Advanced Geometry

20th FLUKA Beginner's Course
Stellenbosch University
28 May – 01 June 2018

Contents

- Parenthesis
- Body Transformations
  Expansion, translate, transform
- ROT-DIFIni
- LATTICE
- Examples and Exercises

Parenthesis

Parentheses can be used to group together combinations of bodies. Parentheses can be used in free format only.

- Parentheses expansion is almost like converting
  from: product of sums
  to: sum of products
- Product operators are: +/-, Sum operator: |
- The final result will be an expression in the normal form. Unions of all possible combinations of the bodies in the expression!
- Initially the code removes all repeated terms:
  A + A = A
  A – A = ∅
  expA | expA = expA

Parentheses Expansion

Nested parentheses are supported, however:

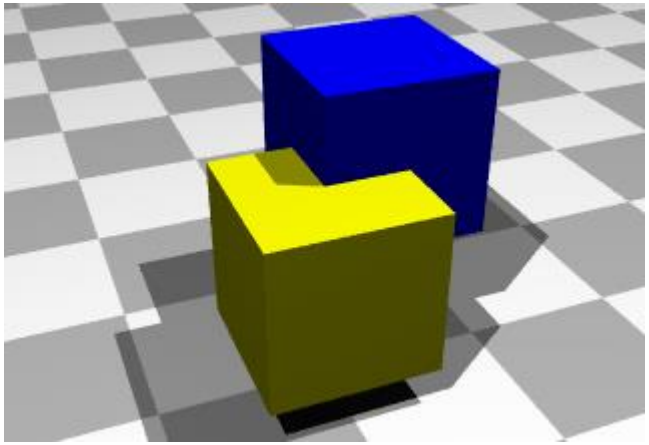
parentheses should be used with care since their expansion can generate a quickly diverging amount of terms. A partial optimization is performed on planes (aligned with the axes) and bounding boxes only

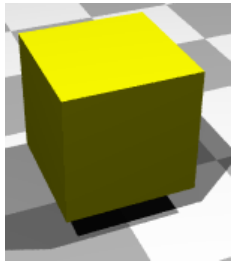
Use parentheses with care!!!

It can easily create an expansion with thousands of terms!

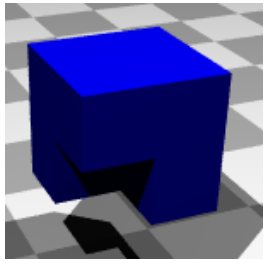
Proposed exercise

```
TITLE
        Two cubes
*
GLOBAL                                                1.
DEFAULTS                                                                PRECISIO
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
GEOBEGIN                  0.01
   0    0          Two cubes geometry
*== Body Definitions ================================================
* define external void
SPH EXTVOID  0.0 0.0 0.0 1000000.0
* First cube planes
XYP Front1      -10.0
XYP Back1        10.0
XZP Bottom1     -10.0
XZP Top1         10.0
YZP Right1      -10.0
YZP Left1        10.0
* Second cube planes
XYP Front2      -20.0
XYP Back2          0.0
XZP Bottom2        0.0
XZP Top2          20.0
YZP Right2         0.0
YZP Left2         20.0
END
*== Region Definitions ==============================================
BLACK   5 +EXTVOID
          -(+Back1 -Front1 +Top1 -Bottom1 +Left1 -Right1)
          -(+Back2 -Front2 +Top2 -Bottom2 +Left2 -Right2)
*
CUBE1   5 +Back1 -Front1 +Top1 -Bottom1 +Left1 -Right1
CUBE2   5 +Back2 -Front2 +Top2 -Bottom2 +Left2 -Right2
          -(+Back1 -Front1 +Top1 -Bottom1 +Left1 -Right1)
END
GEOEND
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
* Blackhole
ASSIGNMAT   BLCKHOLE     BLACK
ASSIGNMAT      EARTH     CUBE1
ASSIGNMAT   ALUMINUM     CUBE2
*START         500.  999999999.   180000.             0.0
STOP
```





CUBE1



CUBE2

Geometry directives

Special commands enclosing body definition:

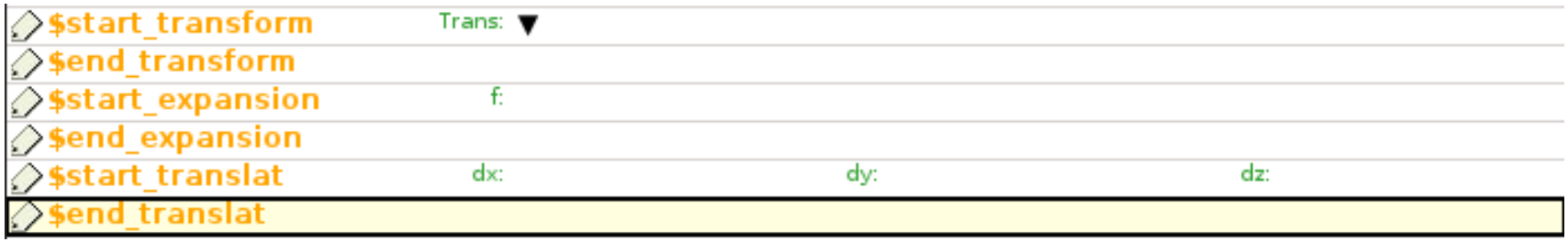
$Start_xxx

.....

$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.



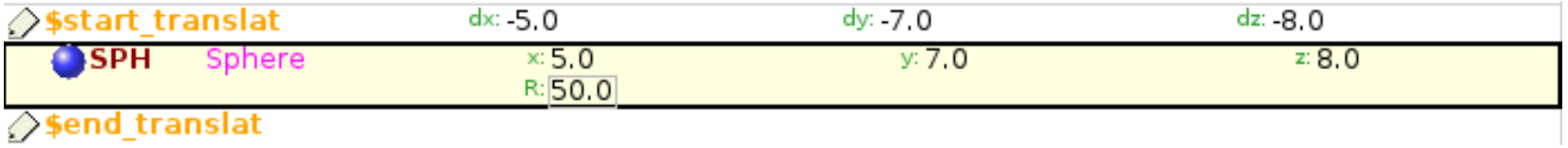
Directives in geometry: translation

- $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)



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

Example: Translation

- Open the file ex_geometry2.inp in Examples/example_geometry2
- Define a sphere centered in (0.0, 0.0, 5.0), with radius = 5 cm, and made of WATER

```
SPH target1      0.0      0.0      5.       5.
```

SPH target1 x: 0.0 y: 0.0 z: 5. R: 5.

```
* Void around
VOID         5 +void-target1
* Target
TARGET1      5 +target1
```

Void around
REGION VOID Neigh: 5 expr: +void-target1

Target
REGION TARGET1 Neigh: 5 expr: +target1

```
ASSIGNMA      WATER   TARGET1
```

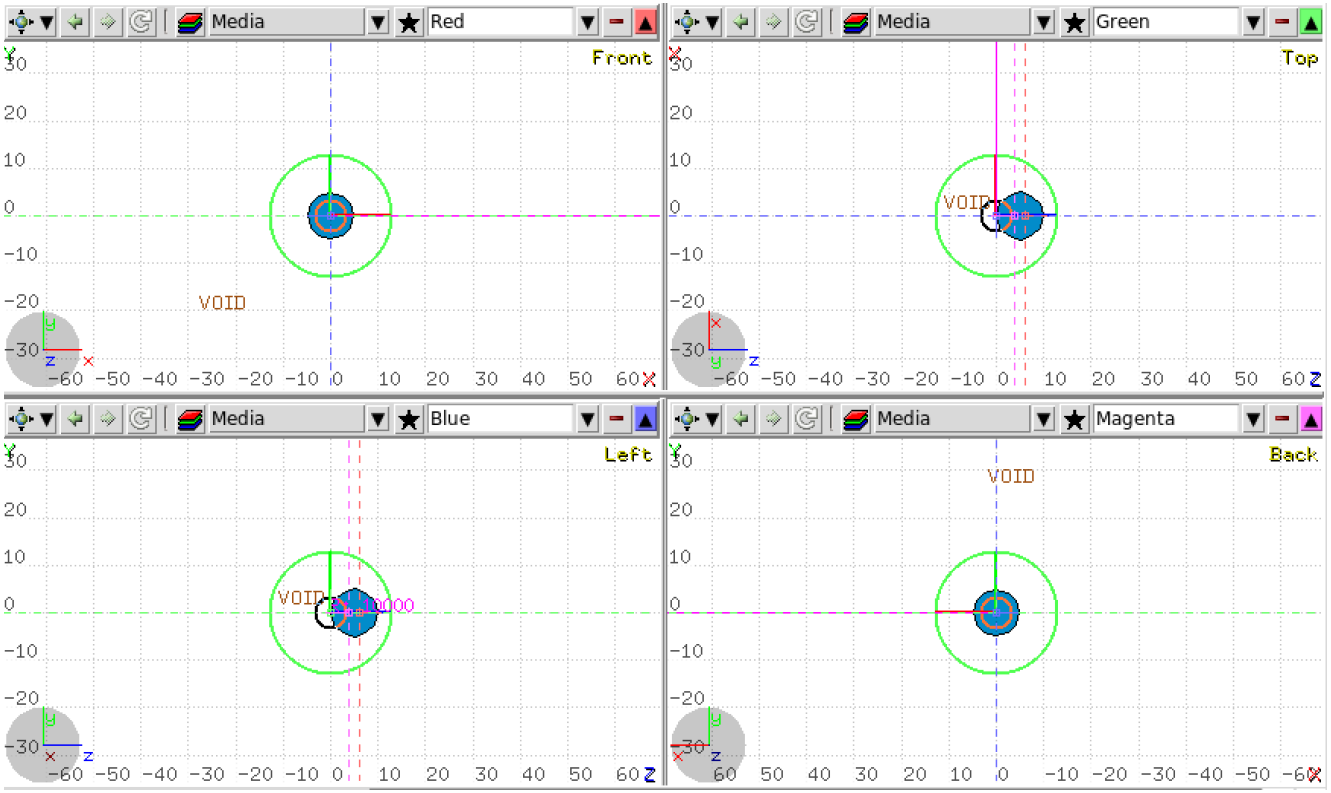
ASSIGNMA Mat: WATER Reg: TARGET1 to Reg: Mat(Decay): Step: Field:

Example: Translation

- Define a sphere centered in (0.0, 0.0, 5.0), with radius = 5 cm, and made of WATER

```
SPH target1      0.0      0.0      5.       5.
```

SPH target1 x: 0.0 y: 0.0 z: 5. R: 5.

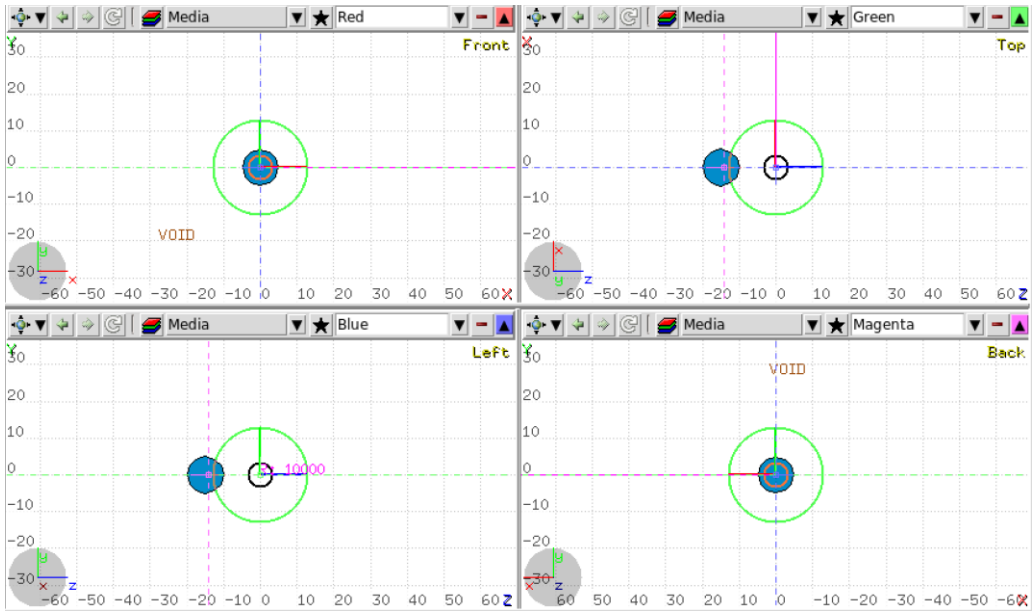


Example: Translation

- Apply a translation of -10 cm along z to the previously defined sphere.

```
$start_translat     -20
SPH target1    0.0 0.0 5. 5.
$end_translat
```

$start_translat		dx:	dy:	dz: -20
SPH	target1	x: 0.0 R: 5.0	y: 0.0	z: 5.0
$end_translat				

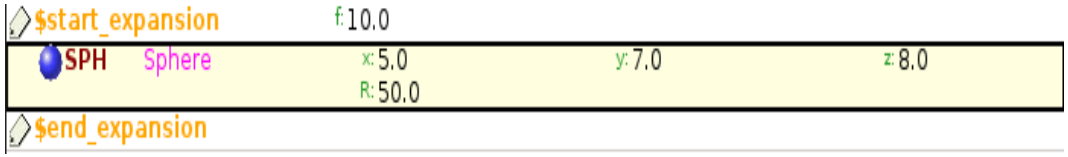


Directives in geometry: expansion/reduction

- $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)

Putting the body in its quadric form

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

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

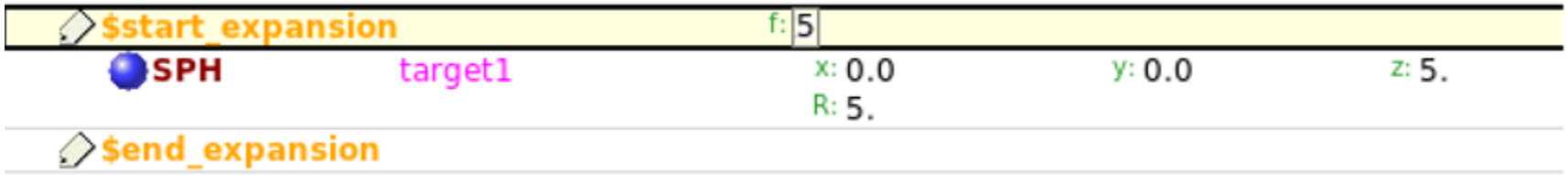
the expansion/reduction matrix is $T = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

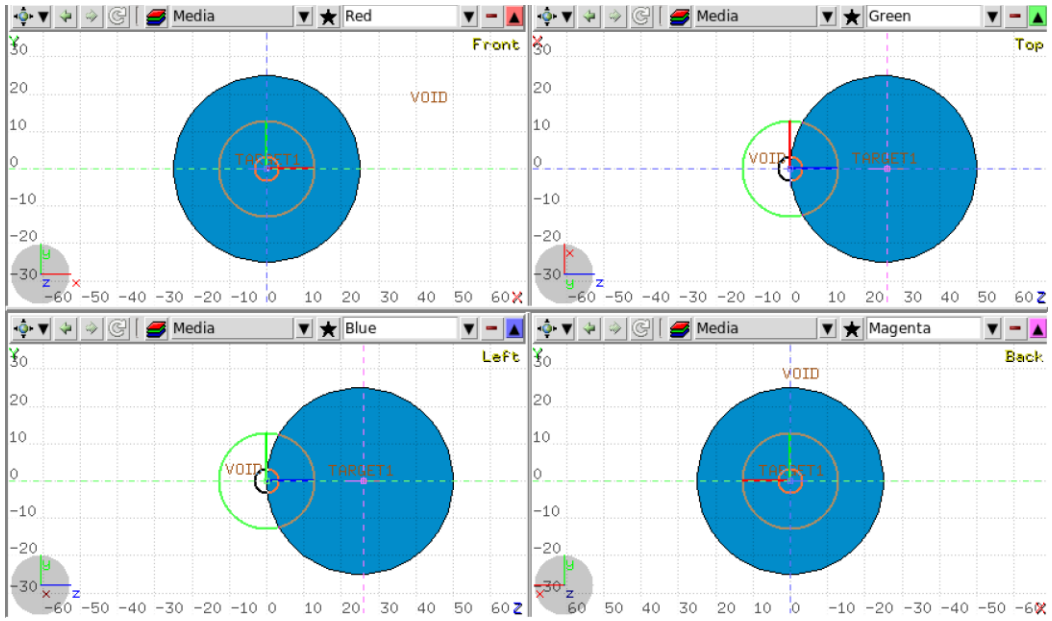
and the transformed body equation is $\mathbf{r}^T (T^{-1})^T M_{QUA}\, T^{-1}\,\mathbf{r} = 0$

Example: Expansion

- Apply an expansion factor equal to 5 to the sphere. See the effect on the geometry.

```
$start_expansion 5
SPH target1      0.0      0.0      5.      5.
$end_expansion
```

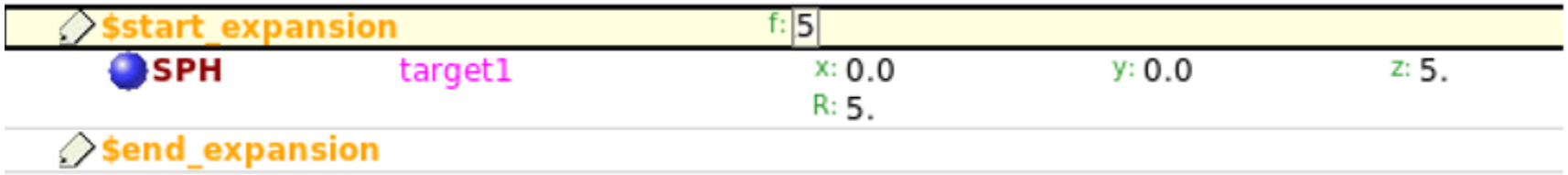


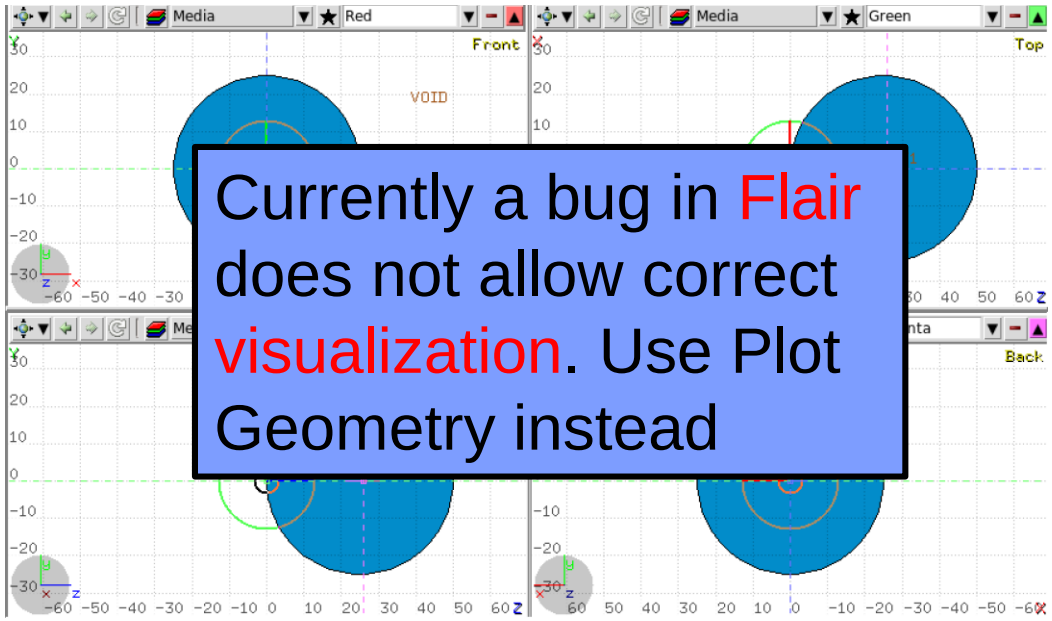


Example: Expansion

- Apply an expansion factor equal to 5 to the sphere. See the effect on the geometry.

```
$start_expansion 5
SPH target1      0.0      0.0      5.      5.
$end_expansion
```





Directives in geometry: transform

$Start_transform ... $End_transform

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

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

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

Note that also the **inverse** transformation can be used, T^{-1}, using the sign minus in front of the name of the transformation, defined in the ROT-DEFI

Directives in geometry: ROT-DEFIni

The ROT-DEFIni card defines roto-translations that can be applied to i. bodies, to ii. USRBIN & EVENTBIN and iii. LATTICE.

WHAT(1): assigns a transformation index and the corresponding rotation axis

I + J * 100 or **I * 1000 + J**

I = index of rotation

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

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

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

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

SDUM: Optional (but recommended) name for the transformation

You can always divide a transformation into many ROT-DEFI cards for easier manipulation of complex roto-translations, by using different names/numbers in the SDUM ("ROT#", "Rot#", "rot#", "RO#", "Ro#", "ro#")

Example: roto-translation

- Change the previous sphere target1 into a rectangular parallelepiped (RPP) with x_{min}, y_{min}, z_{min}=-5 cm and x_{max}, y_{max}, z_{max}=5 cm

RPP target1 -5.0 5.0 -5.0 5.0 -5.0 5.0

target
RPP target1

Xmin: -5.0 Xmax: 5.0
Ymin: -5.0 Ymax: 5.0
Zmin: -5.0 Zmax: 5.0

- Apply a coordinate roto-translation to the body, using a transform directive + the ROT-DEFI card:

$start_transform Trans: rot1

target
RPP target1

Xmin: -5.0 Xmax: 5.0
Ymin: -5.0 Ymax: 5.0
Zmin: -5.0 Zmax: 5.0

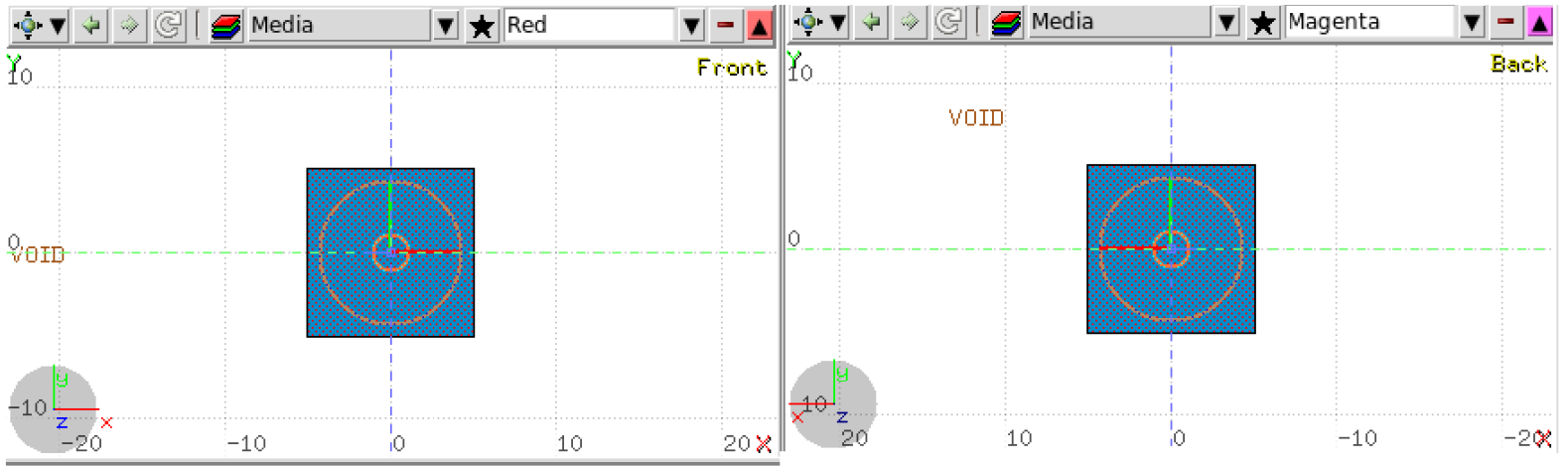
$end_transform

ROT-DEFI Axis: X Id: 0 Name: rot1
Polar: 110 Azm:
Δx: Δy: Δz:

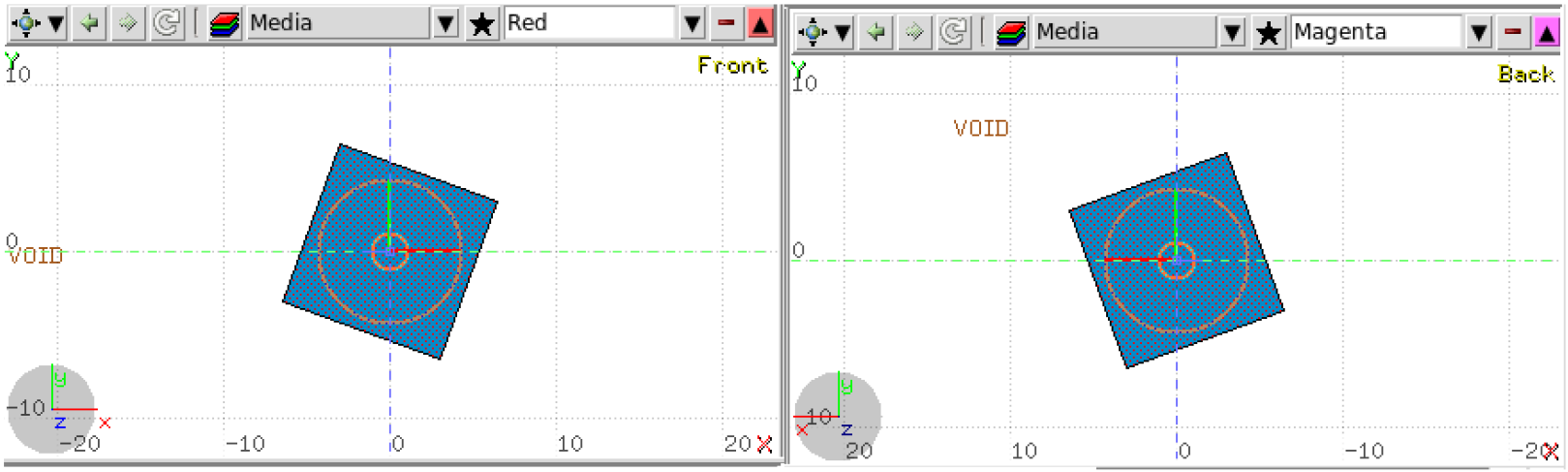
IMPORTANT: ROT_DEFI outside the geometry definition!!

Example: roto-translation

Before roto-translation



After roto-traslation



Directives in geometry: roto-translation

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.

$$\mathbf{X_{new} = M_{polar} \times M_{az} \times (X + T)}$$

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 inizialization)

  $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

  $Start_translat –

  $Start_expansion

  .....

  $End_expansion

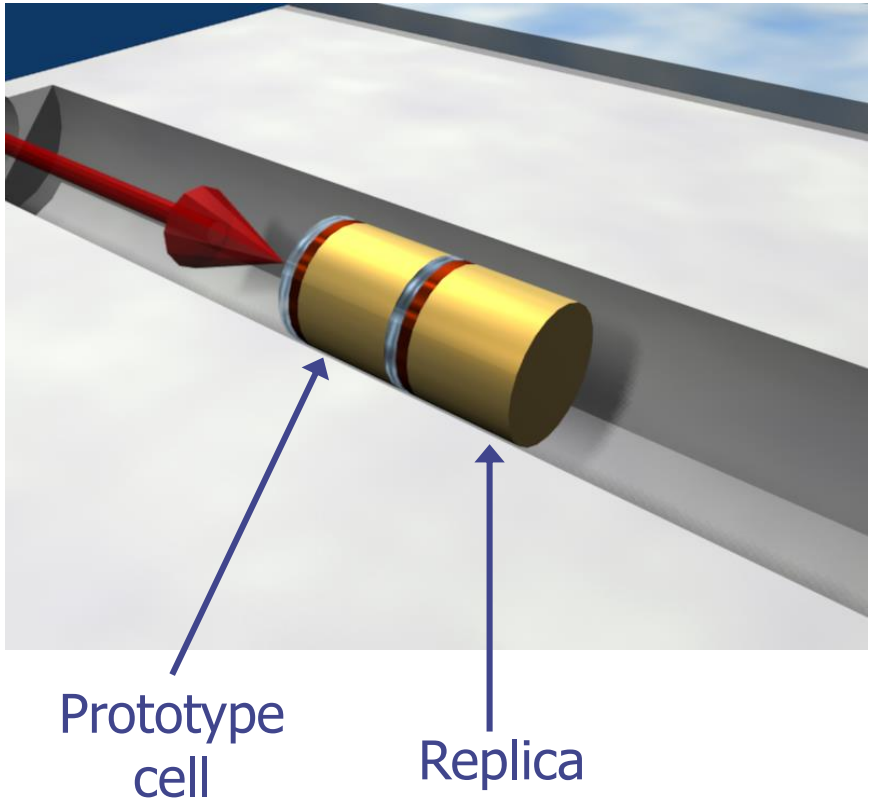
  $End_translat

  $End_transform

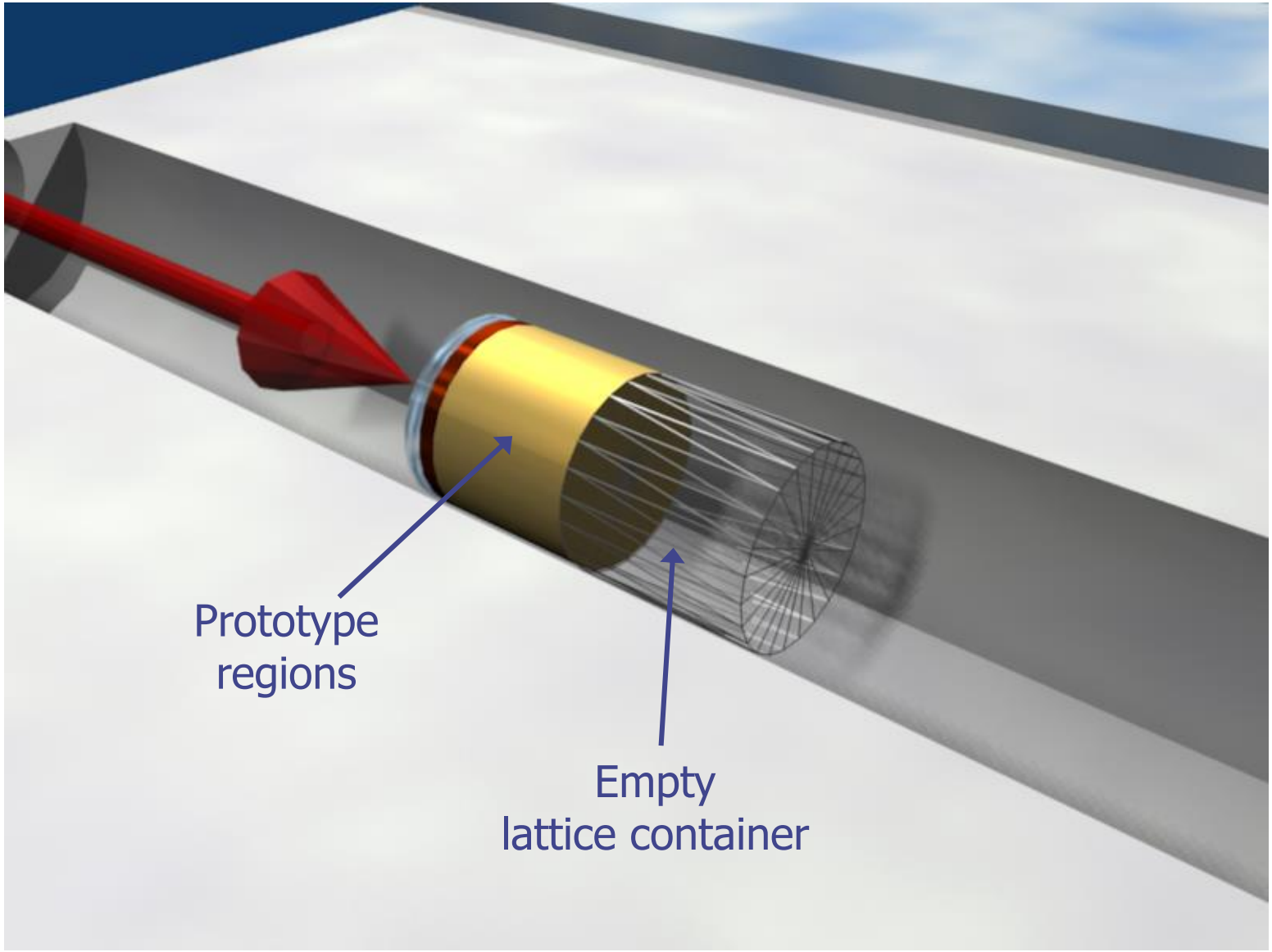
- Directives are not case sensitive (whereas roto-translation names are)

Lattice

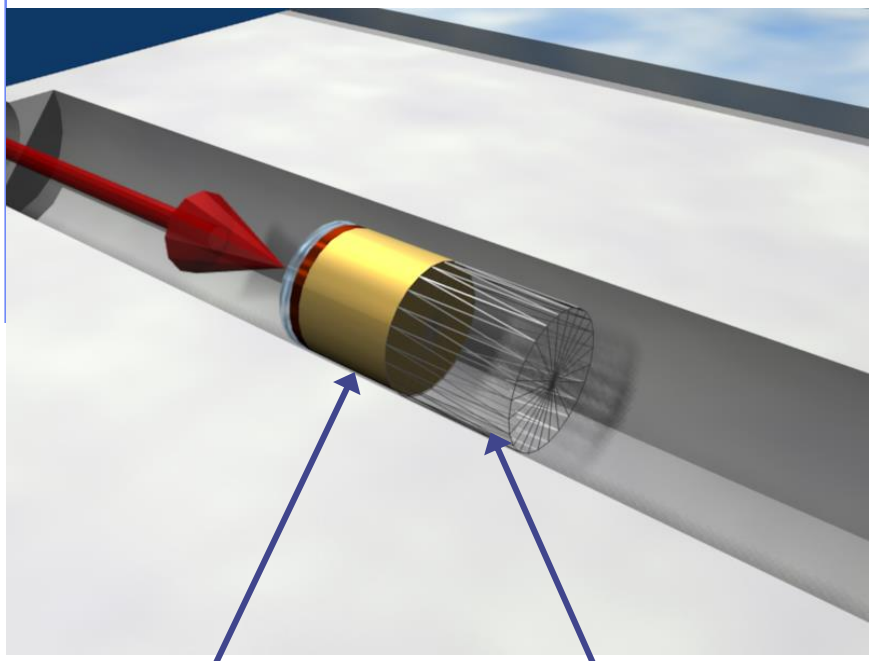
FLUKA geometry has *replication* (lattice) capabilities

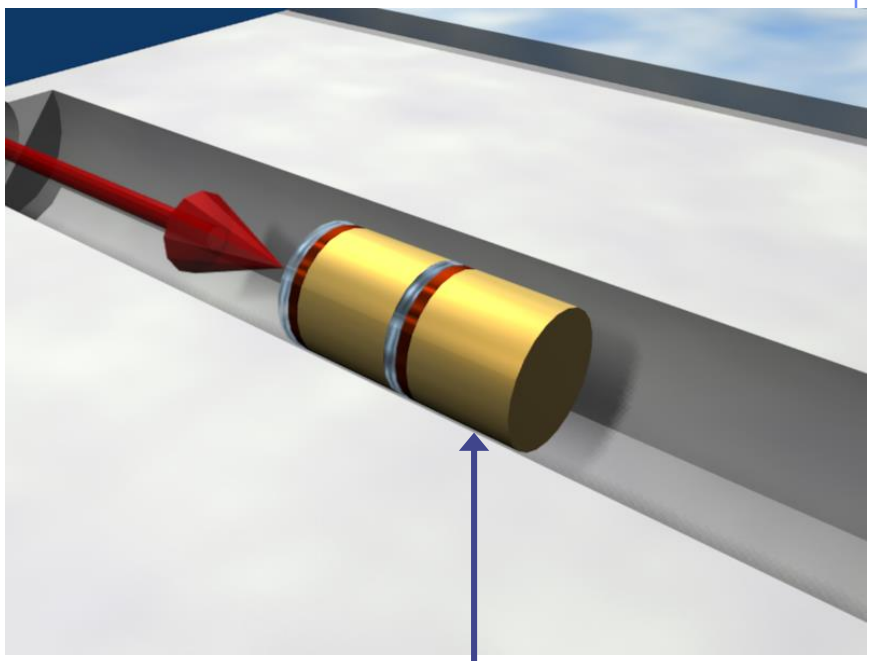


Example



Example



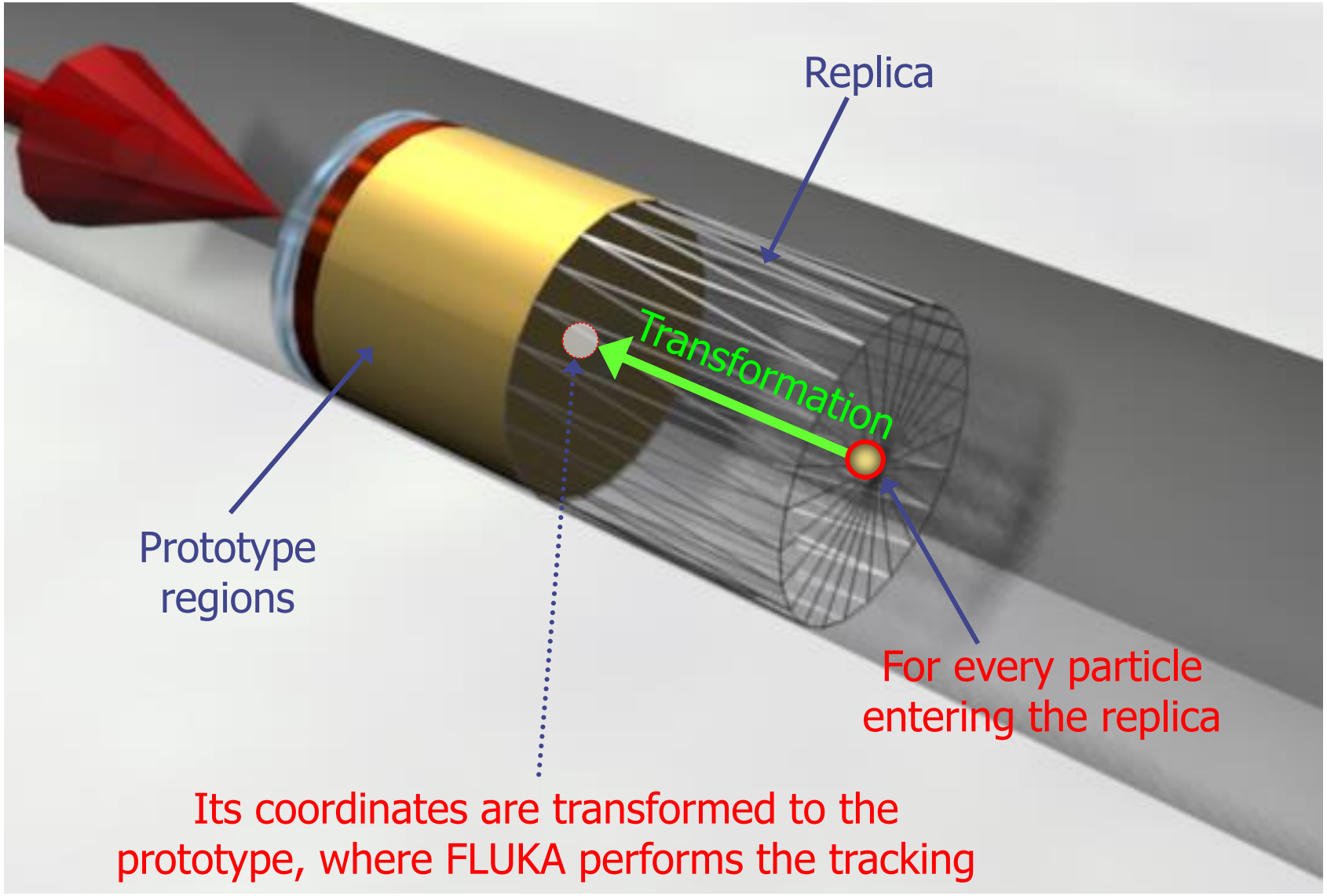


Prototype cell

Empty lattice cell

Final replica

Example

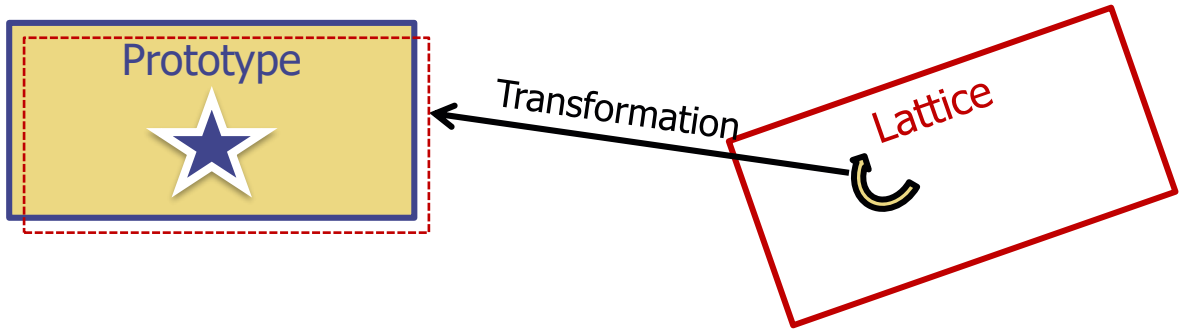

Replica
Transformation
Prototype
regions
For every particle
entering the replica
Its coordinates are transformed to the
prototype, where FLUKA performs the tracking

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

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

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 (lattic.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: 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 is defined by name

- Always remember that the transformation must bring the container onto the prototype and not viceversa!

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)

WARNING:

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

Tips & Tricks

- 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.

LATTICE - Example:

- Open the input file example_geometry2_rotdefi.inp
- Look at the:

BODIES:

Definition of the RPP target2 (*replica*) and RPP Box (*prototype*)

REGIONS:

Definition of the REGIONS TARGET2 and BOX

LATTICE:

Application of the Transformation transl1, defined by ROT_DEFI, to the TARGET2 REGION

ROT-DEFI:

Definition of a Transformation called transl1 corresponding to a translation of -45 cm along Z, which brings the *replica* to the *prototype*

ASSIGNMA:

Assignment of a material to the REGION BOX (*prototype container*) (not to TARGET2, which is the *empty lattice container*)

Output file (1)

Interpreted body echo

```
Body n.       1  SPH  blkbody    Rot.     0
       0.000000          0.000000          0.000000          100000.0
Body n.       2  SPH  void       Rot.     0
       0.000000          0.000000          0.000000          10000.00
Body n.       3  RPP  target1    Rot.     1
      -5.000000          5.000000         -5.000000          5.000000         -5.000000          5.000000
Body n.       4  RPP  target2    Rot.     0
      -5.000000          5.000000         -5.000000          5.000000          20.00000          25.00000
Body n.       5  RPP  Parking    Rot.     0
      -600.0000          600.0000         -600.0000          600.0000          29000.00          31000.00
Body n.       6  RPP  Box        Rot.     0
      -5.000000          5.000000         -5.000000          5.000000          30020.00          30025.00
```

Output file (2)

```
 Reg.      1          2          3          4          5          6
Volume   1.000E+00  1.000E+00  1.000E+00  1.000E+00  1.000E+00  1.000E+00

 *** Geometry lattice capabilities activated ***


 *** Geometry lattice capabilities activated ***


*****NEXT CONTROL CARD *****   LATTICE    4.000      4.000      1.000      1.000      1.000      1.000
transl1
```

Proposed Exercises:

- Change the size of the RPP target2 from 5 cm to 10 cm along Z (Z_{max}=30 cm)

  Change the size of the RPP Box accordingly, for an exact match between prototype and replica

- Build a new replica of your prototype centered in Z=45 cm and rotated by 30 degrees with respect to the Z axis

- Build a new spherical prototype, of radius R=10 cm and made of Tin

  Define a replica of the spherical prototype centered in Z=70 cm

Suggestion: use GeoViewer to check for possible mistakes

Additional information

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

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

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 (lattic.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


```
*...+....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*

Transformation by input

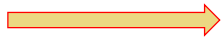
- Rotations/Translations can be defined with the ROT-DEFIni card
- Can be assigned to a lattice by name or with ROT#nnn SDUM in the LATTICE card
- ROT-DEFIni cards can be consecutive (using the same index or name) to define complex transformations

WARNING:

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

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 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
1st	integer	region number	MUSRBR
2nd	integer	lattice cell number	LUSRBL
3rd	float	zero	FUSRBV