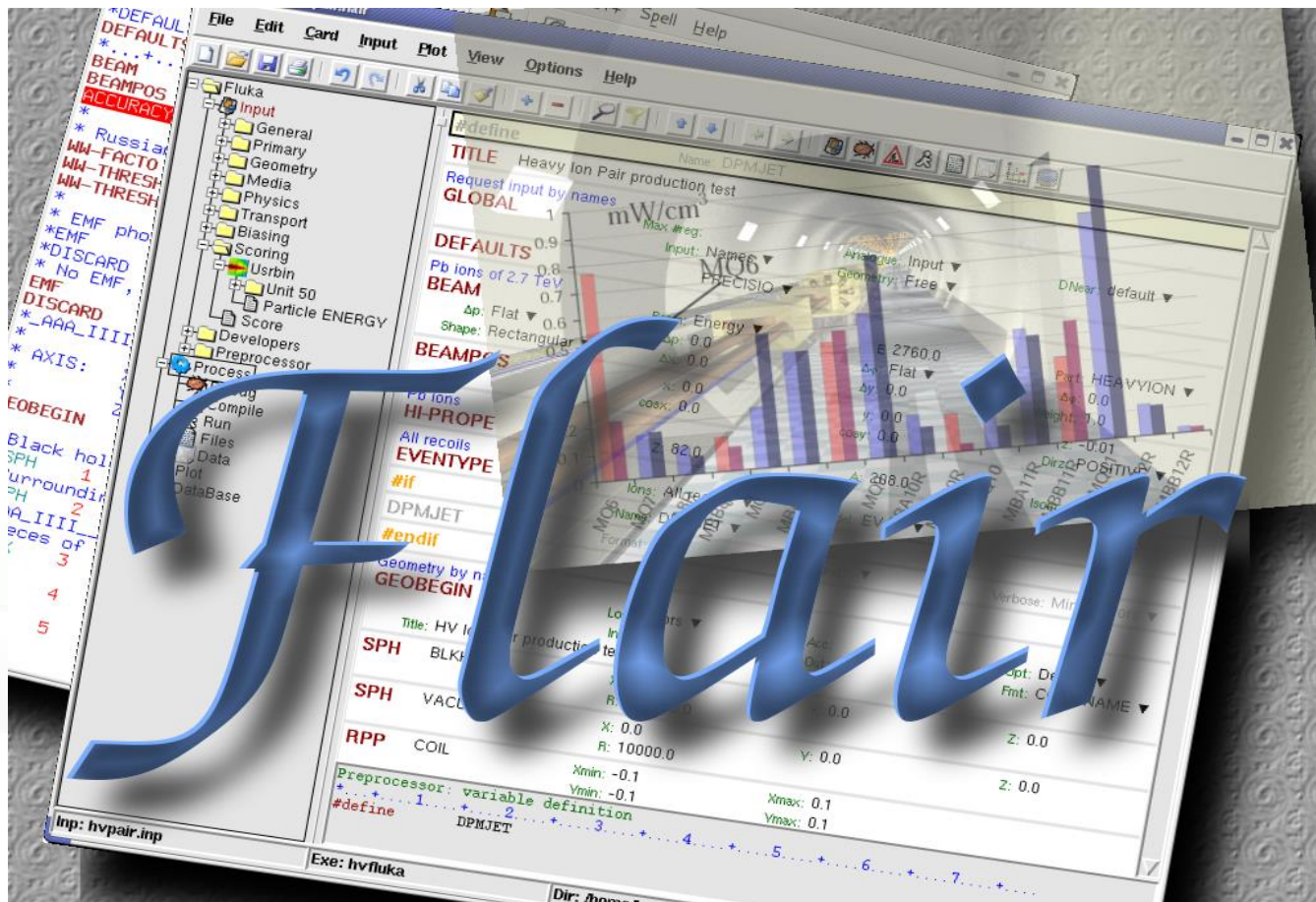# Introduction to Flair

OMA MC School

/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

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 (see Geometry Editor talk)
- **Compilation** of the FLUKA Executable
- **Running** and **monitoring** of the status of a/many run(s)

3

What is flair

Back-End interface:

- Inspection of the output files (core dumps and directories)
- Output file(s) viewer divided in sections
- Post processing (merging) the output data files
- Plot generation through an interface with gnuplot (matplotlib under development)

Other Goodies:

- Access to FLUKA manual as hyper text
- Checking for release updates of FLUKA and flair
- Import export to various formats: MCNP/X, GDML, Povray…
- Nuclear wallet cards
- Library of materials
- Database of geometrical objects (Not yet completed)
- Programming python **API**
- Everything is accessible with keyboard shortcuts

Concepts: Flair Project

- Store in a single 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 as 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 format 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!

Installation

- Flair web site to download code and documentation

  http://www.fluka.org/flair

- Installation procedures:

  - RPM/DEB method (Linux): strongly recommended! on systems that support the RPM/DEB. The package will create all file association, menu items and keep track of updates and files installed.
    The package will install the program to: /usr/local/flair
    and will create the following launcher programs:
    - /usr/local/bin/flair   flair program
    - /usr/local/bin/fm   FLUKA manual
    - /usr/local/bin/pt   Periodic Table
    - /usr/local/bin/fless   FLUKA output viewer

  - tar.gz method (MacOS, MS-Windows). Please follow the instructions on:
    http://www.fluka.org/flair/download.html
    and for special instruction on the FAQ:
    http://www.fluka.org/flair/faq.html

Starting flair

Programs Menu (Linux)

- Click the icon of Flair from the programs menu;
- Flair is registered under the **Science/Physics** category but depending your Linux distribution and window manager it might appear in different sub-menus (i.e. Applications, Education, Science or Others).

Window Manager (Linux, only via RPM or DEB installation)

- Flair makes an association of the following extensions:

*.flair   ***.fluka *.inp**

Console

- Type the command flair. Remember to add to your **$PATH** the directory where flair is installed!

Startup

Unless you specify any option during startup flair will perform the following operation:

- Check for the existence of a FLUKA installation (FLUPRO environment variable), and if not found will open the Preferences dialog to set explicitly the FLUKA path;
  WARNING: Window managers (GNOME, KDE…), as well command shells (bash, tcsh, ash…) have a different configuration file where they expect the environment variable.

- Open the "Check for Updates" dialog (every 30 days interval).

Basic Preferences
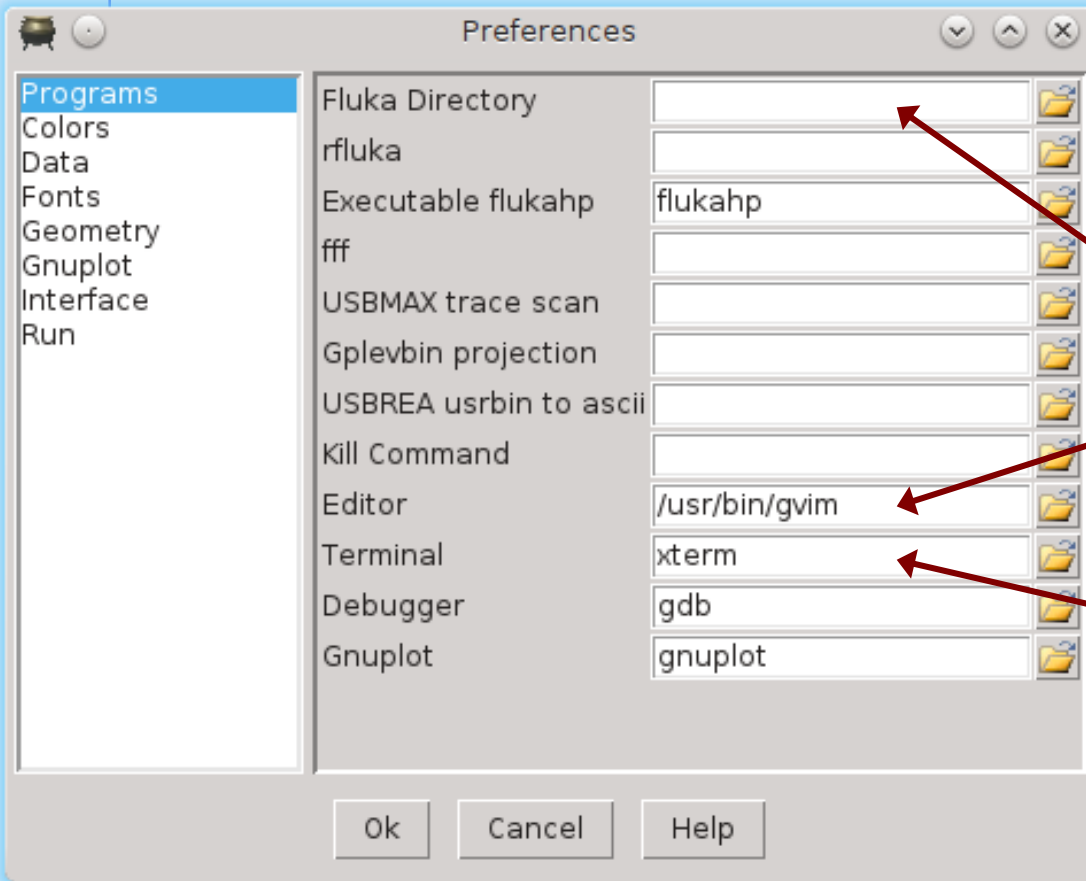
The program tab of preferences allows the user to set the default programs and directories

Set your FLUKA directory, to override $FLUPRO

Set your favorite editor

Set your favorite console program (xterm, nxterm, kconsole…)

Interface

Tabs drag to rearrange or to undock

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

Interface - Multi docking

Input Templates

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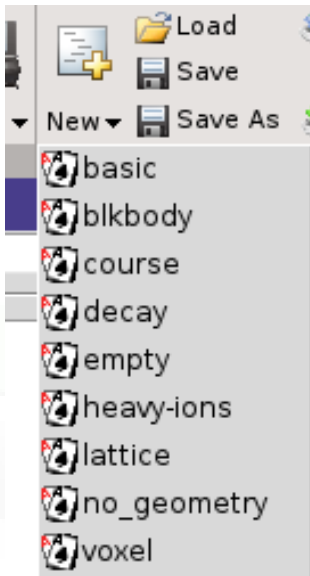
```
TITLE

GLOBAL                                                  1.0        1.0
DEFAULTS                                                              NEW-DEFA
BEAM
BEAMPOS
GEOBEGIN                                                              COMBNAME
     0     0
* Black body
SPH blkbody      0.0 0.0 0.0 10000000.0
* 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
```

Menu items shown:
- Load
- Save
- New
- Save As
- basic
- blkbody
- course
- decay
- empty
- heavy-ions
- lattice
- no_geometry
- voxel

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)

12

Input Editor

- With the input editor the user can manipulate the input cards:
  - Add card to input
  - Edit existing ones
  - Copy & Paste
  - Clone (Duplicate)
  - Import from other input files
  - Validate the correctness of the cards
  - Error filtering
  - Rearrange order
- The editor will try to rearrange the input cards (if needed) to create a valid FLUKA input file
  e.g. body cards outside the GEOBEGIN/GEOEND parts will be moved inside

Note: Automatic rearranging of cards cannot work if "**#include**" cards are present. The user have to do it manually

Card Categories

For easier access, cards are grouped in the following categories:

- **General**   General purpose (TITLE, DEFAULTS, GLOBAL…)
- **Primary**   Definition of the primary starting particles
- **Geometry**   Cards related to the definition of the geometry bodies/regions/lattices plotting and rotations/translations

  - **Bodies**   Subcategory containing only the bodies definition
  - **Transformations** Subcategory containing only the geometrical directives

- **Media**   Definition and assignment of materials
- **Physics**   Setting physics properties of the simulation
- **Transport**   Modify the way particles are transported in FLUKA
- **Biasing**   Cards for importance biasing definition
- **Scoring**   Cards related to scoring
- **Flair**   flair special cards
- **Preprocessor** Definitions for creating conditional input files

Concepts: Extended Cards

- Flair is treating the input file as a list of extended cards;
- Each extended card contains:
    - Comment: All commented lines preceding the card(s) as well the inline comments
    - Tag: The 8 character word identifying the card. All tags not recognized by flair will be converted to #error
    - WHATs: Multiple number of WHATs (0=sdum, 1-6 first line, 7-12 continuation line...)
    - Extra: multi line string of extra information for special cards like REGION, TITLE, PLOTGEOM etc.
    - State (Enable/Disable)
- Flair recognize automatically (and separates them from the comments) all the disabled valid FLUKA cards

Concepts: Extended Cards

- The region definition in the in geometry is emphasized by the presence of a card named "REGION"

- All COMPOUND cards related to one material are joined in one card

- Cards are edited with the flair editor through the use of the mini-dialogs, forcing the user to enter a *proper* information

- The user gets full control of the card using the Edit dialog
(See button at *lower right corner)*

- Flair will try to find the best floating point representation of each number, to ensure the maximum accuracy; number of digits that fits in the specific width (10 for the fixed format, 22 for the free format)

- Function evaluation: a field value starting with **=** will force flair to evaluate its content as a function e.g.

  BEAMPOS   x: =2*10+length

  Flair will create a valid fluka input containing the evaluation of the formula and keep the formula inside the comments as

  !@what.1 =2*10+length

WARNING: If you edit with an external editor the input flair will overwrite the changes of the functions if any!

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
```

Comment

Label

Interpreted Value of WHAT(1)

Drop down list box with possible options

Beam characteristics			
BEAM			
Δp: Gauss ▼	Δp(FWHM): 0.082425	E: 20.0	Part: PROTON ▼
Shape: Rectangular ▼	Δx:	Δφ: Gauss ▼	Δφ: 1.7
		Δy:	Weight: 1.0

Beam: Energy ▼

Grey box
Shows currently editing item

Tag
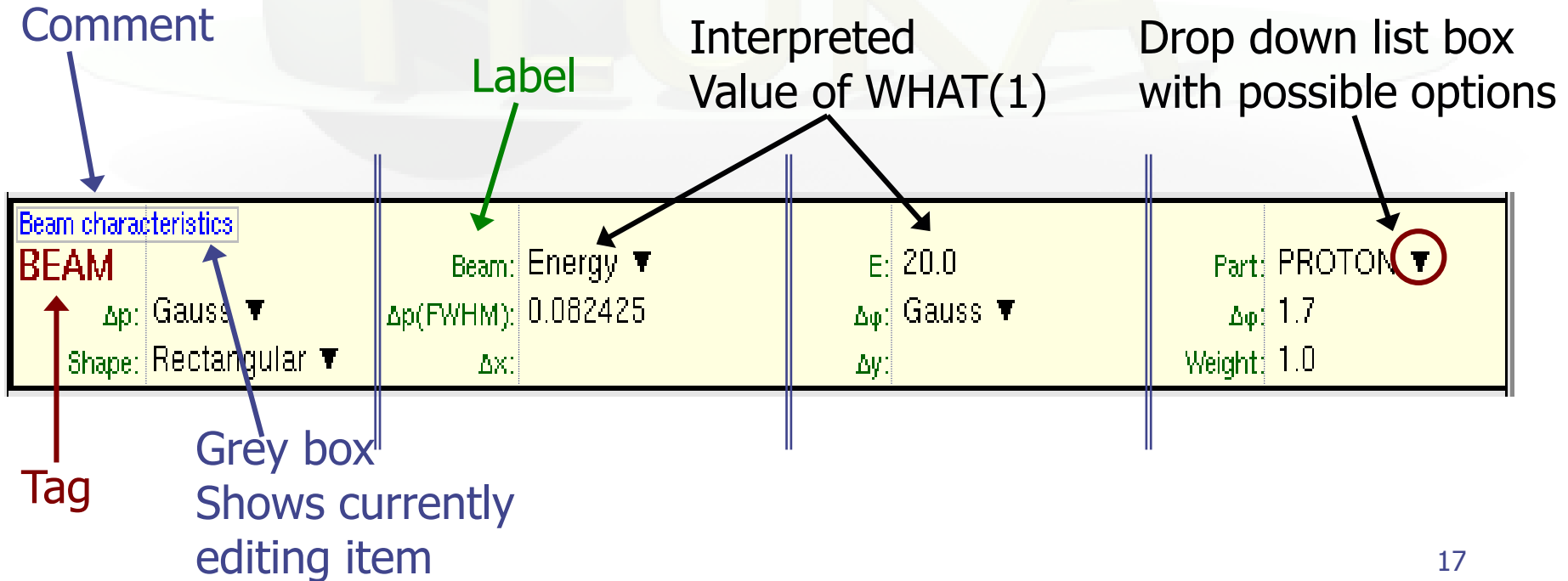
Anatomy of a card mini-dialog

```
* Energy deposition in 3D binning
USRBIN            10.0    ENERGY      -50.0      45.0      54.0      36.0EneDep
USRBIN           -45.0    -54.0       -33.0     100.0     100.0     100.0&
```

Anatomy of a card mini-dialog

```
* Polypyromellitimide Polyimide, Kapton
* Chemical        O = C    H-C    C = O
* Formula         / \   // \ / \       H-C - C-H         H-C - C-H
*                /   \C     C    \       //      \\       //      \\
*       ---- N    |    ||   N - C       C - O - C       C ----
*               \   /C     C    /       \      /       \      /
*   C  H  N O      \ /  \\ / \ /      H-C = C-H       H-C = C-H
*    22 10 2 5     O = C    H-C     C = O
```
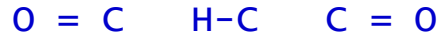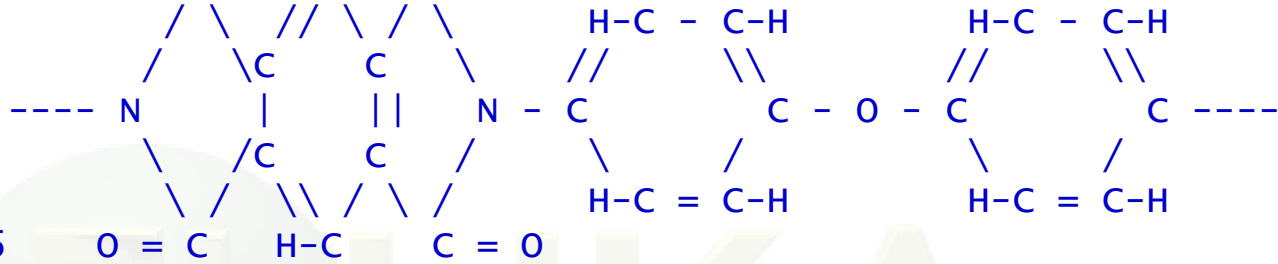
MATERIAL			1.43			Polyimid
COMPOUND	10.0	HYDROGEN	22.0	CARBON	2.0	NITROGENPolyimid
COMPOUND	5.0	OXYGEN				Polyimid

Editing Cards

While in input editor you can work in two modes:

1. Card mode: manipulate the cards as a unit (e.g. to copy, paste, delete, change order of cards)

2. Edit mode: manipulate the contents of the card

Edit mode is activated immediately after **adding** a new Card, by hitting **Enter** or with the **mouse click**

To leave edit mode click the **Esc** or with the mouse click somewhere else

The active item (what) is highlighted with a **grey rectangle** and highlighted also in the card viewer below the editor

A range of cards can be selected with:

● Shift + Mouse
● Shift + Up/Down arrows

Validating input and Error correction

- Flair validates the input file while loading and each card during editing

- Errors are highlighted in **red**

- For the moment only syntactical errors are checked, and a few logical errors

- Popup-menu option "Show errors" displays a short message on what is expected as correct value

- Menu item "Input / Filter Invalid" shows only the invalid cards from the last filtered view

Material Database

- Flair contains an **internal database** of ~500 predefined materials and/or compounds
- Some (~300) with the Sternheimer parameters

Please use these data as Reference only!

- Validate **always** the correctness of the data
- If errors found please contact the author

- The database can be edited, and populated with your own materials. In this case a local copy of the database will be made in ~/.flair directory

Starting a Run

- Flair can start a simulation (single run) based on the input file

- Multiple runs can be started by overriding some options, like #defines, title, random number seed and number of starting particles (primaries)

- Flair will try to "attach" to a run. Using only the information from the output files generated by FLUKA, flair will try to identify the directory where the run takes place and monitor the progress of the selected run

- During the execution of the run the user can view the output files in the "Files Frame" under the "Run Tab"

Tips & Tricks

- **Mouse**
  - right-click
opens the popup-menu with the most important actions

- **Keyboard**
Check the accelerators on the menus
  - Ctlr-Enter
Performs the default action in every frame.
e.g. Add a card in the Input Editor
   start a run in the Run Frame

  - Ctrl-Space
Access popup-menu (like right-clicking)
  - Listboxes
All listboxes in flair are searchable and case insensitive. Type the first characters of the string you are searching and the closest match will be highlighted
Ctrl-G repeats the search. Space selects/deselect item

  - +, -, Ins
While editing the REGION expression shows a list of all available bodies

Known Bugs / Limitations

- **Unicode / International** characters do not work well and should be avoided
- **Gnuplot:**
  - **<4.2** has a bug in the number of palette colors, and on the cblabel for the wxt terminal
  - **4.4.3** has a non linear behavior on the palette colors for the xterm terminal
- **Inline comments,** and comments inside REGION definition are treated as one comment preceding the input card

- **REMEMBER** always that the .flair and .inp are different thing
  Do not save the project as .inp or the input file as .flair

Other goodies

Flair has a lot of functionalities that are not shown in this tutorial

Most of these will be shown during the exercises of the course

We would advice the users to go through the various menus and help page and try it out

Flair – Geometry Editor – Part I

Beginners' FLUKA Course

Starting the Geometry Editor

Click on "Geometry" Tab

Geometry editor

- Working on 2D cross sections of the geometry;
- Interactive visual editing of the geometry in 2D;
- Debugging bodies/regions in a graphical way;
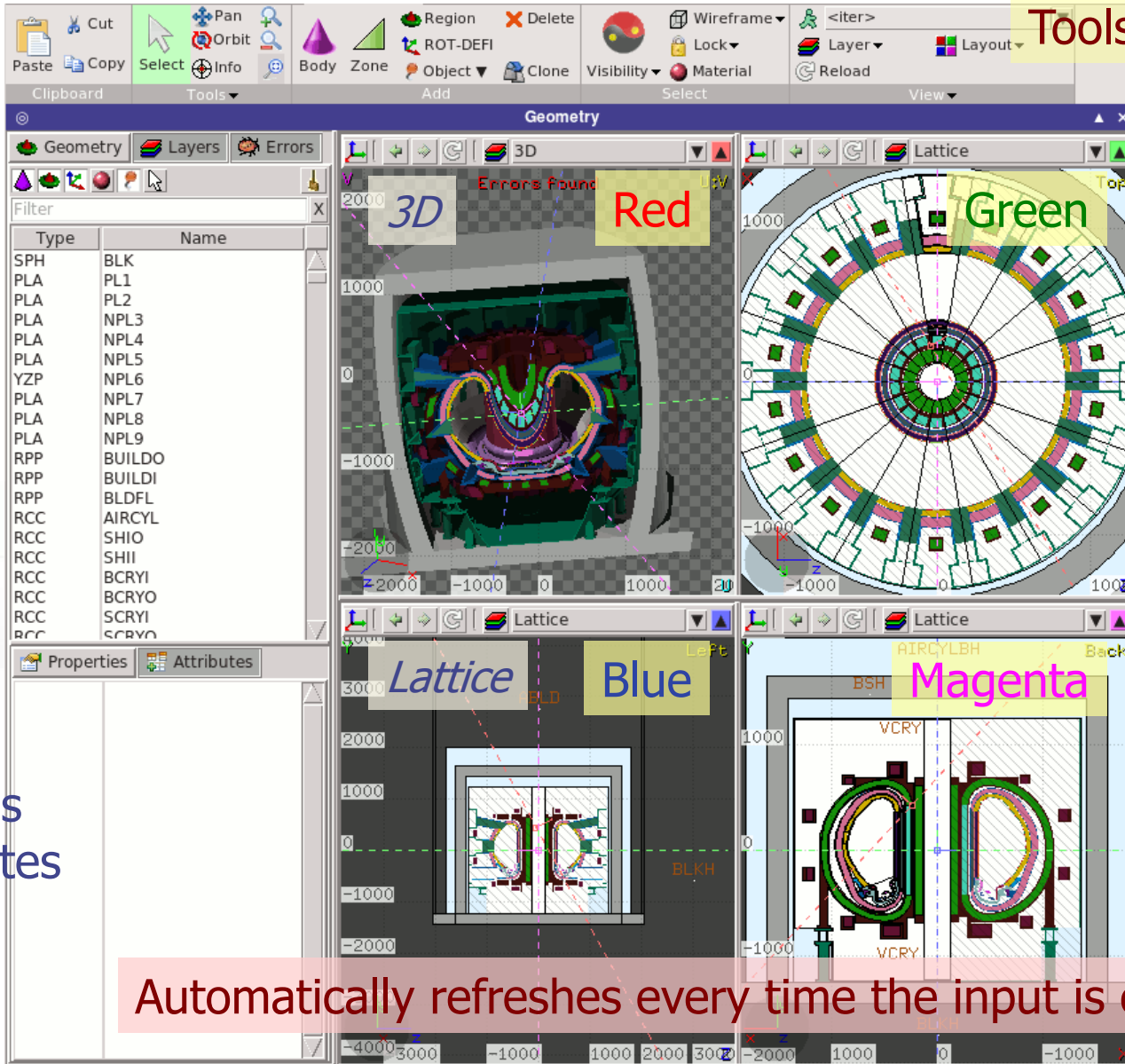- Fast 3D rendering of the geometry;

Pros

- Fast display of complex geometries;
- Many user-customizable layers;
- Graphical editing of the bodies with snapping mechanism to generate accurate coordinates;
- Visual selection and editing of zones w/o the need to know the orientation of bodies;
- Use real curve of bodies with no conversion to vertices/edges;
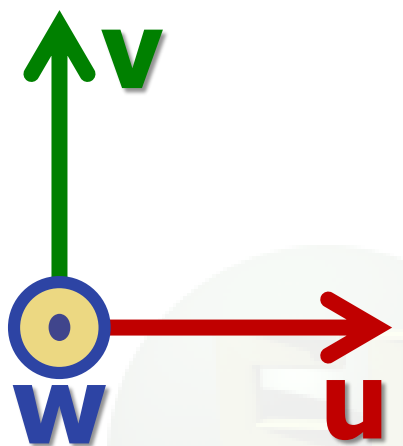- Interactive debugging with information of problematic bodies, regions and/or zones;

Cons

- Tricky to orientate in an unknown geometry;
- Difficult to find region using the expression;

Geometry Editor: Interface

Filter

Filtered Objects

Properties & Attributes

Tools

3D   Red

Green

Lattice   Blue

Magenta

Automatically refreshes every time the input is changed

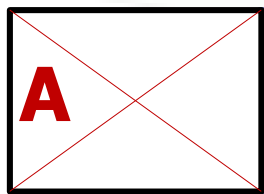# Viewport axes System

v

w   **u**

A

w

B

Each viewport is defined by:

- Origin   center of viewport
- Basis   relative axes system u, v, w. w is coming out of the screen towards the user
- Extends   zooming

Note:

- Each viewport is facing towards negative **–w**
- If bodies A, B are touching the viewport like on the plot.
- Only body B will be visible

Navigation - Keyboard

- [*arrows*]   pan viewport
- Ctrl + [*arrows*]   orbit viewport around **u,v** axes
    + [Shift]   rotates by 90°
- Page Up/ Page Down   pan viewport front/back
- Ctrl + PgUp/PgDn   rotate viewport around **w** axis
- = / -   zoom in / zoom out
- o   open projection dialog to set the **o**rigin/basis/save/recall etc…
- Ctrl-0 (zero)   Center to origin
- Ctrl-1, Ctrl-2   **front [X:Y] / back [-X:Y]**
- Ctrl-3, Ctrl-4   **left [Z:Y] / right [-Z:Y]**
- Ctrl-5, Ctrl-6   **top [Z:X] / bottom [-Z:X]**

  Assuming: *Z = direction of the beam (horizontal)*
                  X = horizontal
                  Y = vertical

Navigation – Mouse [1/2]

With the left mouse button:

1. Select the appropriate action pan/orbit/zoom with:
   I. Menu → Tools
   II. Toolbar
   III. Keyboard shortcut
2. Click and drag the desired viewport

	function	key	description
	Pan	x	Pan viewport
	Orbit	t	Orbit viewport using a virtual **t**rackball
	Zoom	z	Drag area to **z**oom In ([**Ctrl**] to zoom out)
		Shift-Z	Zoom viewport on selected items
		Alt-Left	Go to previous in history projection
		Alt-Right	Go to next in history projection

Navigation – Mouse

- With the **middle** mouse button
    - click-only   Center position and set pivot point for rotations
    - drag   Pan/Move viewport
    - Ctrl   orbit projection using a virtual trackball
    - Ctrl-Middle-Shift   orbit projection using a virtual trackball with steps of 15 degrees
    - Shift   select rectangle region and zoom into
    - Shift-Middle-Ctrl   select rectangle region and zoom out
- Wheel (if any)   zoom in/zoom out
    - Ctrl-Wheel   pan/move forward or backward
    - Ctrl-Shift-Wheel   smoother pan/move forward/backward
- With the **right** mouse button
    - click   opens popup menu
    - Shift   pan/move viewport
   - Ctrl   orbit projection using a virtual trackball

When <u>laptop mode</u> is enabled in the <u>Preferences/Geometry</u> then the <u>middle</u> and <u>right</u> buttons are <u>swapped</u>

Navigation – Viewport lines [1/2]

3D Viewing direction

Description:

- Dashed lines represent other viewports (the intersection of other viewports with the current one);

- The center is represented with a square;

- Viewing direction **w** is indicated by a short line;

- When another viewport is outside the view window, the viewport-line will be displayed on the closest edge;
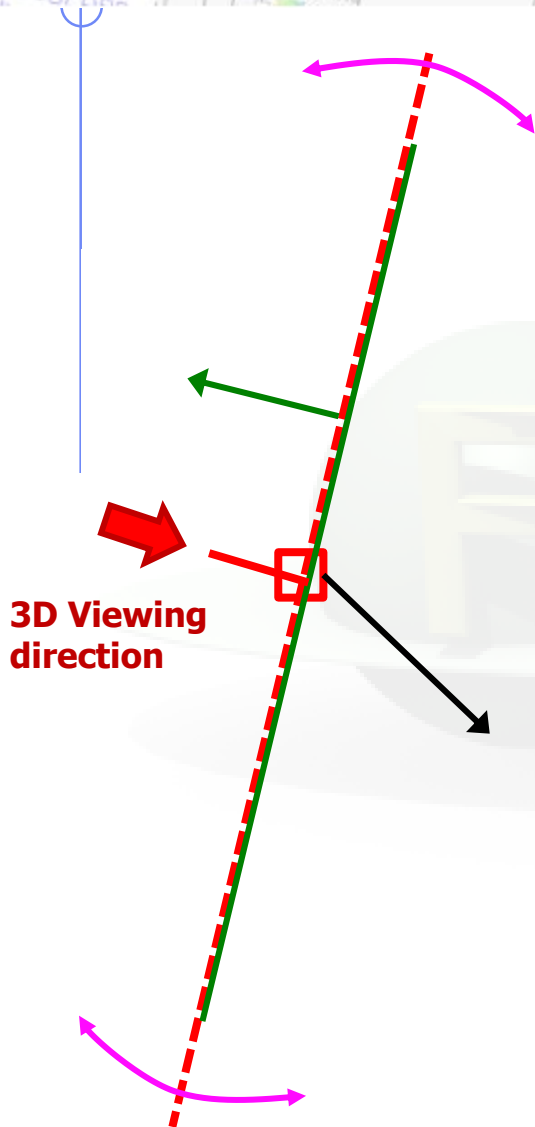
Actions: Select ⌖ + left mouse button

- <u>Drag the center</u> square to reposition the viewport

- <u>Drag the line close to the center</u> to reposition the viewport along the vertical **w** axis

- <u>Drag the extremities</u> to rotate it

- Hold [Shift] when dragging to snap on grid or angle intervals

35

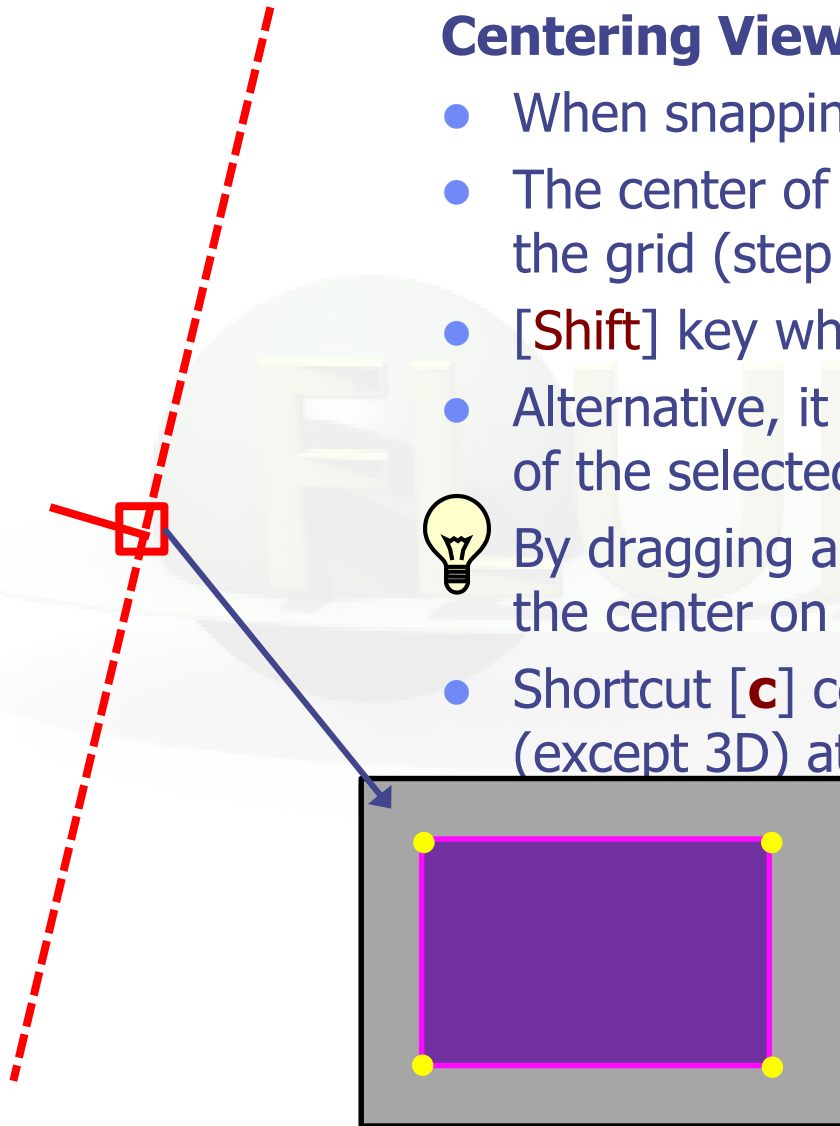# Navigation – Viewport lines

Centering Viewports

- When snapping to grid is activated
- The center of the viewport will be aligned to the grid (step of 1/10 of the main grid)
- [Shift] key while toggle the snapping action;
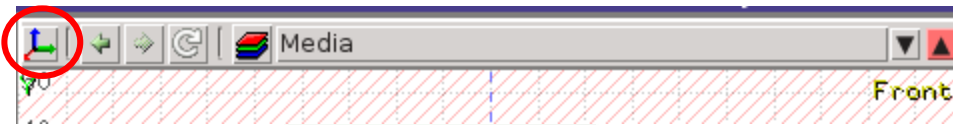- Alternative, it can be centered on the vertices of the selected bodies;

By dragging a viewport center it always moves the center on the current viewing plane.

- Shortcut [c] centers all other viewports (except 3D) at the mouse pointer

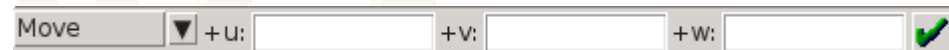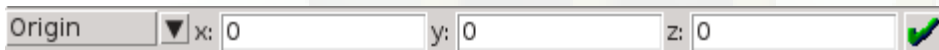# Navigation – Projection dialog

With the projection [**o**] ⚹ button you can change, move, shift, rotate, save and reload the projection of a viewport

| 🔴 | ← | → | ⟳ | 🔴 Media | ▼ ▲ |

Front

Shift the coordinate system

| Move | ▼ | +u: | +v: | +w: | ✔ |

Set the **o**rigin of the viewport

| Origin | ▼ | x: 0 | y: 0 | z: 0 | ✔ |

Change the reference axis

| Basis | ▼ | ux: 1.0 | uy: 0.0 | uz: 0.0 | x-y | x-z | y-z | ↻ ↔ ⚹ | ✔ |
| | | vx: 0.0 | vy: 1.0 | vz: 0.0 | -v | -u | swap | ⚹ ↔ ⚹ | |

Rotate around the Cartesian axis

Shortcut: Ctrl + (1-6)

| Euler | ▼ | Rx: 0 | Ry: -0 | Rz: 0 | ✔ |
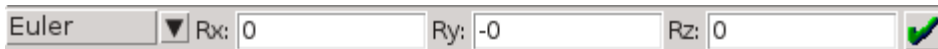
Add and Select Bookmark

| Bookmarks | ▼ | Plot: Red | ▼ ✚ − 💾 ✔ |
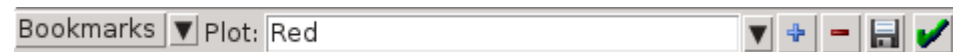
Select Transformation

| Transform | ▼ | Rotdefi: | ▼ | ✔ |

Debugging Geometry Errors

Undefined region →

← Overlapping regions
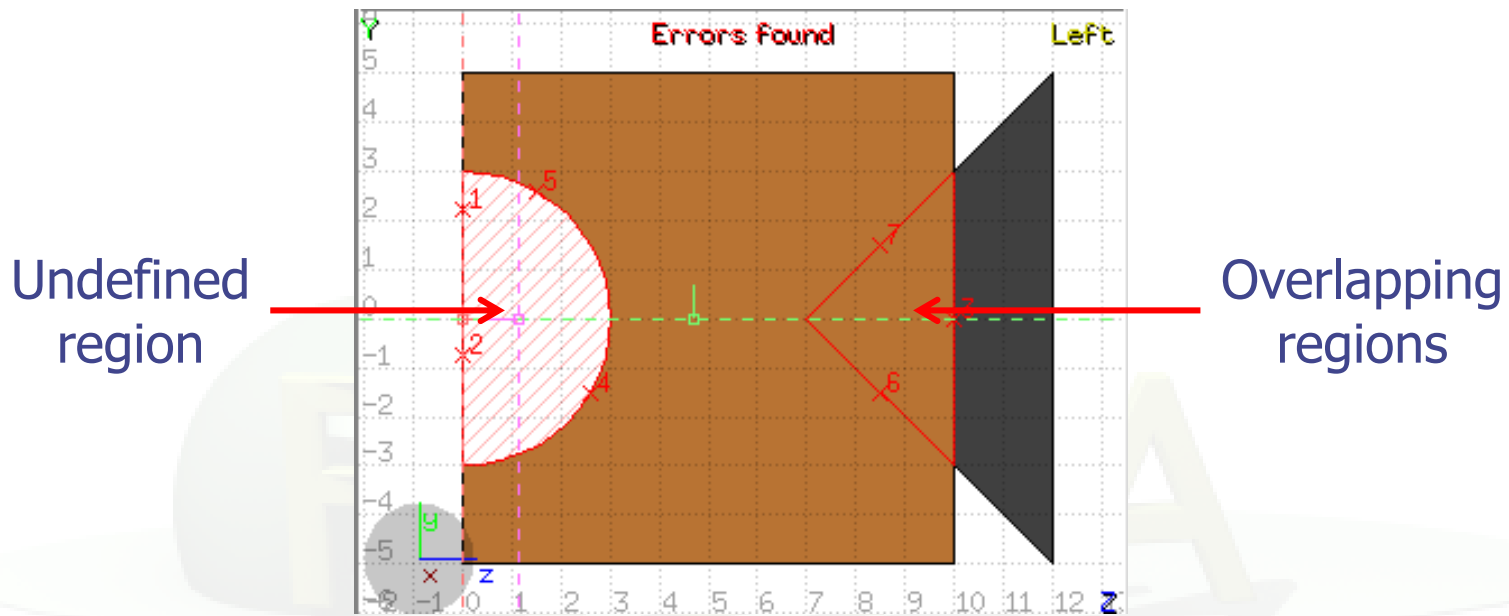
Errors found notifies that are errors in the geometry (on the current projection):

- The areas affected by the errors are outlined with a **Red** stroke:
  - Areas filled with a full color correspond to overlapping regions;
  - Areas filled with red lines correspond to a missing region definition;
  - Body segments that are involved in the errors are numbered;
- Clicking the 🐞 Errors tab (on the left) displays the dialog with the errors.
- Touching surfaces are checked against **10** significant digits
- Non-strictly geometrical errors (i.e. missing Material Assignment to a region, non recognized cards) are also notified;