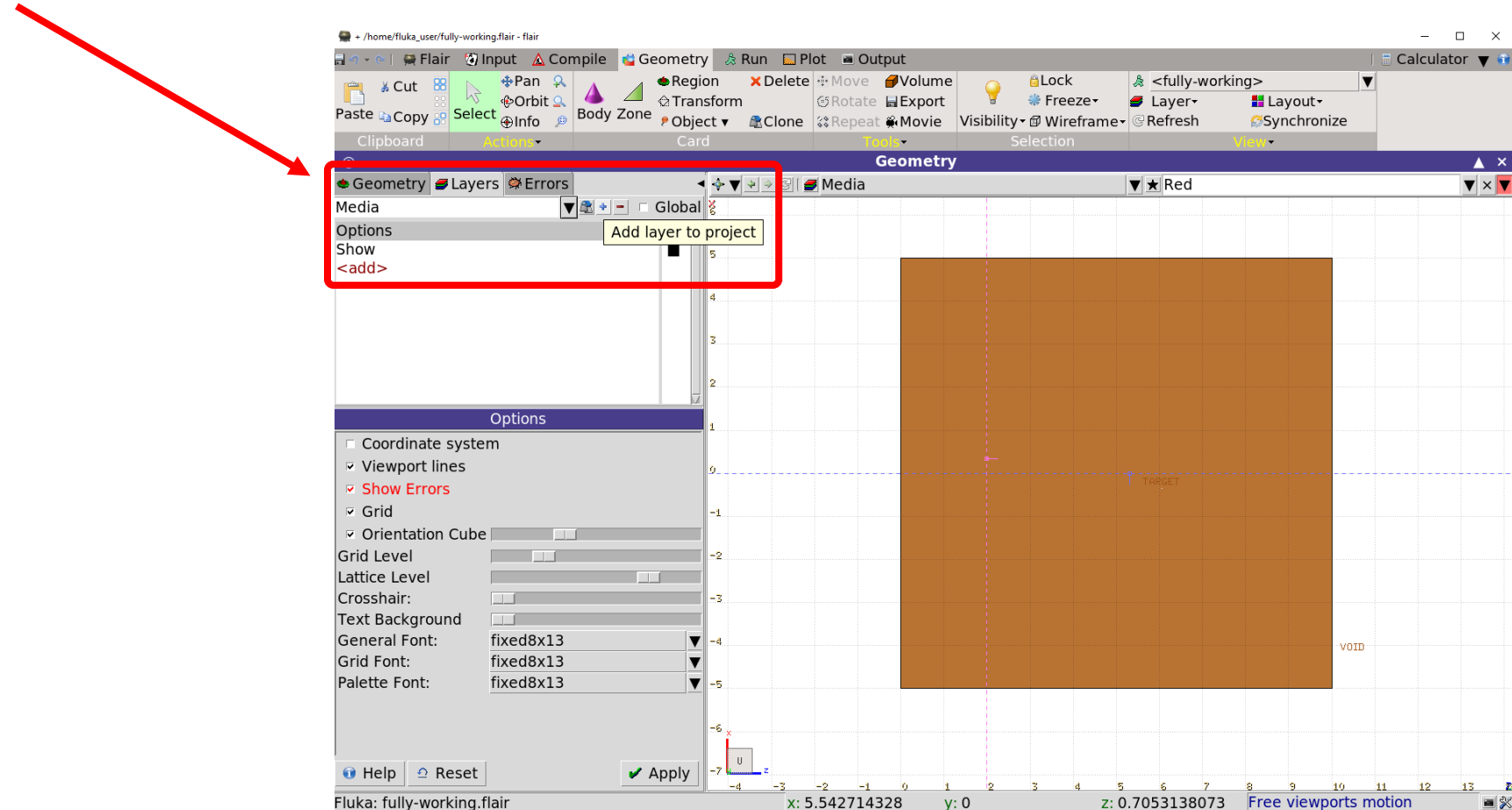
Get

swap

errors

Plotting results in the Geometry tab – 1

- It is possible to superimpose USRBIN results on the geometry
- A new layer has to be created or cloned from an existing one



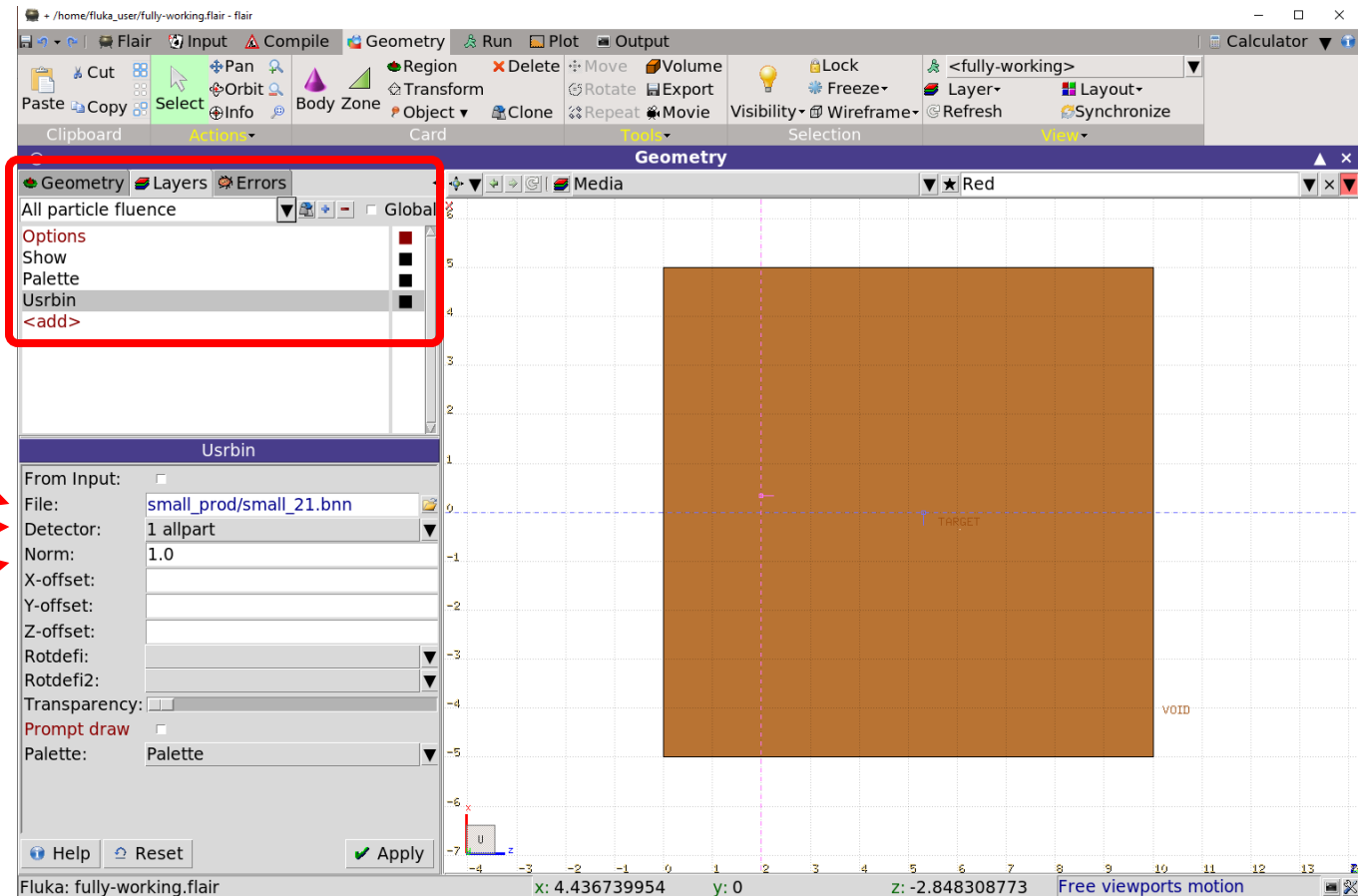
Plotting results in the Geometry tab – 2

- It is possible to superimpose USRBIN results on the geometry
- A new layer has to be created or cloned from an existing one

- `<add>` “Usrbin”

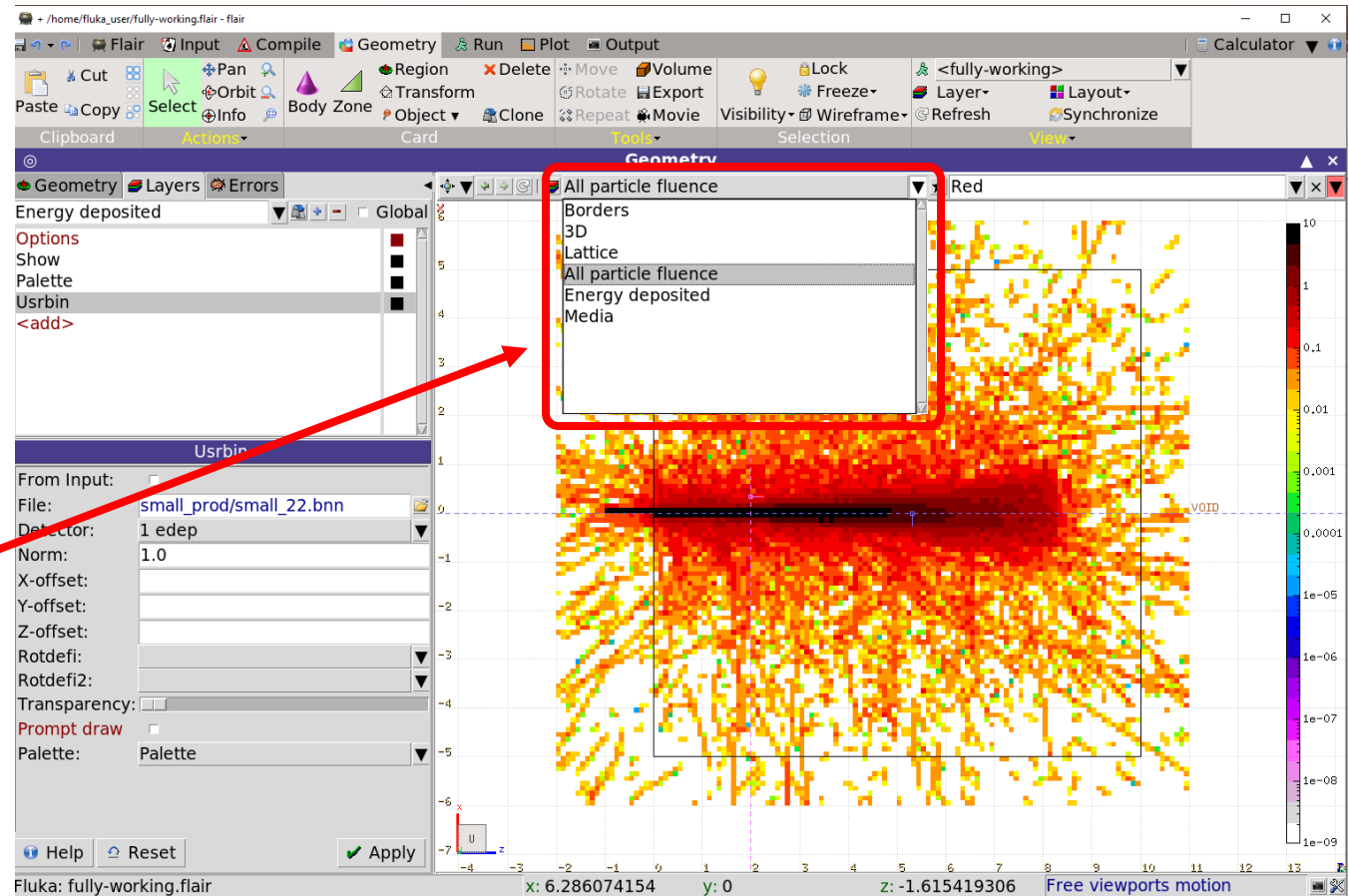
(possible to add more than one)

- Select the file with the results
- Select the detector
- Play with normalization, palette and other options



Plotting results in the Geometry tab – 3

- It is possible to superimpose USRBIN results on the geometry
- A new layer has to be created or cloned from an existing one
- <add> “Usrcin”
- Select the file with the results
- Select the detector
- Play with normalization, palette and other options
- Select the layer to visualize



Plotting result in the Geometry tab – 4

- WARNING: if the USRBIN used in a layer is missing, an error message is issued
- Not necessarily something to be worried about
- This will happen in the hands-on that follows this lecture! Don't worry!

```
TITLE course basic input hands on
Set the defaults for precision simulations
DEFAULTS : PRECISIO
Define the beam characteristics
BEAM Beam: Momentum p: 1.0 Part: PROTON
      Delta phi: Flat
Shape(X): Rectangular Delta x: Shape(Y): Rectangular Delta y:
Define the beam position
BEAMPOS x: y: z: -5.0
        cosx: cosy: Type: POSITIVE
GEOBEGIN Accuracy: Option: Out: Paren: Fmt: COMBNAME
Title:
Black body
SPH blkbody x: 0.0 y: 0.0 z: 0.0
            R: 100000.0
Void sphere
SPH void x: 0.0 y: 0.0 z: 0.0
         R: 10000.0
Cylindrical target
RCC target x: 0.0 y: 0.0 z: 0.0
           Hx: 0.0 Hy: 0.0 Hz: 10.0
           R: 5.0
END
Black hole
REGION BLKBODY Neigh: 5
expr: +blkbody -void
Void around
REGION VOID Neigh: 5
expr: +void -target
Target
REGION TARGET Neigh: 5
expr: +target
END
GEOSND
```

Error loading USRBIN
Unable to load meshdata/usrbin
detector 'small_prod/small_22.bnn'
1

Summary of the work flow

- Create your **input** in the Input tab and Geometry tab (see future lectures)
- Verify your geometry in the Geometry tab
- **Run** the simulations and **merge** the output files in the Run tab
- **Plot** your results in the Plot tab and Geometry tab (see future lectures)

Time to do some practice!

- Let's start from the example file
and run a simulation step by step



