



Introduction to Flair and basic input

A very basic introduction to perform your first simulation

A very short introduction

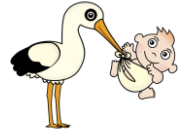
- Fluka's story begun a long time ago (1970s)...
 - ...no graphical interfaces, input and output via text file
- Inputfile can be very long > 50k lines
- Inputfile based on "cards": `.inp` file
- Each card has 1 name, 6 values (called WHATs), 1 string (called SDUM)
- Two examples of cards (the actual 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

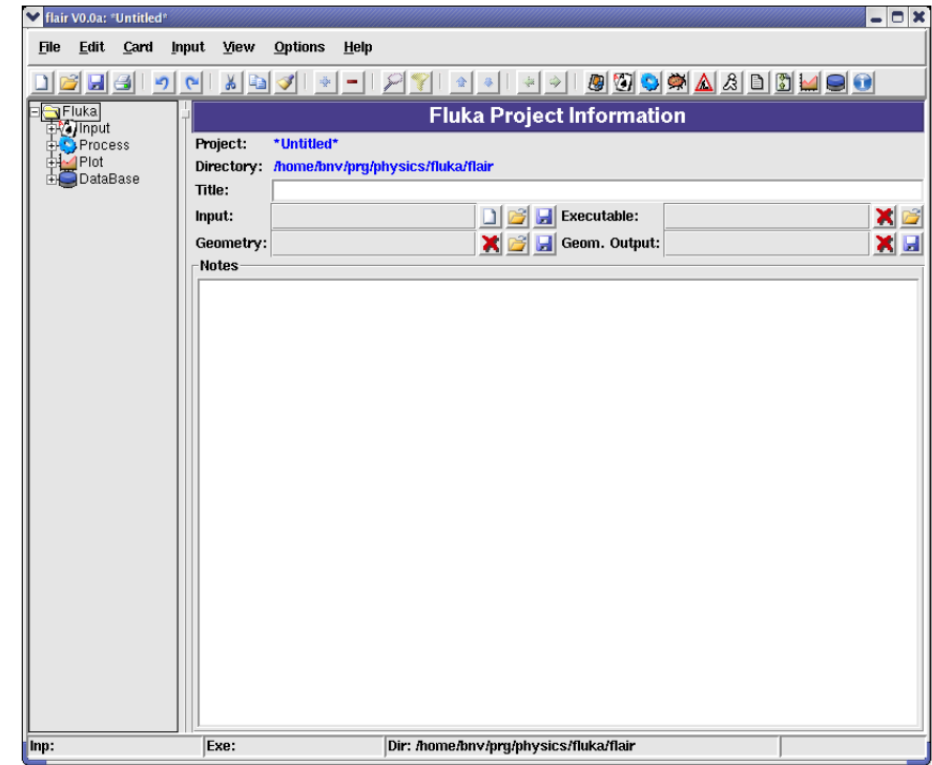
Geometry visualisation and construction

Simulation execution

Results visualisation

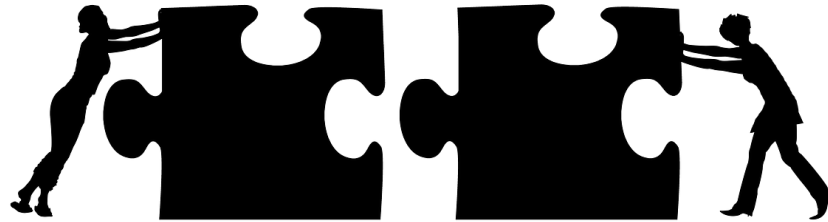
- Flair acts as an intermediate layer between the user and the inputfile
- It allows a user friendly editing of the Fluka input
- Based on a `.flair` file and generates the `.inp` file that is run by Fluka

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

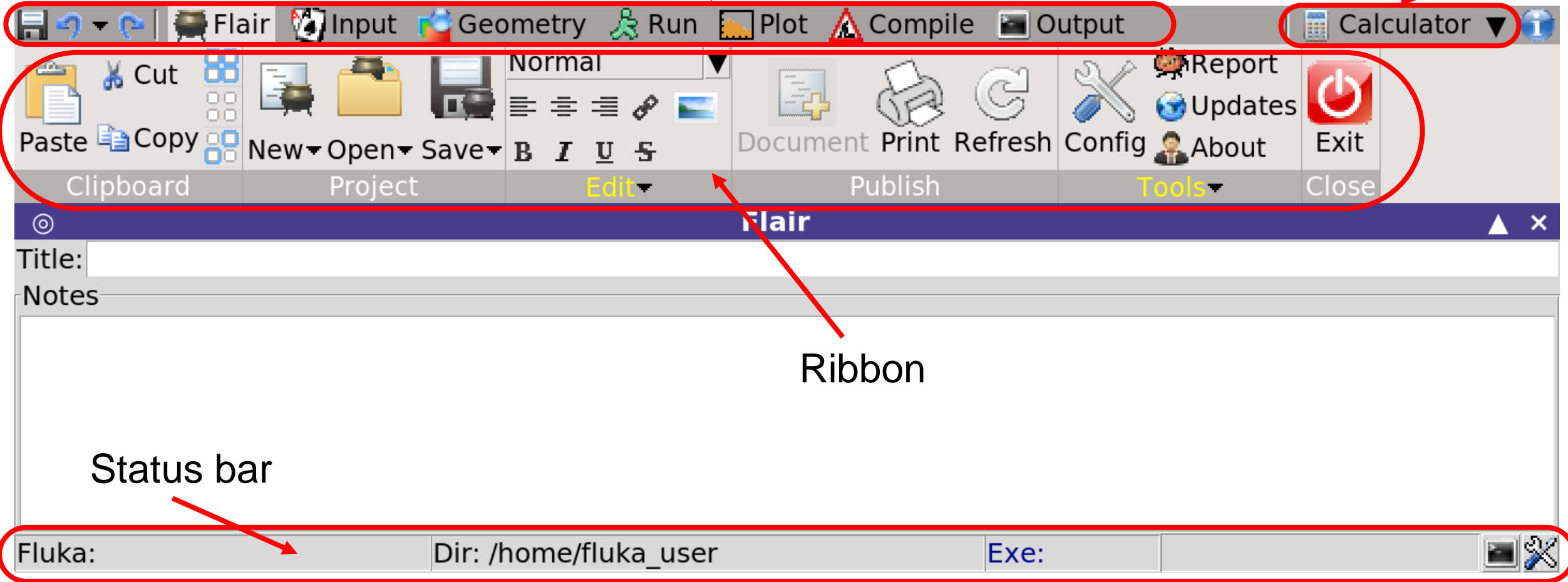


- It is possible to work with Fluka only using text editors (for expert or old users)
- Flair is not just a graphical interface for text editing
- Flair has a lot of features very useful for expert users
- This entire course will be based on Flair

Starting Flair and basic nomenclature

“Ribbon tab” or “Program tab”

Dynamic tab



What's each tab for?

The screenshot shows the Flair software interface with several menu tabs circled in red. 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. At the top, there is a menu bar with options like 'New', 'Load', 'Save', 'Export', 'Preprocessor', 'Delete', 'Change', 'Clone', 'Show', 'Comment', 'Edit Card', 'State', 'Move Up', and 'Move Down'. Below the menu bar is a toolbar with icons for 'Cut', 'Copy', 'Paste', 'New', 'Load', 'Save', 'Export', 'Preprocessor', 'Delete', 'Change', 'Clone', 'Show', 'Comment', 'Edit Card', 'State', 'Move Up', and 'Move Down'. The main window is titled 'Input' and contains a file tree on the left side with categories like 'General', 'Primary', 'Geometry', 'Media', 'Physics', 'Transport', 'Biasing', 'Scoring', 'Flair', and 'Preprocessor'. The main area of the window is currently empty.

Open and save input file

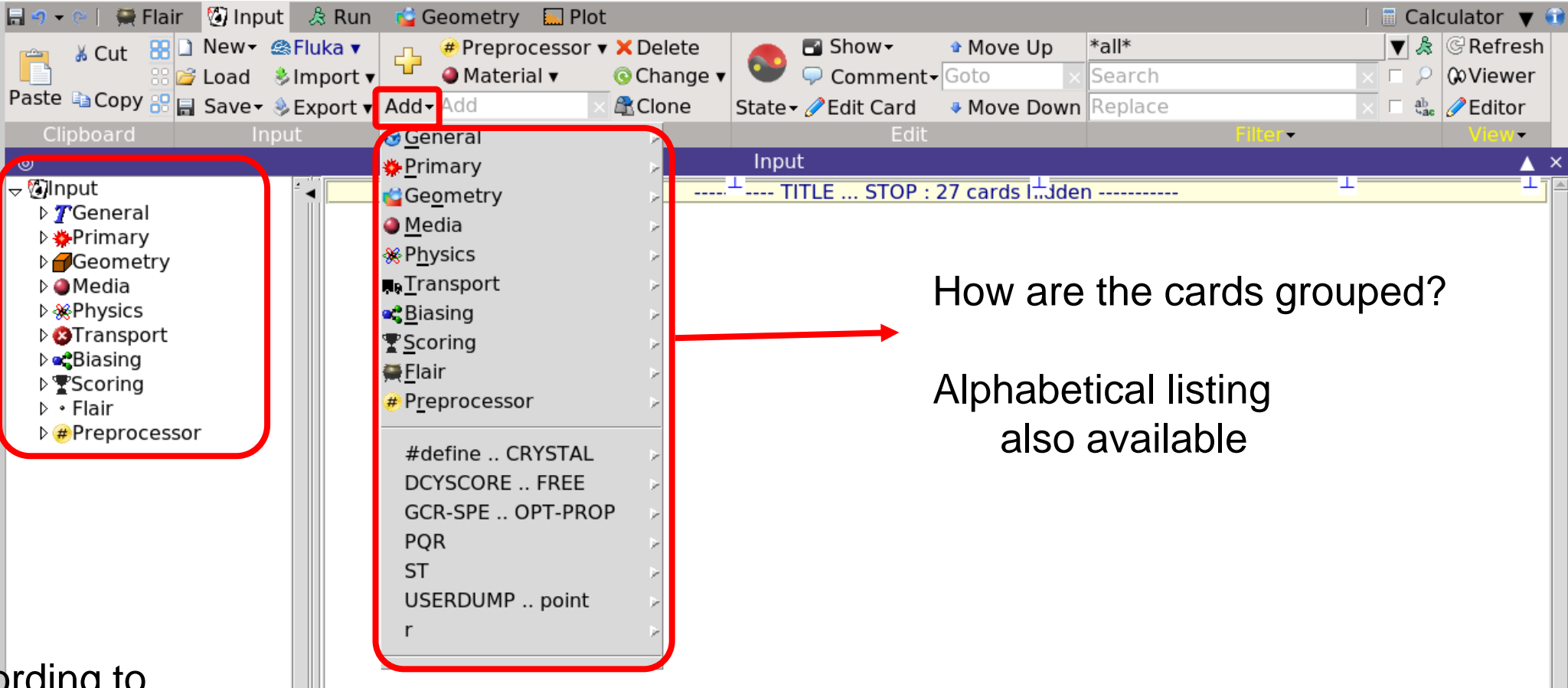
Add, clone, delete cards

Move cards up and down (some limitations present)
Recommended to use the suggested structure

Input file tree
Cards grouped according to their “field of action”

Input tab – 2: input file tree and card grouping

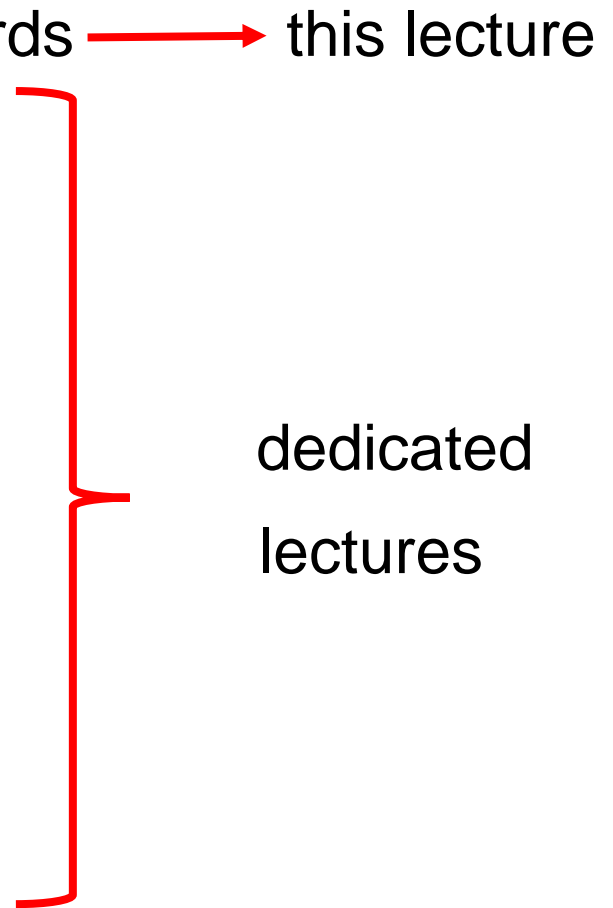
- Input file tree and card grouping



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

How are the cards grouped?
Alphabetical listing also available

Input tab – 3: input file tree and card grouping

- General: defaults selection and other general cards → 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

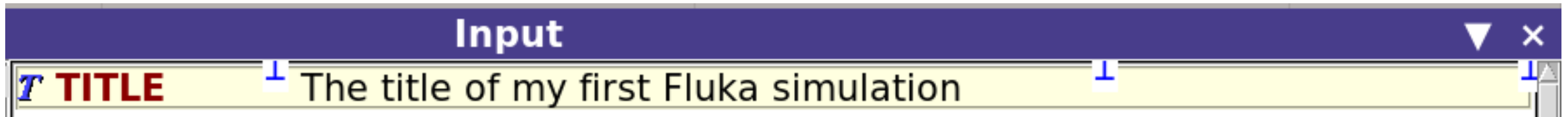
STOP

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: General cards

TITLE

START

STOP

RANDOMIZe

DEFAULTS

START

- Actually listed among the “Primary” cards
- 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.

Core: ▼

Time:

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

RANDOMIZe

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: ▼	Field: ▼
Mat(Decay): ▼	Step:		Imp: ▼
Type: ▼	RR:		Step:
Reg: ▼	to Reg: ▼		Set: ▼
Type: ▼	Part: ▼		
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 ▼
Mat(Decay): ▼	Step: 2
	to Reg: region_3 ▼
	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 ▼
Mat(Decay): ▼	Step: 2
	to Reg: region_4 ▼
	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.

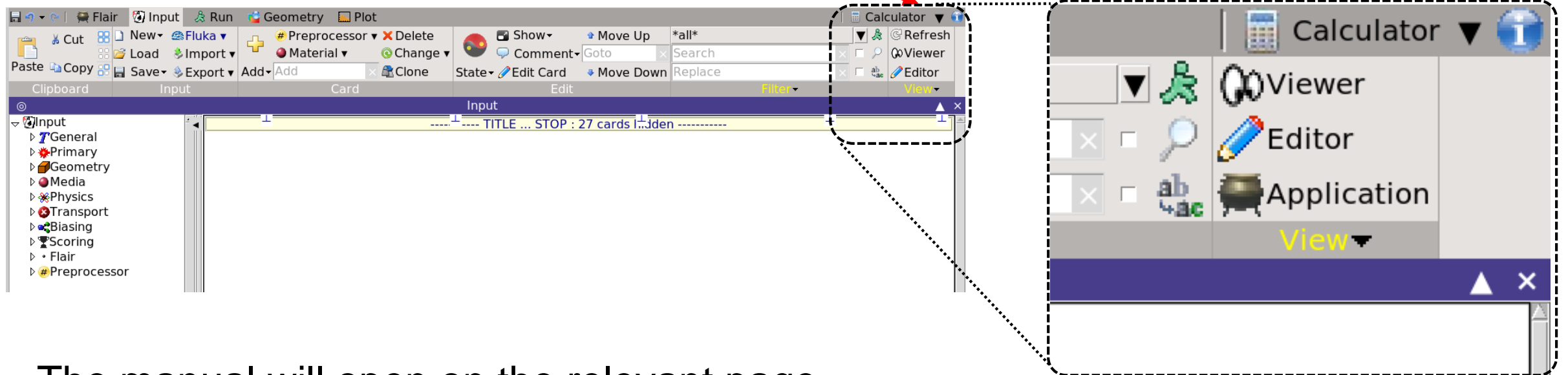
PHOTONUC E>0.7GeV: off ▼	Type: ▼ Δ resonance: off ▼ Mat: CARBON ▼	Quasi D: off ▼ to Mat: ▼	All E: off ▼ Giant Dipole: off ▼ Step: 2
REGION region_1 expr: +reg1		Neigh: CARBON	
REGION region_2 expr: +reg2		Neigh: NITROGEN	
REGION region_3 expr: +reg3		Neigh: OXYGEN	
REGION region_4 expr: +reg4		Neigh: 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

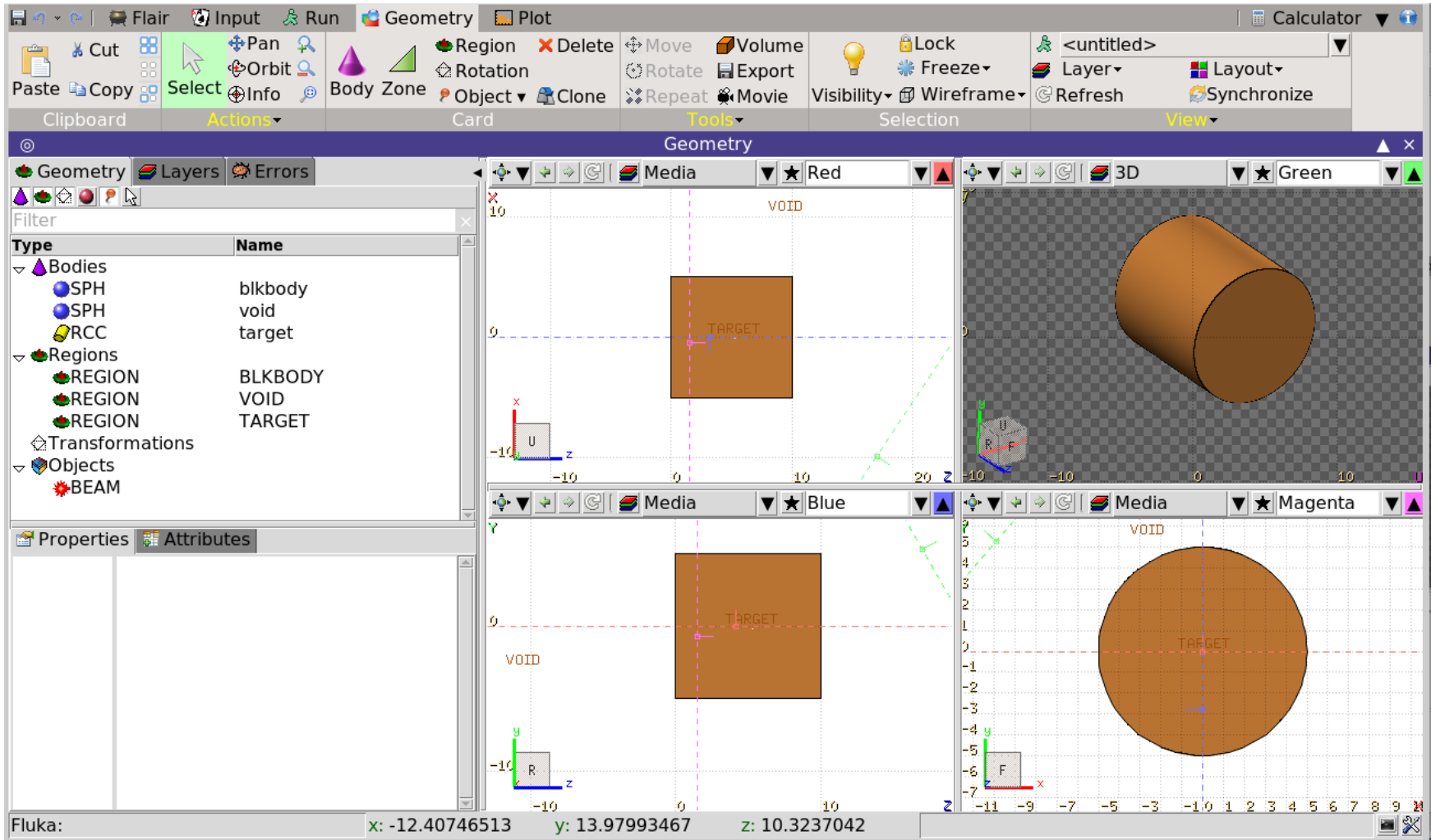
- Can be accessed using F1 button
- Can be accessed clicking on the “info” button



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

Geometry tab – 1

- Visualise 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. The top toolbar includes buttons for Cut, Copy, Paste, Select, Pan, Orbit, Info, Body Zone, Object, Clone, Repeat, Rotate, Move, Volume, Export, Lock, Freeze, Layer, Layout, Synchronize, and Trans. The main window is divided into several panes:

- Filter:** A toolbar with icons for Geometry, Layers, and Errors.
- Objects Listbox:** A tree view showing the hierarchy of objects. The selected object is 'target' (RCC).
- Properties & Attributes Listbox:** A table showing the properties of the selected object.
- Viewports:** Three 3D viewports showing the geometry in different media (Red, Blue, Magenta).

Type	Name
Bodies	blkbody
SPH	void
SPH	target
RCC	target
Regions	
REGION	BLKBODY
REGION	VOID
REGION	TARGET
Transformations	
Objects	
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
Hx	10.0
R	5.0
@xmid	0.0
@ymid	0.0

Geometry tab – 3

Geometry

Media

Media

Media

Media

Red

Green

Blue

Magenta

Red viewport

Green viewport

Blue viewport

Magenta viewport

Geometry

Layers

Errors

Filter

Type	Name
Bodies	
SPH	blkbody
SPH	void
RCC	target
Regions	
REGION	BLKBODY
REGION	VOID
REGION	TARGET
Transformations	
Objects	
BEAM	

Properties

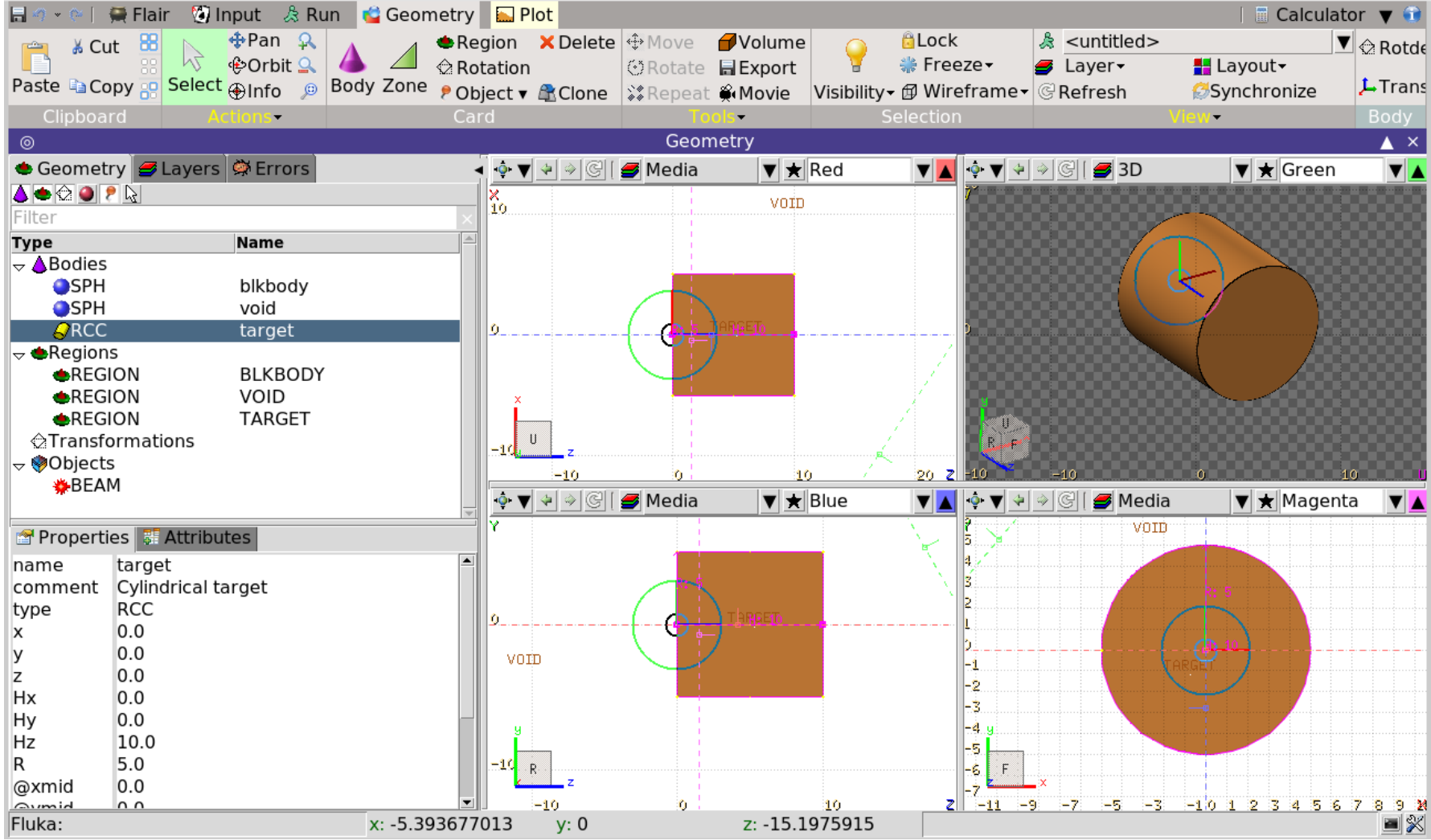
Attributes

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

Fluka: x: -5.393677013 y: 0 z: -15.1975915

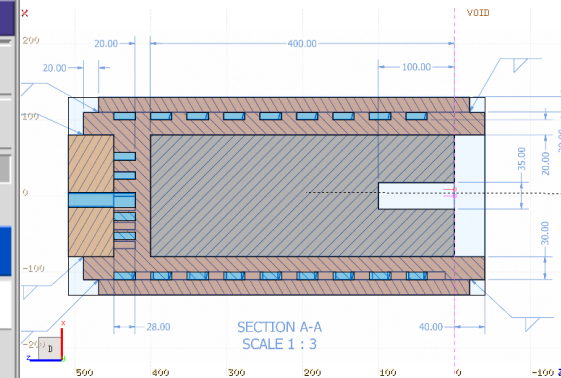
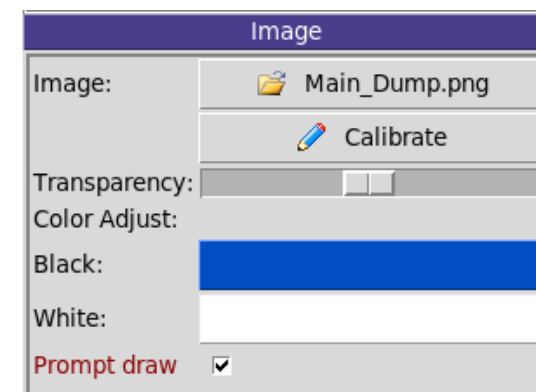
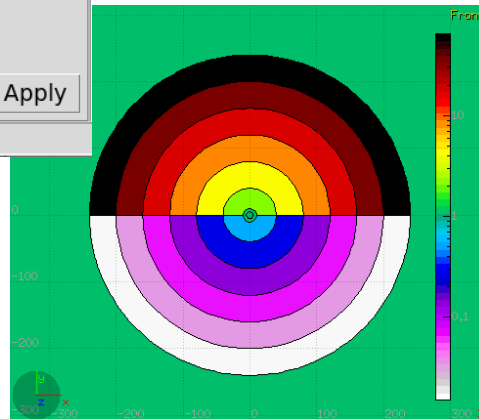
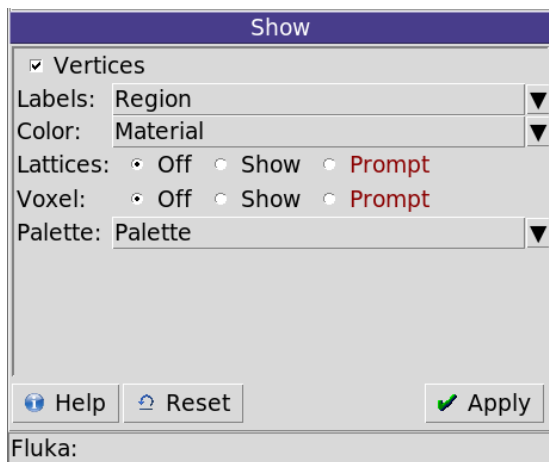
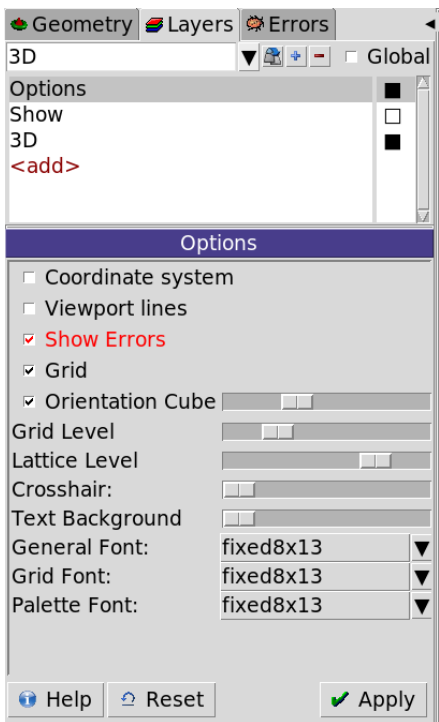
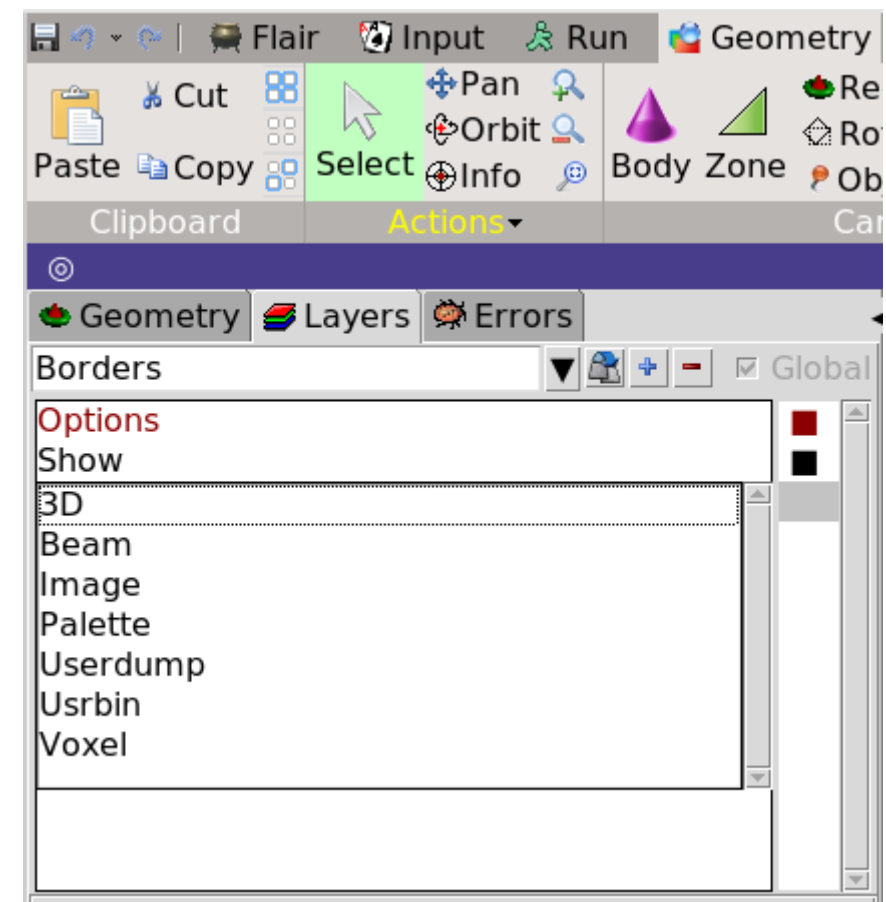
Geometry tab – 4

- 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”

• Shared listbox:
List of simulations
associated with the project

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

Progress

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

Fluka: fully-working.flair Running 0 out of 14

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

The screenshot shows the Flair software interface. The top toolbar includes icons for Run, Files, and Data. The main panel displays a tree view of simulation runs. The tree structure is as follows:

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

Red arrows point from the text on the left to the following elements in the interface:

- Management of the various simulations: Points to the Run tab and the Run icon in the toolbar.
- Basic inputfile of the Flair project: Points to the <fully-working> run.
- Different simulations associated with the Flair project: Points to the test, small_prod, and large_prod folders.
- Number of spawns: Points to the number '4' next to the small, large, and exe runs.

The right panel shows the Override section with fields for Title, Primaries (0.0), Mode, and Defines (Default). Below this is a table with columns Name and checkboxes:

	Name
<input type="checkbox"/>	pointless_define_1
<input checked="" type="checkbox"/>	pointless_define_2

The bottom panel shows the Progress section with fields for Status (Not Running), Started, Elapsed, Cycles, and Primaries. The status bar at the bottom indicates 'Fluka: fully-working.flair' and 'Running 0 out of 14'.

Run tab – Runs view – 2

- Override of inputs

- Number of primaries

- Executable

- #define

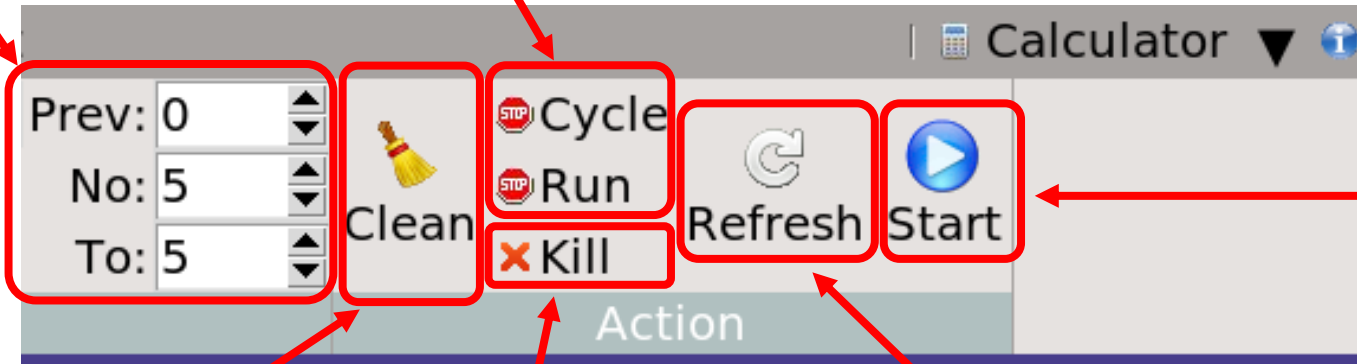
Dedicated lecture

	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 a 'Run' window with a tree view on the left and a detailed progress bar on the right. The tree view shows a hierarchy of runs: <fully-working>, test, small_prod, small, large_prod, large, and example-spawn. The 'small' directory is expanded, showing 'small_01' through 'small_04'. The 'small_01' run is selected and highlighted. The progress bar for 'small_01' shows the following information:

- 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%

Red arrows point from text labels to specific elements in the screenshot:

- Overall job progress bar (points to the tree view)
- Status (points to 'Running')
- Estimated end of the simulation (points to 'ETA')
- Input file actually running (points to 'Input')
- Temporary directory (points to 'Dir')
- Time per primary (points to 'Time/prim')
- Cycle progress bar (points to the 'Cycles' progress bar)
- Time since the start of the cycle (points to 'Elapsed')
- Time until the end of the cycle (points to 'Cycle')
- Time until the end of the simulation (points to 'Run')

• Overall job progress bar

• Cycle progress bar

• Status

• Time since the start of the cycle

• Estimated end of the simulation

• Time until the end of the cycle

• Input file actually running

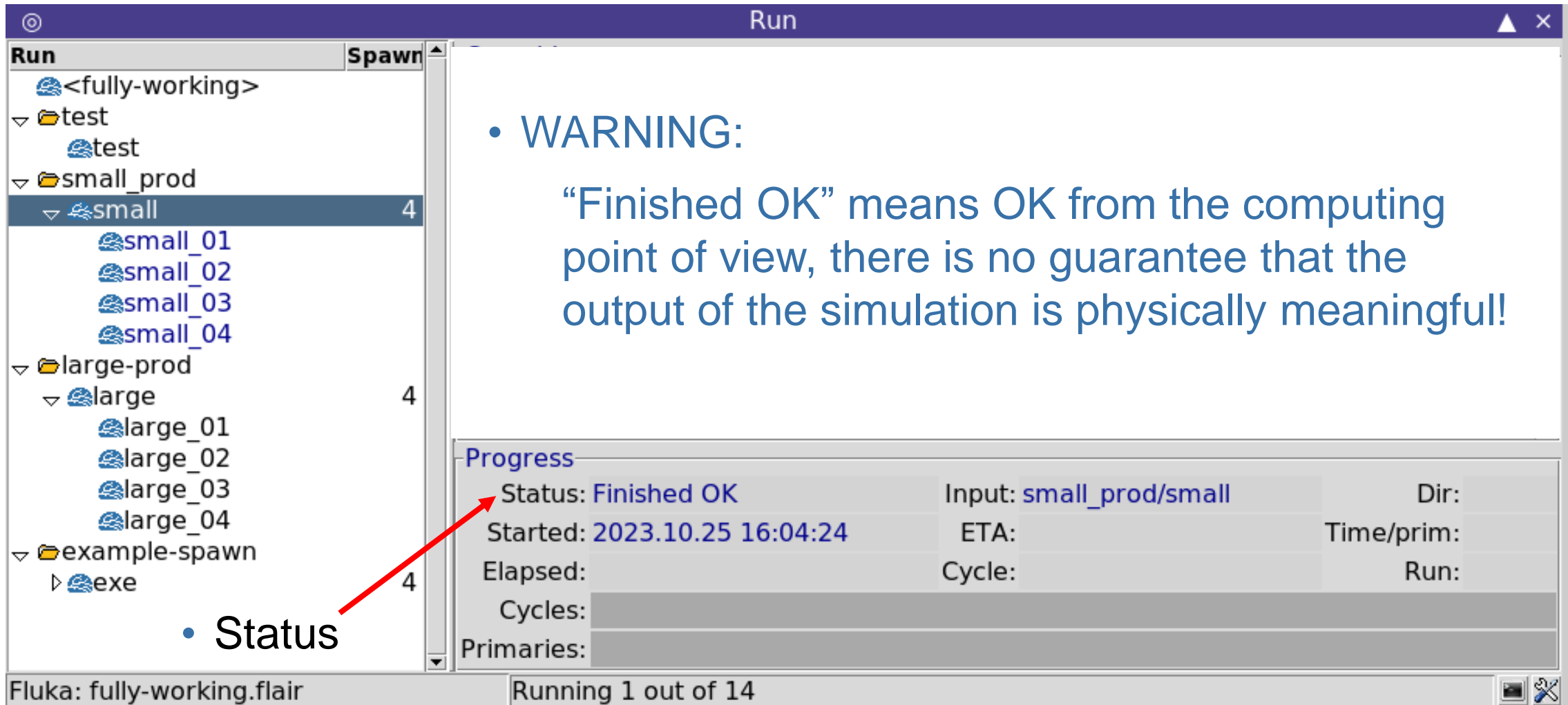
• Time per primary

• Time until the end of the simulation

• Temporary directory

Run tab – Runs view – 5

- At the end of the simulations...



The screenshot shows a window titled "Run" with a tree view on the left and a progress table on the right. The tree view shows a hierarchy of simulation runs: <fully-working>, test, small_prod, small, large-prod, and example-spawn. The "small" folder is selected, showing sub-items small_01 through small_04. The "large-prod" folder shows a "large" sub-folder with items large_01 through large_04. The "example-spawn" folder shows an "exe" sub-folder. The progress table shows the status of the selected run as "Finished OK".

- **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:			

- Status

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_fort.21  small_03002_fort.22  small_04001_fort.22  
ransmall_01004  small_01.out        small_01005.log      small_02004_fort.22  small_03002_fort.22  small_04001_fort.22  
ransmall_01005  small_01001.err     small_01005.out      small_02004_fort.22  small_03002_fort.22  small_04001_fort.22  
ransmall_01006  small_01001.log     small_01005_fort.21  small_02004_fort.22  small_03002_fort.22  small_04001_fort.22  
ransmall_02001  small_01001.out     small_01005_fort.22  small_02004_fort.22  small_03002_fort.22  small_04001_fort.22  
ransmall_02002  small_01001_fort.21  small_02005.err      small_02005.log      small_03004_err      small_04003_err  
ransmall_02003  small_01001_fort.22  small_02005.log      small_02005.out      small_03004_log      small_04003_log  
ransmall_02004  small_01002.err     small_02001_err      small_02005_fort.21  small_03004_out      small_04003_out  
ransmall_02005  small_01002.log     small_02001_log      small_02005_fort.22  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_02005_fort.22  small_03004_err      small_04004_err  
ransmall_03002  small_01002_fort.22  small_02001_fort.22  small_02005_err      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_04005_err      small_04005_err  
ransmall_04001  small_01003_fort.22  small_02002_fort.22  small_03001_fort.22  small_04005_log      small_04005_log  
ransmall_04002  small_01004.err     small_02003_err      small_03002_err      small_04005_out      small_04005_out  
ransmall_04003  small_01004.log     small_02003_log      small_03002_log      small_04005_fort.21  small_04005_fort.21  
ransmall_04004  small_01004.out     small_02003_out      small_03002_out      small_04005_fort.22  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

Run	Spawr	Cycles	File	Type▲	Size	Date
<fully-working>		001	small_prod/ransmall_01001	-file-	1651	2023.10.25 16:13:54
test		002	small_prod/small_01001_fort.21	21	242	2023.10.25 16:13:57
test		003	small_prod/small_01001_fort.22	22	242	2023.10.25 16:13:57
small_prod		004	small_prod/small_01001.err	Error	714	2023.10.25 16:13:57
small	4	005	small_prod/small_01001.log	Log	0	2023.10.25 16:13:53
small_01		006	small_prod/small_01.out	Output	2193	2023.10.25 16:14:13
small_02		compile	small_prod/small_01001.out	Output	56913	2023.10.25 16:13:57
small_03		data				
small_04		input				
large_prod		plot				
large	4	temporary				
large_01						
large_02						
large_03						
large_04						
example-spawn						
exe	4					

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 inputfile

The screenshot displays the Flair software interface in the 'Run' tab. The left sidebar shows a tree view of the project structure. The 'small' folder under 'small_prod' is selected, showing its sub-items. The main panel displays a table of detected logical units.

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

The status bar at the bottom indicates the current file is 'Fluka: fully-working.flair' and there are 40 files in the project.

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 options like Flair, Input, Run, Geometry, and Plot. The toolbar contains various icons for file management and execution. The main window is titled 'Run' and is divided into two panes. The left pane shows a tree view of the project structure, including folders like 'test', 'small_prod', 'small', 'large-prod', 'large', and 'example-spawn'. The right pane is split into two sections: 'Detectors' and 'Files'. The 'Detectors' section contains a table with columns 'Run', 'Type', 'Output', and 'Name/Unit'. The 'Files' section contains a table with columns 'File', 'Type', 'Size', and 'Date'.

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
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

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. The toolbar at the top has icons for Scan (3), Refresh (2), and Process (4). The Run tab is active, and the Data view is shown. The tree view on the left shows a hierarchy of runs, with 'small' selected (1). The 'Detectors' table on the right shows the following data:

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

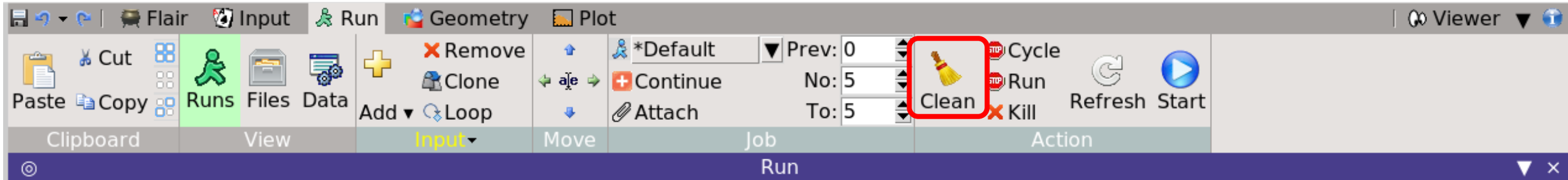
Below the Detectors table, the 'Files' tab is active, showing a list of files:

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

At the bottom of the interface, the status bar shows 'Fluka: fully-working.flair' and 'Files: 20'.

Run tab – Cleaning – 1

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



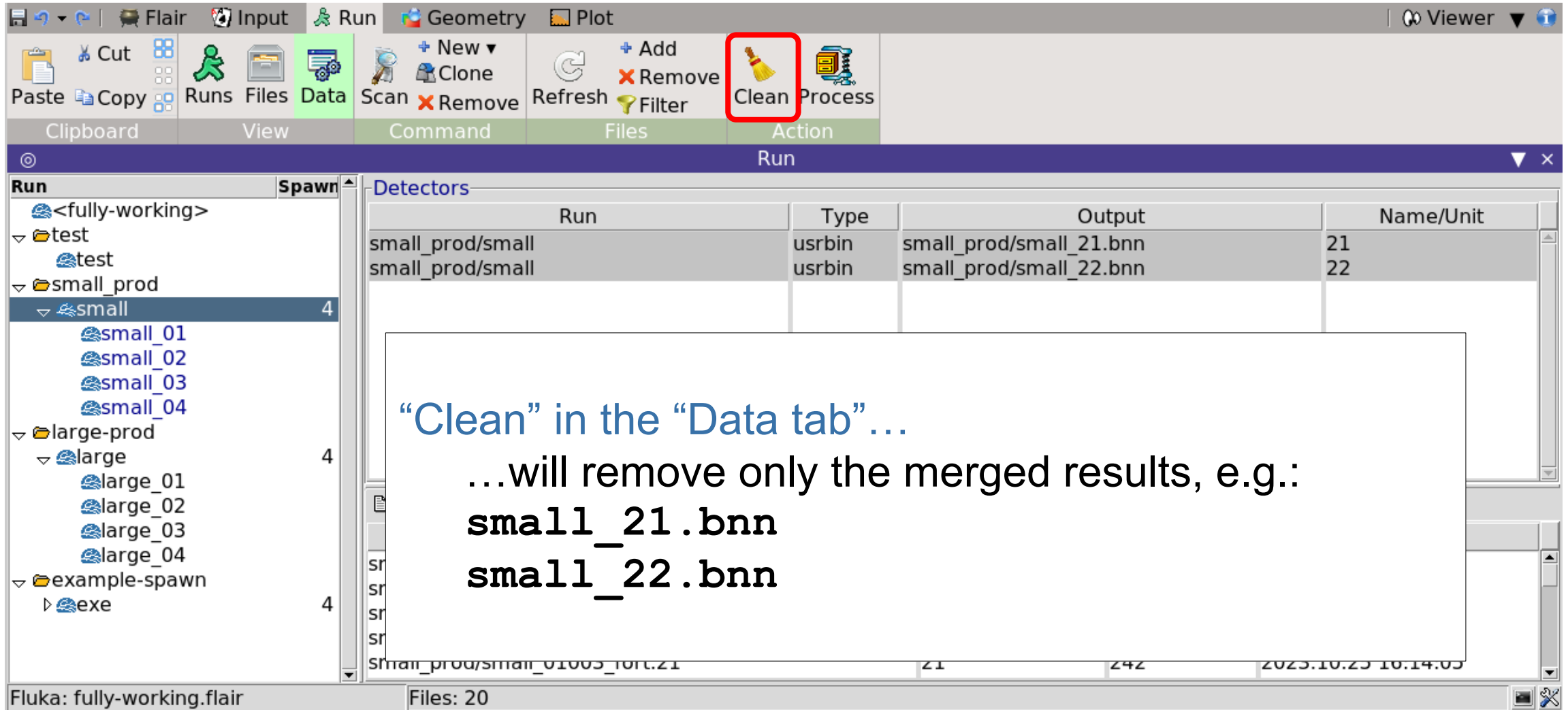
The screenshot shows the Flair software interface. The 'Run' window is open, and the 'Action' tab is selected. The 'Clean' button, represented by a broom icon, is highlighted with a red box. The 'Run' window displays a tree view of the current run structure, including folders like 'test', 'small_prod', 'large-prod', and 'example-spawn'. The 'Clean' button is located in the 'Action' tab, which also contains buttons for 'Cycle', 'Run', 'Kill', 'Refresh', and 'Start'.

“Clean” in the “Runs tab”...
...will remove all and only the files generated within a specific run (or spawn), e.g.:

<code>large_01001_fort.21</code>	<code>large_02001_fort.21</code>
<code>large_01001_fort.22</code>	<code>large_02001_fort.22</code>
<code>large_01001.err</code>	<code>large_02001.err</code>
<code>large_01001.log</code>	...
<code>large_01001.out</code>	...
<code>ranlarge_01001</code>	

Run tab – Cleaning – 2

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



The screenshot shows the Flair software interface. The 'Run' tab is active, and the 'Data' menu is open. The 'Clean' button, represented by a broom icon, is highlighted with a red box. Below the interface, a text box contains the following text:

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

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

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 selected. The toolbar contains several icons, with 'Add Database', 'lfluka' (linker), and 'Build' highlighted with red boxes. Red arrows point from the text labels above to these elements. The Compile window shows a table of files to be compiled:

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 advanced courses

Do you remember slide 6?

The screenshot shows the Flair software interface. The top menu bar includes: Flair, Input, Geometry, Run, Plot, Compile, Output, Calculator. Below the menu bar are several toolbars: Clipboard (Paste, Copy), Project (New, Open, Save), Edit (B, I, U), Publish (Document, Print, Refresh), Tools (Config, Report, Updates, About), and Close (Exit). The main window has a title bar 'Flair' and a 'Notes' section. Red arrows point from the following menu items to text in the Notes section: 'Input' to 'Build input and geometry', 'Geometry' to 'Build geometry and plot results', 'Run' to 'Run and merge results', 'Plot' to 'Plot results', 'Compile' to 'Compile own executable', and 'Output' to 'Visualise output files and messages'. The status bar at the bottom shows 'Fluka: Dir: /home/fluka_user Exe:'.

Build input and geometry

Build geometry and plot results

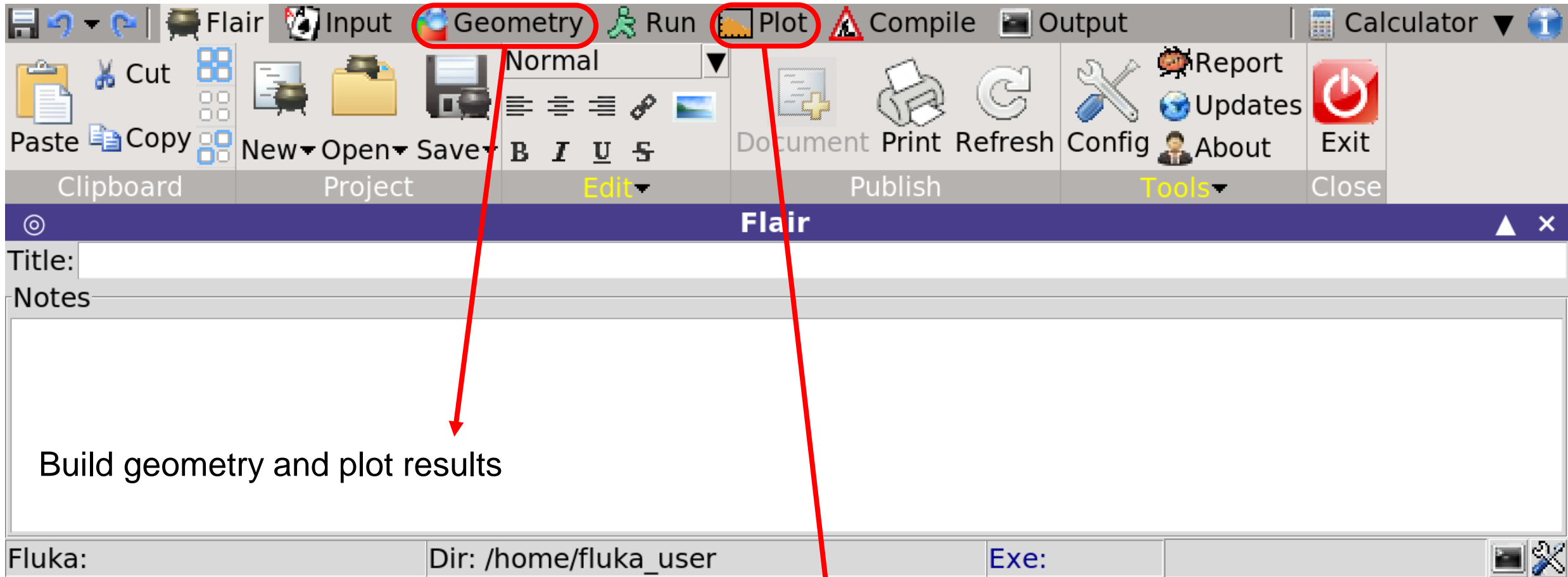
Run and merge results

Plot results

Compile own executable

Visualise output files and messages

Do you remember slide 6?

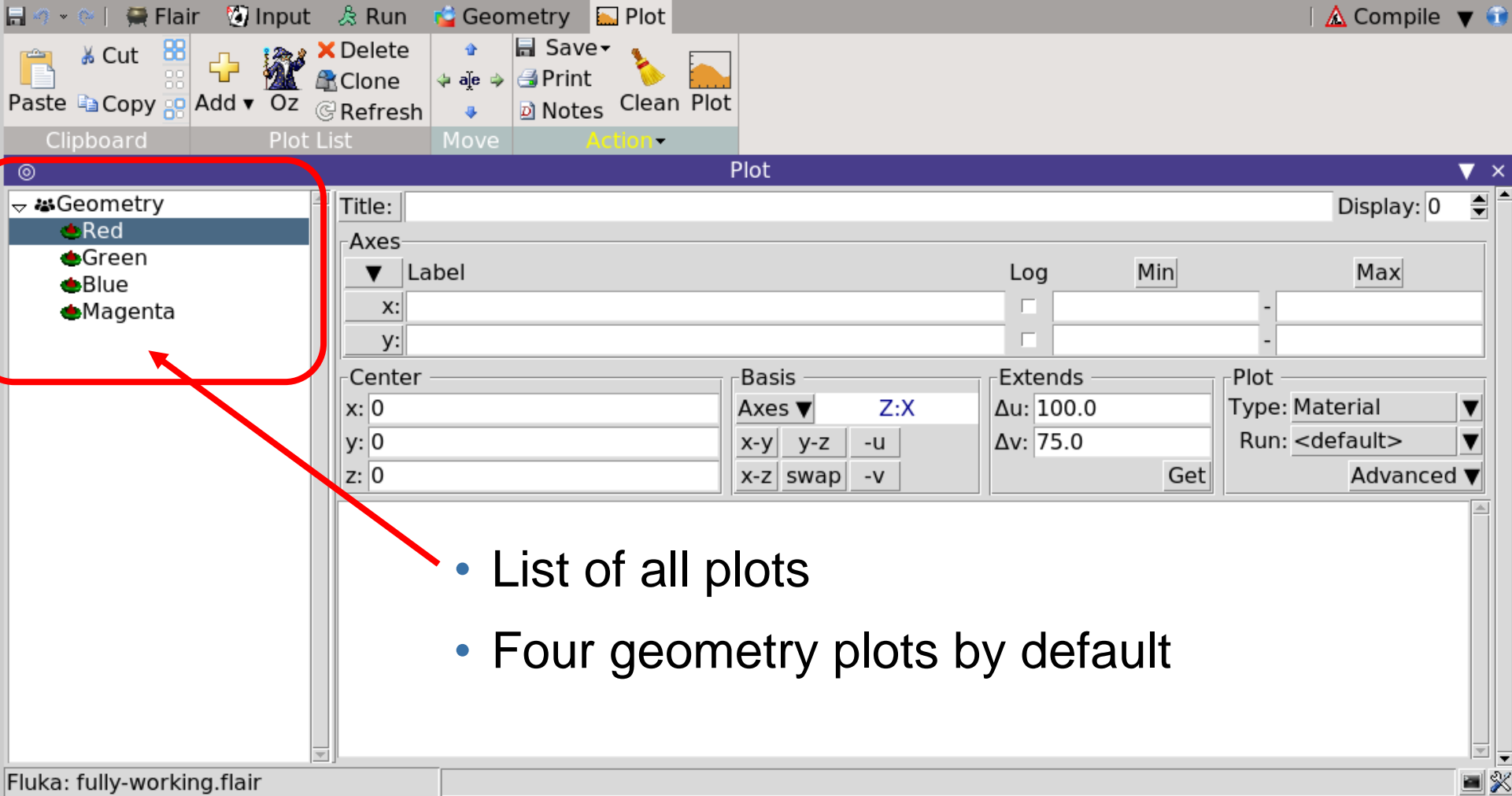


Build geometry and plot results

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 plot configuration area. The plot configuration area includes the following fields:

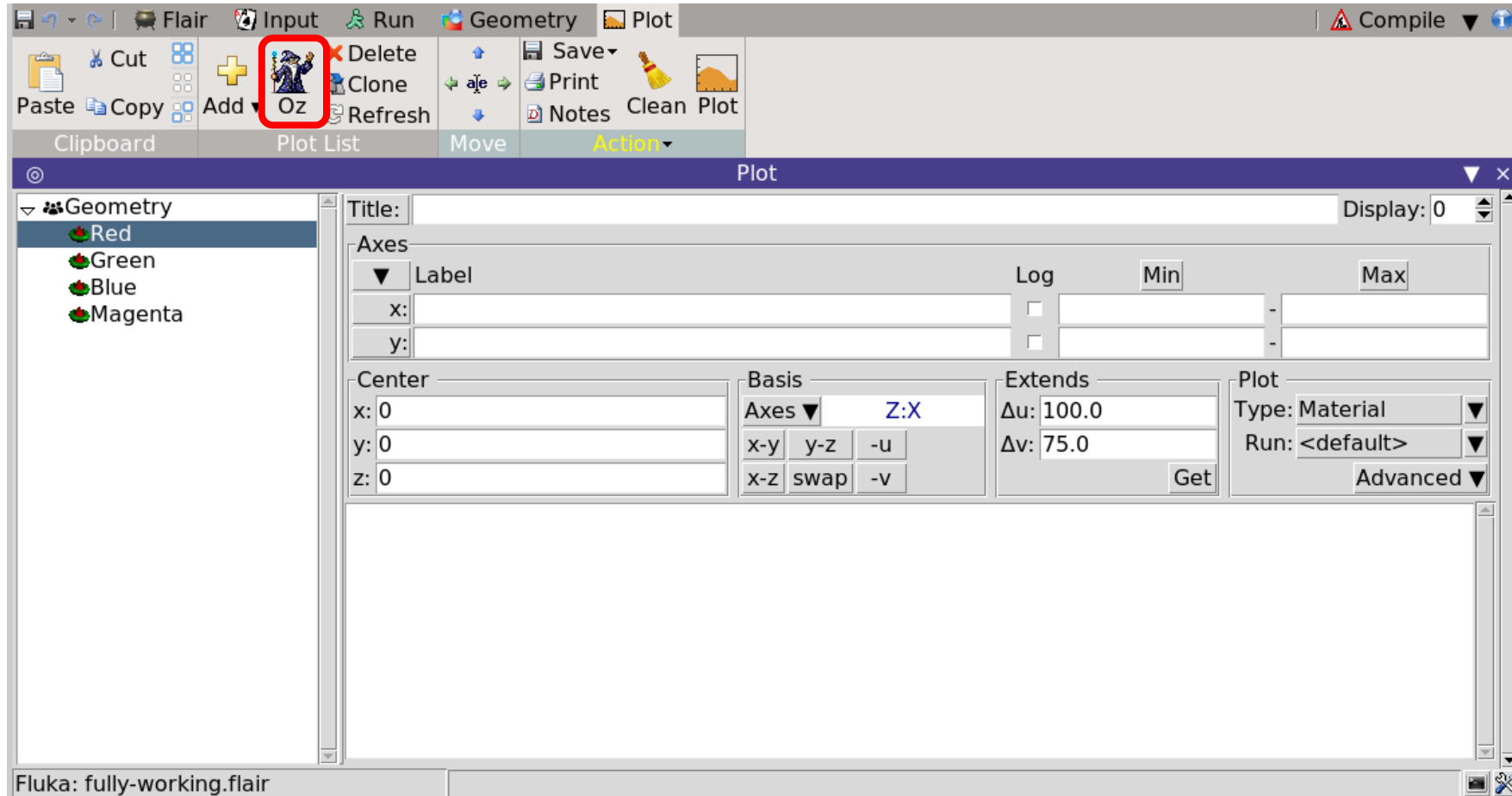
- Title: [] Display: 0
- Axes: x: [] y: []
- Center: x: 0 y: 0 z: 0
- Basis: Axes ▼ Z:X (with sub-options x-y, y-z, -u, x-z, swap, -v)
- Extends: Δu: 100.0 Δv: 75.0
- Plot: Type: Material Run: <default>

At the bottom of the window, the status bar shows 'Fluka: fully-working.flair'.

- List of all plots
- Four geometry plots by default

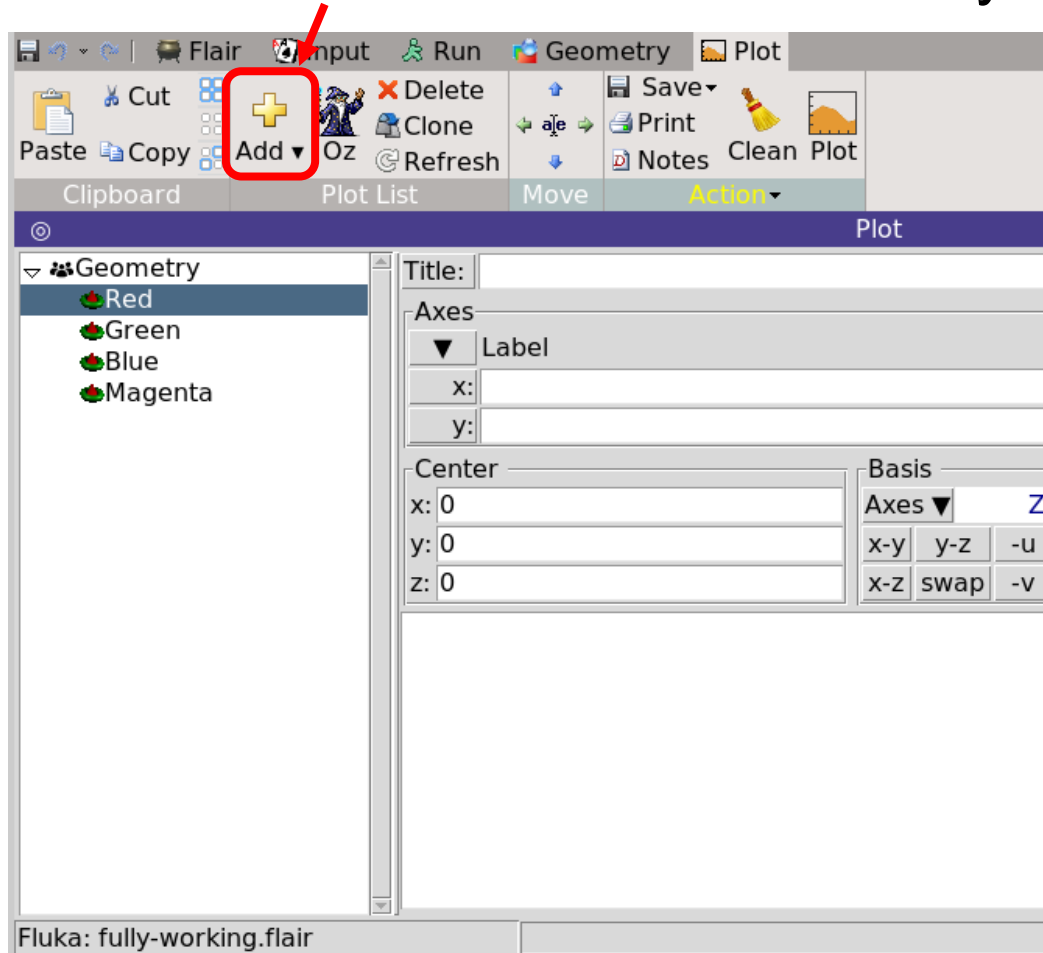
Plotting results in the Plot tab – 2

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

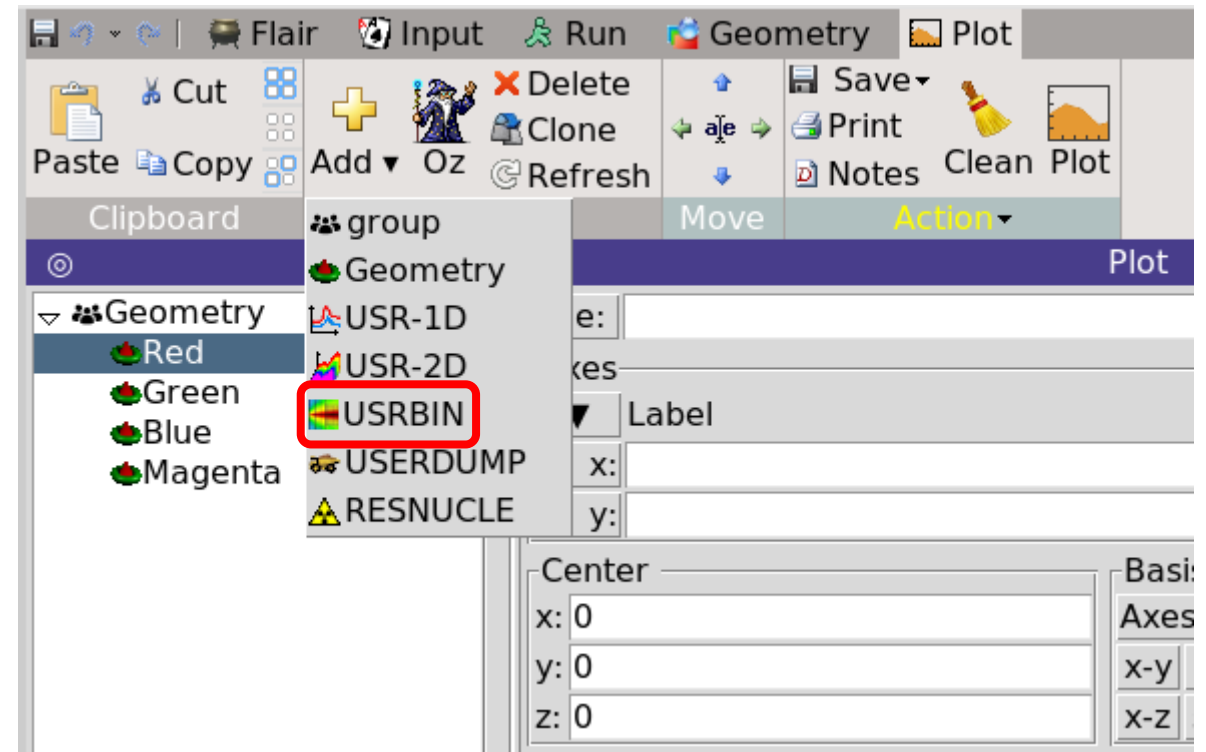


Plotting results in the Plot tab – 3

- 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

The screenshot shows the Flair software interface with the Plot tab active. The Plot List on the left contains several files, with 'fully-working_plot05' selected and highlighted in blue. A red box is drawn around this file name. A red arrow points from the text 'Name of the file that will be saved' to this box. The main Plot configuration area has a 'Title' field containing 'Plot #5', which is also highlighted with a red box and pointed to by a red arrow from the text 'Title of the plot'. To the right of the title field is a 'Display' dropdown menu set to '0', also highlighted with a red box and pointed to by a red arrow from the text 'Display ID'. The interface includes a menu bar at the top with options like 'Flair', 'Input', 'Geometry', 'Run', 'Plot', 'Compile', and 'Output'. Below the menu bar is a toolbar with icons for 'Cut', 'Copy', 'Add', 'Oz', 'Rename', 'Clone', 'Notes', 'Clean Plot', 'Save', 'Print', and 'Notes'. The main configuration area is divided into sections: 'Axes' (with fields for X, Y, and Z), 'Binning Detector' (with fields for File, Title, Cycles, Primaries, Weight, and Time), 'Binning Info' (with fields for Det, X, Y, Z, Min, Max, and Int), 'Projection & Limits' (with fields for X, Y, Z, and Norm), and 'Options' (with fields for Type, Color, Line width, and Point type). The status bar at the bottom shows 'Fluka: fully-working.flair' and 'Plot completed'.

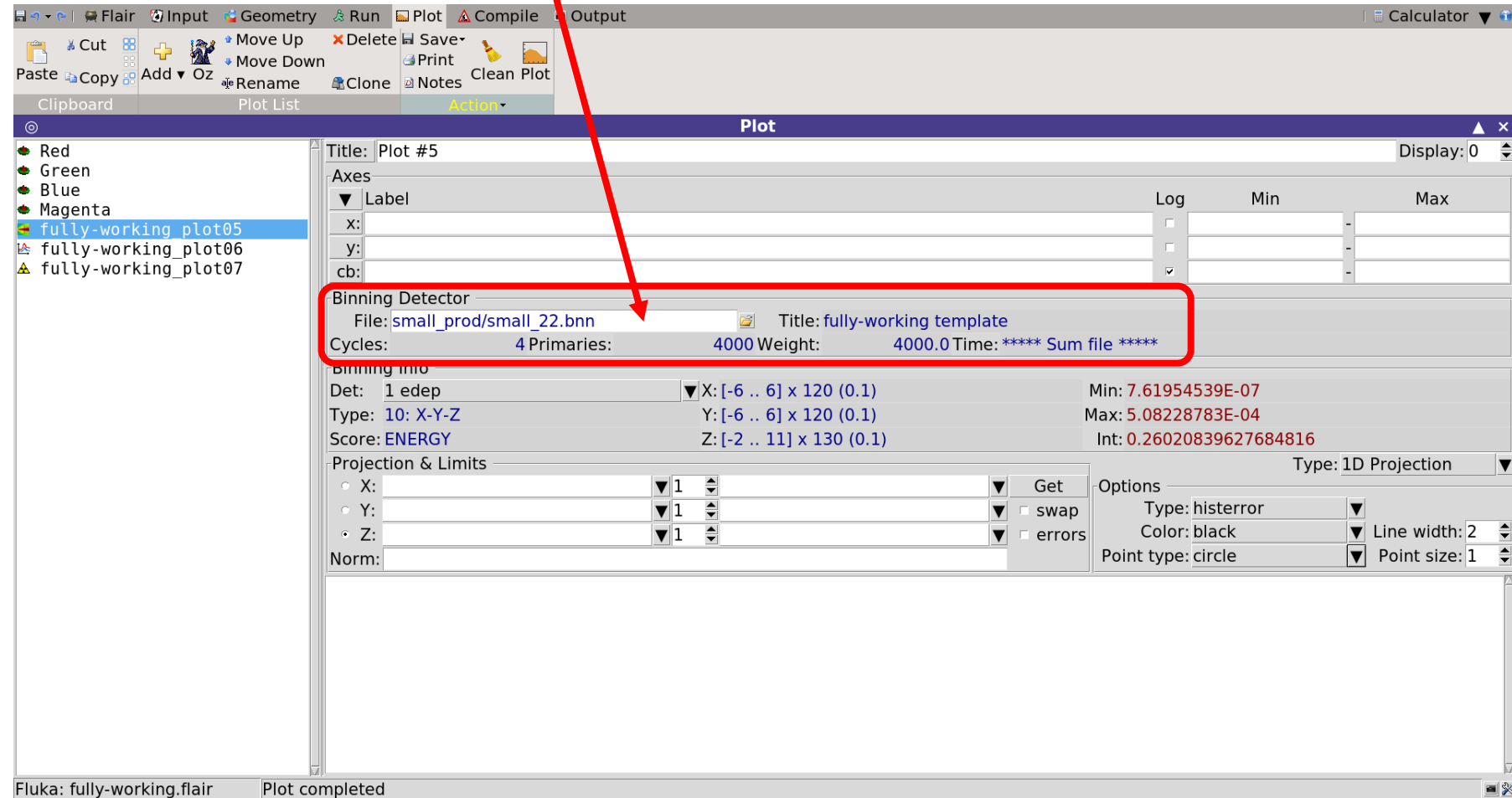
Plotting results in the Plot tab – 5

The screenshot displays the Flair software interface with the Plot tab active. The main window shows a list of plots on the left and a detailed configuration panel for the selected plot. The configuration panel includes options for font, color, grid, aspect, and lines, as well as axis labels and tick marks. Two inset windows show the resulting plots: one with a rectangular plot and one with a circular plot. Both plots use a logarithmic color scale for the data values.

• Plenty of options for plot customisation

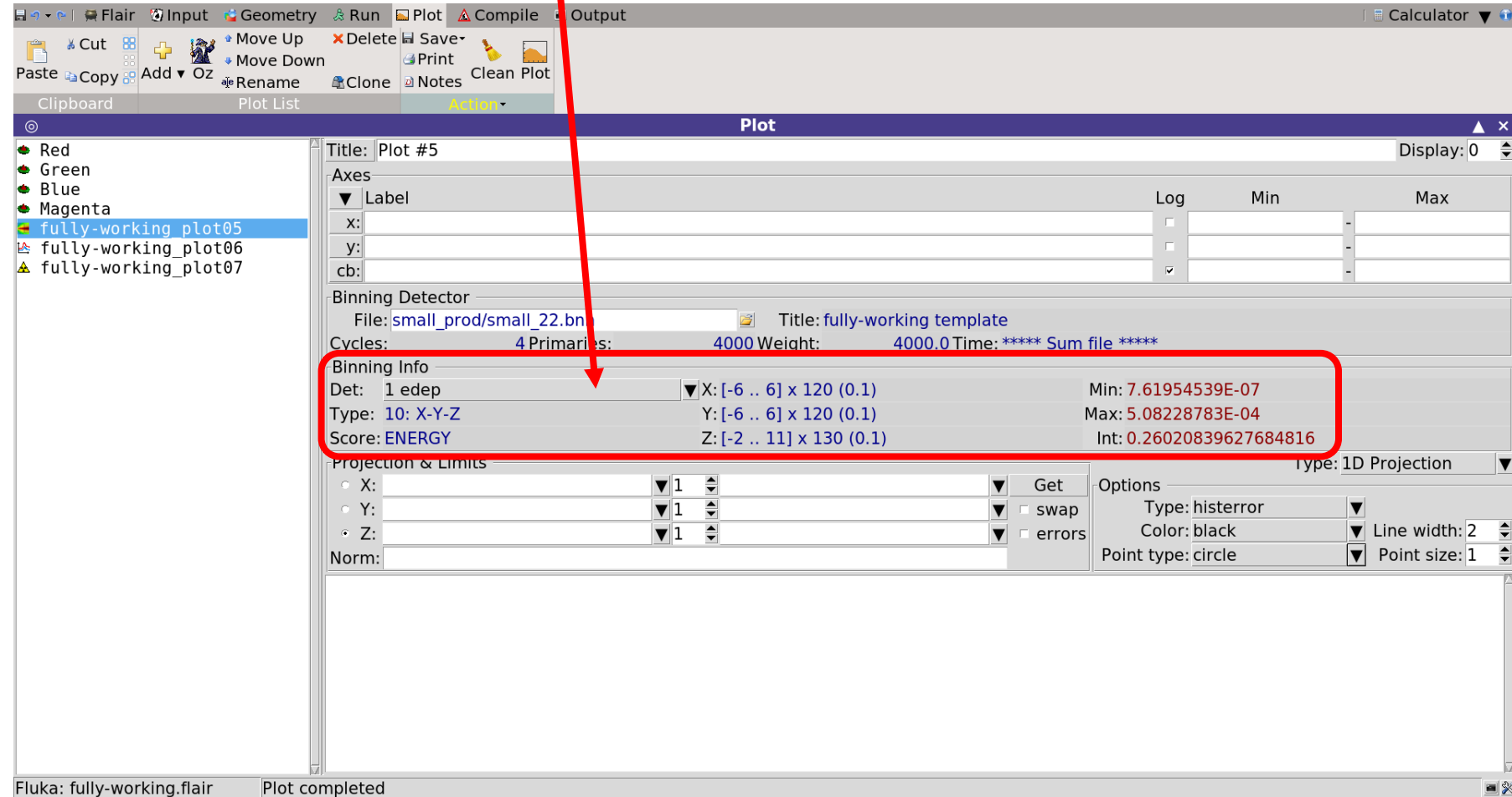
Plotting results in the Plot tab – 6

- 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

The screenshot shows the FLUKA Plot tab interface. A red arrow points to the 'Projection & Limits' section, which is highlighted with a red box. 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 icons for 'Cut', 'Copy', 'Paste', 'Add', 'Move Up', 'Move Down', 'Delete', 'Save', 'Print', 'Notes', 'Clean Plot', 'Rename', 'Clone', and 'Action'. The main window is titled 'Plot' and contains several sections: 'Axes' (with a table for X, Y, and Z axes), 'Binning Detector' (with fields for File, Title, Cycles, Primaries, Weight, and Time), 'Binning Info' (with fields for Det, Type, Score, and Int), 'Projection & Limits' (with radio buttons for X, Y, and Z, and a 'Norm' field), and 'Options' (with fields for Type, Color, Point type, Line width, and Point size). The status bar at the bottom indicates 'Fluka: fully-working.flair' and 'Plot completed'.

Label	Log	Min	Max
X:	<input type="checkbox"/>		
Y:	<input type="checkbox"/>		
Z:	<input checked="" type="checkbox"/>		

Projection & Limits

- X: 1
- Y: 1
- Z: 1

Options

- Type: histerror
- Color: black
- Point type: circle
- Line width: 2
- Point size: 1

Plotting results in the Plot tab – 9

Projection & Limits

X: 1 Y: 1 Z: 1

Norm:

Type: 2D Projection

Geometry

Use: -Auto-

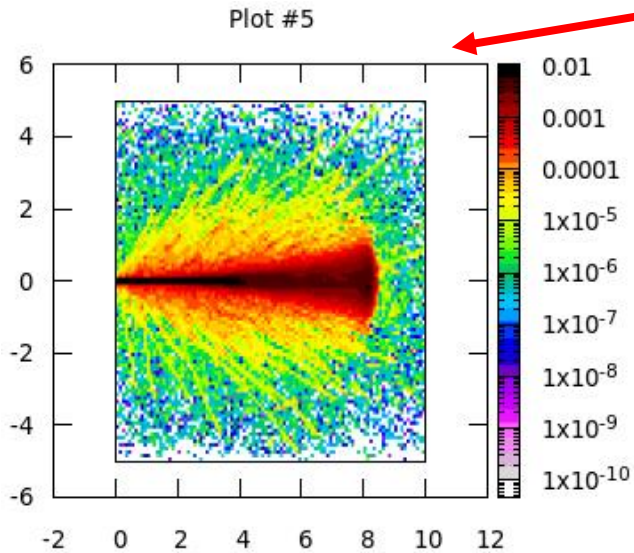
Pos:

Axes: Auto

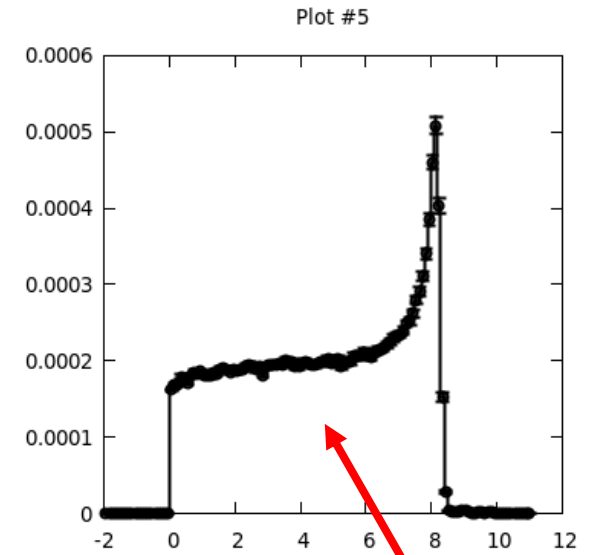
Get

swap

errors



- 2D vs 1D projections



Projection & Limits

X: 1 Y: 1 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 – 10

Projection & Limits

X: ▼ 1 ▼ Get

Y: ▼ 1 ▼ swap

Z: ▼ 1 ▼ errors

Norm:

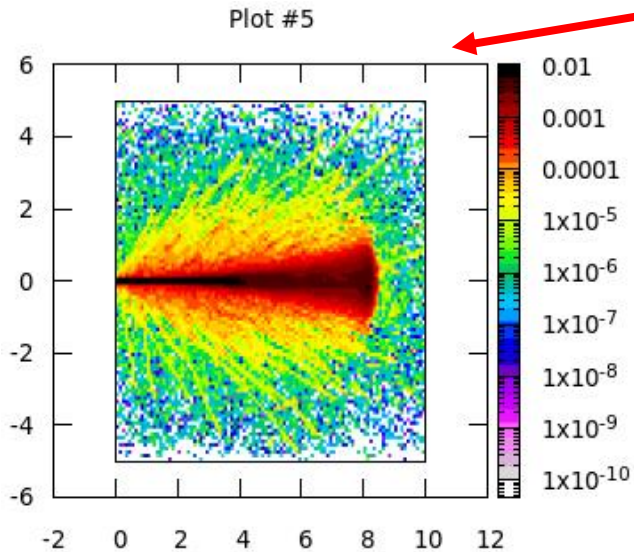
Type: 2D Projection ▼

Geometry

Use: -Auto- ▼

Pos:

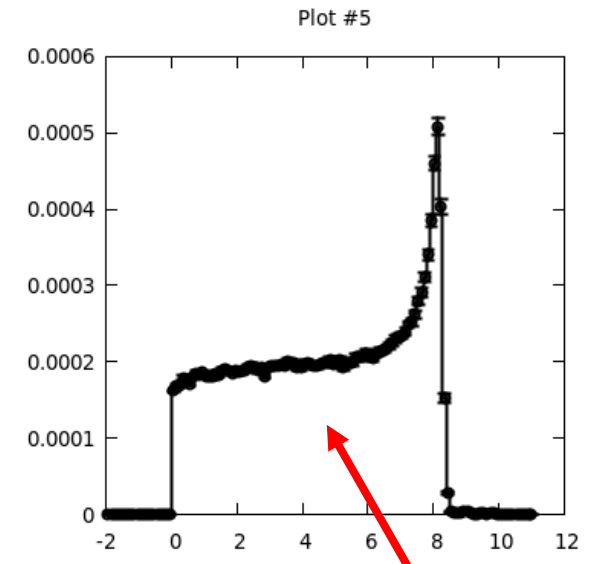
Axes: Auto ▼



- 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 – 11

- 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

<input type="radio"/> X:	▼ 1 ▲	▼	Get
<input type="radio"/> Y:	▼ 1 ▲	▼	<input type="checkbox"/> swap
<input checked="" type="radio"/> Z:	▼ 1 ▲	▼	<input type="checkbox"/> errors

Norm:

Type: 1D Projection ▼

Options

Type: histerror ▼	Line width: 2 ▲
Color: black ▼	Point size: 1 ▲
Point type: circle ▼	

Plotting results in the Plot tab – 12

Projection & Limits

X: 1

Y: -0.2

Z: 1

Norm:

Type: 2D Projection

Geometry

Use: -Auto-

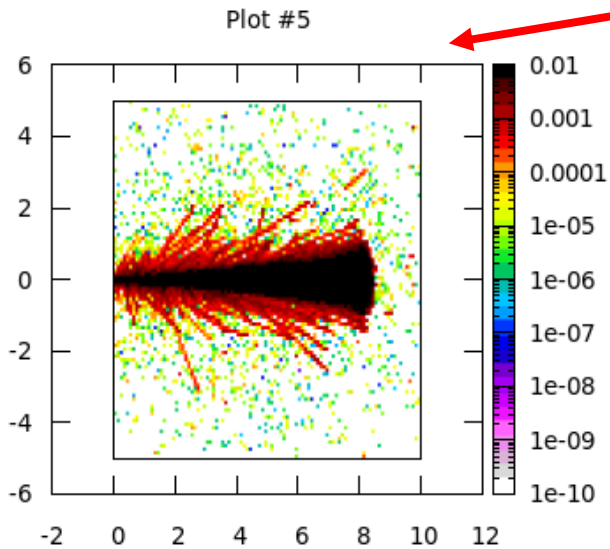
Pos:

Axes: Auto

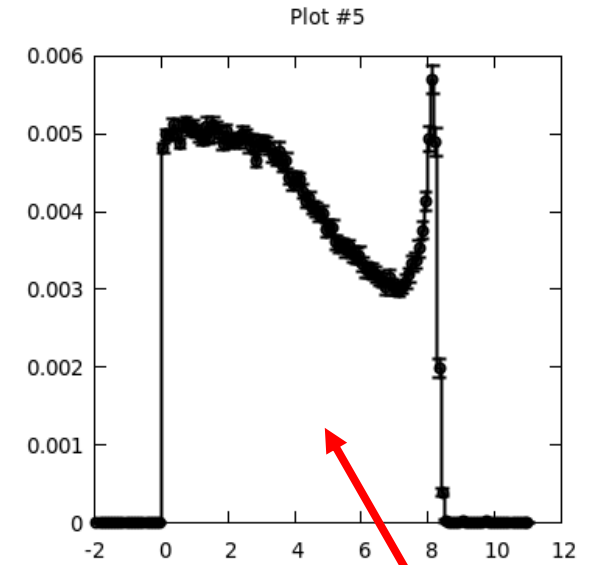
Get

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 – 13

Projection & Limits

X: 1

Y: 1

Z: 1

Norm:

Type: 2D Projection

Geometry

Use: -Auto-

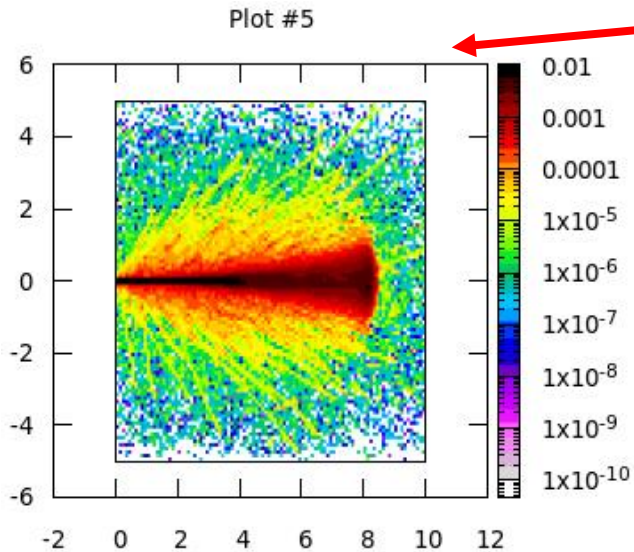
Pos:

Axes: Auto

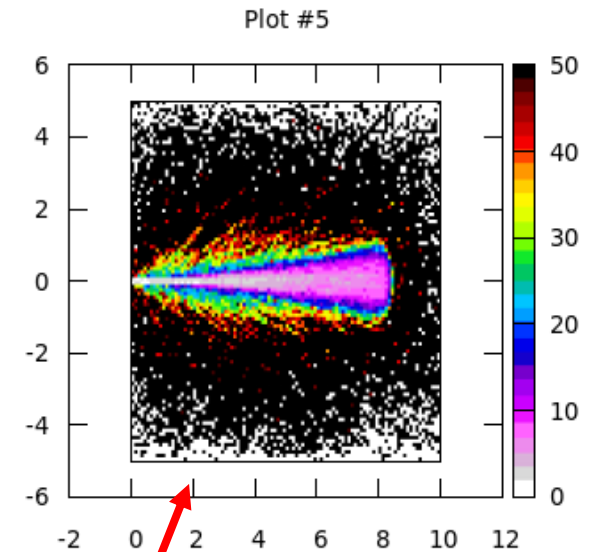
Get

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

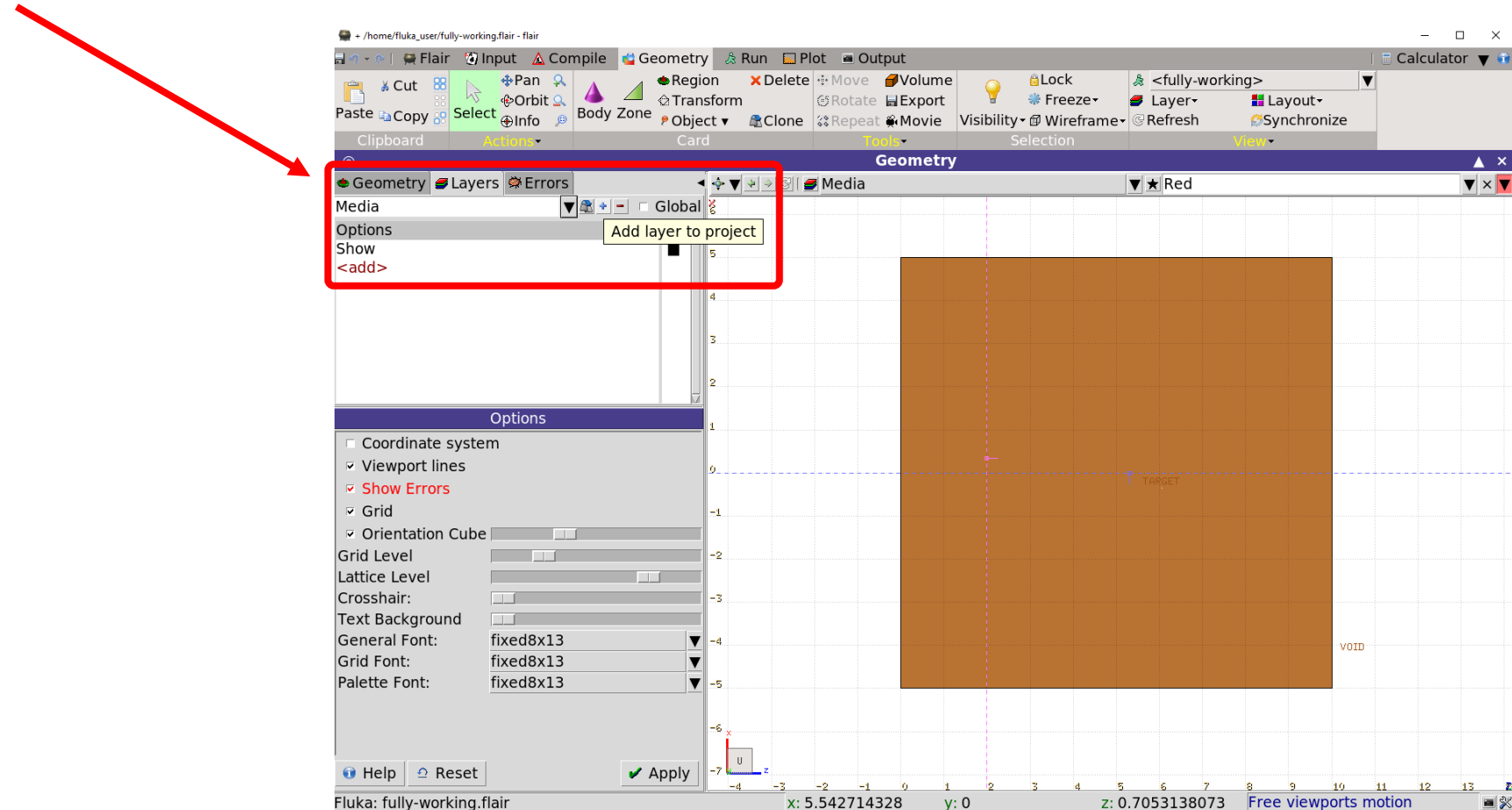
Get

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



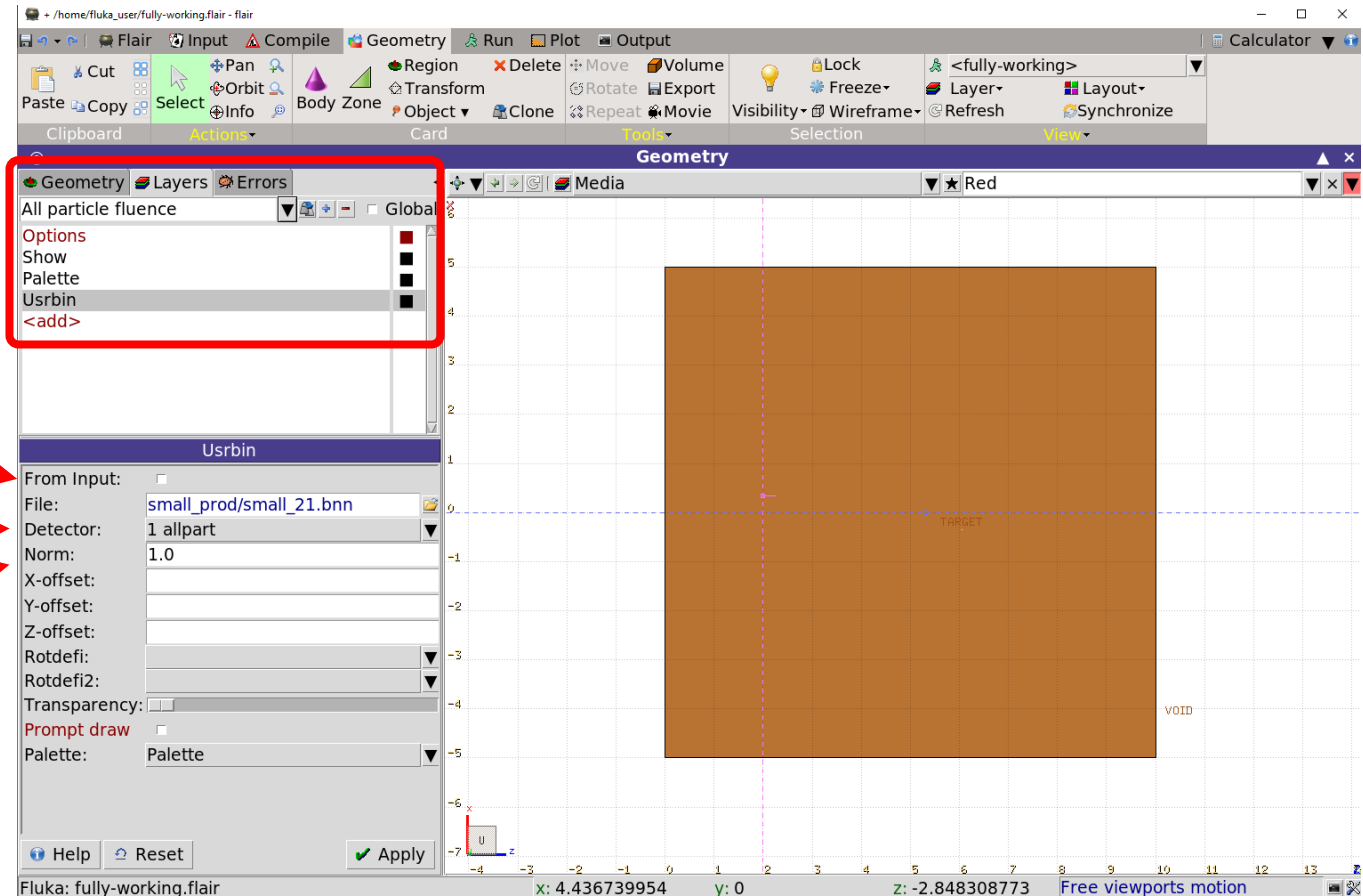
Plotting results in the Geometry tab – 2

- 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”

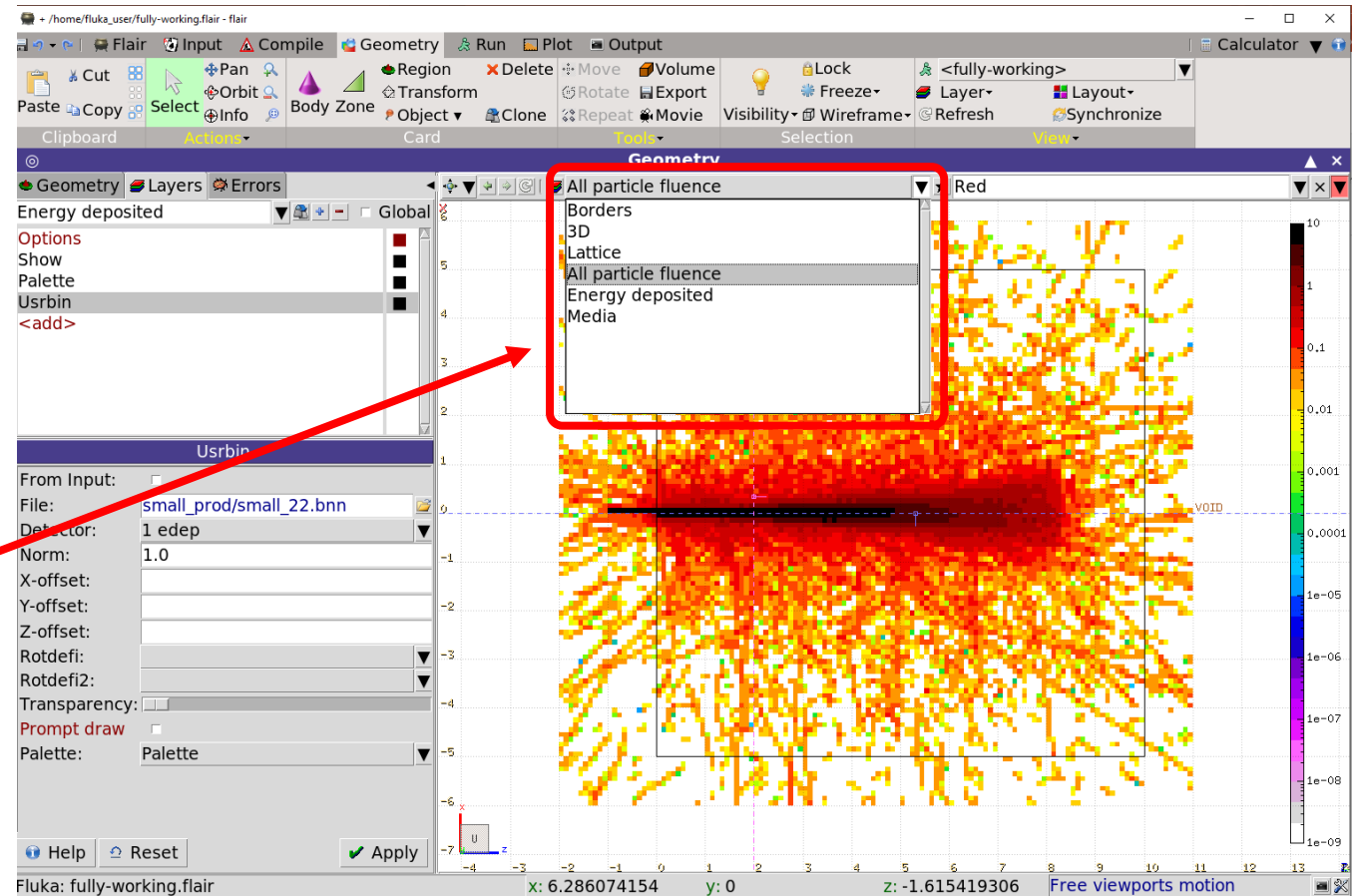
(possible to add more than one)

- 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!

The screenshot shows the Fluka software interface with the 'Input' tab selected. The main window displays the input file content, which includes a title, defaults, beam characteristics, beam position, and geometry definitions. The error message in the bottom right corner reads: 'Error loading USRBIN: Unable to load usrbn detector 'test-dir/test_22.bnn' 1'. The status bar at the bottom indicates 'Fluka: course_basic_handson; Current:1 Selected:1 Total:23'.

```
TITLE course basic input hands on
Set the defaults for precision simulations
DEFAULTS : PRECISIO
Define the beam characteristics
BEAM
  Beam: Momentum
  p: 0.8
  Part: PROTON
  Δp: Flat
  Δφ: Flat
  Shape(X): Rectangular
  Δx:
  Shape(Y): Rectangular
  Δy:
Define the beam position
BEAMPOS
  x: 0.
  y: 0.
  z: -1.
  cosx:
  cosy:
  Type: POSITIVE
  Paren:
  Geometry:
  Out:
  Fmt: COMBNAME
GEOBEGIN
  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
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
TITLE
course basic input hands on
```

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



