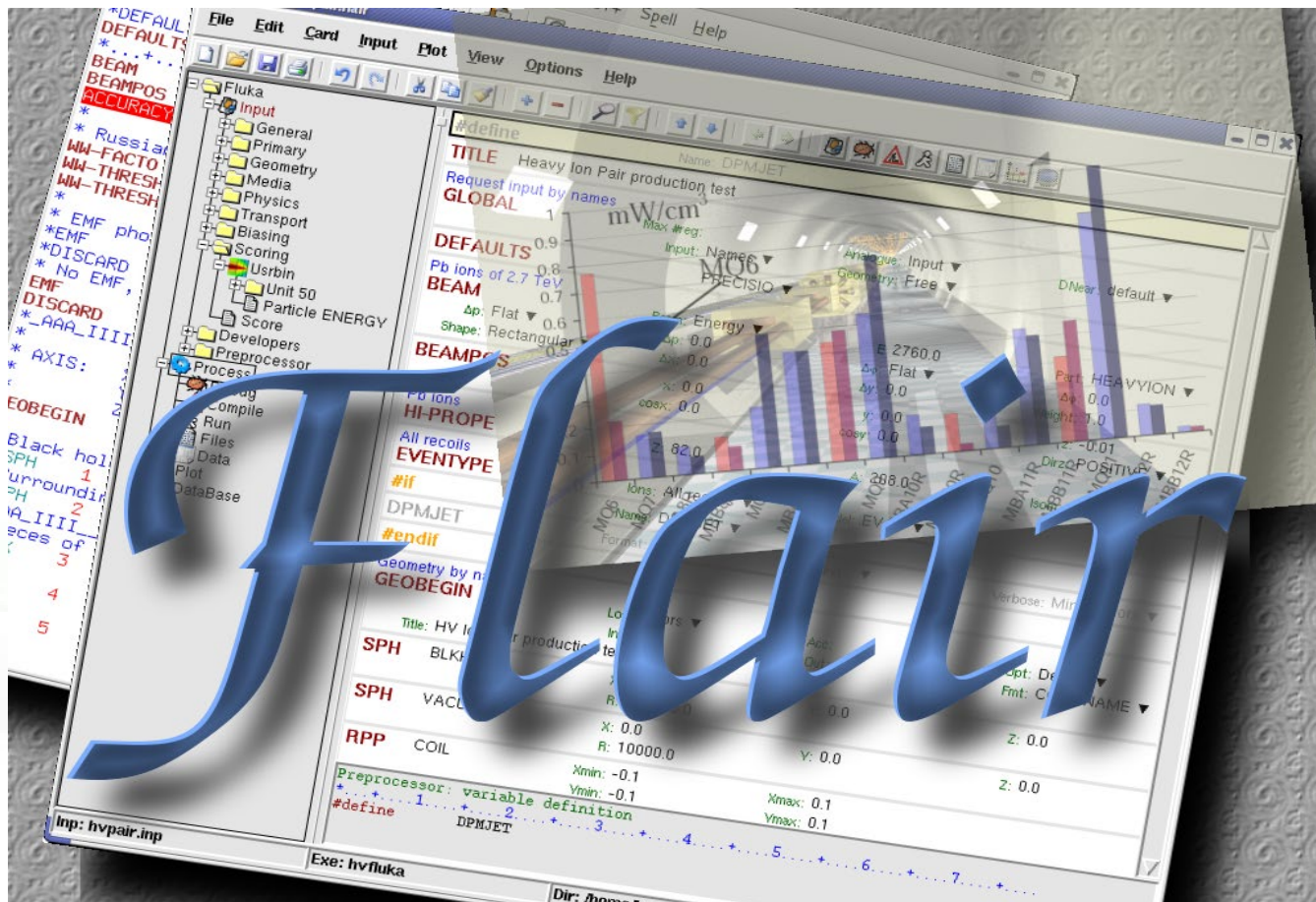




# Flair Advanced Features

Advanced FLUKA Course

# About



/fleə(r)/ n [U,C] natural or instinctive ability (to do something well, to select or recognize what is best, more useful, etc.

[Oxford Advanced Dictionary of Current English]

# What is flair [1/2]

**FLUKA Advanced Interface** [<http://www.fluka.org/flair>]

- **All-in-one** User friendly graphical Interface;
  - Minimum requirements on additional software;
  - Working in an intermediate level
- Not hiding the inner functionality of FLUKA**

## Front-End interface:

- Fully featured **Input file Editor**
  - Mini-dialogs for each card, allows easy and almost error free editing
  - Uniform treatment of all FLUKA cards
  - Card grouping in categories and card filtering
  - Error checking and validation of the input file during editing
- **Geometry:** interactive visualization editing, transformation, optimizations and debugging (next talk);
- **Compilation** of the FLUKA Executable;
- **Running** and **monitoring** of the status of a/many run(s)

# What is flair [2/2]

## Back-End interface:

- Inspection of the output files (core dumps and directories)
- Output file(s) viewer dividing into sections
- Post processing (merging) the output data files
- Plot generation through an interface with **gnuplot**;

## Other Goodies:

- Access to FLUKA manual as hyper text
- Checking for release updates of FLUKA and flair
- Nuclear wallet cards
- Library of materials
- DICOM images conversion to VOXEL geometries
- PET automatic geometry creator
- Everything is accessible with keyboard shortcuts

# Concepts: Flair Project

- Store in a **single text file** all relevant information:
  - Project notes
  - Links to needed files: **input file**, **source routines**, **output files** ...
  - **Multiple runs** from the same input file, as well running status
  - Procedures on how to **run the code**
  - **Rules** on how to perform **data merging**
  - Information on how to post process and **create plots** of the results
- You can consider Flair as an **editor** for the project files.
- Can handle any FLUKA input format (reading & writing), but internally it works using the **names format** for the input, **free with names** for the geometry (Recommended way of working)
- The format is plain ASCII file with extension: **.flair**

**Note:** If you want to copy a project you need to copy also all linked files especially the input and source routines!

# Command line options

Usage: flair [options] <filename | filename.flair | filename.inp>

Options:

- **-l** List recent projects
- **-r** Load most recent project
- **-R #** Load recent project (number 1..10 or filename)
- **-i inputfile** Fluka input file (w/o the .inp extension)
- **-1** Load the first flair file in the folder
- **-e exe** Use exe as fluka executable
- **-s** Skips the startup-dialog-splash screen
- **-d/D** Activate/Deactivate the beta-development features  
(use with care...)



# Interface

Tabs drag to rearrange or to undock

The screenshot displays the Flair software interface. At the top, a ribbon menu contains various tool icons and options. Below the ribbon is the 'Input' window, which shows a list of parameters and their values, including #define statements, TITLE, GLOBAL, BEAMPOS, and GEOBEGIN. A 'Dynamic Tab' is visible in the top right corner. Below the input window is the 'Flair' window, which displays a heatmap visualization of the loaded project. The heatmap shows a distribution of values across a grid, with a color scale ranging from 1 to 100. A green notification box at the bottom right of the heatmap area provides project information: Project: ntof33.flair, Input: ntof33, Dir: /home/bnv/prg/physics/fluka/flair/examples. The status bar at the bottom of the interface shows 'Inp: ntof33.inp', 'Card:1', 'Displayed:80', and 'Total:82'. A 'Quick access' icon is also present in the bottom right corner.

- Common interface for all frames/pages
- Dockable windows + Possibility to open as external window
- Fully User customizable

# Interface - Multi docking

The screenshot displays the ntof33.flair - flair software interface, which is multi-docked. The main window is titled "ntof33.flair - flair" and features a menu bar with "Flair", "Input", "Geometry", "Run", and "Plot". Below the menu bar is a toolbar with various icons for file operations, editing, and execution.

The interface is divided into several docked panels:

- Input Panel:** Contains a tree view on the left showing a hierarchical structure of folders like "General", "Primary", "Geometry", "Media", "Physics", "Transport", "Biasing", "Scoring", "Flair", and "Preprocessor". The main area shows a list of input parameters and their values, including:
  - #c:flair: 10
  - #definet2: 1
  - #definet3: 1
  - #definet4: 2
  - #definet5: 2
  - GLOBAL: Max #reg: Analogue: DNear: Input: Names Geometry: Free
  - DEFAULTS: NEW-DEFAULT
  - BEAM: Beam: Energy: E: 0.3 Part: P; Δp: Gauss Δp(FWHM): 0.082425 Δφ: Gauss Δφ: 1; Shape(X): Rectangular Δx: =2\*fwhm Shape(Y): Rectangular Δy: ; BEAMPOS: x: 2.2632 y: -0.5 z: ; cosx: .017364818 cosy: Type: P
  - GEOBEGIN: Log: Acc: Opt: ; Inp: Out: Fmt: C
  - Title: implied nTOF geometry
  - Black body: SPH BLKBODY: x: 0.0 y: 0.0 z: 0; R: 10000000.0
  - \$start\_translat: dx: =5\*2 dy: 3.0 dz: 2
  - Void sphere: SPH VOID: x: 0.0 y: 0.0 z: 0; R: 1000000.0
  - \$end\_translat
  - Water container: RPP WATERCNT: Xmin: -43.0 Xmax: 43.0; Ymin: -53.6 Ymax: 53.6; Zmin: -32.5 Zmax: 35.0
  - Lead Target: RPP LEADTARGET: Xmin: 40.0 Xmax: 40.0
  - \*+...+1...+2...+3...+4...+5...+

- Run Panel:** Contains a table with columns "Run", "Spawn", and "Usrxxx". The "Run" column lists "1/ntof33". The "Usrxxx" column lists various detector types and their units:
 

Run	Detector	Type	Unit
<ntof33>	ntof33_usrbin_50	usrbin	50
<ntof33>	ntof33_resnuclei_51	resnuclei	51
<ntof33>	ntof33_usrbdx_52	usrbdx	52
<ntof33>	ntof33_usrcoll_53	usrcoll	53
<ntof33>	ntof33_usrtrack_54	usrtrack	54
<ntof33>	ntof33_usrtrack_55	usrtrack	55
<ntof33>	ntof33_usrbin_56	usrbin	56
- Output Panel:** Contains a table with columns "Type", "Process", and "Status". The "Type" column lists various components like "flair", "Data", "Geometry", "USRBIN", and "ntof33\_usrbin\_50". The "Process" column lists "flair", "Merge", "ntof\_geom", "ntof\_geom2", "enedep", "ntof\_smallbin", and "ntof33\_usrbin\_50". The "Status" column lists "Idle", "Finished", and "Idle".
 

Type	Process	Status
flair	flair	Idle
Data	Merge	Finished
Geometry	ntof_geom	Idle
Geometry	ntof_geom2	Idle
USRBIN	enedep	Idle
USRBIN	ntof_smallbin	Idle
ntof33_usrbin_50	ntof33_usrbin_50	Idle

The status bar at the bottom shows "Input: ntof33.inp" and "Files: 45".



# Interface

## Keyboard:

Almost everything is possible with the keyboard see manual for shortcuts

**Ctrl-Enter:** Execute most important action

**Ins/Del:** Add or Delete

## Mouse:

**right-click** anywhere to get a popup menu

## Listboxes:

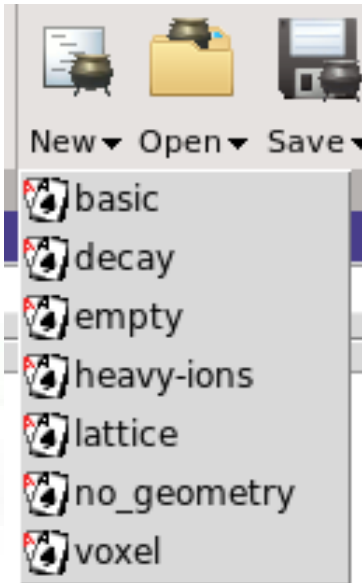
all listboxes are searchable. Typing only the characters (A-Z) and numbers (0-9) all other are ignored

## LabelFrames:

can be collapsed/expanded by clicking on the label

# Input Templates

- When requesting a new input or a new project flair will prompt to select an input template:




Default template: **basic.inp**

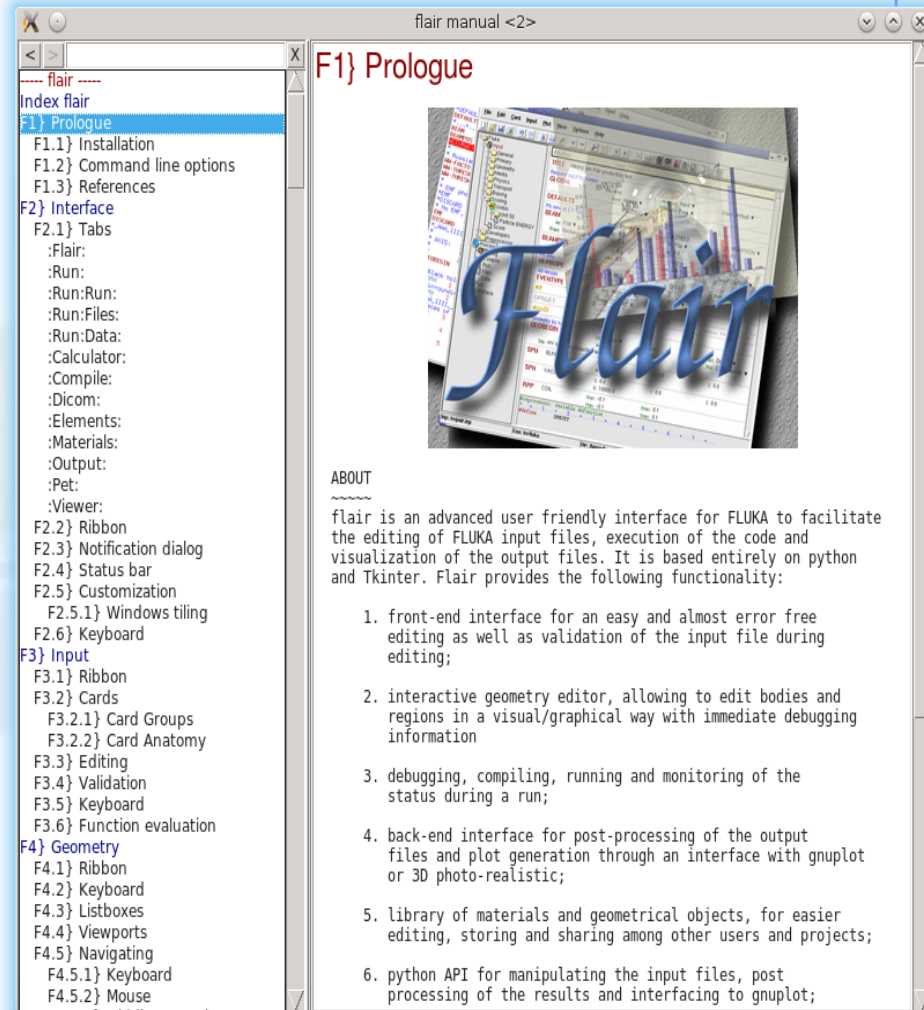
```

TITLE
GLOBAL                                1.0      1.0
DEFAULTS                                NEW-DEFA
BEAM
BEAMPOS
GEOBEGIN                                COMBNAME
      0      0
* Black body
SPH blkbody      0.0 0.0 0.0 1000000.0
* Void sphere
SPH void         0.0 0.0 0.0 1000000.0
* 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
RANDOMIZ     1.0
START
STOP
    
```

The user can create his own set of input templates. They are normal FLUKA input files and they have to be placed in the directory `~/flair/templates` (create the directory if not existing)

# Help

- [F1] or the help icon 
- Displays both the flair and the installed fluka manual
- Searchable:
  - **Filter** pages only with specific keywords (keeps history of keywords)
  - Type searching in the listbox
  - **Ctrl-F** to search for a keyword inside the displayed page



The screenshot shows a window titled "flair manual <2>". On the left is a table of contents with sections like "F1) Prologue", "F2) Interface", "F3) Input", and "F4) Geometry". The "F1) Prologue" section is selected and displayed in the main window. The main window contains the text "F1) Prologue" in red, followed by an image of the Flair software interface with the word "Flair" in large blue letters. Below the image is an "ABOUT" section describing Flair as an advanced user-friendly interface for FLUKA, listing its various functionalities.

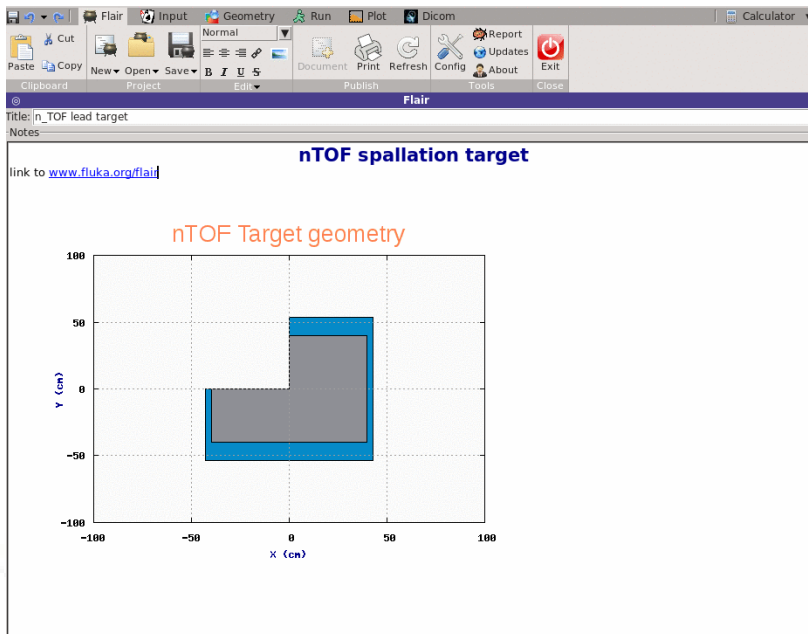
**F1) Prologue**

**ABOUT**

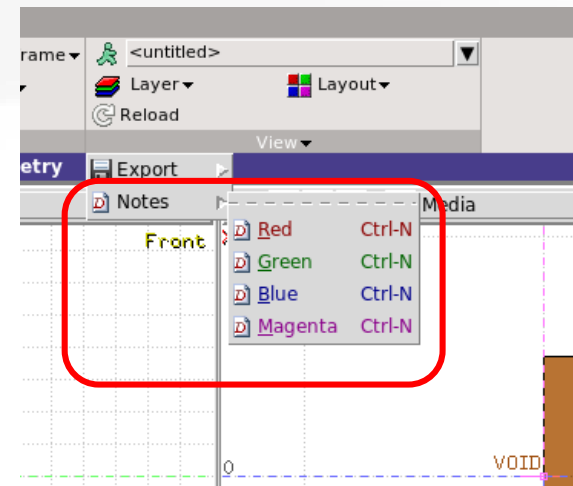
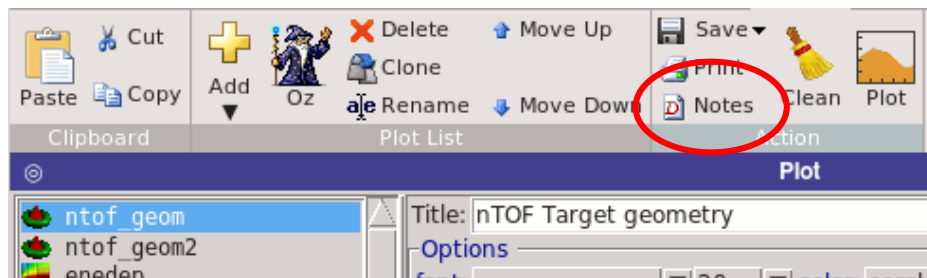
flair is an advanced user friendly interface for FLUKA to facilitate the editing of FLUKA input files, execution of the code and visualization of the output files. It is based entirely on python and Tkinter. Flair provides the following functionality:

1. front-end interface for an easy and almost error free editing as well as validation of the input file during editing;
2. interactive geometry editor, allowing to edit bodies and regions in a visual/graphical way with immediate debugging information
3. debugging, compiling, running and monitoring of the status during a run;
4. back-end interface for post-processing of the output files and plot generation through an interface with gnuplot or 3D photo-realistic;
5. library of materials and geometrical objects, for easier editing, storing and sharing among other users and projects;
6. python API for manipulating the input files, post processing of the results and interfacing to gnuplot;

# Page: Flair



- Notes, like a simple word processor. Accepts basic formatting, images and URLs to external documents.
- Any plot or geometry viewport can be inserted as an image inside the notes



# Input

## Material Database

## Display Active Cards of A Run Configuration

## Variables for parametric runs

## Hide Card Interpretation

## Manual Editing

The screenshot displays the Geant4 Input Editor interface. The top toolbar includes icons for File, Edit, and View operations. The main window shows the 'Input' card with the following content:

```
##define lang : -10
##define test2 :
##define test3 : 1
TITLE n_TOF lead target
##define test4 : 2
##define CUT :
GLOBAL Max #reg: Analogue: DNear:
Input: Names Geometry: Free
DEFAULTS NEW-DEFAULT
BEAM Beam: Energy E: 0.3 Part: PROTON
Delta p: Gauss Delta p(FWHM): 0.082425 Delta phi: Gauss Delta phi: 1.7
Shape(X): Rectangular Delta x: =2*fwhm Shape(Y): Rectangular Delta y:
BEAMPOS x: 2.2632 y: -0.5 z: -10.0
cosx: .017364818 cosy: Type: POSITIVE
GEOBEGIN Log: Acc: Opt:
Inp: Out: Fmt: COMBNAME
Title: implified nTOF geometry
Black body
SPH BLKBODY x: 0.0 y: 0.0 z: 0.0
R: 1000.0
$start_translat dx: =5*2 dy: 3.0 dz: 2.0
Void sphere
SPH VOID x: 0.0 y: 0.0 z: 0.0
R: 100.0
$end_translat
Water container
RPP WATERCNT Xmin: -43.0 Xmax: 43.0
Ymin: -53.6 Ymax: 53.6
Zmin: -32.5 Zmax: 35.0
Lead Target
RPP PBTARGET Xmin: -40.0 Xmax: 40.0
Ymin: -40.0 Ymax: 40.0
Zmin: -30.0 Zmax: 30.0
RPP NICHE Xmin: -25.0 Xmax: 15.0
Ymin: -40.1 Ymax: 15.0
Zmin: -30.0 Zmax: 30.0
```

The bottom status bar shows the active cards: BEAM -0.3 -0.082425 -1.74.70964009 1.0PROTON.

# Input Editor – 2: Import / Export

## Importing

- **Input:** merge parts or entire input file with the current
- **Mcnp:** import mcnp geometry into FLUKA. (experimental)
- **Gdml:** import gdml geometries into FLUKA. Use of parenthesis on Macro bodies (e.g. polycone, xrtu...) which can lead to complicated geometry for FLUKA to handle. (experimental)

## Exporting

- **Gnuplot:** save active plot to a gnuplot script
- **Makefile:** create a makefile for compiling the executable
- **Mcnp:** save input in MCNP format: Geometry, Materials, Importances
- **Povray:** save geometry into povray 3D format



# Input Editor - 3

- Drag'n'drop from the TAG of the cards
- Double click on card TAG to select all similar cards
- Editing multiple cards: select cards and modifying the value in one card will propagate the change to all similar selected cards
- Ctrl-Double-Click Show/Hide selected cards
- #if..#endif, \$transform, \$translat or \$expand flair will enclose the selected cards with the #if #endif, or \$start\_xxx, \$end\_xxx transformation cards
- Popup Balloon tooltip displays short help:
  - for every option on every card
  - body description in the REGION expression
- Right-click: shows popup-menu
  - Quick filtering by REGION, MATERIAL, scoring etc...
- Easter Eggs: AWARI by Double-Right-Click on dialog showing the card representation as text at the bottom of the screen

# Input Editor – 4

- Automatic indentation of nested **#if..#endif** directives.
- To refresh the display press **Ctrl-R**
- Each **REGION** can be split into many cards if needed to be used with preprocessor commands.
- Use as a name **"&"**

Void around

**REGION VOID**

Neigh: 5

Volume:

Expr: +void -target

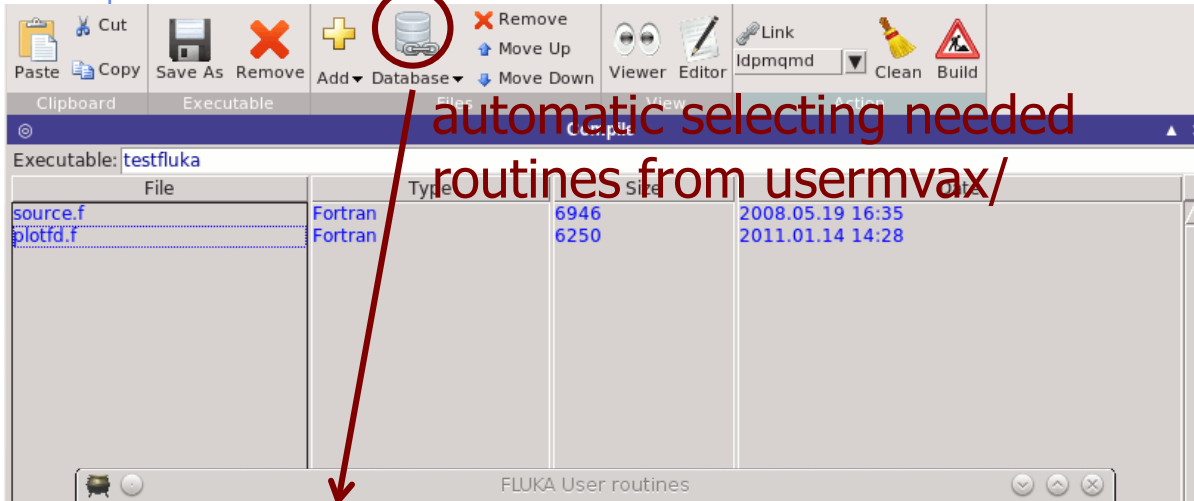
**#if** BIAS ▼

**REGION &**

cont: -bias

**#endif**

# Compiling



automatic selecting needed routines from usermvax/

File	Size	Date	Desc
comscw.f	5257	2011.03.24 13:19	response functions, user dependent selection for density-like quantities
dffcff.f	1469	2011.03.24 13:19	diffusion coefficient (for optical photons)
endscp.f	4055	2011.03.24 13:19	energy density distributed - change of positions
fldscp.f	3418	2011.03.24 13:19	fluence distributed - change of positions
fluscw.f	4201	2011.03.24 13:19	response functions, user dependent selection for flux-like quantities
formfu.f	2488	2011.03.24 13:19	nuclear charge form factors
frghns.f	1463	2011.03.24 13:19	material roughness (for optical photons)
ftelos.f	1553	2011.03.24 13:19	this routine is called at the very end of a FLUKA run
fusrbv.f	1379	2011.03.24 13:19	defines a continuous variable for 3-D binnings
lattic.f	8417	2014.02.03 18:33	symmetry transformation for lattice geometry
lusrbl.f	1369	2011.03.24 13:19	defines a discrete variable for 3-D binnings
magfld.f	2301	2011.03.24 13:19	to use a magnetic field map
mdstck.f	1746	2014.02.03 18:33	management of secondary stack
mgdraw.f	14935	2014.02.03 18:33	to dump trajectories, etc.
musrbr.f	1367	2011.03.24 13:19	defines a discrete variable for 3-D binnings
ophbdx.f	1799	2011.03.24 13:19	boundary crossing properties (for optical photons)
pshckp.f	1274	2011.03.24 13:19	Push Cerenkov photon
queffc.f	1605	2011.03.24 13:19	quantum efficiency (for optical photons)
rflectv.f	1469	2011.03.24 13:19	reflectivity (for optical photons)
rfrndx.f	1469	2011.03.24 13:19	refraction index (for optical photons)

Filetypes accepted:

- Fortran: .f, .F, .for, .FOR
- C/C++: .c, cpp, .cxx, .cc
- Libraries: .a, .so

Automatic scanning of necessary user routines and copying them to project folder.

**Build:** behaves like a "makefile" compiles based on files timestamp when are newer

**Compile:** Forces compile of the selected files

**Clean:** cleanup of all produced files

When you are unsure, click on "Clean" before "Build"

# Running

Run loops over a variable

Clean all run generated files

The screenshot shows the FLUKA GUI. The 'Run' dialog is open, displaying the following fields:

- Title: <foo>
- Primaries: 5000
- Rnd: 2
- Time: 0
- Exe: (empty)
- Defines: Default Defines

The 'Defines' table is as follows:

Name	Value
ENERGY	1.0

The 'Run' list shows the following runs:

Run	Spawn
<foo>	3
footest	
footest_01	
footest_02	
footest_03	
footest2	2
footest2_01	
footest2_02	

The 'Progress' bar at the bottom shows:

- Status: Running
- Input: footest\_02
- Dir: fluka\_13710
- Started: 2014.01.23 11:24
- ETA: 2014.01.23 11:24
- Time/prim: 3.31429 ms
- Elapsed: 4.64 s
- Cycle: 11.9314 s
- Run: 11.9314 s
- Cycles: Current: 2 [0 - 2] Completed: 50%
- Primaries: Current: 1401 [0 - 5000] Completed: 28%

Multi core runs

Custom preprocessor #defines even with value/function

Multiple editing is Allowed

<inputname> refers to the input file AS IT IS in the input editor.

Create additional runs based on the same input file by overriding:

- Title
- Preprocessor definitions
- Random number seed
- Starting particles
- Execution timeout
- Executable

- Monitors the status of the run by inspecting the FLUKA output files. If **timeout** occurs try to re-**Attach** to the running process.
- The timeout is user-definable in the **Preferences** dialog

# Running: How to use multicore CPU's

- Create clones of the current input e.g. **test.inp** named: test**1**.inp, test**2**.inp, test**3**.inp ...
- Assign a **different random number seed** on each run (Rnd entry). Incremental from the parent run.
- Select all in the listbox and click Run
- Check "**Preferences**" for changing the naming format

## Multiple Selection:

- To modify **many runs** at the same time, select them in the listbox
- The options will be "*disabled*"
- **Right-click** on the options you want to **enable** and modify them
- Modify the filters in Data processing for summing up all cycles from all runs (see later)

# Output Files

USRBIN to ASCII format

Files pattern

The screenshot shows the FLUKA GUI interface. The 'Files' menu is open, showing options like 'Paste', 'Copy', 'Run', 'Data', 'Add', 'Delete', 'Clone', 'Loop', 'Move Up', 'Move Down', 'Rename', 'Refresh', and 'Filter'. The 'Filter' text box is circled in red. The 'Run' window is also visible, showing a table of output files.

Run	Spawn	Cycles	File	Type	Size	Date
<foo>		001	footest_02001_fort.22	22	2514	2014.01.23 11:
footest	3	002	ranfootest_02001	-file-	1651	2014.01.10 14:
footest_01		003	footest_02001.err	Error	6930	2014.01.23 11:
footest_02		compile	footest_02001_fort.21	21	160238	2014.01.23 11:
footest_03		data	footest_02001.log	Log	9519	2014.01.23 11:
footest2	2	fluka_19879	footest_02001.out	Output	99326	2014.01.23 11:
footest2_01		input	footest_02.out	Output	1129	2014.01.23 11:
footest2_02		plot				
		temporary				

Inspect Output files generated by FLUKA classified per: **Run/Cycle**

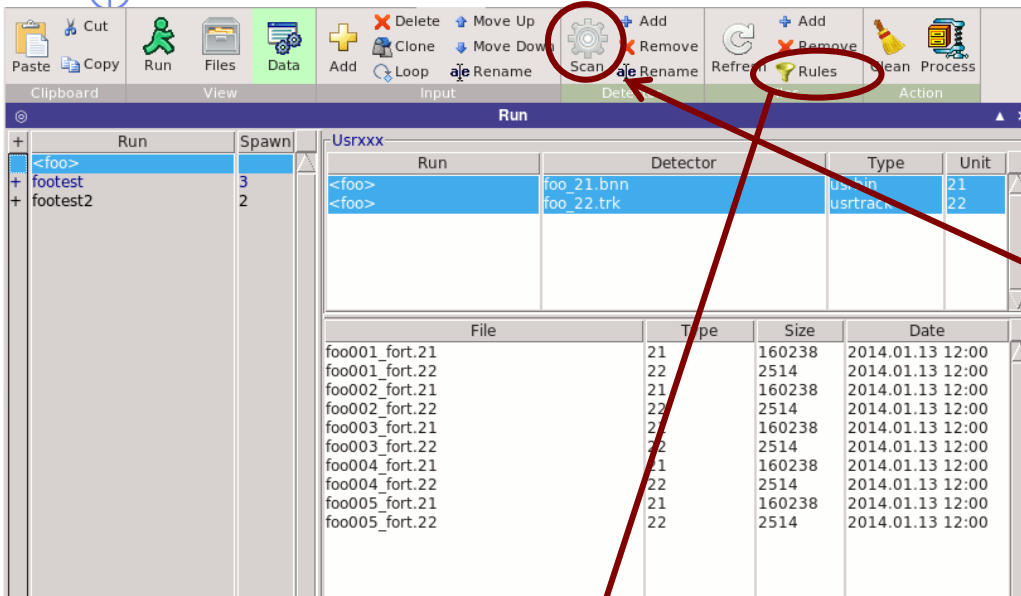
As well special output files from **compilation data processing plotting and temporary**

Double clicking opens:

- Files in the file Viewer
- coredumps in debugger



# Data Processing



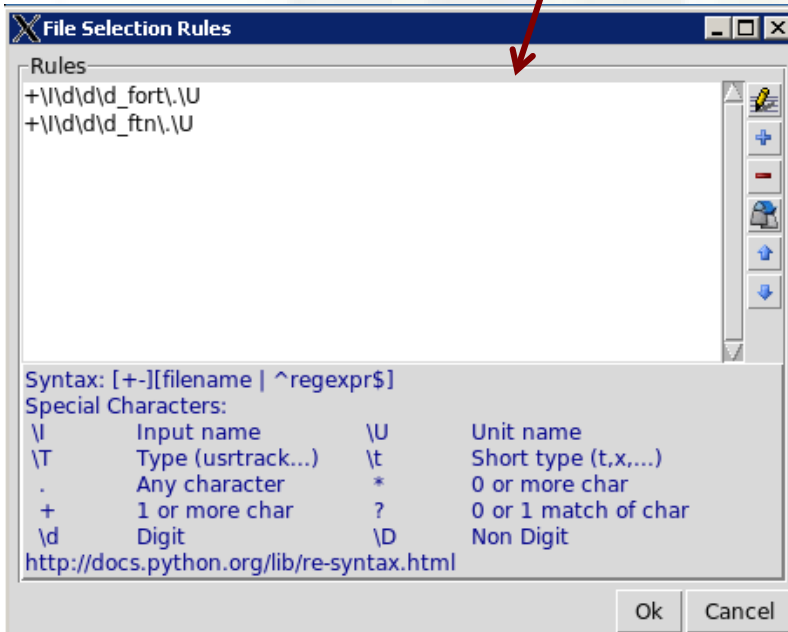
Process all scoring BINARY output files for each Run.

Name rules are defined in Preferences

Automatically scan input for scoring cards

+/- Modify file list by adding / removing items

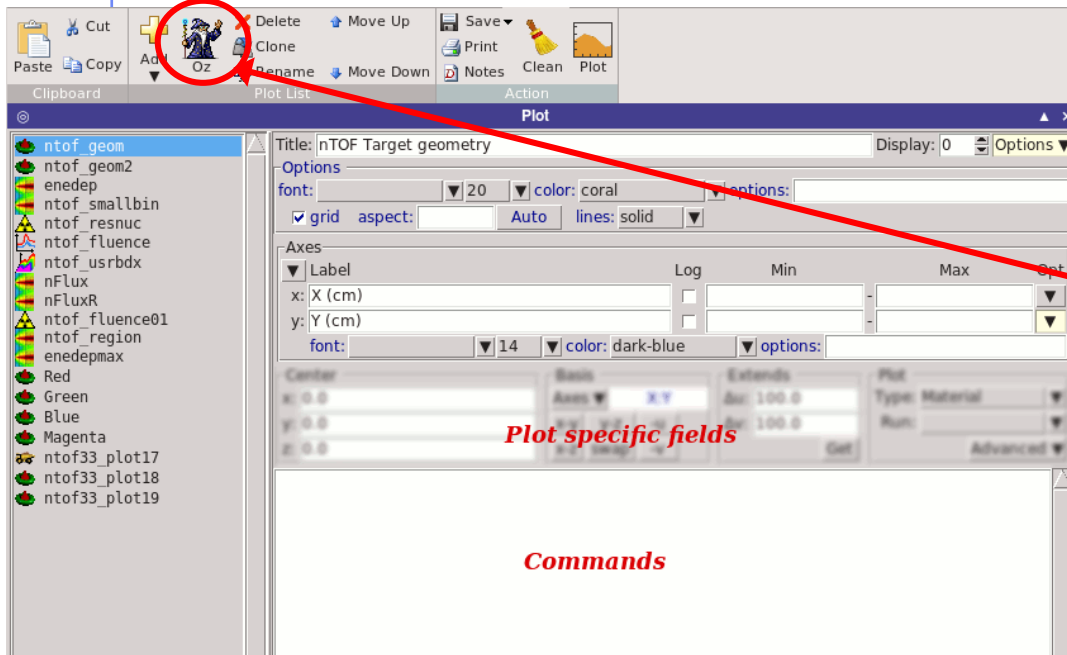
Dialog for editing scanning rules for files.



💡 To modify the rules for multiple scoring cards, select all Usrxxx before

💡 The default rules can be modified in the **Preferences Dialog**

# Plots



- It is important to set a **unique** filename for each plot. This filename will be used for every auxiliary file that the plot needs (the extension will change)

- The **Wizard** button creates automatically one plot for each processed unit (From the default input file)

- Multiple plots can be selected and modified. All changes on enabled fields will be propagated on selected plots

- Right-click** on disabled items to enable/disable the field during multiple editing

- All plots share the same header and footer

- Commands:** allows to enter gnuplot commands for further customization

## Plot Types

- **Geometry** For geometry plots
- **USRBIN** For plotting the output of USRBIN
- **USR-1D** To plot single differential quantities from cards **USRBDX, USRTRACK, USRCOLL, USRYIELD**
- **USR-2D** To plot double differential from **USRBDX**
- **RESNUCLE** To plot 1d or 2d distributions of **RESNUCLEI**

# General Tips

- In the Configuration Dialog you can set global commands to execute before or after any plot
- The **output page** displays all the commands that are sent to gnuplot. As well as the errors. In case of problem always consult the output window!
- Multiple displays can be used to compare plots
- Advanced options are displayed by clicking the **"Options"/"Advanced"** buttons
- Additional axes can be found in the **"Axes"** button
- In the **Gnuplot commands** you can fully customize the plot by adding manually gnuplot commands:
- Special commands:
  - **plot, splot** with no options, defines the order where flair should insert the plot or splot command.
  - **replot <plot-cmd>** append extra plots to the one generated by flair

# USRBIN Plots

- Normalization could be plotted:
- 2D projection, 1D projection
  - Trace of the maximum
  - Full width at half maximum

The screenshot shows the 'Binning Detector' window with the following settings:

- File:** ntof33\_usrbin\_50
- Title:** n\_TOF lead target
- Cycles:** 4
- Primaries:** 2432
- Weight:** 2432.0
- Time:** \*\*\*\*\*
- Sum file:** \*\*\*\*\*
- Binning Info:**
  - Det:** 1 EneDep
  - Type:** 10: X-Y-Z
  - Score:** ENERGY
  - X:** [-40 .. 40] x 100 (0.8)
  - Y:** [-40 .. 40] x 100 (0.8)
  - Z:** [-30 .. 35] x 100 (0.65)
  - Min:** 6.93416879E-10
  - Max:** 0.204449102
  - Int:** 6.18527894
- Projection & Limits:**
  - Type:** 2D Projection
  - Geometry:**
    - Use:** -Auto-
    - Pos:** -15
    - Axes:** Auto
  - Y-axis:** -0.8
  - Y-axis limits:** 1, 0.8
  - Buttons:** Get, swap, errors
  - Norm:** (empty)

Usrbin file

Detector

Rebinning

Swap axes

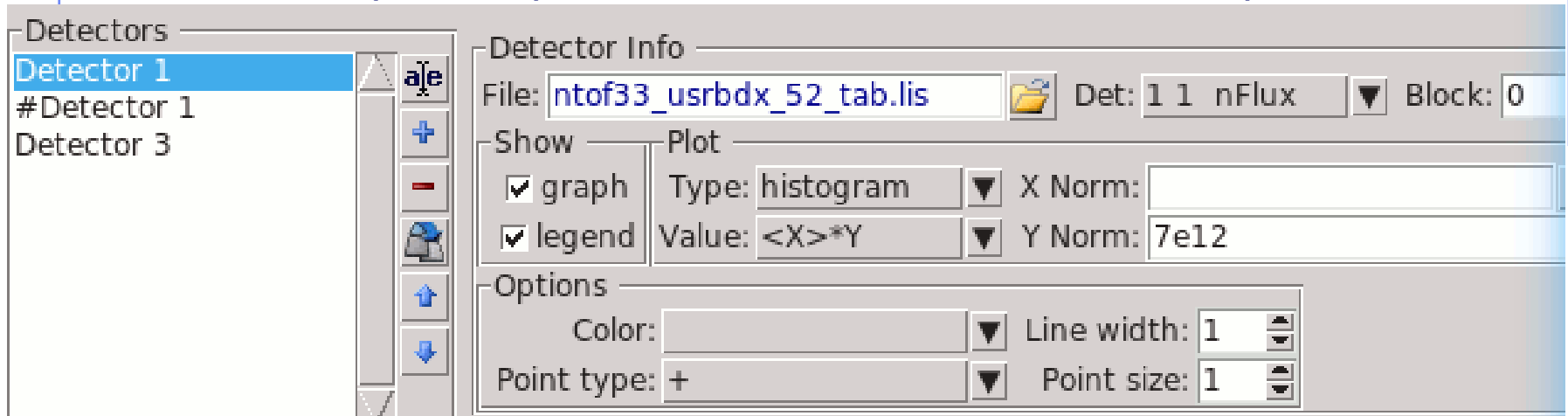
Get limits from gnuplot using right-mouse

Normalization could be used as:  
 number or expression evaluating in a number  $65e-3/2.7$   
 function with x as variable. e.g  $E2T(x*65e-3/2.7)-293$   
 with the function defined in the Gnuplot commands  
 $E2T(x) = ((3.00629e-08*x-0.000108436)*x+1.01097)*x+311.839$

Draw errors. (combined with log)  
 Correct only if one slice is used

# USR-1D

- Multiple 1D plots can be displayed: USRBDX, USRTRACK, USRYIELD as well the "plot" output from USRBIN, RESNUCLEI,... or experimental data



- Detectors:** listbox to add graphs
- File:** detector file name (`_tab.lis` or `.dat`)
- Det:** Detector name or index
- Block:** multi-dimensional information, splitted into blocks
- Type:** drawing type (histogram, histerrors, lines,...)
- Value:**  $Y$ ,  $\langle X \rangle * Y$ ,  $Dx * Y$
- X Norm:** to change units (1/eV,...)
- Y Norm:** normalization function like in USRBIN

# A final warning

- Pay attention to differences in new/old expressions format:  
Flair is not fully up-to-date with the latest Fluka and might raise warnings when loading a new .inp file
- Known «features»/bugs:
  - PHYSICS card issues:
    1. WHAT(1)=3 for SDUM=EVAP
    2. WHAT(6)=3 for SDUM=IONSPLITThese have to be manually edited if input with Flair
- Make sure you explicitly link to ldpmqmd (if needed), since Flair defaults to lfluka:  
ldpm2qmd and ldpm3qmd are the same as ldpmqmd!