FLUKA

Advanced geometry

Transformations and modular geometries

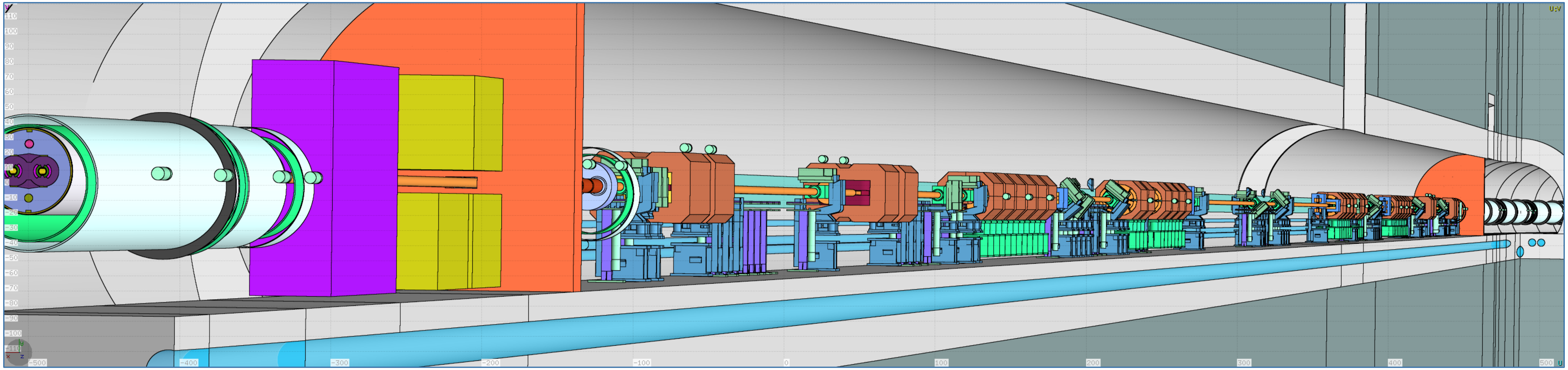
Basic geometry concepts

Three concepts are fundamental in the FLUKA Combinatorial Geometry, which have been described earlier in the course:

- **Bodies**: basic convex objects + infinite planes & cylinders + generic quadric
- **Zones**: portion of space defined by intersections **(+)** and subtractions **(-)** of bodies (used internally)
- **Regions**: union of multiple zones **(|)**
  (or a single zone)

Complex and modular geometries

3D rendering of LHC IR7



Complex and modular geometry models like the one shown here are built with LineBuilder [A. Mereghetti et al., IPAC2012, WEPPD071, 2687]

Such a geometry model heavily depends on **`LATTICEs`** (i.e. duplication of existing regions) which are not covered here

In this lecture

- Roto-translation transformations
  - `ROT-DEFIni` card

- Geometry directives
  - `translat`
  - `transform`
  - `expansion`

- Additional card related to a transformation
  - `ROTPRBIN` card

- Tips for building a modular geometry

The `ROT-DEFI` card

ROT-DEFI card – Introduction

The **ROT-DEFI** card defines roto-translations that can be applied to:

- Bodies:
  To move and rotate geometry
- **USRBIN** and **EVENTBIN** cards (see **ROTPRBIN** card later)
  To move and rotate scorings
- **LATTICE** (not covered here)

The **ROT-DEFI** card must be outside the geometry description, e.g. after **GEOEND**

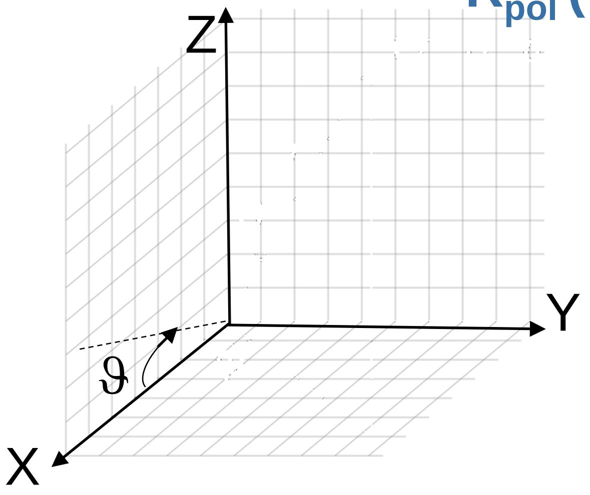
The roto-translation places the body (or USRBIN etc) in the **lab** frame of reference.

ROT-DEFI card – Definition

ROT-DEFI	Axis: Z ▼	Id: 0	Name:
	Polar:	Azm:	
	Δx:	Δy:	Δz:

Axis:	reference axis
Id:	transformation index. If set to 0, then Id is automatically assigned
Name:	transformation name. Optional, but recommended for easy referencing
Polar:	polar angle of the rotation $\mathbf{R}_{pol}$ ($0 \le \vartheta \le 180$ degrees) [clockwise]
Azm:	azimuthal angle of the rotation $\mathbf{R}_{azm}$ ($-180 \le \varphi \le 180$ degrees) [clockwise]
Δx, Δy, Δz:	vector components for the translation **T**

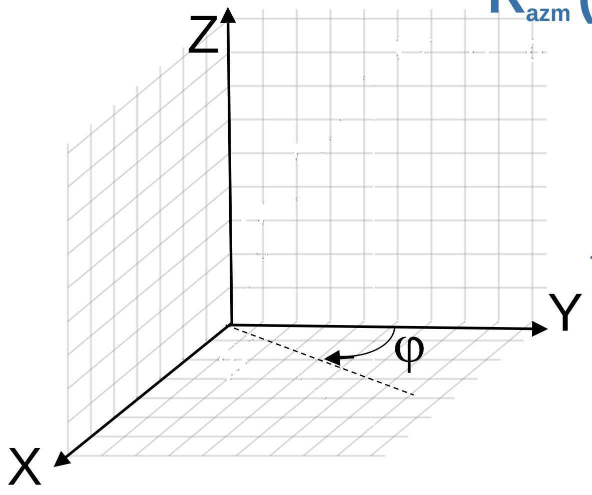
$\mathbf{R}_{pol}\ (\vartheta)$



When reference axis is Z:
Clockwise rotation around Y with angle ϑ

* More generally, with ref. axis X_0:
Clockwise rotation around X_2 with angle ϑ
"X_1 goes towards X_0"

$\mathbf{R}_{azm}\ (\varphi)$



When reference axis is Z:
Clockwise rotation around Z with angle φ

* More generally, with ref. axis X_0:
Clockwise rotation around X_0 with angle φ
"X_2 goes towards X_1"

** Let (X0, X1, X2) be a right-handed orthogonal system in a 3D space. For example: **(Z, X, Y)**, or (X, Y, Z), or (Y, Z, X).*

ROT-DEFI card – Definition

ROT-DEFI	Axis: Z ▾	Id: 0	Name:
	Polar: ϑ value	Azm: φ value	
	Δx: X_{offset} value	Δy: Y_{offset} value	Δz: Z_{offset} value

The ROT-DEFI card roto-translation is defined as:

$$\mathbf{R}_{\mathbf{pol}}(\vartheta) \circ \mathbf{R}_{\mathbf{azm}}(\varphi) \circ \mathbf{T}$$

3. 2. 1.

Composition order matters!
First T, then R_{azm}, then R_{pol}

For example, for a ROT-DEFI card with **Axis = Z**, the roto-translation is:

$$\begin{vmatrix} X_{\text{new}} \\ Y_{\text{new}} \\ Z_{\text{new}} \end{vmatrix} = \begin{vmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{vmatrix} \begin{vmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} X_{\text{old}} + X_{\text{offset}} \\ Y_{\text{old}} + Y_{\text{offset}} \\ Z_{\text{old}} + Z_{\text{offset}} \end{vmatrix}$$

rotation around Y axis with clockwise angle ϑ

rotation around Z axis with clockwise angle φ

See ROT-DEFI in manual!

It is preferable to define rotations through the azimuthal angle.

ROT-DEFI cards – “Chaining” / Inverse

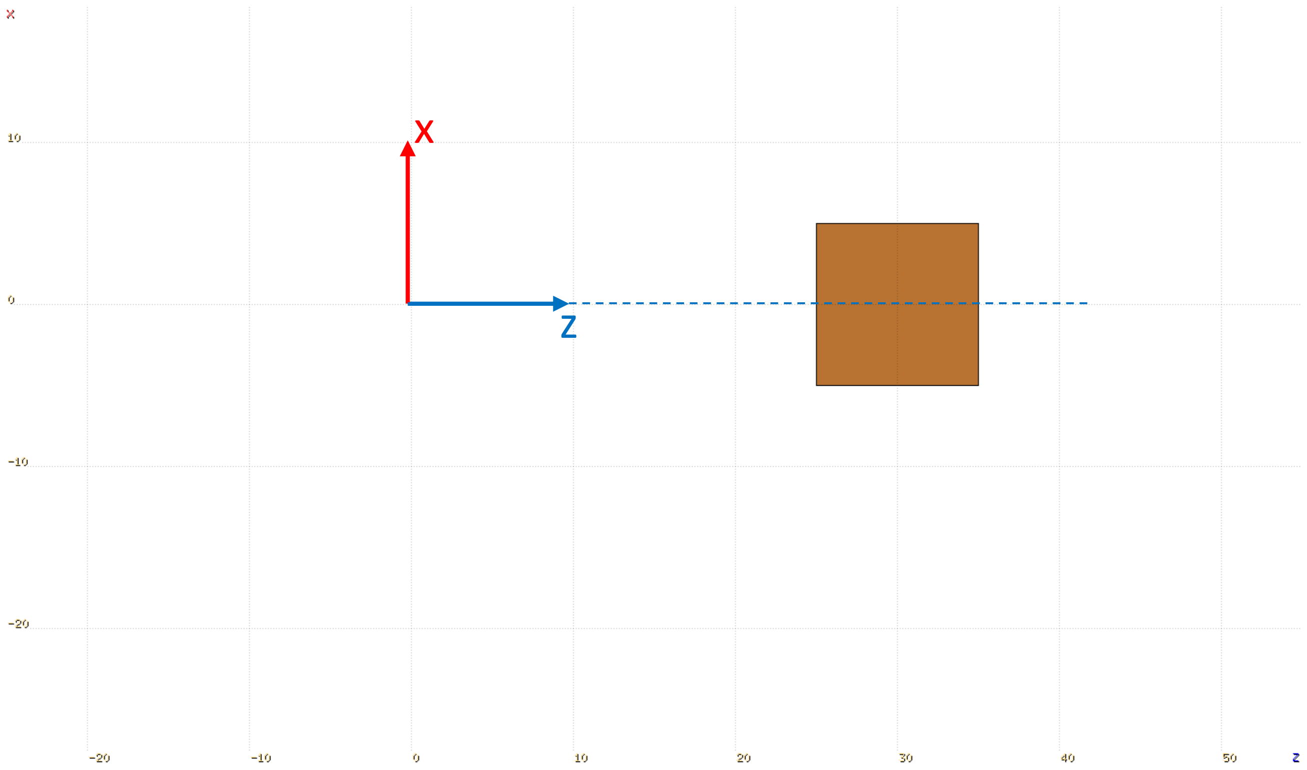
- It is possible to use multiple **ROT-DEFI** cards to define a single transformation (**compositon, or "chaining"**):
  - The Name (or Id) on the “chained” **ROT-DEFI** cards has to be the same.
  - The transformations associated with the **ROT-DEFI** cards are applied from top to bottom.

- It is also possible to access the **inverse** of the transformation associated with a **ROT-DEFI** card.
  - Just refer to the existing **ROT-DEFI** card with a minus sign (“**-**”) before its name or Id number.
  - Example use with **ROTPRBIN** card later in the lecture.

ROT-DEFI card – Example 1

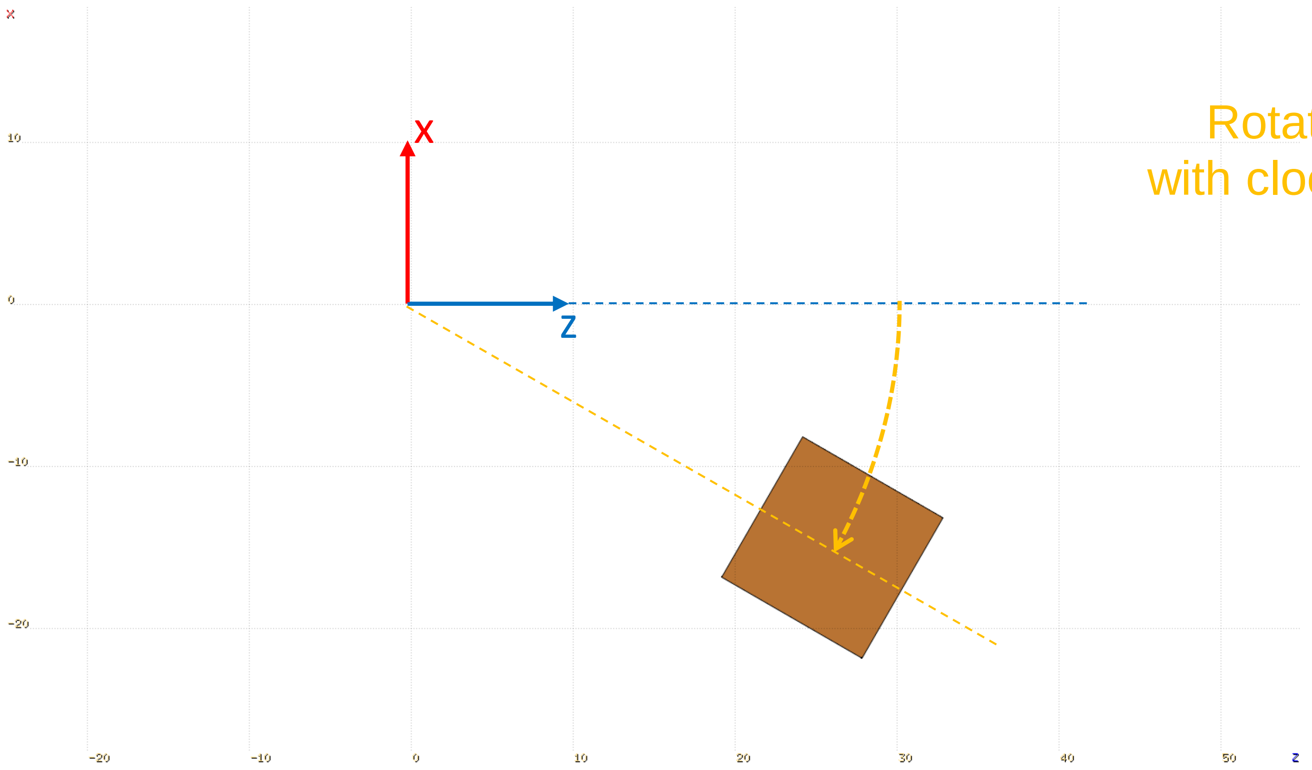
Body located away from the origin of the coordinate system.

Initial state



ROT–DEFI card – Example 1

ROT-DEFI	Axis: Y ▾	Id: 0	Name: Rot
	Polar:	Azm: 30	
	Δx:	Δy:	Δz:

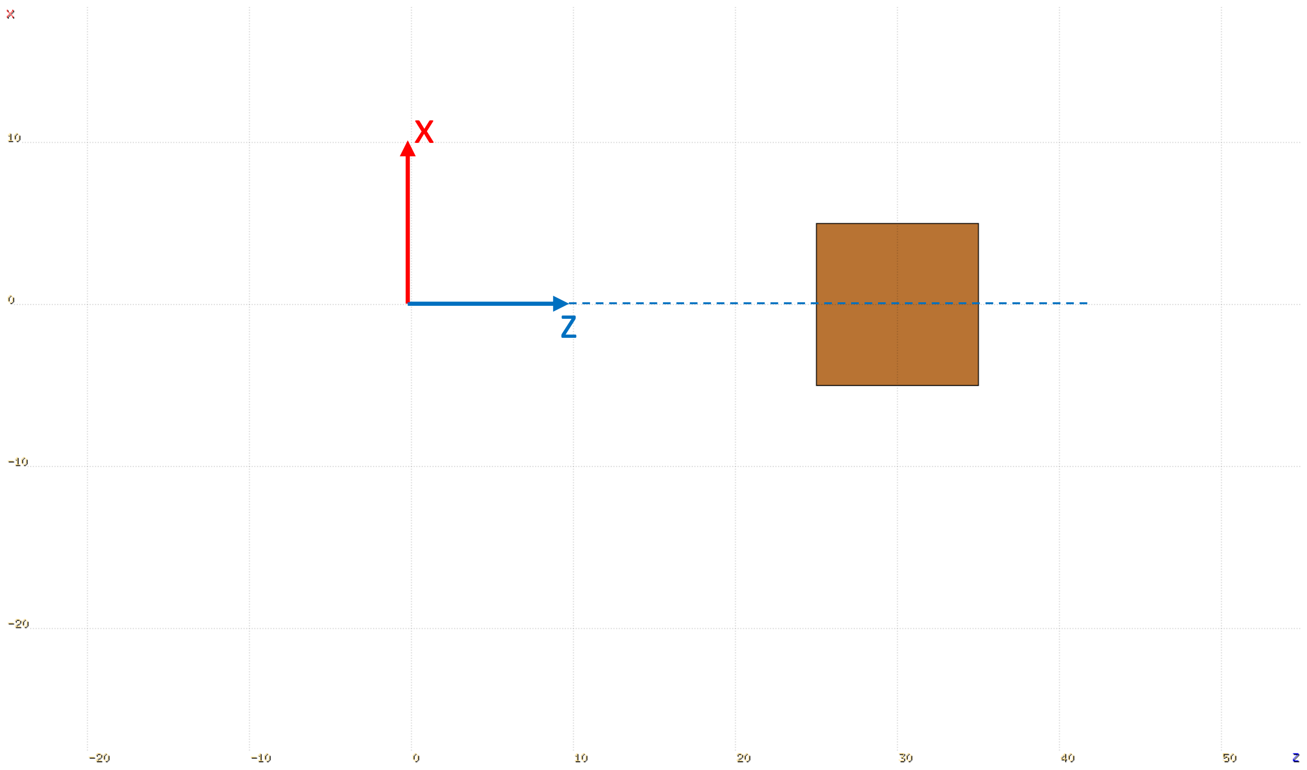


Rotation around Y axis
with clockwise angle $\varphi = 30°$

ROT-DEFI card – Example 2

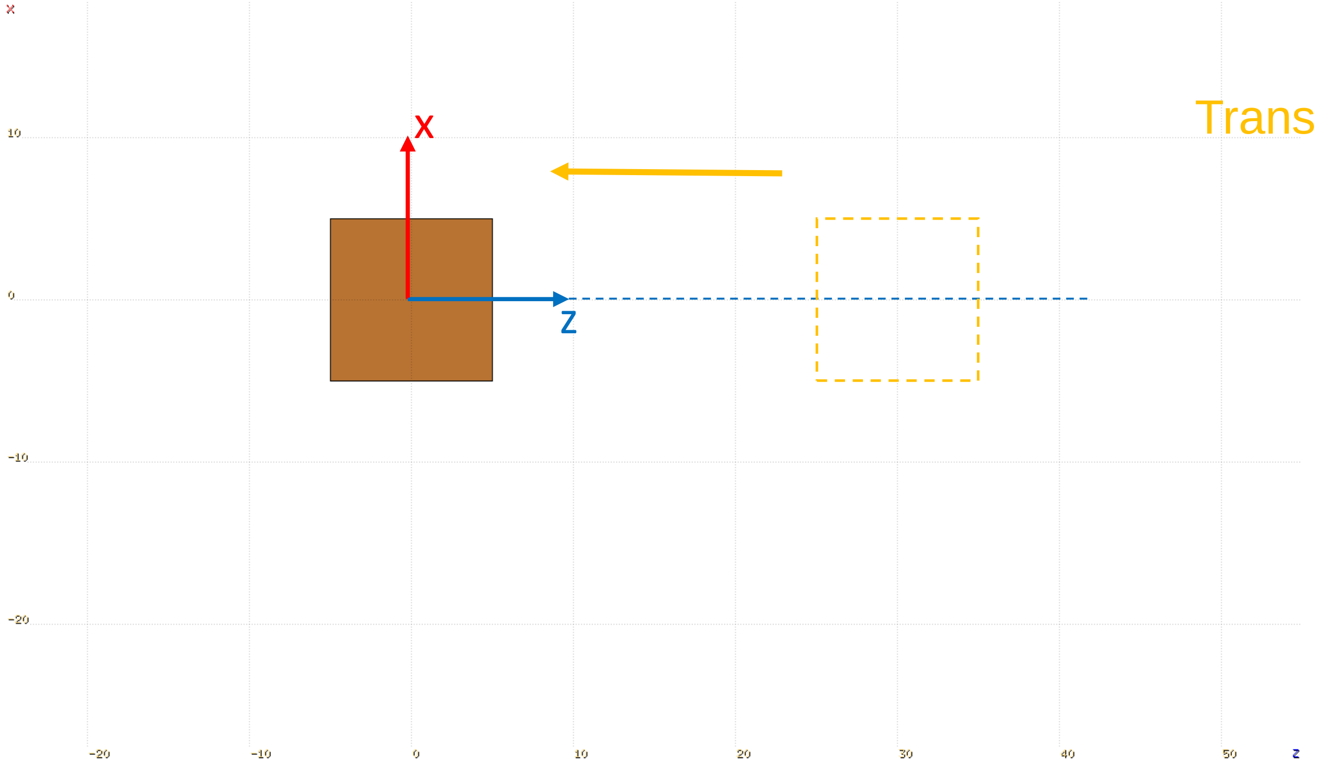
Body located away from the origin of the coordinate system.

Initial state



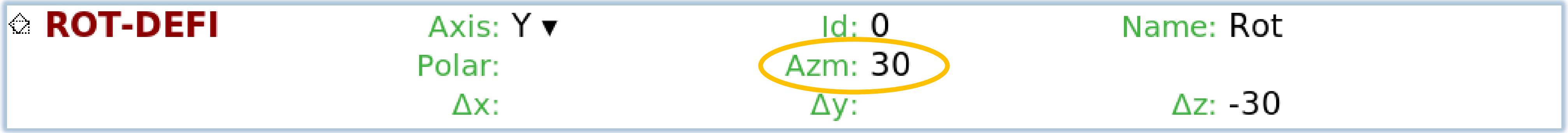
ROT−DEFI card – Example 2

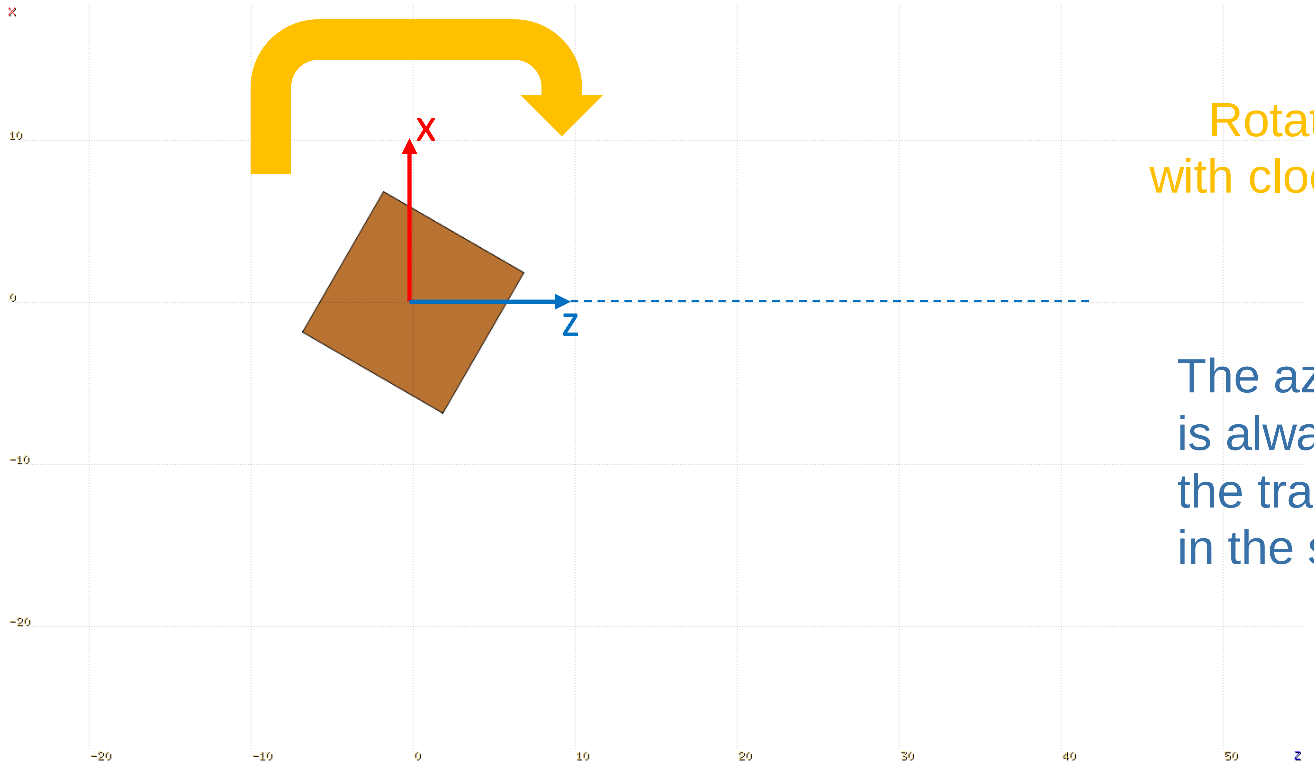
ROT-DEFI	Axis: Y ▾	Id: 0	Name: Rot
	Polar:	Azm: 30	
	Δx:	Δy:	Δz: -30



Translation along Z axis

ROT–DEFI card – Example 2



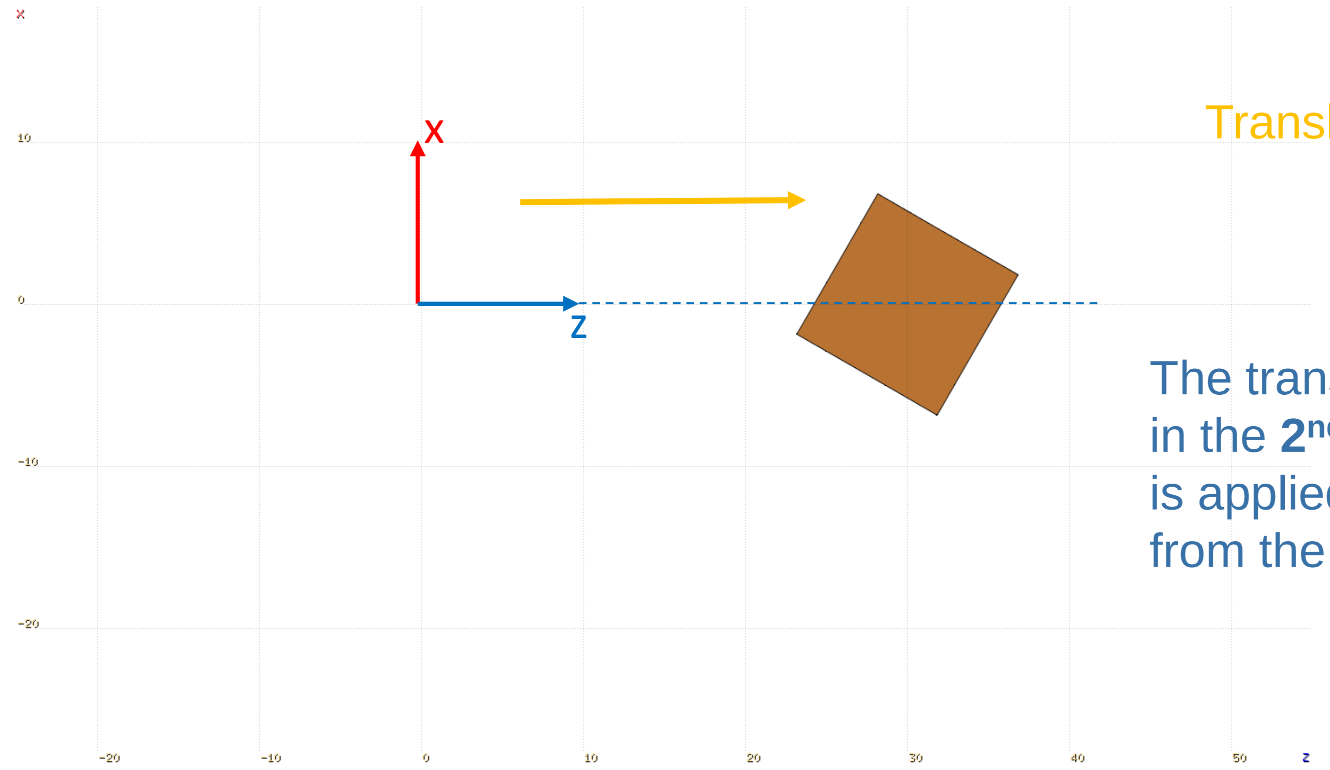


Rotation around Y axis
with clockwise angle $\varphi = 30°$

The azimuthal rotation
is always performed **after**
the translation defined
in the same ROT-DEFI card.

ROT−DEFI card – Example 2

1. **ROT-DEFI** Axis: Y ▾ Id: 0 Name: Rot
   Polar: Azm: 30
   Δx: Δy: Δz: -30
2. **ROT-DEFI** Axis: Y ▾ Id: 0 Name: Rot
   Polar: Azm:
   Δx: Δy: Δz: 30



Translation along Z axis

The transformation defined in the 2^{nd} **ROT-DEFI card** is applied **after** the roto-translation from the 1^{st} ROT-DEFI card.

Geometry directives

Geometry directives

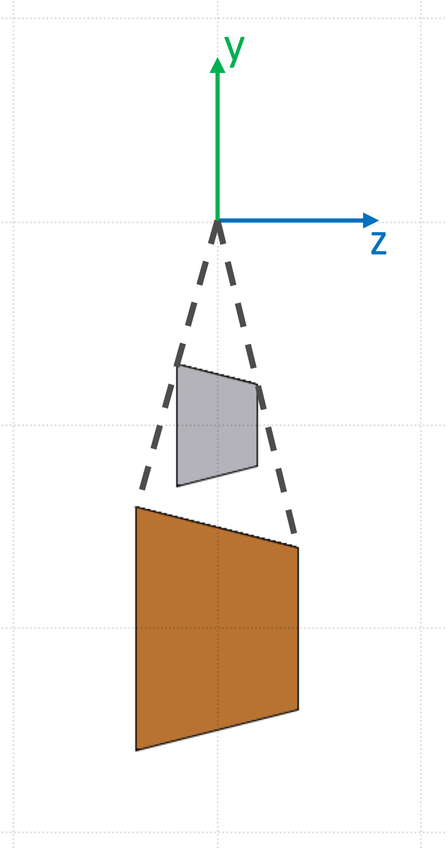
- Special commands enclosing a body (or a list of bodies) definition:

```
$start_xxx
...
$end_xxx
```

- Where "**xxx**" stands for "**translat**", "**transform**" or "**expansion**"
- The directive is applied to the list of the bodies embedded between the starting and the ending directive lines

Directives in geometry: expansion

```
$start_expansion
...
$end_expansion
```

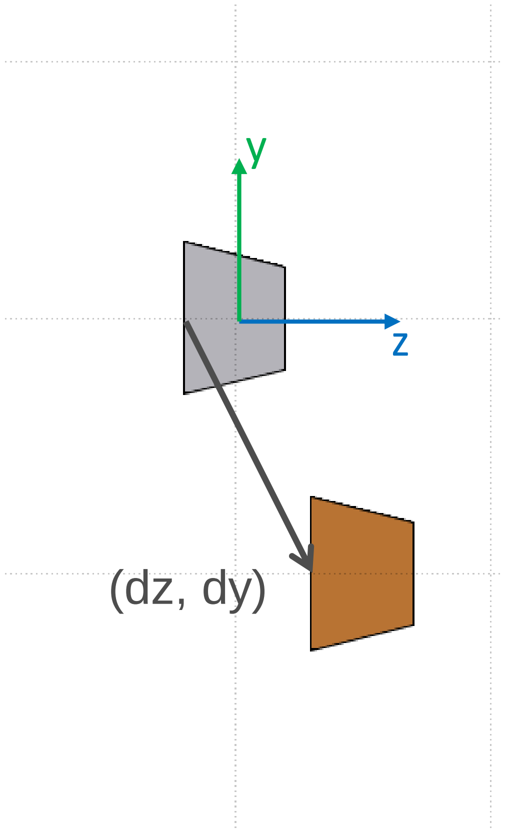


provides an expansion (or reduction)
of all body components (dimensions and placement)
by a defined scaling factor (f),
for all bodies included in the directive

```
$start_expansion     f: 2
    TRC  target       x: 0.0           y: -10.0         z: -2.0
                     Hx: 0.0          Hy: 0.0          Hz: 4.0
                  Rbase: 3.0      Rappex: 2.0
$end_expansion
```

Directives in geometry: translation

```
$start_translat
...
$end_translat
```



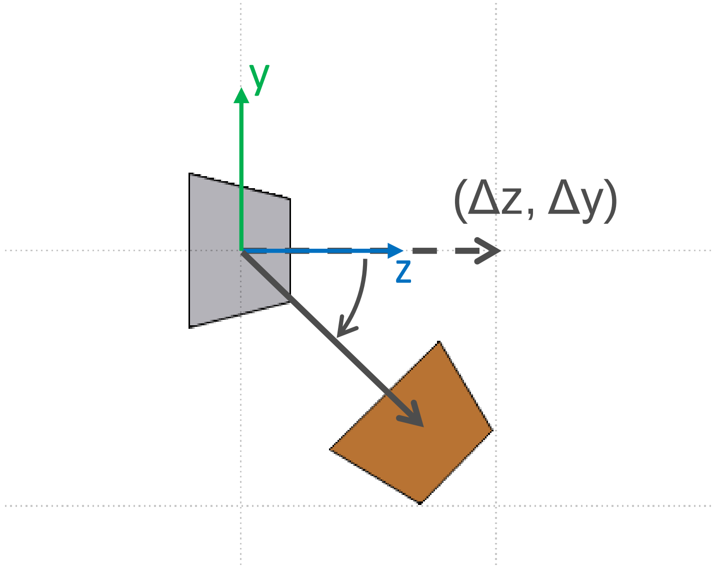
provides a coordinate translation (dx, dy, dz)
for all bodies embedded within the directive

$start_translat		dx:	0.0	dy:	-10.0	dz:	5.0
	TRC target	x:	0.0	y:	0.0	z:	-2.0
		Hx:	0.0	Hy:	0.0	Hz:	4.0
		Rbase:	3.0	Rappex:	2.0		
$end_translat							

Directives in geometry: transform

```
$start_transform
...
$end_transform
```

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



$start_transform	Trans: Rot ▾		
TRC target	x: 0.0	y: 0.0	z: -2.0
	Hx: 0.0	Hy: 0.0	Hz: 4.0
	Rbase: 3.0	Rappex: 2.0	
$end_transform			

ROT-DEFI	Axis: X ▾	Id: 0	Name: Rot
	Polar:	Azm: -45	
	Δx:	Δy:	Δz: 10

Directives in geometry: warnings

- `$start_expansion` and `$start_translat` are applied at intialisation
  → no CPU penalty

  `$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
```

The `ROTPRBIN` card

The ROTPRBIN card

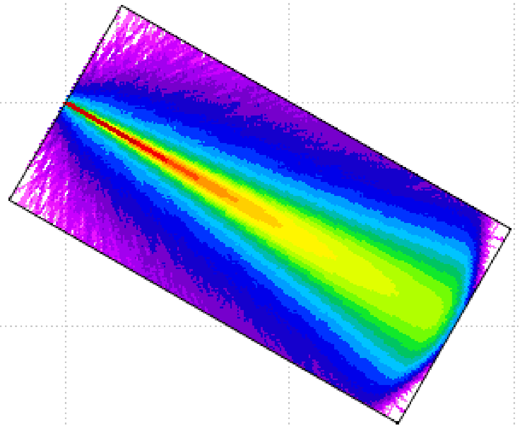
- Consider the following problem:
  - Pencil beam impinging on a cylindrical target
  - Using the R-Φ-Z USRBIN scoring, for symmetry
  - The beam and the target are rotated by 30 degrees around the **y** axis

- Solution: **ROTPRBIN** card
  - Allows to apply a roto-translation transformation (**ROT-DEFIni** cards) to **USRBIN** or **EVENTBIN** scorings

  - **Important:** In the **ROTPRBIN** card, the **transformation which is specified is NOT the usual placement of the mesh in the lab frame of reference** (i.e., the transformation: lab frame of reference → mesh frame of reference), but its **inverse**.

The `ROTPRBIN` card

- Example: **Both** the "target" solid and the "Fluence" mesh are rotated with "Rot":



ROT-DEFI	Axis: Y ▾	Id: 0	Name: Rot
	Polar:	Azm: 30	
	Δx:	Δy:	Δz:

$start_transform	Trans: Rot ▾		
RCC target	x: 0.0	y: 0.0	z: 0.0
	Hx: 0.0	Hy: 0	Hz: 2.0
	R: 0.5		
$end_transform			

Solid placement:
Call "Rot"

USRBIN		Unit: 21 BIN ▾	Name: Fluence
Type: R-Φ-Z ▾	Rmin: 0.0	Rmax: 0.5	NR: 50
Part: PROTON ▾	X: 0.0	Y: 0.0	NΦ: 1
	Zmin: 0.0	Zmax: 2.0	NZ: 200
ROTPRBIN	Type: ▾	Storage:	# Events:
	Rot: -Rot ▾	Rot2: ▾	
	Bin: Fluence ▾	to Bin: ▾	Step:

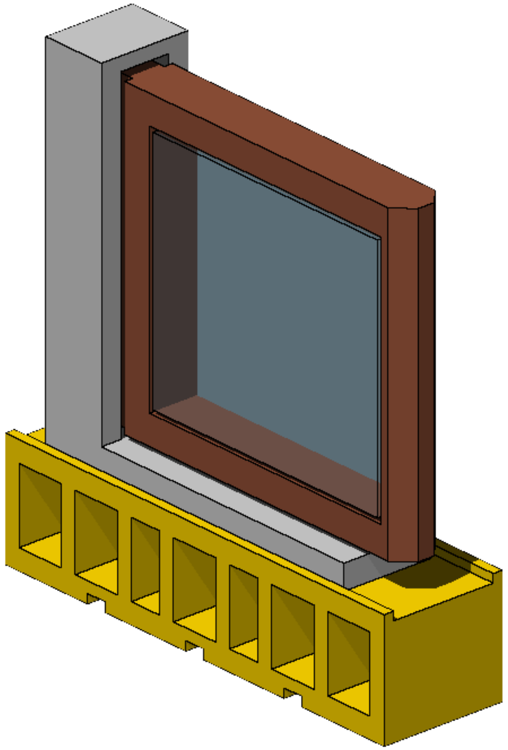
Mesh placement:
Call "- Rot"

Building modular geometries

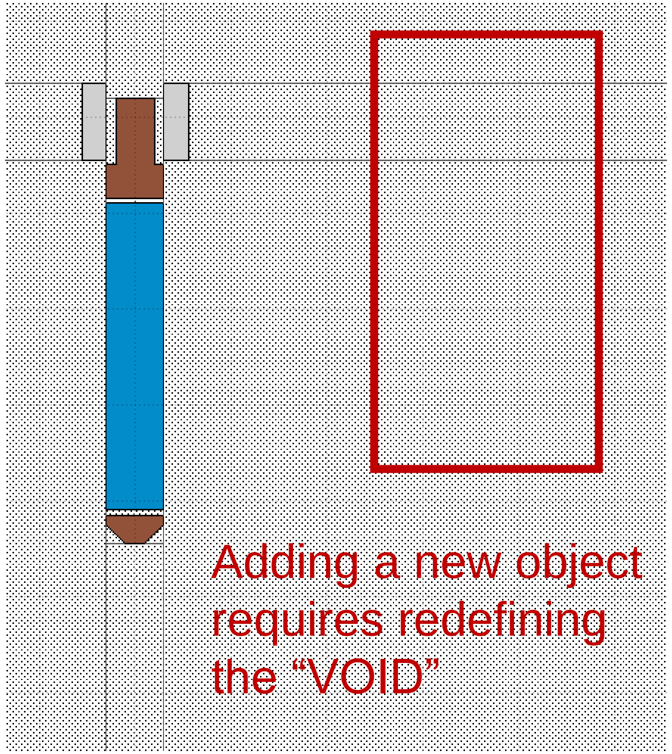
Bounding box

In the geometry lectures we saw that defining the “VOID” around objects can be quite difficult

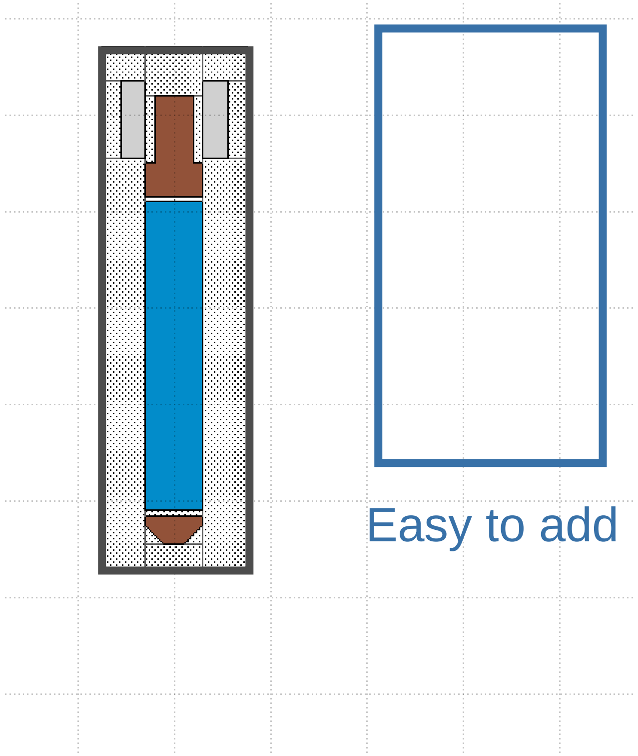
Complex object



Complex “VOID”

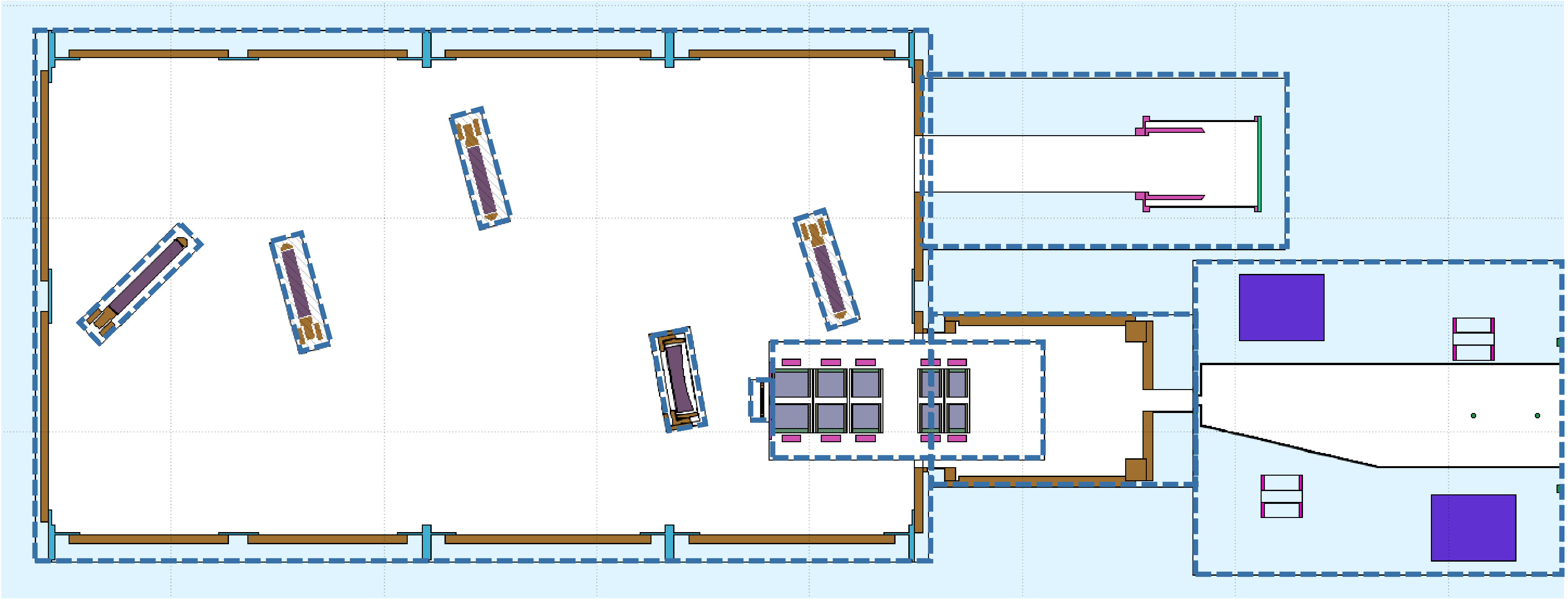


Solution: the Bounding Box



Good practice: use a finite body (**RPP**, **RCC**, etc.) as a **container** for the whole object

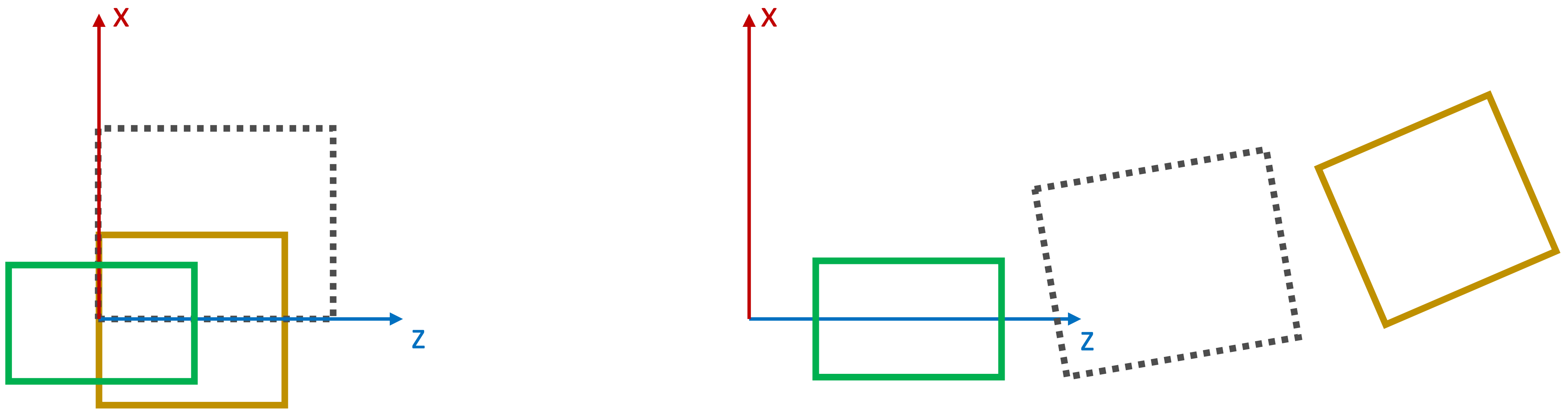
Bounding box



Only the Bounding Boxes have to be subtracted from the surrounding regions

Object location

- It is always easier to build an object around the origin:
  - It makes possible to use measurements from technical drawings directly
  - The final object can be translated / rotated into its final position with geometry directives



Naming conventions

- If multiple people are working on a complex geometry (multiple experimental halls and beamlines) it could happen that a body or region name is used twice, which leads to geometry errors

- Solution: agree on a naming convention, e.g. set prefixes for each object

- For example:
  - 1st character: Beamline
  - 2nd character: Object type
  - 3rd character: Object number
  - 4th-8th character: Free

Summary

- The **ROT-DEFI** card defines roto-translations

- Geometry directives (inside the geometry input) manipulate bodies
  - **$start_translat** **$end_translat**
    $start_transform **$end_transform**
    $start_expansion **$end_expansion**

- The **ROTPRBIN** card sets the correspondence between a roto-translation transformation and selected **USRBIN** and **EVENTBIN** scorings

- Tips on how to more easily build complex geometries

FLUKA