Flair Advanced Features

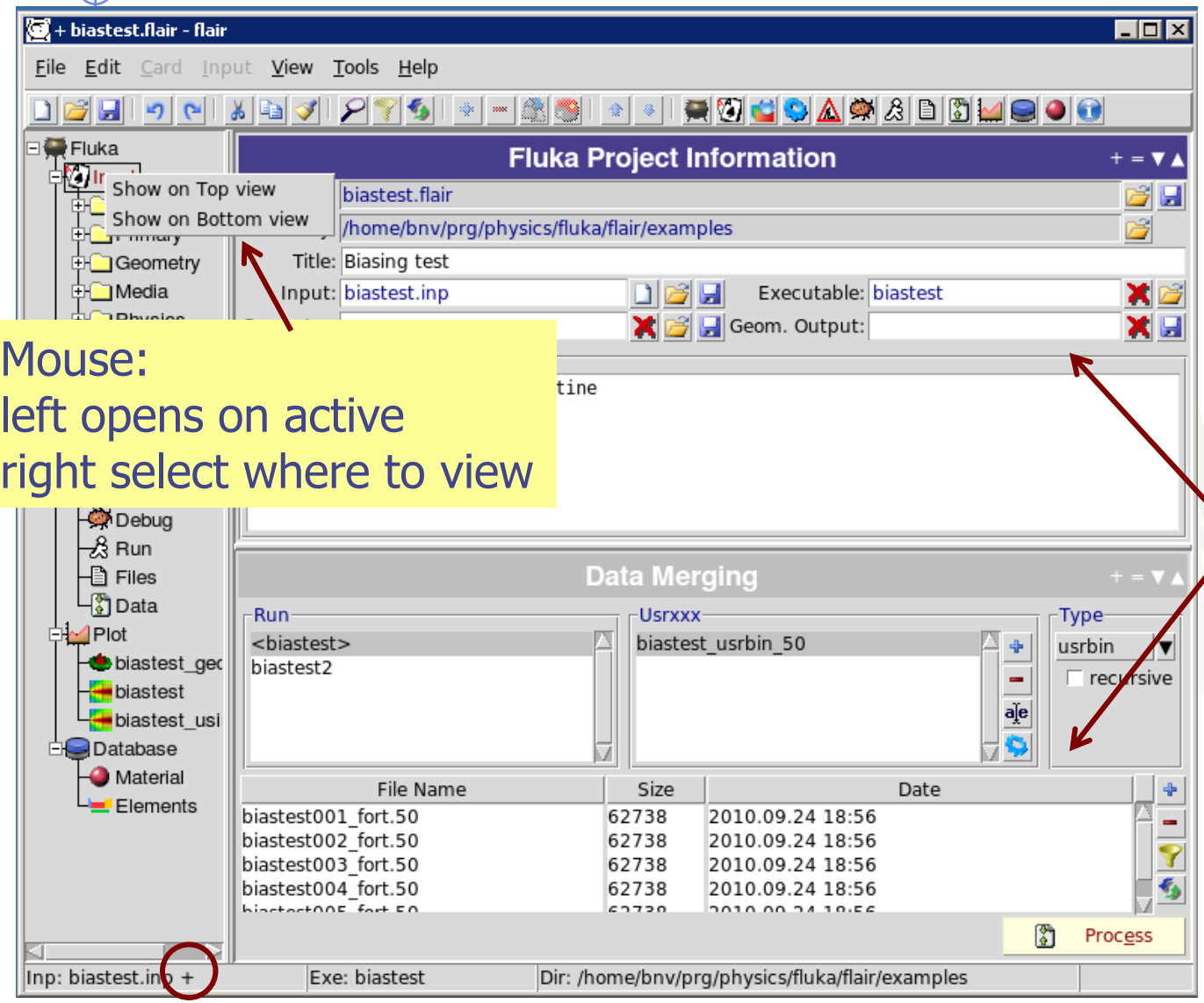
Advanced FLUKA Course

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*]

Interface



Mouse:
left opens on active
right select where to view

active

+ vertical/horizontal
= equalize
▼ minimize
▲ maximize

2 working frames

inactive
click to activate

input modified and not saved

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

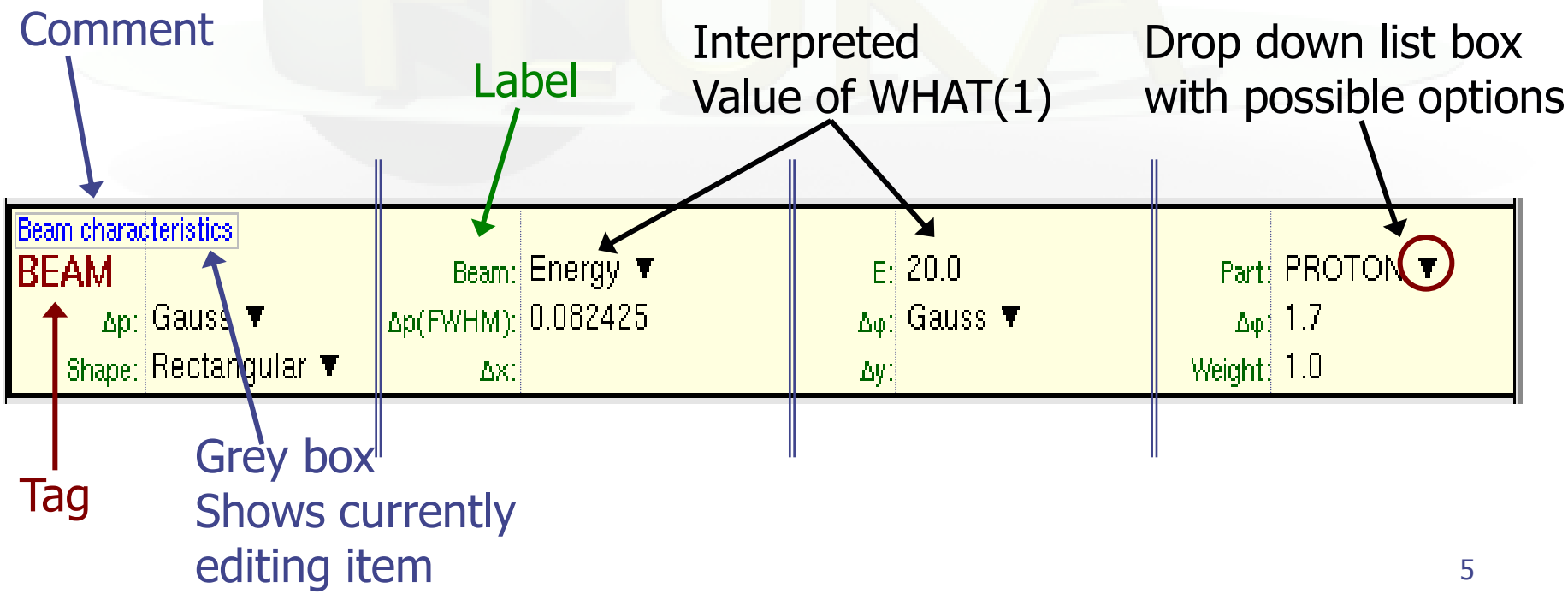
LabelFrames:

can collapse/expand by clicking on the label

Anatomy of a card mini-dialog

- For each extended card flair has a mini dialog (currently in 4 columns), interpreting all information stored in the card

```
* Beam characteristics
BEAM              -20.0 -0.082425      -1.7                              1.0PROTON
```



Input Editor - 1

highlight differences during editing

Input Editor - 2

- 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 modify 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

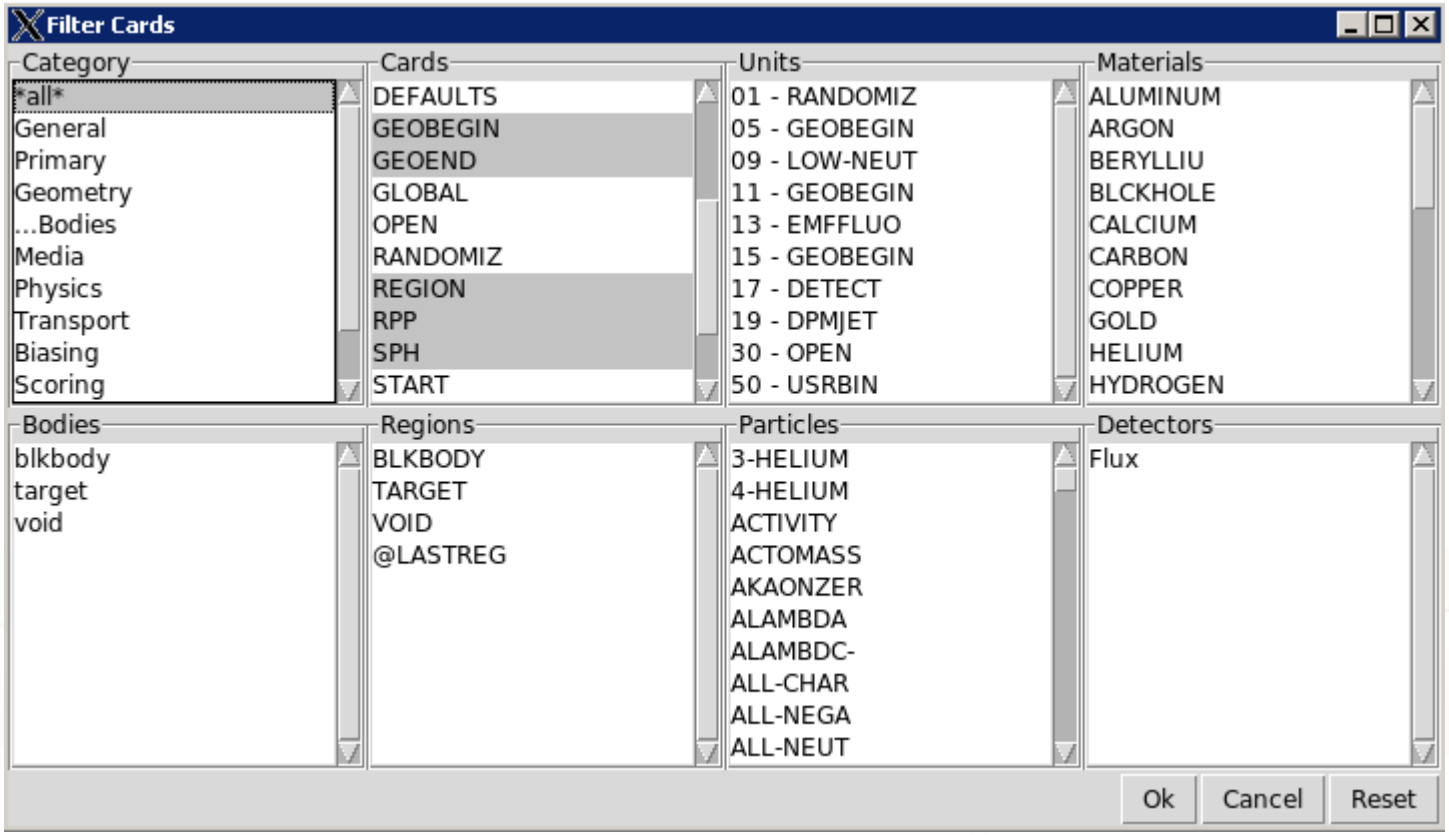
Input Editor - 3

- Automatic indentation of nested #if..#endif and $start..$end directives.
- To refresh the display type Ctrl-R

- Each REGION can be split into many cards if needed to be used with preprocessor commands.
- Use as a name "&"

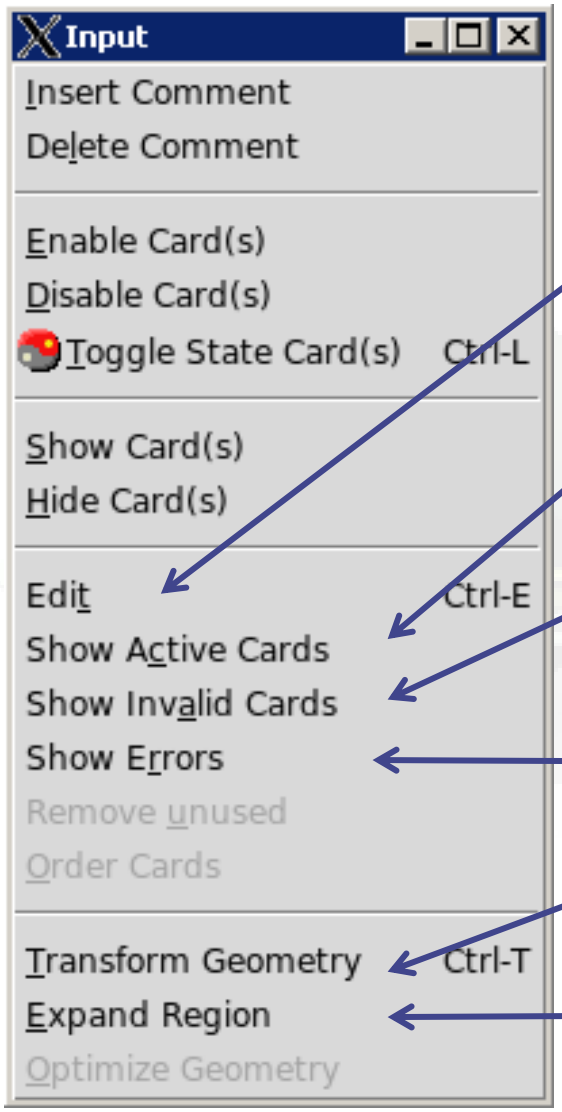
```
Void around
REGION VOID                                   Neigh: 5          Volume:
      Expr: +void -target
#if         BIAS ▼
    REGION &                  cont: -bias
#endif
```

Input Card Filtering



- Filter Cards dialog allows a more advanced selection of cards to be displaced, by showing only the cards that match the selected options

Input Menu



Manual editing of the card

Scan input and display only active cards (not excluded by the preprocessor)

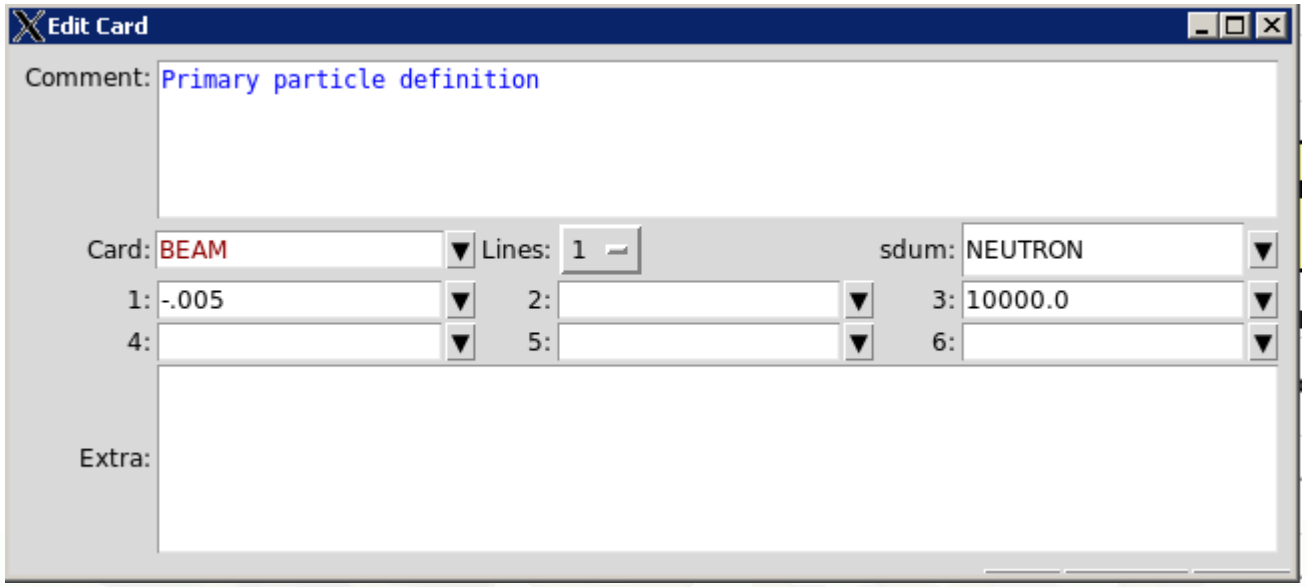
Show cards containing problems/errors

Display a message with the errors identified in the cards

Open the geometry transformation dialog

Expand parenthesis in the region (only logical optimization will performed)

Manual Card Editing



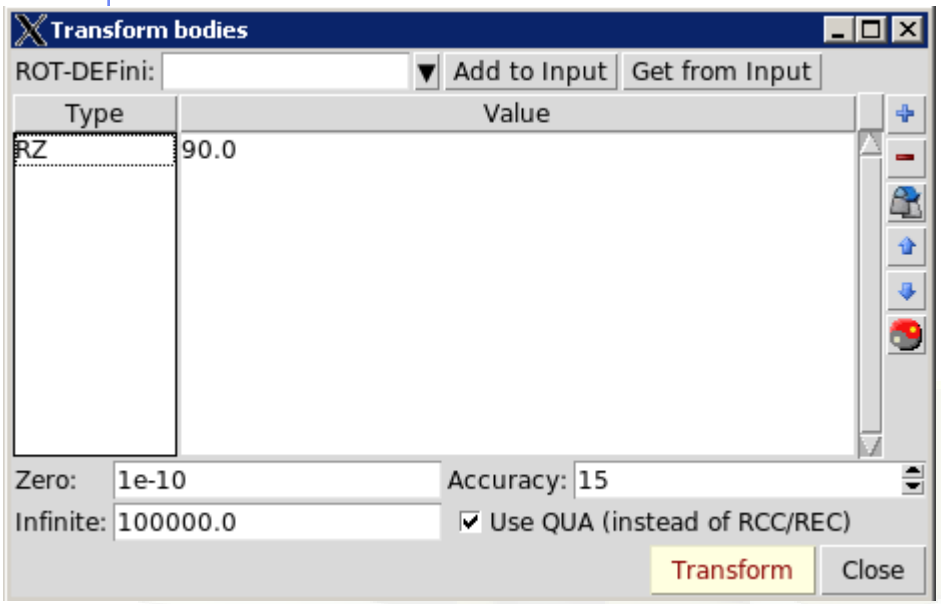
Accessible: Ctrl-E, right-click → Edit, Menu → Input → Edit

Lines: How many lines the card extends

Extra: additional information for a card like title string for TITLE, or region expression for REGION

Dropdown box: shows with categories all items defined in the input (bodies, regions, materials, particles…)

Bodies Transformation



Transformation Types:

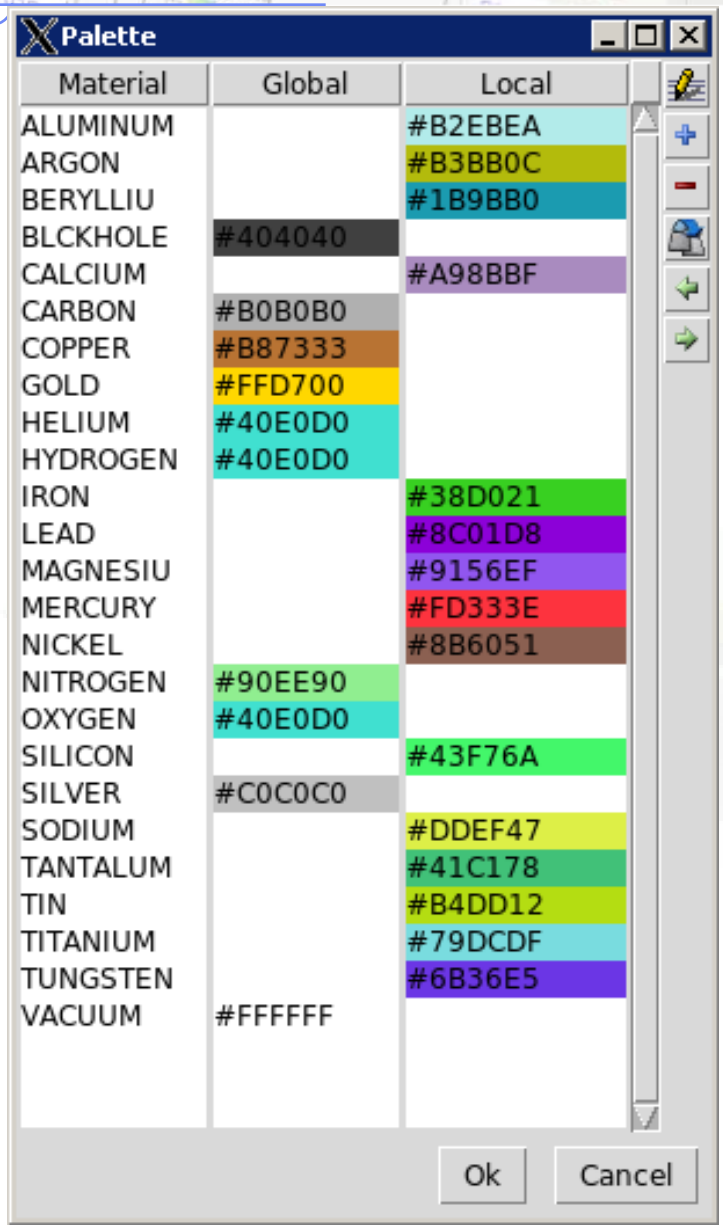
T	translate along a vector
TX TY TZ	translate along axis
RX RY RZ	axis rotation (degrees)
S	scaling

- Applies a user transformation to the selected bodies on the input editor.
- Convert transformations to/from ROT-DEFini cards
- Zero: limit below which to be considered as zero
- Accuracy: Numeric digits
- Infinite: infinite bodies when converted to which size to use
- Use QUA: convert infinite cylinders to infinite QUAdrics

Remember:

When transforming bodies for use with LATTICE card, use the maximum precision

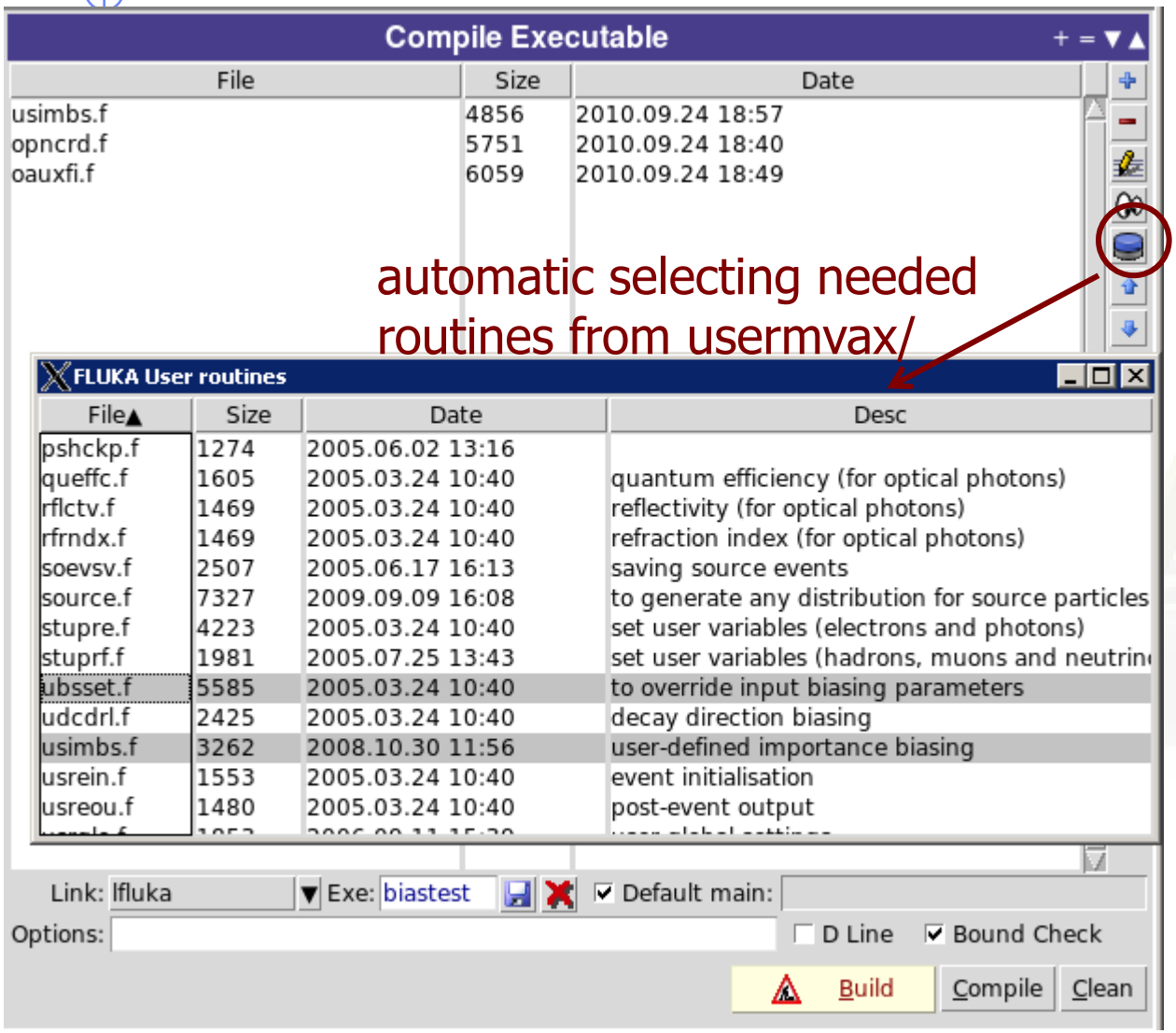
Color Palette



Accessible: Menu → View → Palette

- Edit colors used for material display in Geometry plots and GeometryEditor
- Global colors are saved inside flair.ini and are shared between all projects
- Local colors are initially randomly assigned and saved inside the project file

Compiling



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.

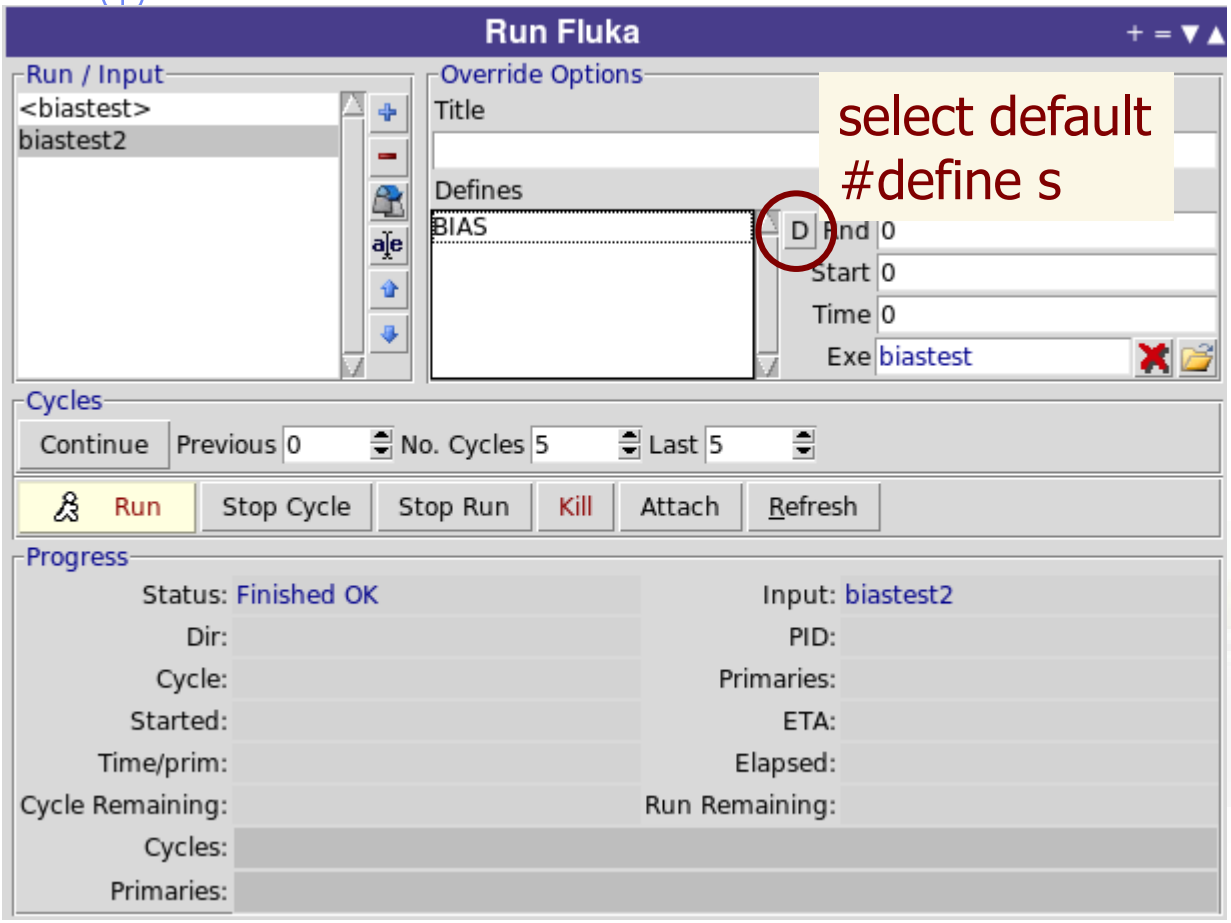
Build: behaves like a "makefile" compiles based on files timestamp when are newer

Compile: Forces compile of the selected files

Clean: cleanup of all produced files

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

Running



<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

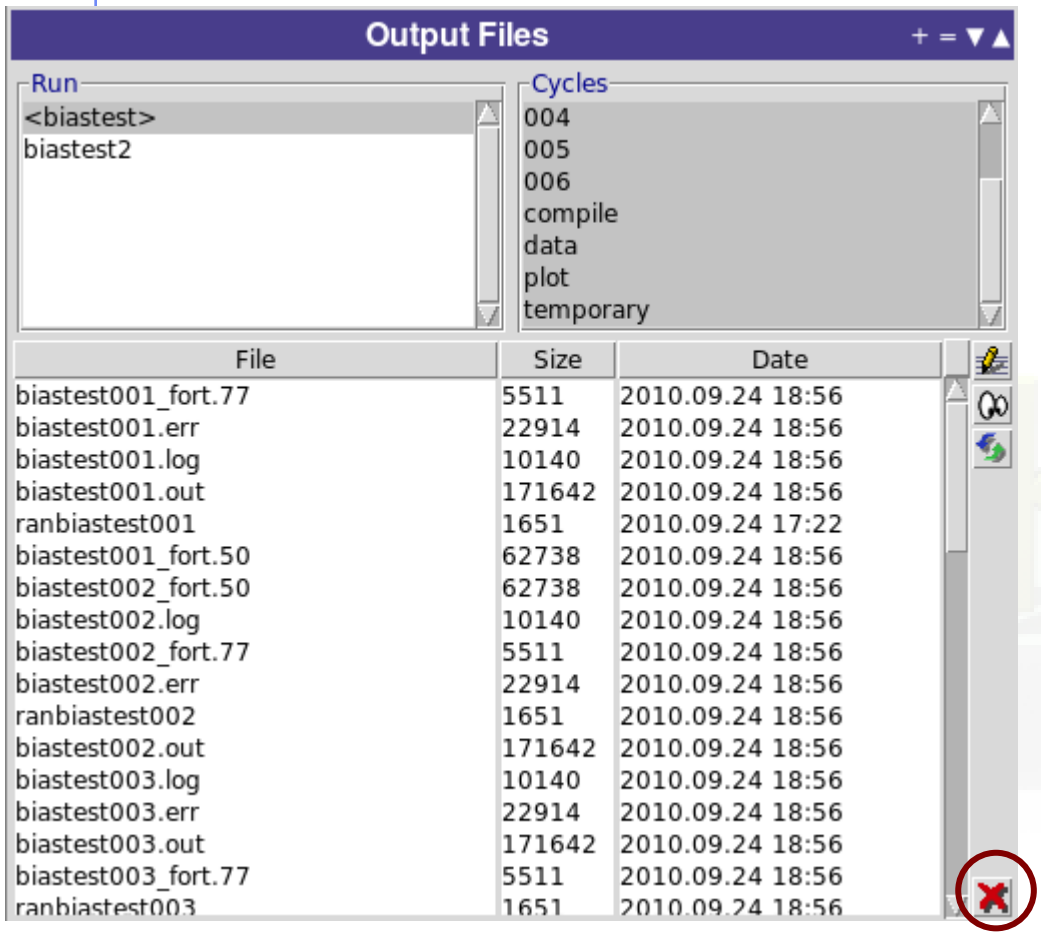
Running: How to use multicore CPU's

- Create clones of the current input e.g. test.inp named: test1.inp, test2.inp, test3.inp …
- Assign a different random number seed on each run (Rnd entry)
- Select all in the listbox and click Run

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



Delete selected files

Inspect Output files generated by FLUKA classified per:

Run/Cycle

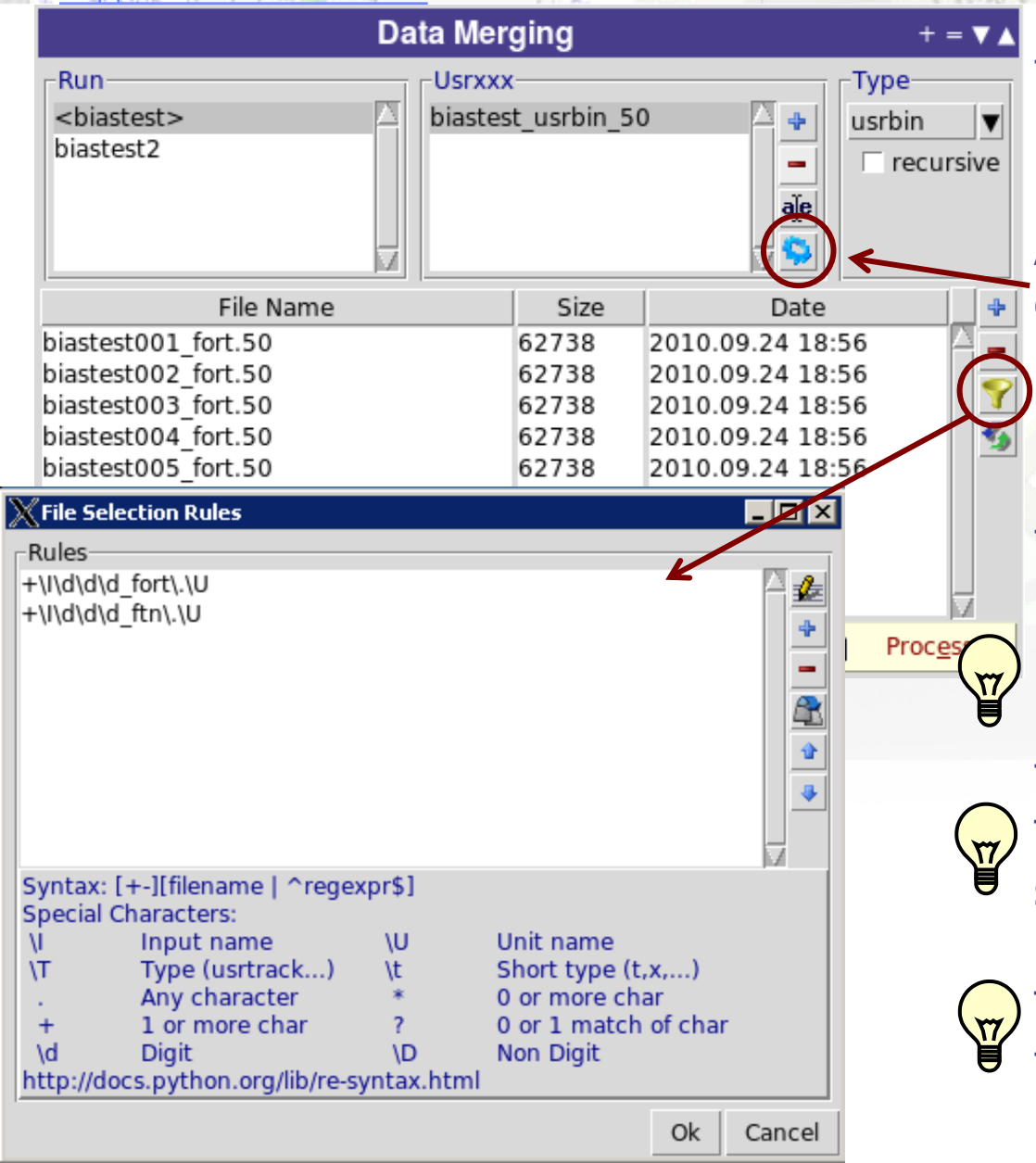
As well special output files from compilation data processing plotting and temporary

Double clicking opens:

- Files in the file Viewer
- coredumps in debugger

Right click can convert USRBIN's from formatted to unformatted

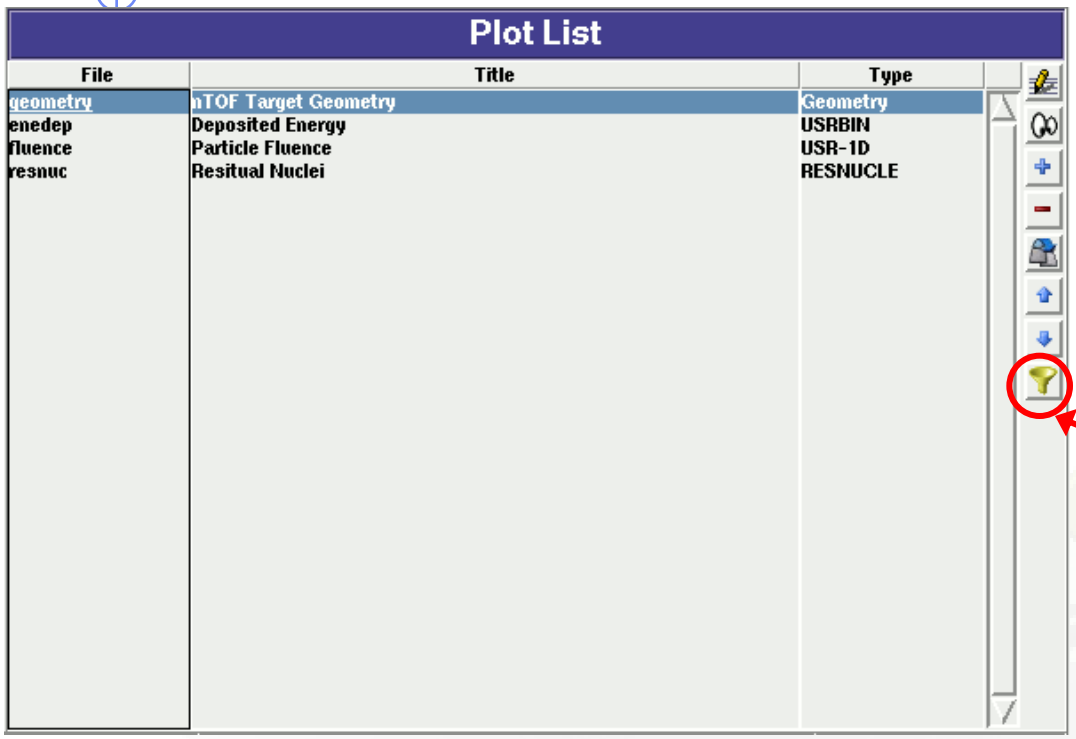
Data Processing



- 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.

- Use the rules to merge from multiple runs. e.g. add a \d in the target like +\I\d\d\d\d_fort.\U
- To modify the rules for multiple scoring cards, select all Usrxxx before
- The default rules can be modified in the Preferences Dialog

Plot List

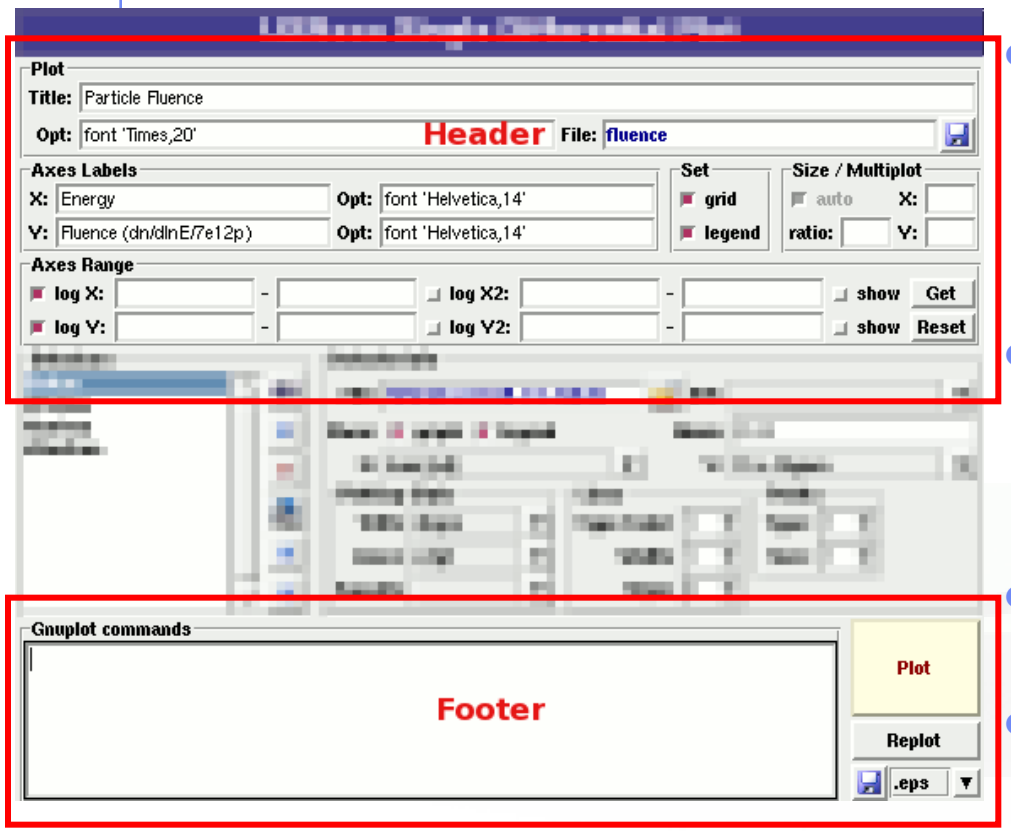


- Plots can be created in the "Plot" list frame. Either Add new plots or Clone from existing ones.
- It is important to set a unique filename for each plot. This filename will be used for every auxiliary file that the plot needs (the extension will change)
- The Filter button creates automatically one plot for each processed unit
- Hit Enter or click the Edit icon to display the plotting dialog
- Fast Double click on item to open the corresponding dialog
- Slow Double click to modify the value

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
- USERDUMP — To plot the output of USERDUMP. Useful for visualizing the source distribution (ToDo)

Plotting Frames



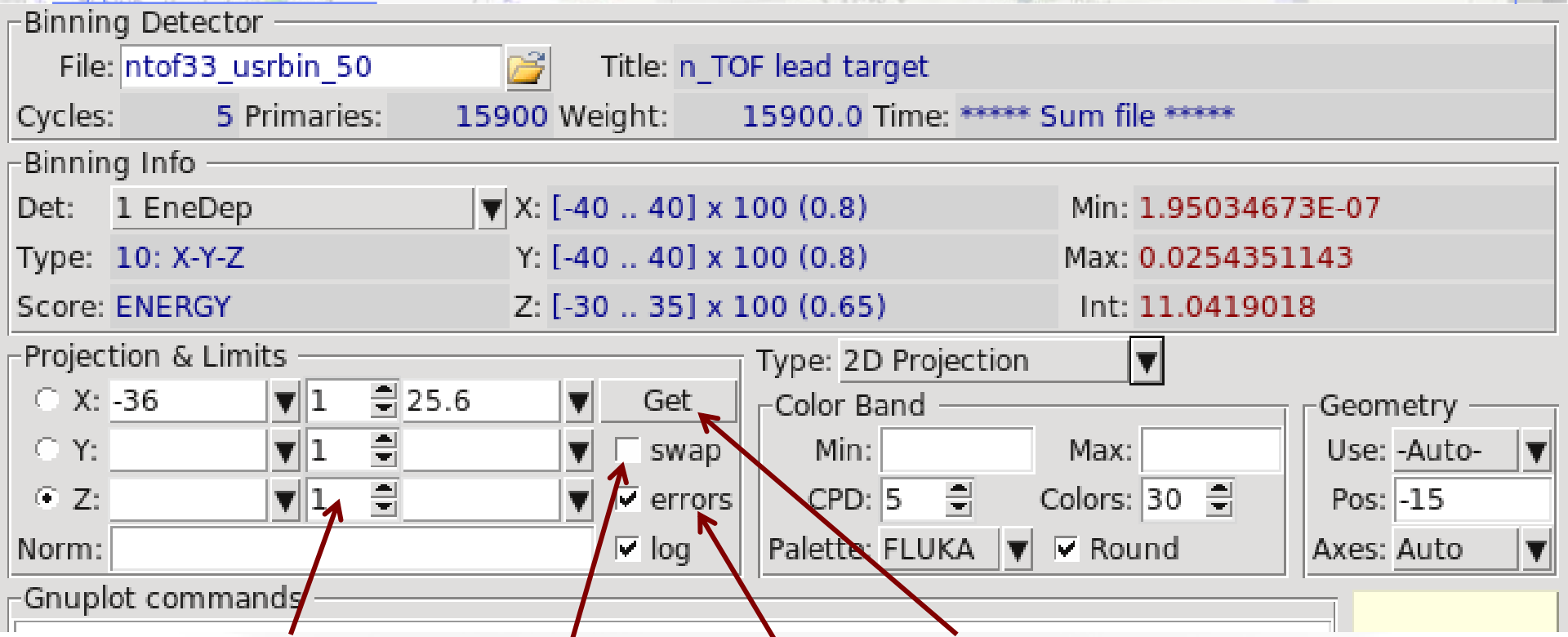
- All plot types share some common fields:
  Title + options, Filename, Axis Labels, Legends (Keys) and Gnuplot Commands.
- Plot button (Ctrl-Enter) will generate all the necessary files to display the plot, ONLY if they do not exist.
- Re-Plot will force the creation of all files regardless their state
- Check the gnuplot manual to provide additional customization commands: e.g. To change the title font to Times size=20, add in the Opt: field the command:
  font 'Times,20'

Look in the flair manual for a short reference of gnuplot commands

General Tips

- In the Configuration Dialog you can set global commands to execute before or after any plot
- The output window displays all the commands that are sent to gnuplot. As well as the errors. In case of problem always consult the output window!
- 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 -1



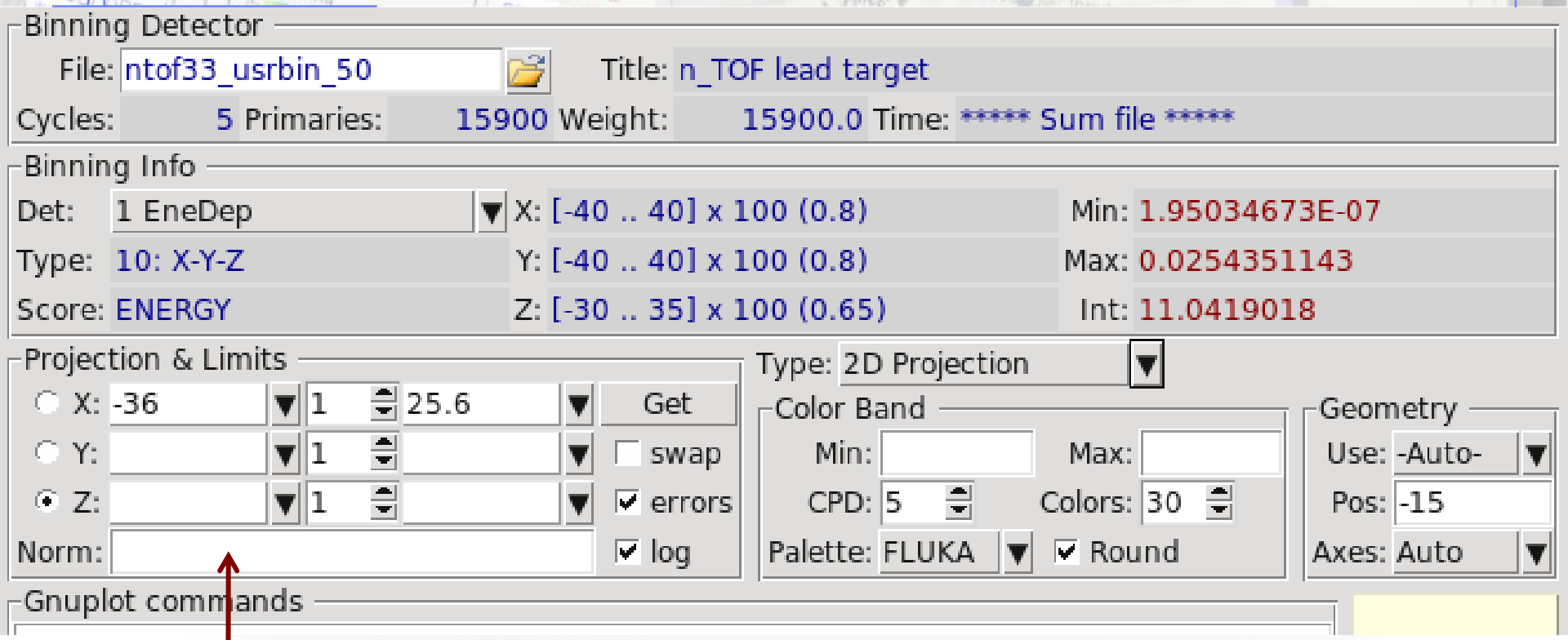
Rebinning

Swap axes

Get limits from gnuplot using right-mouse

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

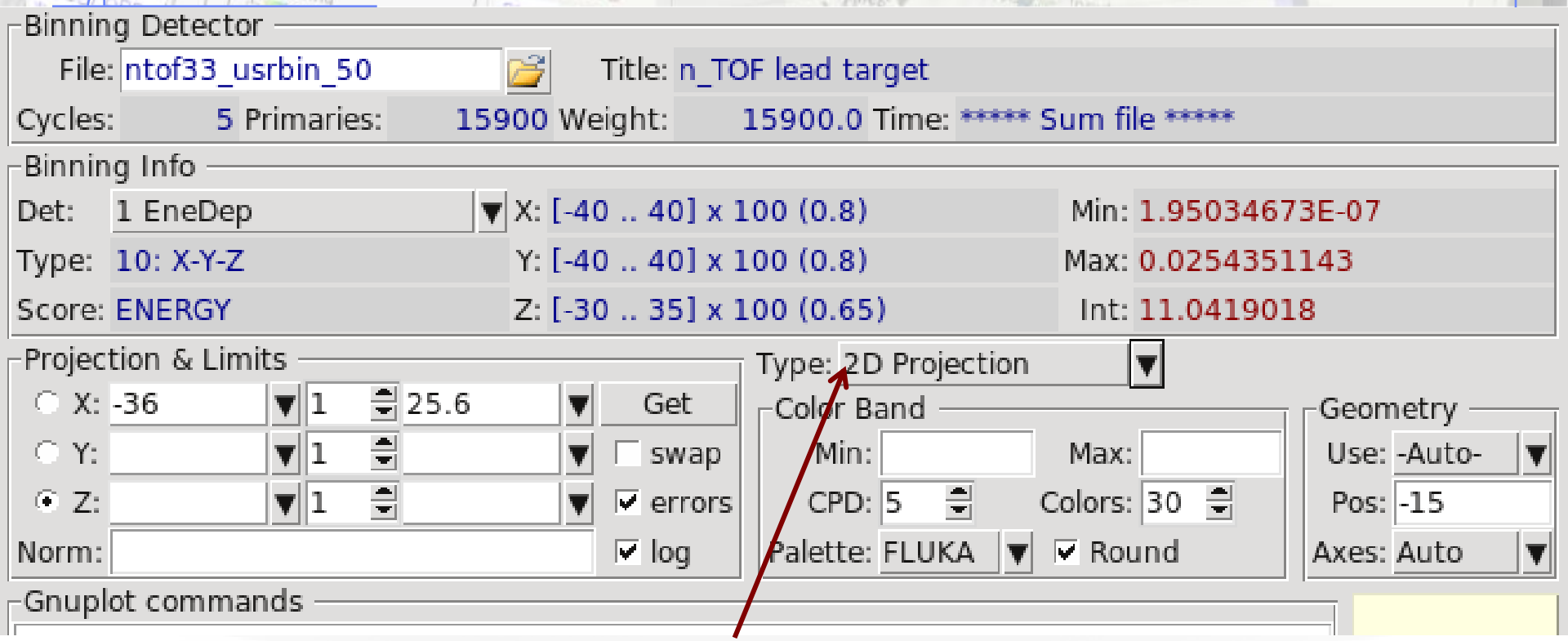
USRBIN Plots - 2



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

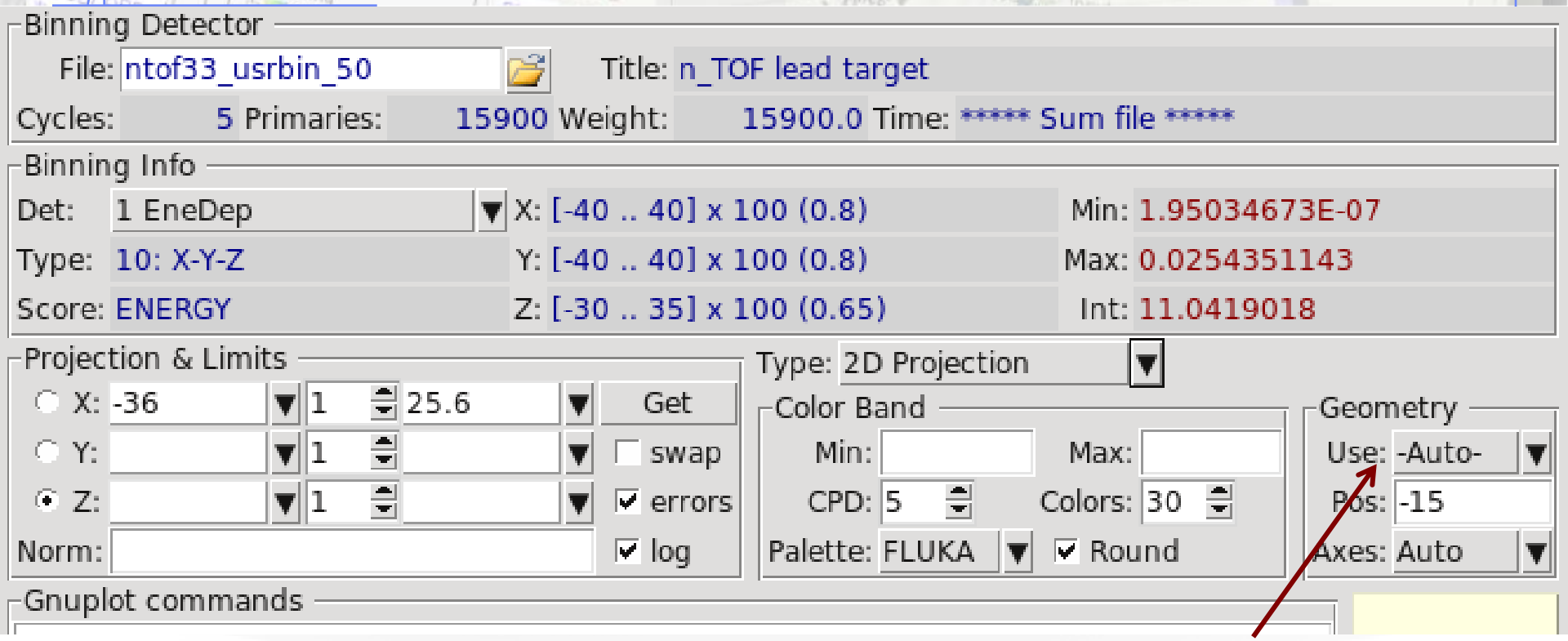
USRBIN Plots - 3



Normalization could be plotted:

- 2D projection, 1D projection
- Trace of the maximum
- Full width at half maximum

USRBIN Plots - 4



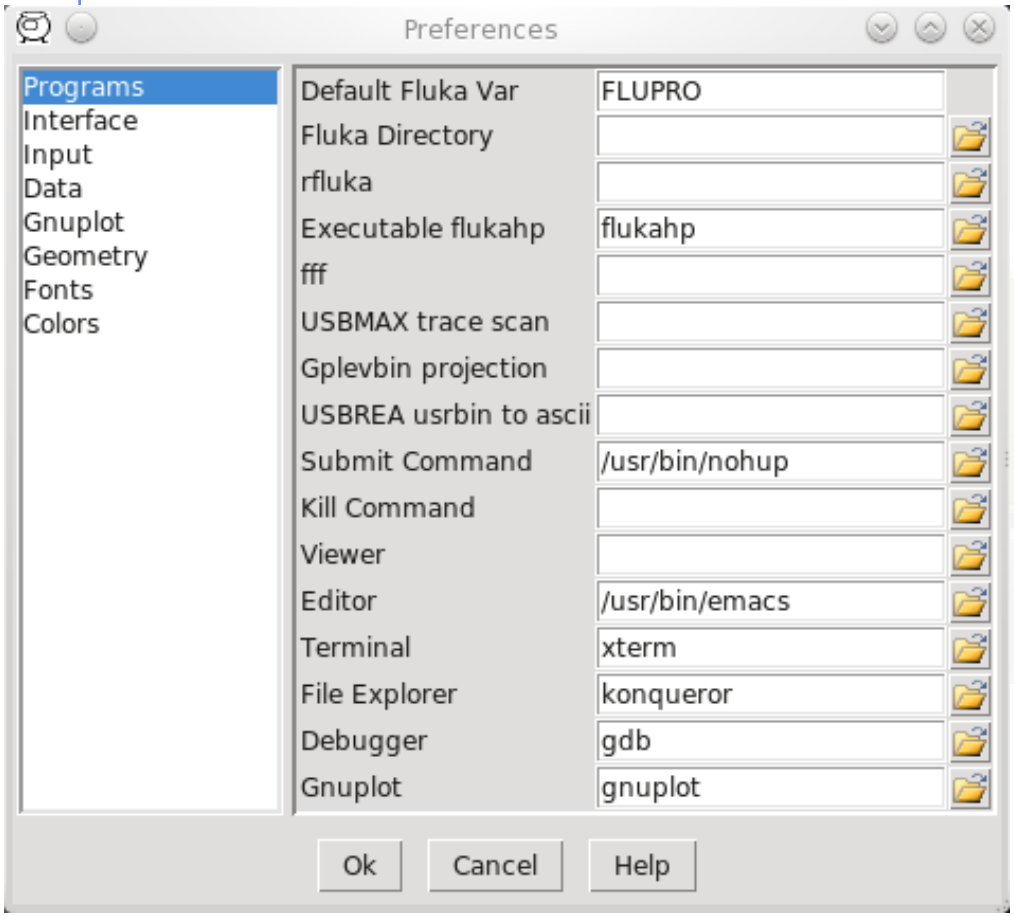
Geometry plot overlay (useful for LATTICE's):

-Auto- generates automatically from FLUKA a geometry at the middle position of the projection

otherwise you can use any existing geometry plot from the drop down list.

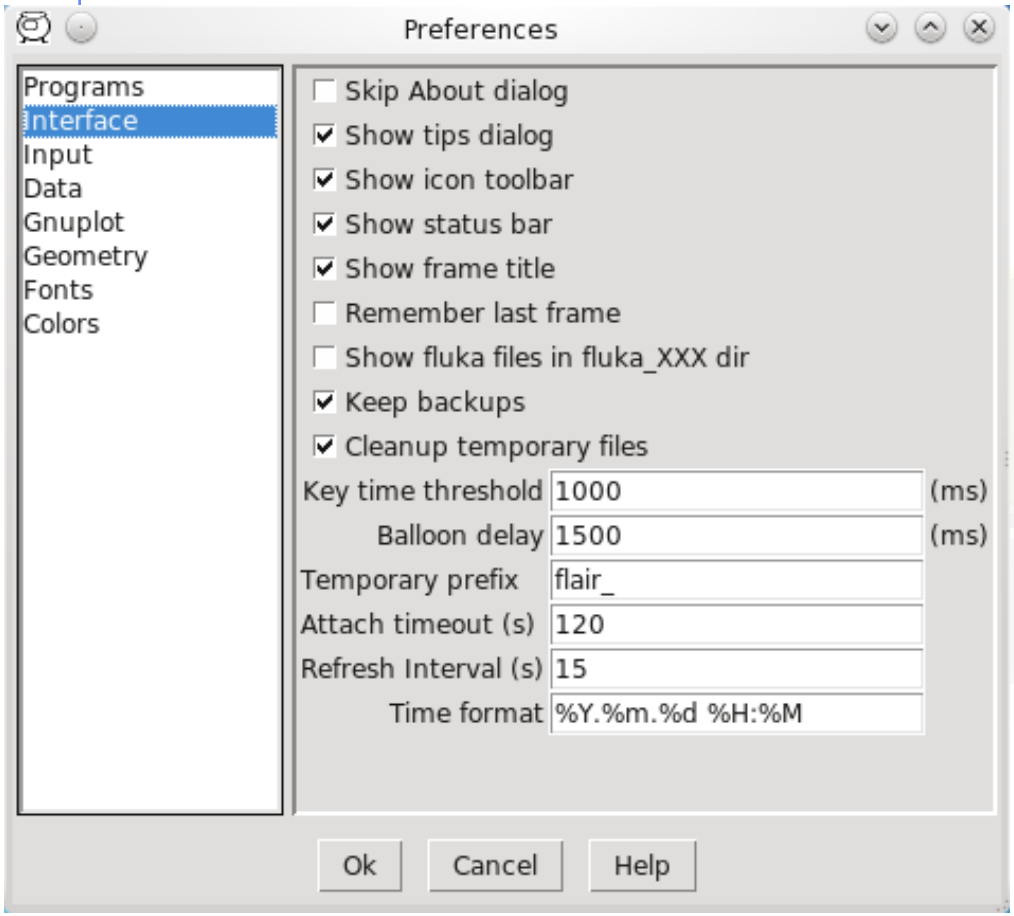
Be carefull to proerly match the axes that you are using

Configuration Dialog: Programs



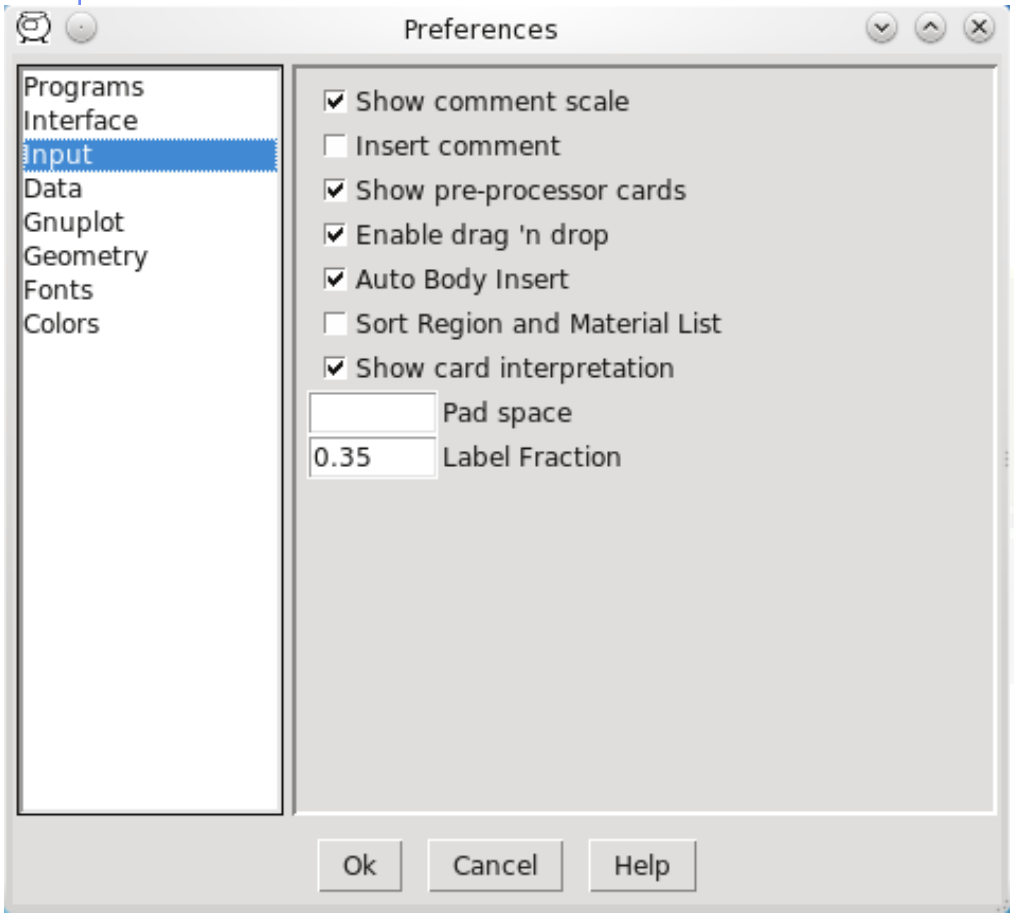
- Set FLUKA directory
- Override default programs to use
- Processing programs are in the “Data” section

Configuration Dialog: Interface



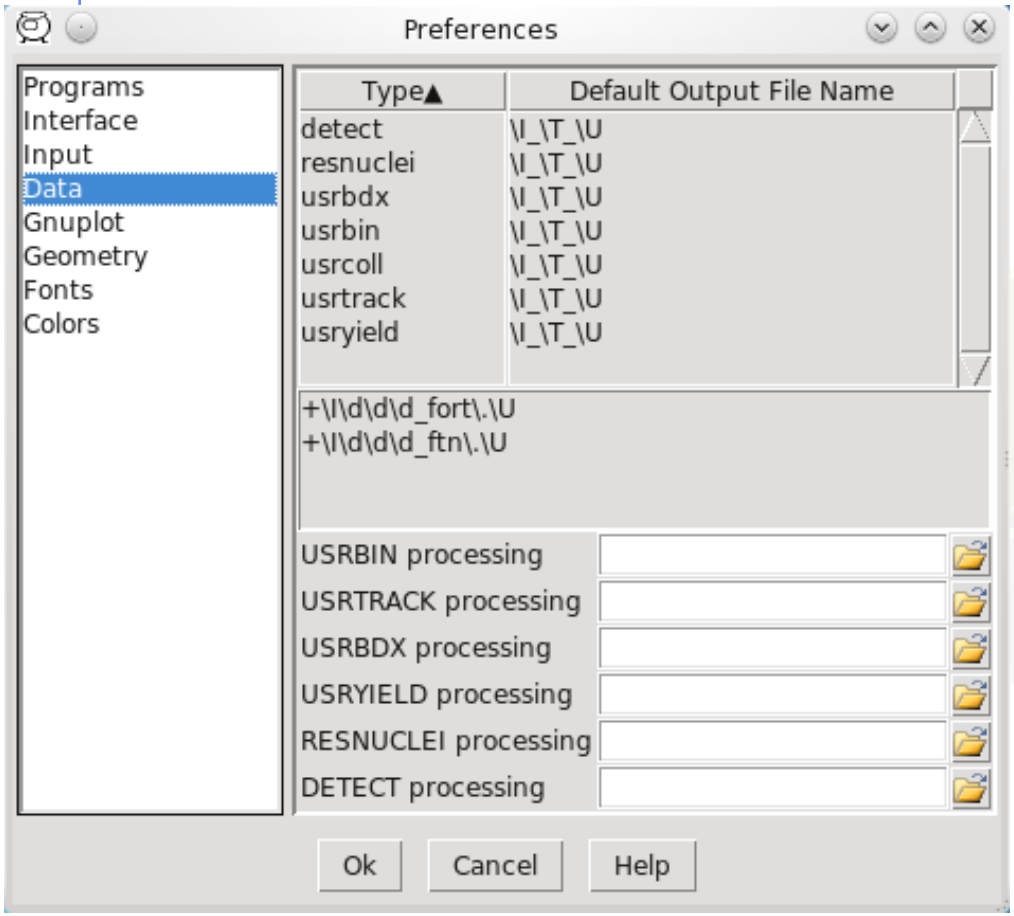
- General interface settings
- Keep backups when files are saved as (file~)
- Automatically Cleanup temporary files. Disable only if you want to inspect files after Debug or Plot when an error occurs
- Key time to reset the type-in search in listboxes
- Balloon delay time
- Time format for files (follows python&C syntax)
- Time out to attach to a running simulation
- Automatic refresh interval of information

Configuration Dialog: Input Editor



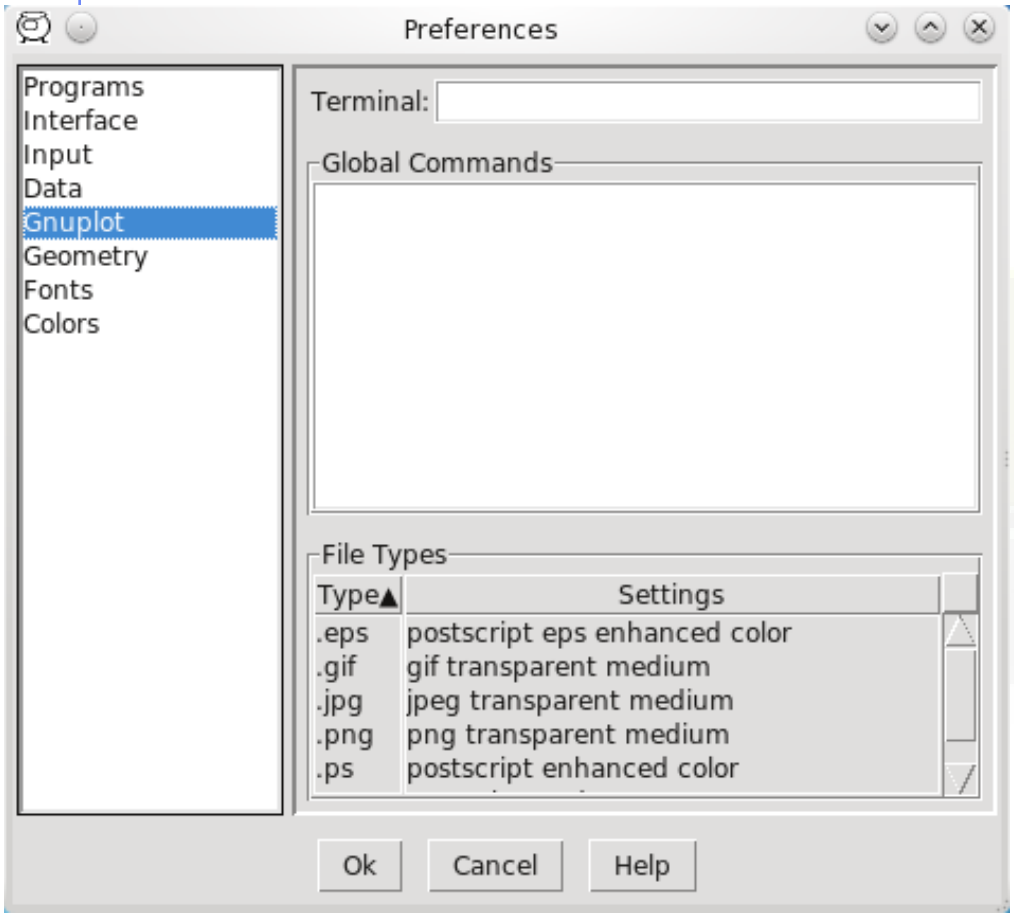
- Show alignment scale
- Automatically insert comment
- Always display preprocessor cards
- Enable drag’n’drop
- Automatic body insertion while editing the region expression
- Sort the region and material list
- Display card interpretation at the bottom of the screen

Configuration Dialog: Data



- Define how to generate the automatic filenames
  - \I will be replaced by input
  - \T by card name
  - \t by card character
    - usrbdx x
    - usrbin b
    - usrcoll c
    - usrtrack t
    - usryield y
    - resnuclei r
  - \U the abs(unit-number)

Configuration Dialog: Gnuplot



Terminal:

additional options to supply to default terminal

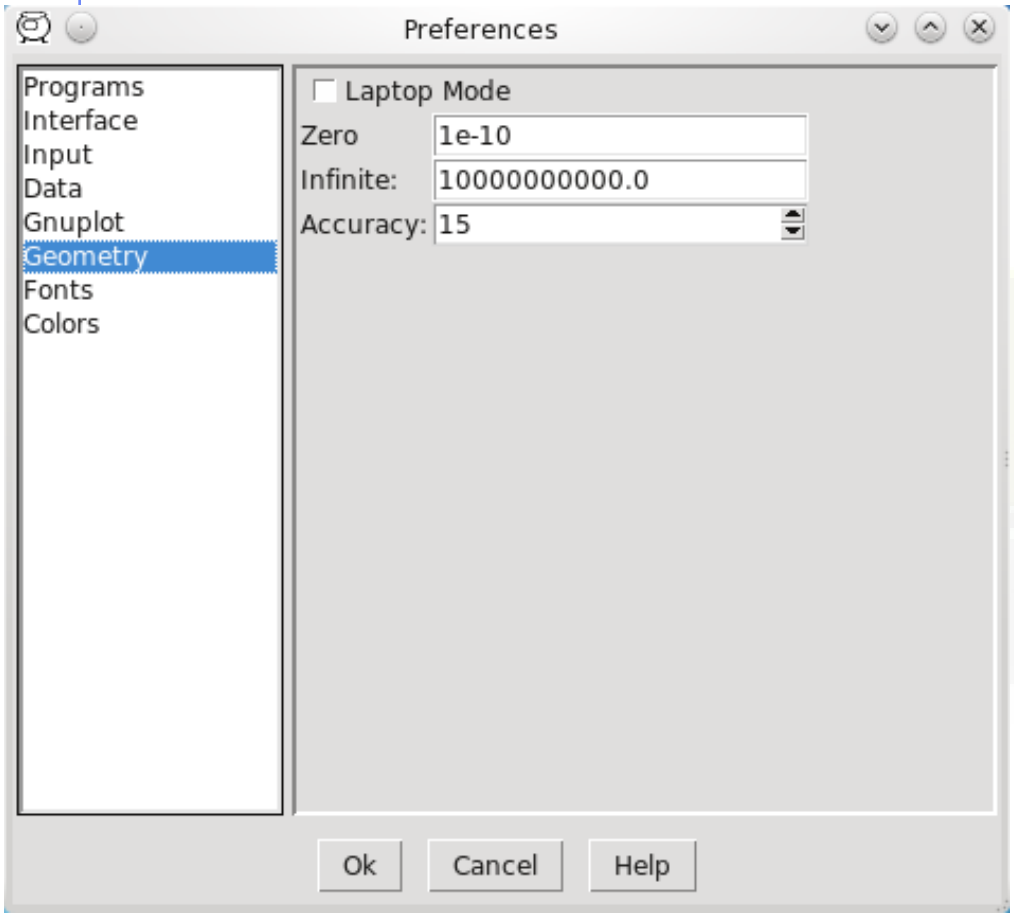
Global Commands:

gnuplot commands to be executed before any plot

File Types:

Right-click: to Add/Delete/Modify file types.

Configuration Dialog: Geometry



Laptop Mode:

check to swap middle with right mouse buttons. Middle button is used in GeometryEditor for panning, zooming, rotating etc…

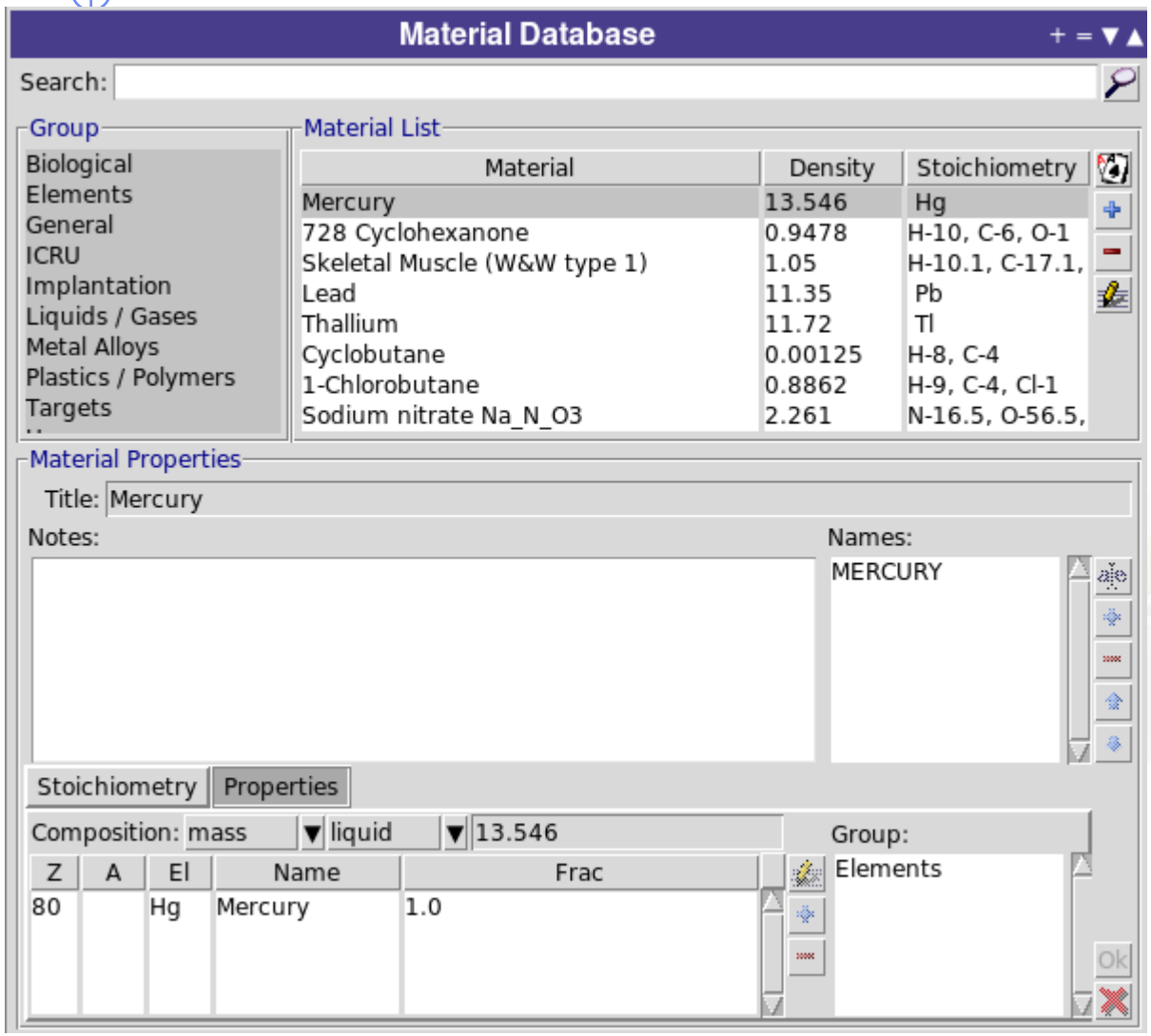
Zero:

Infinite:

Accuracy:

same as in the Bodies Transformation dialog

Materials Database



search database

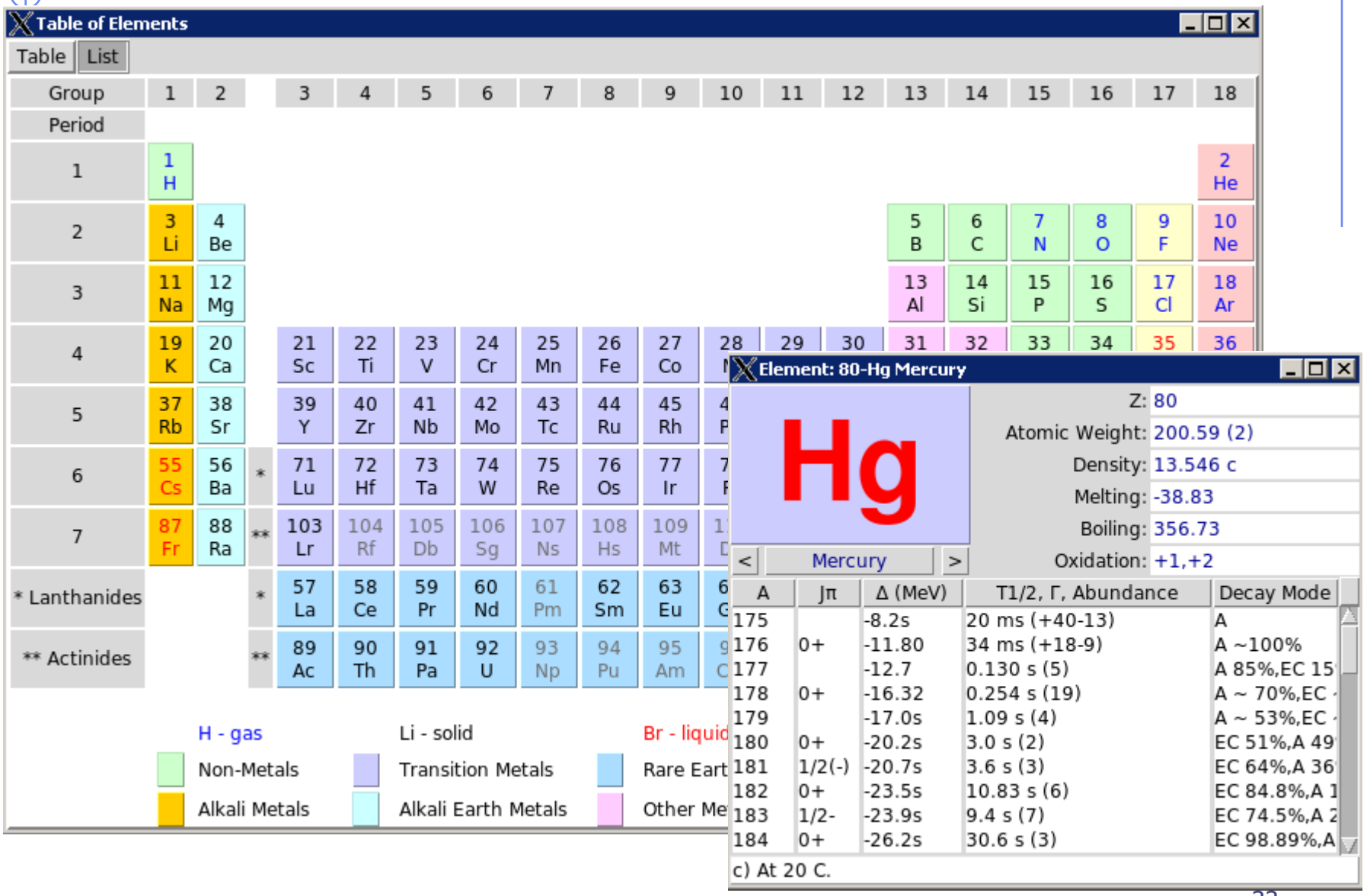
insert material to input
add/del material
edit material

add names to be used by FLUKA

Modify Stoichiometry and properties of material

WARNING: When modifying the database a local copy will be created in ~/.flair folder!!!

Periodic Table



Import / Export

Importing

- Input: merge parts or entire input file with the current
- Mcnp: import mcnp geometry into FLUKA. (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

Geometry Editor 2D

to be done

- Debugging and editing bodies/regions in a graphical way
- Working on 2D cross sections of the geometry. Not a real problem since most of the objects are 2D extruded in the 3rd dimension

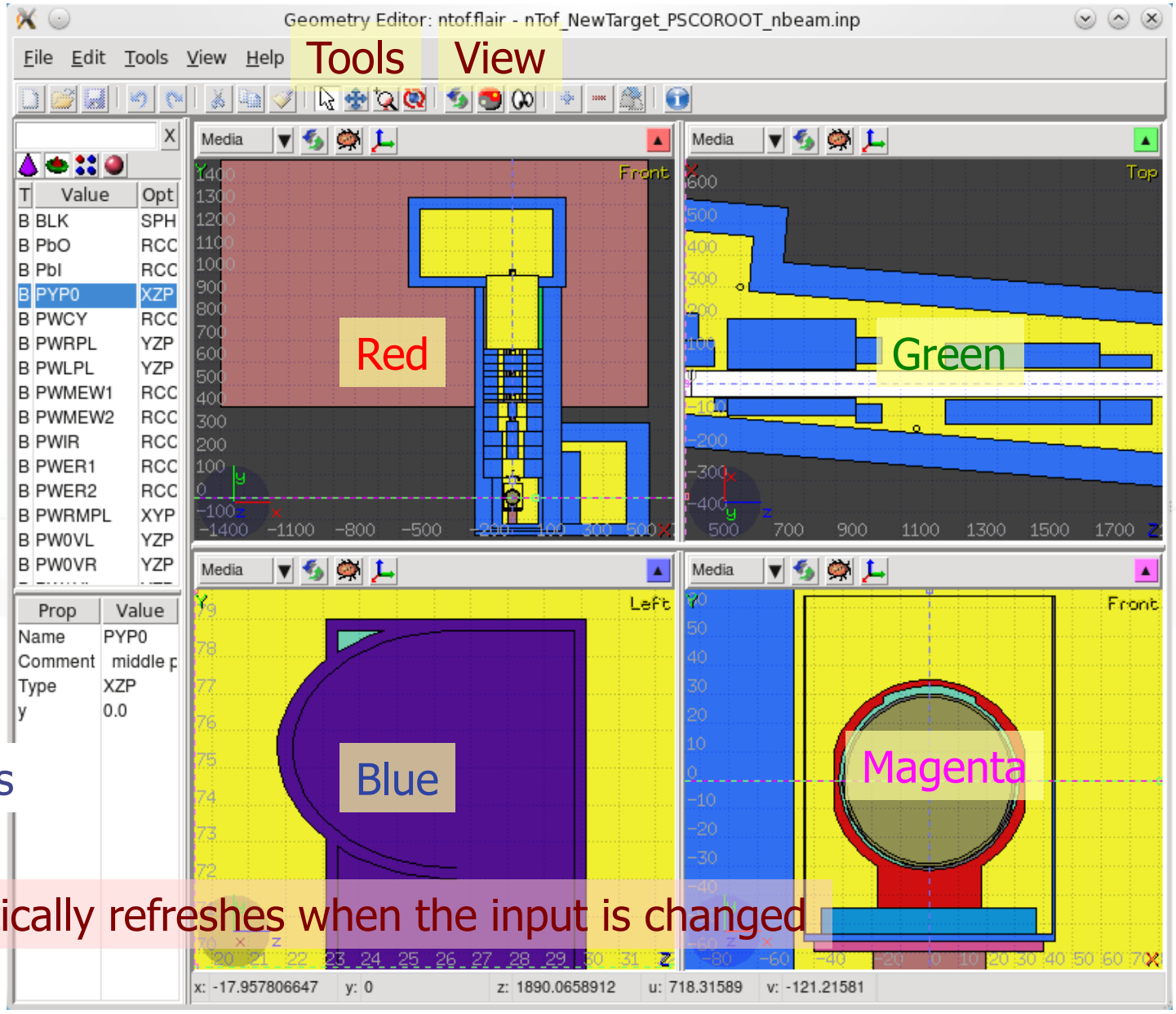
Pros

- Fast display of complex geometries
- Visual selection and editing of zones
- Use real curve of bodies with no conversion to vertices/edges
- Interactive debugging with information of problematic body regions and zones
- No use of any additional hardware (plain X11 libraries)

Cons

- No interactive 3D display
- Blind in 3rd dimension[could be compensated with raytracing]
- Tricky to orientate in an unknown geometry

Geometry Editor: Interface



Geometry Editor: Mouse / Keyboard

General:

- Ctrl "controls" or changes the action
- Shift aligns to grid
- Escape cancels the active action

Mouse:

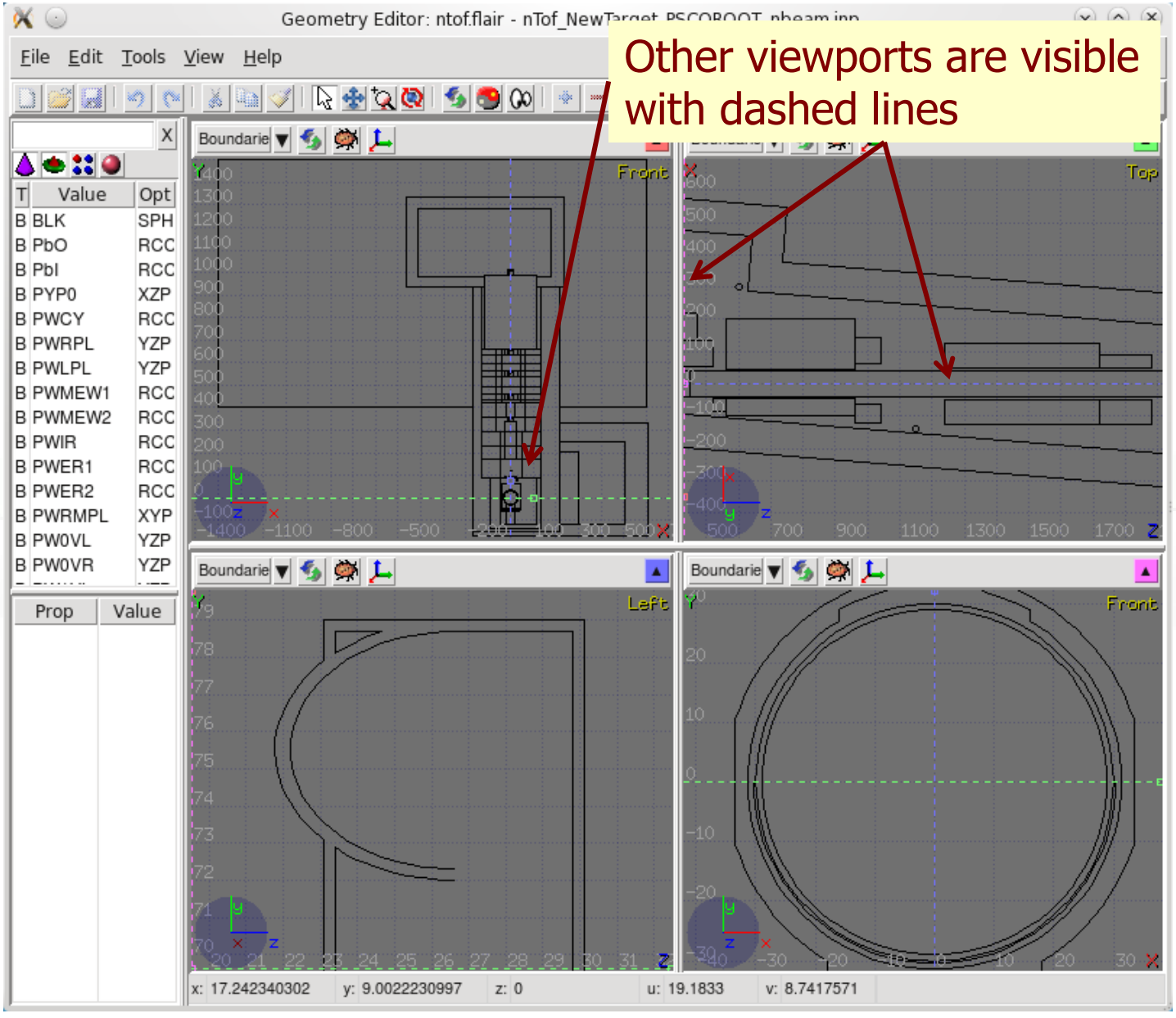
- Left button:
  - User selectable action from the tools
- Middle button
  - default: Pan/Move viewport
  - Shift: select rectangle region and zoom into
  - Shift-Ctrl: select rectangle region and zoom out
  - Ctrl: rotate projection using a virtual trackball
  - Ctrl-Shift: rotate projection using a virtual trackball with steps of 15
- Right button (or Ctrl-Spacebar):
  - pop-up menu

When laptop mode is enabled in the Preferences then the middle and right buttons are swapped

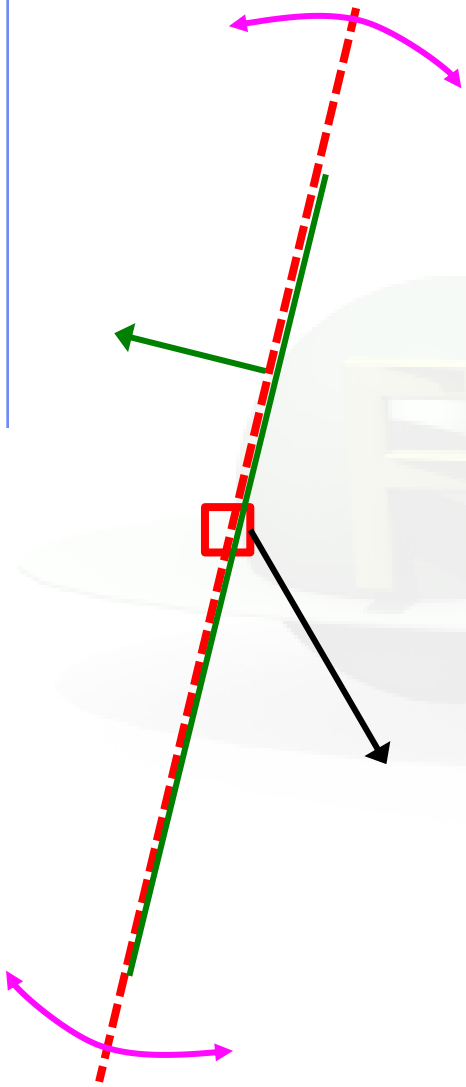
Tools

	select	h	bodies, regions, or modify viewports
	pan	x	move viewport
	zoom	z	zoom in/out. Clicking will zoom by 2 or draw a rectangle. To zoom-out use Ctrl
	trackball	t	rotate viewport
	refresh	Ctrl-R	refresh all viewports
	toggle	Shift-Z	change view type: bodies, region, material
	layout	v	rotate various layouts
	errors		show dialog with geometry errors
	axes	o	show dialog to select projection
	zone	Ctrl-Click	show zone description using selected bodies
	views	1..9	change view projection X-Y, Y-Z,…

Geometry Editor: Viewports



Manipulating Viewport

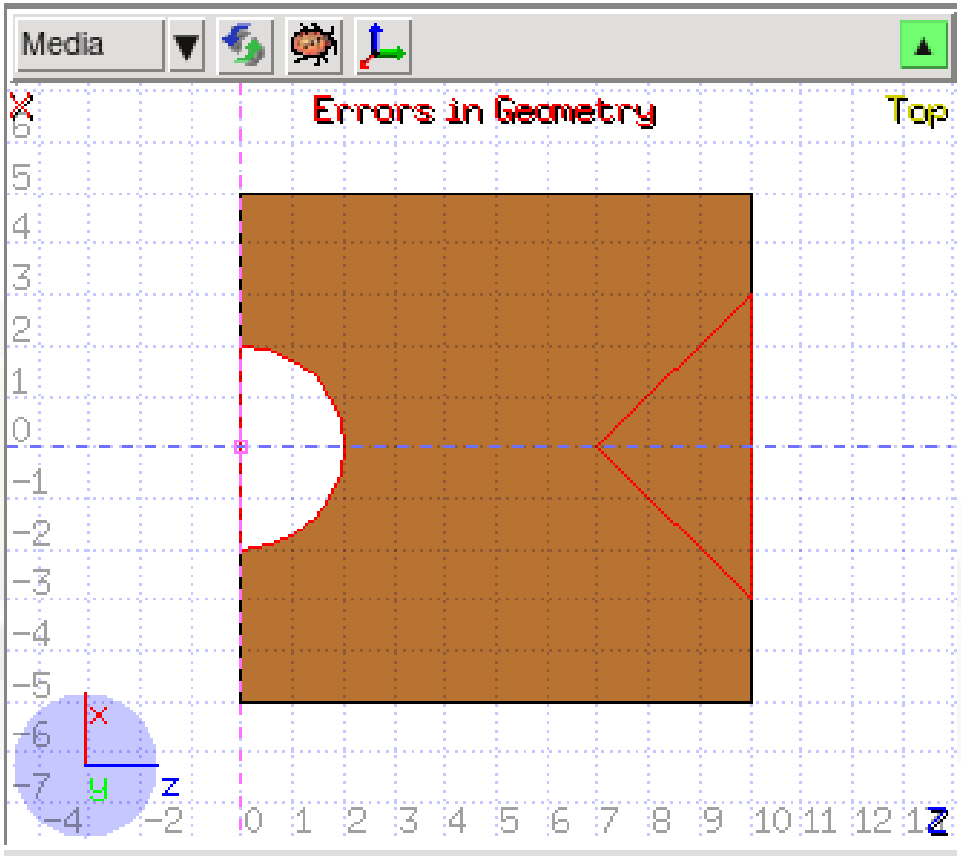


- Dashed lines represent viewports
- Center is represented with a square
- When the other-viewport is outside the view window, the viewport-line will be displayed on the closest edge

Actions (select-tool + left mouse)

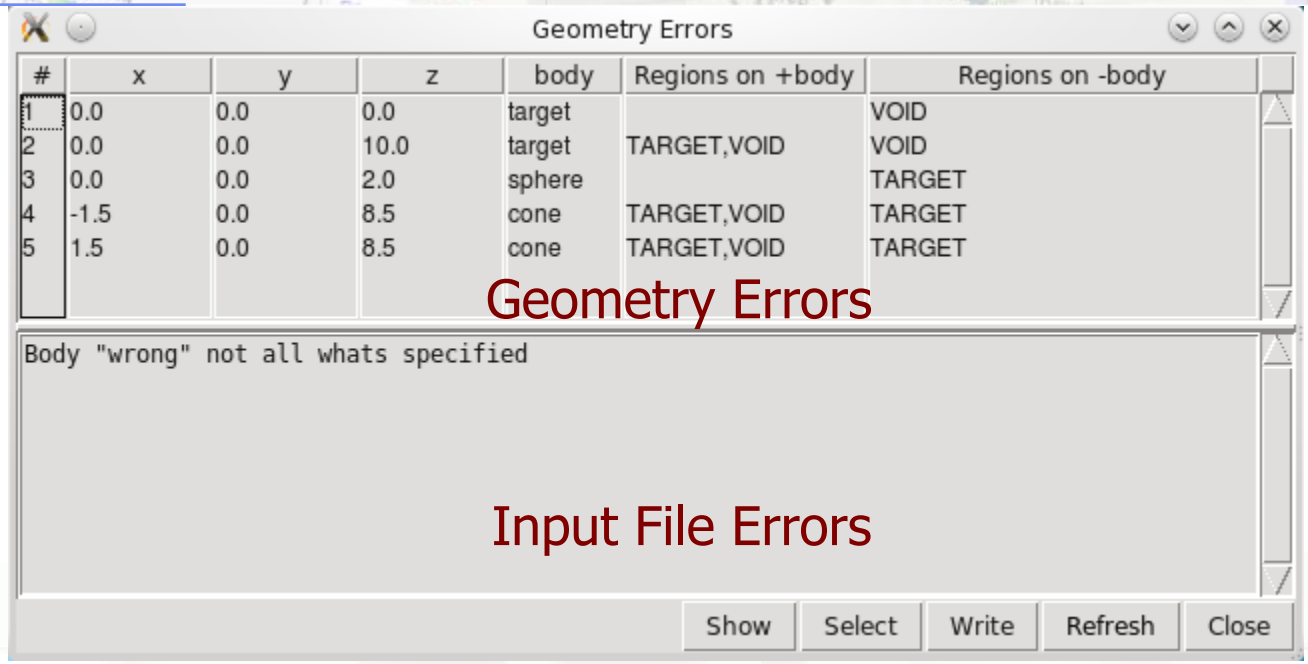
- drag the center square to reposition the viewport
- drag the line close to the center to reposition the viewport along the vertical axis
- drag the extremities of the viewport-line to rotate the viewport

Geometry Errors



- "Errors in Geometry" notifies that are possible errors in the geometry.
- Clicking the icon displays the dialog with the errors.
- Touching surfaces are checked against 10 significant digits

Geometry Errors



x,y,z Coordinates of the error (on the surface of body)

body Body with the x,y,z point on surface generating the error

+body Regions that are on the + side of the body.
Regions where the body should be subtracted to remove the error

-body Regions that are on the – side of the body.
Regions that the body should be intersected to remove the error

+/- of body are defined according to the normal on the surface.
+ refers to outside, - to inside

Programming Interface: API

There is work presently going on to decouple the functionality from the interface, some of the basic classes can be used to input processing

file: Input.py - to manipulate input files

import Input

Input.init([database]) to initialize the database of cards

Most commonly used classes:

Card	containing the description of each card
Input	manipulating the FLUKA input file

file: Project.py - to manipulate project files

API: class Card

Constructor: Input.Card(tag, what [,comment [,extra]])

what is a list starting with what[0]=sdum

Important Methods:

setWhat(n, value)	set value to what#n
nwhats()	return number of whats
what(n)	return value of what#n
numWhat(n)	return numeric value of what#n
intWhat(n)	return integer value of what#n
clone()	return a copy of the card
setEnable(e)	enable/disable card

API: class Input

Constructor: Input.Input()

initialize the structure to hold an input file

Important Variables:

cardlist	a list with pointers to cards
cards	a dictionary with pointers to cards grouped per tag

Important Methods:

read(filename)	read input from file
write(filename)	write input to filename
addCard(card,pos)	add card to position pos (or end of file)
delCard(pos)	delete card from position pos
preprocess()	preprocess input to check for active cards
setEnable(e)	enable/disable card

API: class Project

Constructor: Project.Project()

initialize the structure to hold a project file

Important Methods:

clear()	to re-initialize project
load(filename)	load project from file filename
save([filename])	save project to filename
runCmd(run)	create run command

API: example

Read an input file and modify the random number seed

```
import Input
Input.init()
input = Input.Input()
input.read("test.inp")
try:
    rndcard = self.cards["RANDOMIZ"][0]
    rndcard.setWhat(2,5723)
except:
    print "No RANDOMIZe card found"
    sys.exit(0)
input.write("test2.inp")
```

API: .flair file structure

```
# comments
Variable: Value
Notes:
    multi-line values are terminated with
    Ctrl-L

Run: name
    ...                          Block of Run related information
    Data:
    ...                          ... Including Data processing information
    EndData
EndRun
Plot: name
...                              Plot related informations
EndPlot
```