



Advanced Geometry

FLUKA Beginner's Course

Contents

- Body Transformations
- Lattice

Geometry directives

Special commands enclosing body definition:

`$Start_xxx`

.....

`$End_xxx`

where "xxx" stands for
"expansion", "translat" or "transform"

They provide respectively a coordinate **expansion/reduction**, a coordinate **translation** or a coordinate **roto-translation** of the bodies embedded between the starting and the ending directive lines.

<code>\$start_transform</code>	Trans:	▼		
<code>\$end_transform</code>				
<code>\$start_expansion</code>	f:			
<code>\$end_expansion</code>				
<code>\$start_translat</code>	dx:	dy:	dz:	
<code>\$end_translat</code>				

Directives in geometry: expansion/reduction

➤ `$Start_expansion ... $End_expansion`


it provides a coordinate expansion (reduction) factor **f** for all bodies embedded within the directive

`$Start_expansion 10.0`

SPH Sphere 5.0 7.0 8.0 50.0

`$End_expansion`

transforms a sphere of radius 50 centered in (+5,+7,+8) into a sphere of radius 500 centered in (+50,+70,+80)

<code>\$start_expansion</code>	f:10.0		
 <code>SPH Sphere</code>	x:5.0	y:7.0	z:8.0
	R:50.0		
<code>\$end_expansion</code>			

Putting the body in its quadric form

$$A_{xx} x^2 + A_{yy} y^2 + A_{zz} z^2 + A_{xy} xy + A_{xz} xz + A_{yz} yz + A_x x + A_y y + A_z z + A_0 = 0$$

$$\text{or } \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} A_{xx} & A_{xy}/2 & A_{xz}/2 & A_x/2 \\ A_{xy}/2 & A_{yy} & A_{yz}/2 & A_y/2 \\ A_{xz}/2 & A_{yz}/2 & A_{zz} & A_z/2 \\ A_x/2 & A_y/2 & A_z/2 & A_0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0 \quad \text{i.e. } \mathbf{r}^T \mathbf{M}_{\text{QUA}} \mathbf{r} = 0$$

the expansion/reduction matrix is $\mathbf{T} =$

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the transformed body equation is $\mathbf{r}^T (\mathbf{T}^{-1})^T \mathbf{M}_{\text{QUA}} \mathbf{T}^{-1} \mathbf{r} = 0$

Directives in geometry: translation

➤ `$Start_translat ... $End_translat`


it provides a coordinate translation S_x, S_y, S_z for all bodies embedded within the directive

```
$Start_translat -5.0 -7.0 -8.0
```

```
SPH Sphere 5.0 7.0 8.0 50.0
```

```
$End_translat
```

transforms a sphere of radius 50 centered in (+5,+7,+8)
into a sphere of radius 50 centered in (0,0,0)

<code>\$start_translat</code>	dx: -5.0	dy: -7.0	dz: -8.0
 SPH Sphere	x: 5.0 R: 50.0	y: 7.0	z: 8.0
<code>\$end_translat</code>			

the translation matrix is $T =$

$$\begin{bmatrix} 1 & 0 & 0 & S_x \\ 0 & 1 & 0 & S_y \\ 0 & 0 & 1 & S_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Directives in geometry: roto-translation

➤ `$Start_transform ... $End_transform`

it applies a pre-defined (via **ROT-DEFI**) roto-translation to all bodies embedded within the directive

```
ROT-DEFI , 201.0, 0., +116.5650511770780, 0., 0., 0., cylrot
```

```
$Start_transform cylrot
```

```
QUA cylinder 0.5 1.0 0.5 0.0 1.0 0.0 0.0 0.0 0.0 -4.0
```

```
$End_transform
```

transforms an infinite circular cylinder of radius 2 with axis $\{x=-z, y=0\}$ into an infinite circular cylinder of radius 2 with axis $\{x=z/3, y=0\}$ (**clockwise rotation**)

the roto-translation matrix is $T = \begin{bmatrix} & & & S_x \\ & R & & S_y \\ & & & S_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$T^{-1} = \begin{bmatrix} & & & -(R^{-1}S)_x \\ & R^{-1} & & -(R^{-1}S)_y \\ & & & -(R^{-1}S)_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- it allows to rotate a **RPP** avoiding the use of the deprecated **BOX** !

- note that also the **inverse** transformation can be used T^{-1}

```
$Start_transform -cylrot
```

ROT-DEFIni

The **ROT-DEFIni** card defines roto-translations that can be applied, in addition to bodies, to i. **USRBIN & EVENTBIN** and ii. **LATTICE**. It transforms the position of the tracked particle i. before scoring with respect to the defined binning or ii. into the prototype with the order:

- First applies the **translation**
- followed by the rotation on the **azimuthal angle**
- and finally by the rotation on the **polar angle**.

$$\mathbf{X}_{\text{new}} = \mathbf{M}_{\text{polar}} \times \mathbf{M}_{\text{az}} \times (\mathbf{X} + \mathbf{T})$$

WHAT(1): assigns a **transformation index** and the corresponding rotation axis **I + J ***
100 or **I * 1000 + J**

I = index of rotation (**WARNING: NOTE THE SWAP OF VARIABLES**)

J = rotation with respect to axis (1=X, 2=Y, 3=Z)

WHAT(2): Polar angle of the rotation ($0 \leq \vartheta \leq 180^\circ$ degrees)

WHAT(3): Azimuthal angle of the rotation ($-180 \leq \varphi \leq 180^\circ$ degrees)

WHAT(4), WHAT(5), WHAT(6) = X, Y, Z offset for the **translation**

SDUM: Optional (but recommended) name for the transformation

ROT-DEFI

Id: 1

Axis: Z ▼

Name: 1tra

Polar: 0.0

Azm:

Δx:

Δy:

Δz: -10.0

Directives in geometry: warnings

- `$Start_expansion` and `$Start_translat` are applied when reading the geometry → no CPU penalty (the concerned bodies are transformed once for ever at initialization)

`$Start_transform` is applied runtime → some CPU penalty

- One can **nest** the different directives (*at most one per type!*) but, no matter the input order, the adopted sequence is always the following:

```
$Start_transform StupiRot
```

```
$Start_translat -5.0 -7.0 -8.0
```

```
$Start_expansion 10.0
```

```
QUA whatIsIt +1.0 +1.0 +1.0 0.0 0.0 0.0 -10.0 -14.0 -16.0 -2362.0
```

```
$End_expansion
```

```
$End_translat
```

```
$End_transform
```

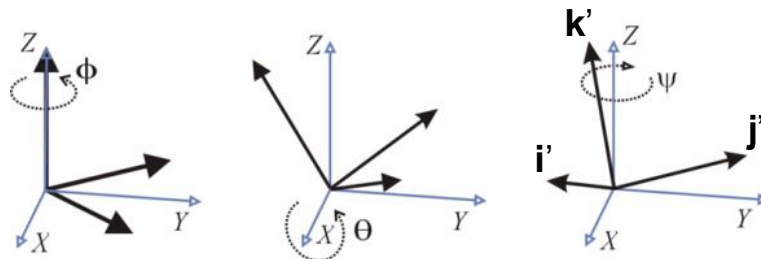
- Directives are not case sensitive (whereas roto-translation names are)

Identifying rotation angles

Let's define the orientation of a body in the space by a system of 3 orthogonal vectors \mathbf{i}' , \mathbf{j}' , \mathbf{k}' , whose coordinates are expressed with respect to the fixed reference frame X, Y, Z

$$\text{Then } [\mathbf{i}' \ \mathbf{j}' \ \mathbf{k}'] = \begin{bmatrix} c_1 c_3 - c_2 s_1 s_3 & -c_1 s_3 - c_3 c_2 s_1 & s_2 s_1 \\ c_2 c_1 s_3 + c_3 s_1 & c_1 c_2 c_3 - s_1 s_3 & -c_1 s_2 \\ s_3 s_2 & c_3 s_2 & c_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \text{ (in the ZXZ convention)}$$

where $c_1 = \cos(\psi)$ $c_2 = \cos(\theta)$ $c_3 = \cos(\Phi)$ $s_1 = \sin(\psi)$ $s_2 = \sin(\theta)$ $s_3 = \sin(\Phi)$



here $\Phi = 45^\circ$ $\theta = 30^\circ$ $\psi = -60^\circ$

The obtained Euler angles can be input as azimuthal angle of three consecutive rotations (**ROT-DEFI**)

Lattice

FLUKA geometry has *replication* (lattice) capabilities

Only one level is implemented (no nested lattices are allowed)

- The user defines lattice positions in the geometry and provides transformation rules from the lattice to the prototype region:
 1. in the input with the ROT-DEFI card
 2. in a subroutine (`latic.f`)

The lattice identification is available for scoring

Transformations include:

Translation, Rotation and Mirroring (the last only through routine).

WARNING:

Do not use **scaling or any deformation** of the coordinate system

Lattice

- The regions which constitute the **elementary cell** (*prototype*) to be replicated, have to be defined in detail
- The **Lattices** (*replicas/containers*) have to be defined as “empty” regions in their correct location.
WARNING: The lattice region **should map exactly** the outer surface definition of the elementary cell.
- The lattice regions are declared as such with a **LATTICE** card at the end of the geometry input
- In the **LATTICE** card, the user also **assigns lattice names/numbers to the lattices**. These names/numbers will identify the replicas in all FLUKA routines and scoring
- Several basic cells and associated lattices can be defined within the same geometry, one **LATTICE** card will be needed for each set
- **Non-replicas carry the lattice number 0**
- Lattices and plain regions can coexist in the same problem

LATTICE card

After the Regions definition and before the GEOEND card the user can insert the LATTICE cards

- WHAT(1), WHAT(2), WHAT(3)
Container region range (from, to, step)
- WHAT(4), WHAT(5), WHAT(6)
Name/number(s) of the lattice(s)
- SDUM
 blank to use the transformation from the **lattic** routine
 ROT#nn to use a **ROT-DEFI** rotation/translation from input
 name the same as above but identifying the roto-translation by the name assigned in the **ROT-DEFI** SDUM (any alphanumeric string you like)

Example

```

LATTICE                   Reg: TARGR1 ▼           to Reg: ▼           Step:
          Id: 1tra ▼           Lat: 1.0           to Lat: 1.0           Step: 1.0
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
LATTICE           6.00000   19.00000                   101.0000   114.00
  
```

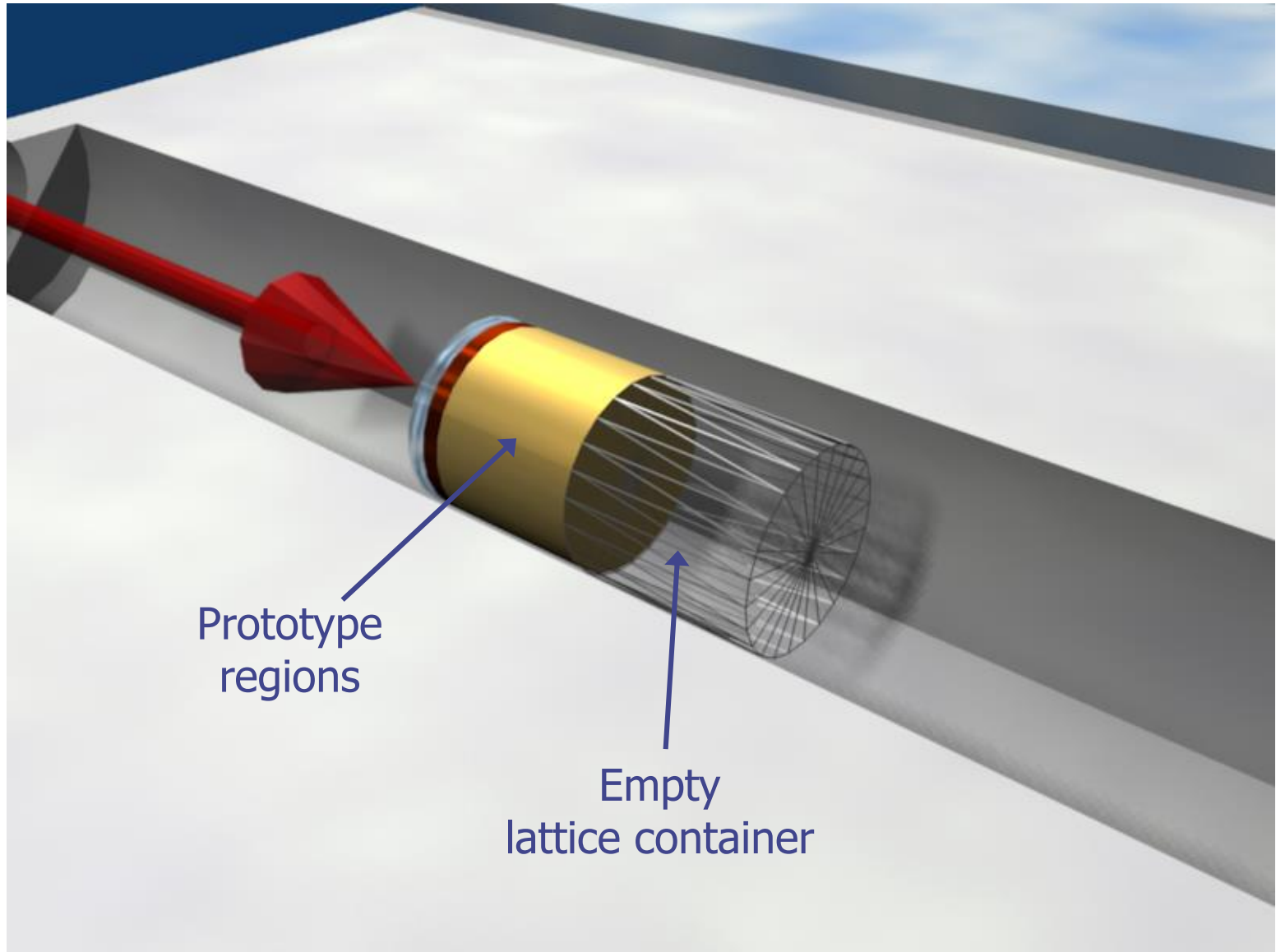
Region # 6 to 19 are the “placeholders” for the first set replicas. We assign to them lattice numbers from 101 to 114

```

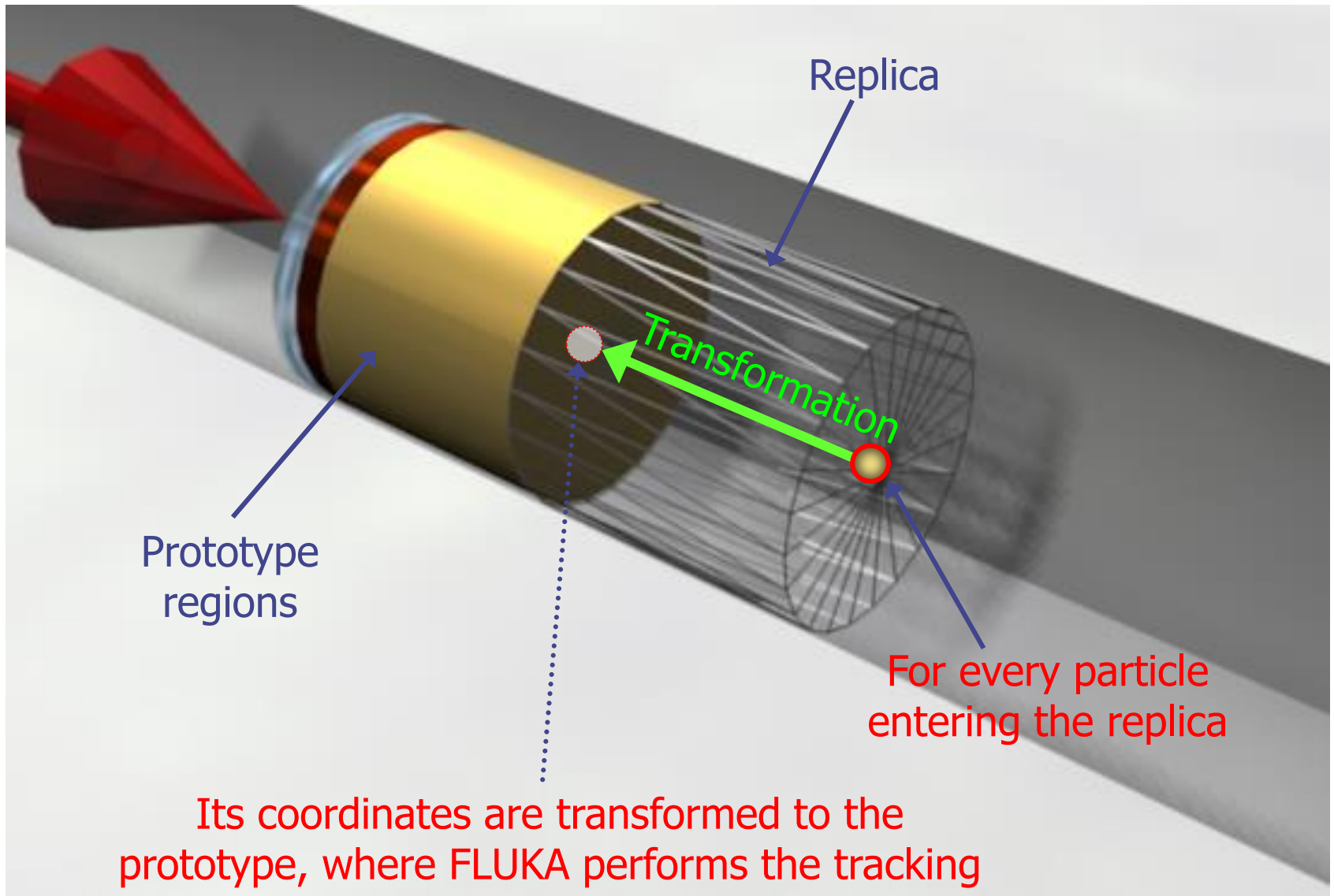
LATTICE           TARGR1                                   TargRep                                   1tra
  
```

TARGR1 is the container region using transformation *1tra*

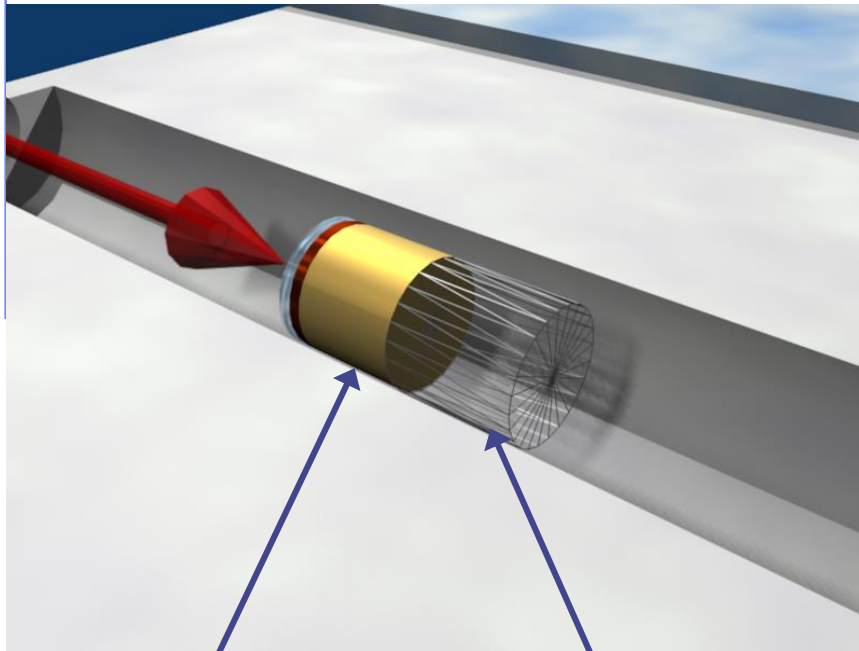
Example



Example

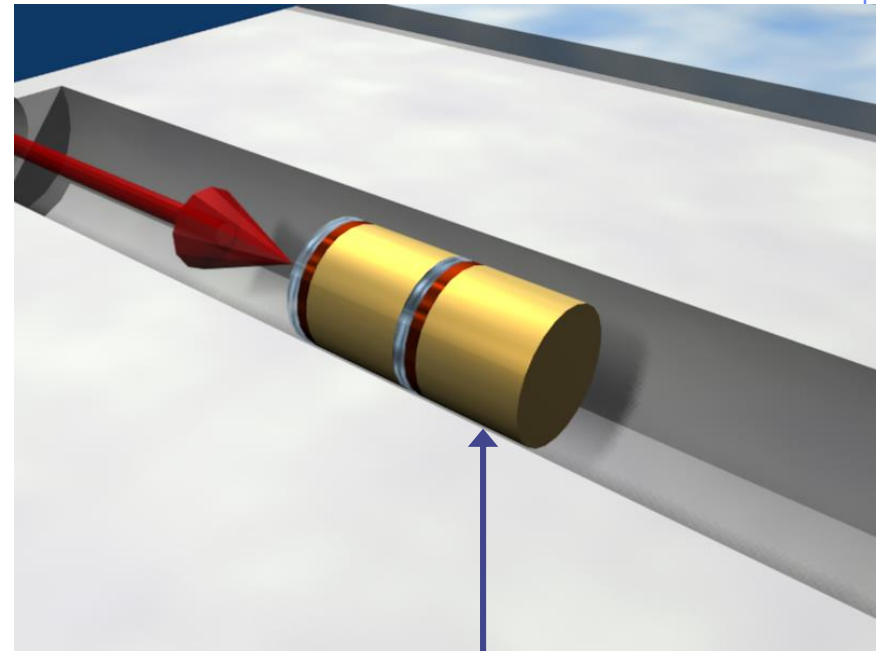


Example



Prototype
cell

Empty
lattice cell



Final
replica

Transformation by input

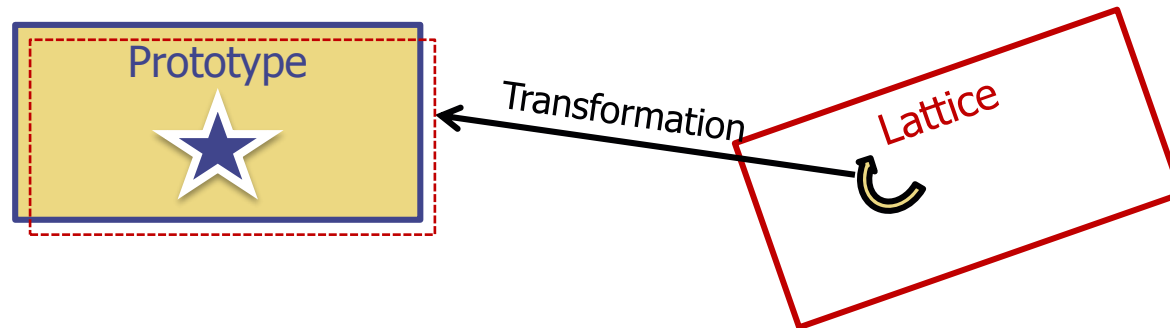
- Rotations/Translations can be defined with the **ROT-DEFIni** card
- Can be assigned to a lattice by **name** or with **ROT#nnn** SDUM in the **LATTICE** card
- **ROT-DEFIni** cards can be consecutive (using the same **index** or **name**) to define complex transformations

WARNING:

Since matrix multiplication is not commutative the **order** of the Rotation/Translation operations in 3D is important.

Numerical Precision

- Due to the nature of the floating point operations in CPU, even if the transformation looks correct the end result could be problematic



This small misalignment between lattice/transformation/prototype could lead to geometry errors

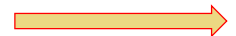
- Use as many digits as possible to describe correctly the prototype and lattice cells as well as the transformation.

It is mandatory that the transformation applied to the container makes the latter **EXACTLY** corresponding to the prototype

- One can use a **FREE** and **FIXED** card before and after the **ROT-DEFI** to input more than 9 digits
- **GEOBEGIN WHAT(2)** allows to relax the accuracy in boundary identification (**USE WITH CAUTION**)

Lattice: Important remarks

- Materials and other properties have to be assigned only to the regions constituting the prototype.
- In all (user) routines the **region number** refers to the corresponding one in the prototype.
- The **SCORE** summary in the **.out** file and the scoring by regions add together the contributions of the prototype region as well as of all its replicas!
- The lattice identity can be recovered runtime by the *lattice number*, as set in the **LATTICE** card, or available through the **GEON2L** routine if it is defined by name
- In particular, the **LUSRBL** user routine allows to manage the scoring on lattices in the special **USRBIN/EVENTBIN** structure.



The USRBIN/EVENTBIN special binning

EVENTBIN or **USRBIN** with **WHAT(1)=8** :

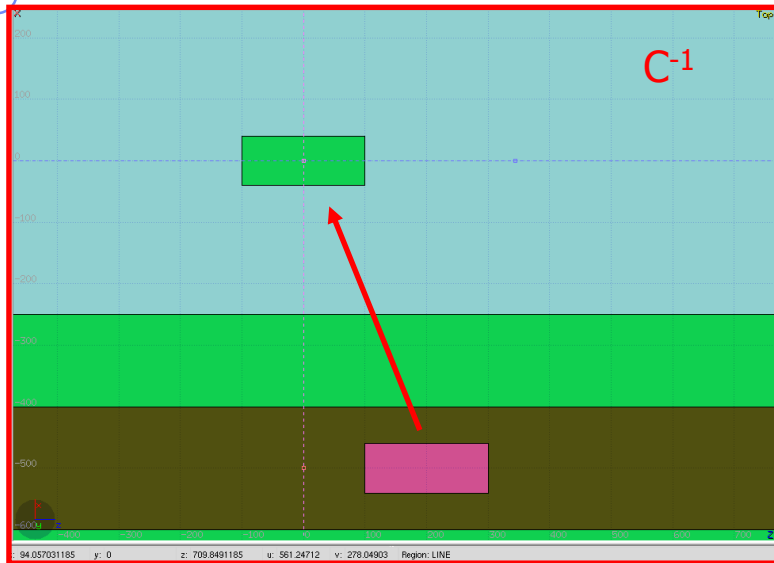
Special user-defined 3D binning. Two variables are discontinuous (e.g. region number), the third one is continuous, but not necessarily a space coordinate.

Variable	Type	Default	Override Routine
1 st	integer	region number	MUSRBR
2 nd	integer	lattice cell number	LUSRBL
3 rd	float	zero	FUSRBV

Tips & Tricks

- Always remember that the transformation must bring the container onto the prototype and not viceversa!
- You can always divide a transformation into many **ROT-DEFI** cards for easier manipulation.
- Rotations are always around the origin of the geometry, and not the center of the object.
 - To rotate an object, first translate the object to the origin of the axes
 - Perform the rotation
 - Move it by a final translation to the requested position.
Of course with the inverse order since everything should apply to the replica
- In order to define the replica body, you can clone the body enclosing the prototype (assigning it a new name!) and apply to it the `$Start_transform` directive with the inverse of the respective **ROT-DEFI** transformation.

Tips & Tricks



GEOBEGIN

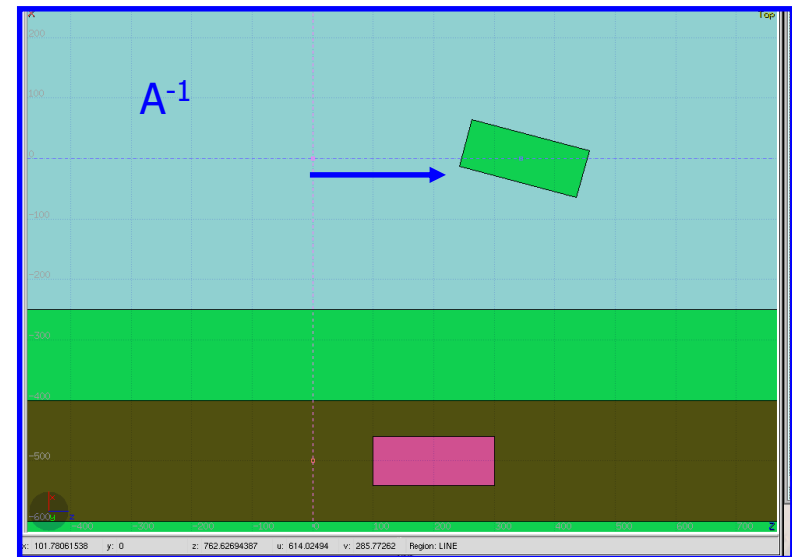
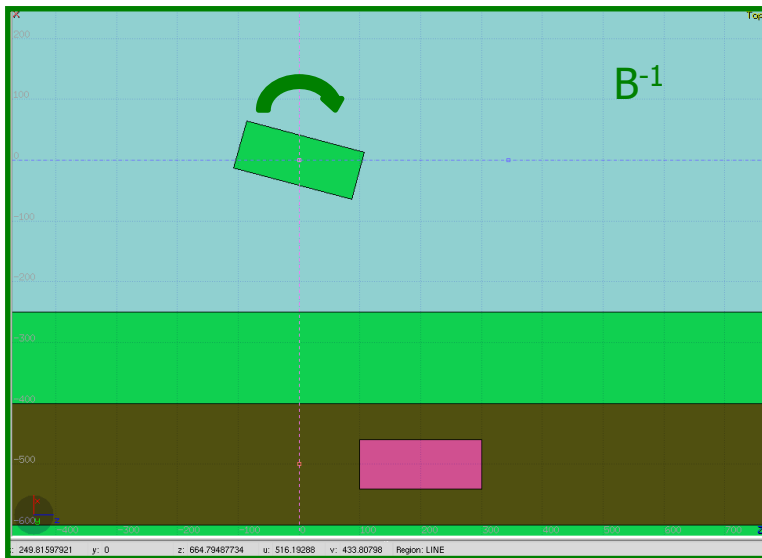
```
...
RPP CollProt -540.0 -460.0 -20.0 20.0 100.0 300.0
$start_transform -rotColl *
RPP CollRepl -540.0 -460.0 -20.0 20.0 100.0 300.0
$end_transform
```

...

GEOEND

```
ROT-DEFI, 1.0, 0.0, 0.0, 0.0, 0.0, -350.0, rotColl [A]
ROT-DEFI, 201.0, 0.0, -15.0, 0.0, 0.0, 0.0, rotColl [B]
ROT-DEFI, 1.0, 0.0, 0.0, -500.0, 0.0, 200.0, rotColl [C]
```

* Remember: if $R=CBA$, then $R^{-1}=A^{-1}B^{-1}C^{-1}$



Tips & Tricks

- The **Geometry transformation editor in flair** can read and write **ROT-DEFI** cards with the transformation requested
- An easy way of creating a replica and the associated transformation is the following:
 1. Select the body defining the outer cell of the prototype
 2. Clone it with (**Ctrl-D**) and change the name of the clones. Click on “**No**” when you are prompted to change all references to the original name.
 3. Open the Geometry transformation dialog (**Ctrl-T**)
 4. Enter the transformation of the object in the listbox
 5. Click on “**Transform**” to perform the transformation on the clone bodies
 6. Click on “**Invert**” button to invert the order of the transformation
 7. Enter a name on the “**ROT-DEFIni**” field and click “**Add to Input**” to create the **ROT-DEFIni** cards
 8. Now you have to create manually the needed **regions** and the **LATTICE** cards