FLUKA advanced geometry

Selected topics to build a modular geometry

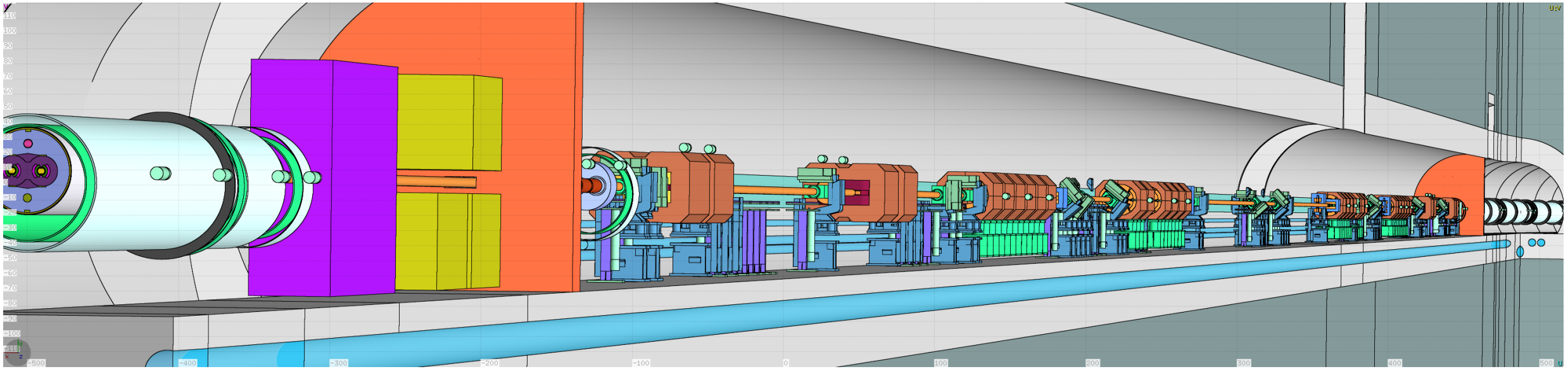
Basic Geometry Concepts

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 **(|)**
  (it can be also be a single zone)

Complex and modular geometry

3D rendering of the LHC IR7 with Collimators and BLM



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

Note that such geometry model heavily depends on **Lattices** (i.e., duplication of existing regions), which is not covered here

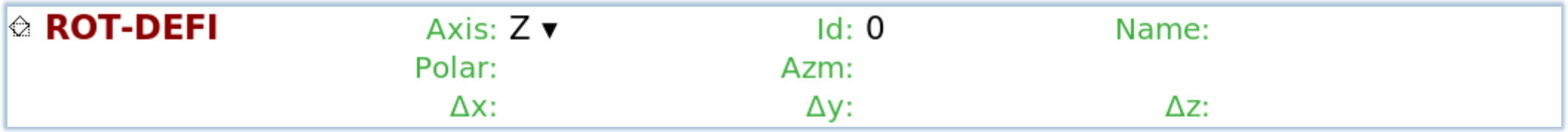
Contents

- Roto-translation transformations
  - `ROT-DEFIni` card
- Geometry directives
  - `expansion`
  - `translat`
  - `transform`
- Additional card related to a transformation
  - `ROTPRBIN` card
- Tips for building a modular geometry with bounding boxes

Contents

- Roto-translation transformations
  - **`ROT-DEFIni`** card
- Geometry directives
  - **`expansion`**
  - **`translat`**
  - **`transform`**
- Additional card related to a transformation
  - **`ROTPRBIN`** card
- Tips for building a modular geometry with bounding boxes

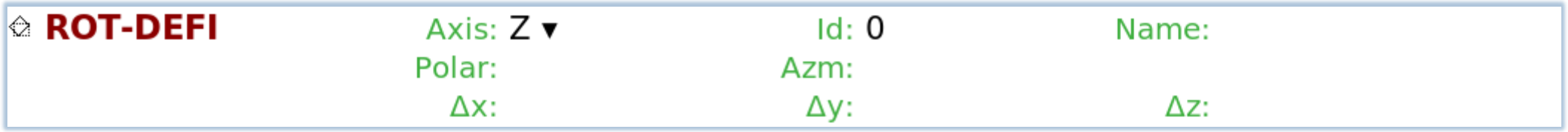
ROT-DEFIni card – Introducion



The **ROT-DEFIni** 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)

ROT-DEFIni card – Definition



Axis: rotation with respect to axis

Id: transformation index

If it is set 0, then Id is automatically assigned

Name: transformation name

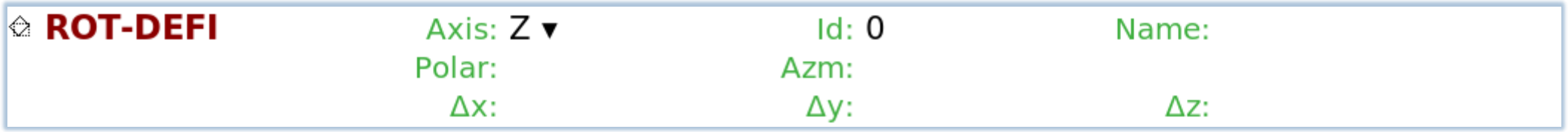
Optional but recommended for easy referencing

Polar: polar angle of the rotation $\mathbf{R_{pol}}$ ($0 \leq \vartheta \leq 180$ degrees)

Azm: azimuthal angle of the rotation $\mathbf{R_{azm}}$ ($-180 \leq \varphi \leq 180$ degrees) [clockwise]

Δx, Δy, Δz: offset for the translation **T**

ROT-DEFIni card – Definition



In a ROT-DEFI, the transformation is defined as $X_{new} = \mathbf{R_{pol}}(\vartheta) \times \mathbf{R_{azm}}(\varphi) \times (X_{old} + \mathbf{T})$ (3. 2. 1.)

The order of translation / rotation is relevant. They are not commutative!

The rotations always done around the origin of the coordinate system

It is preferable to define rotations through the azimuthal angle

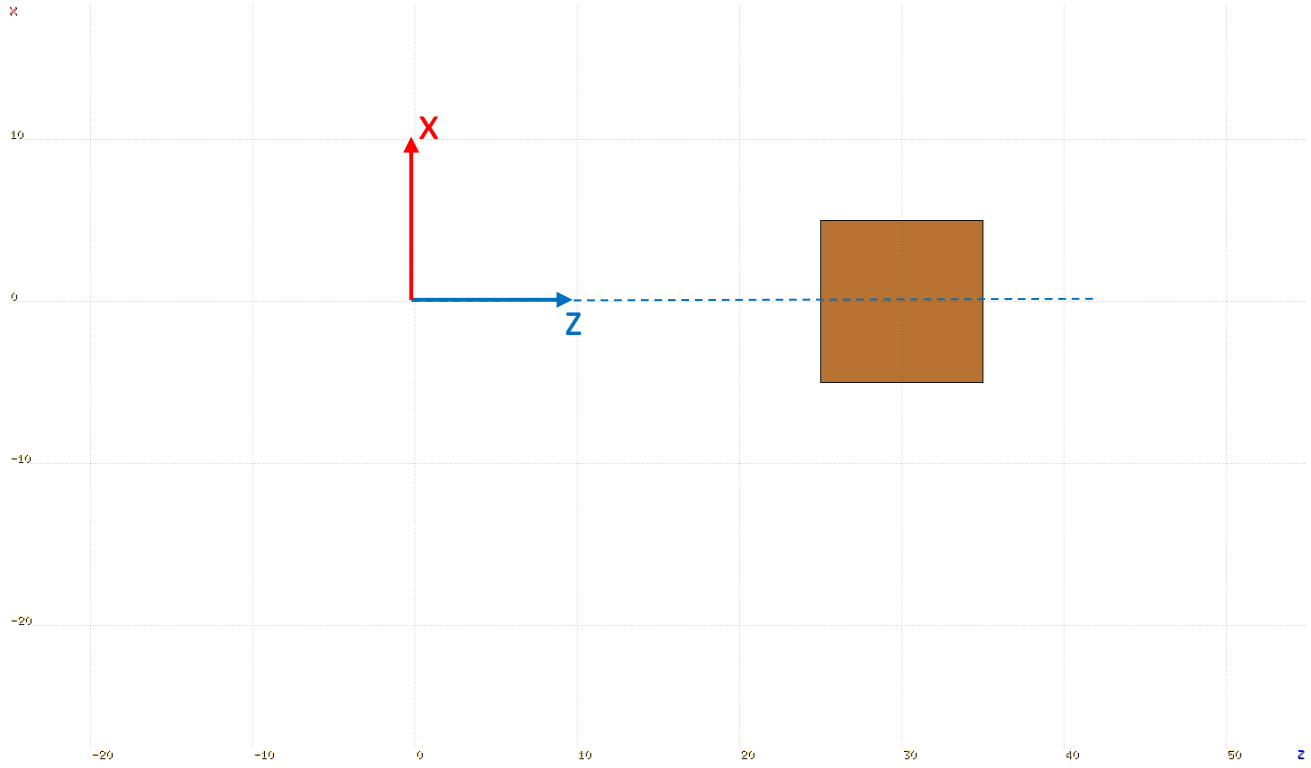
The convention used in the rotation matrices is available in the manual

See: Section 7 – **ROT-DEFIni** – Note 4

ROT-DEFIni card – Example

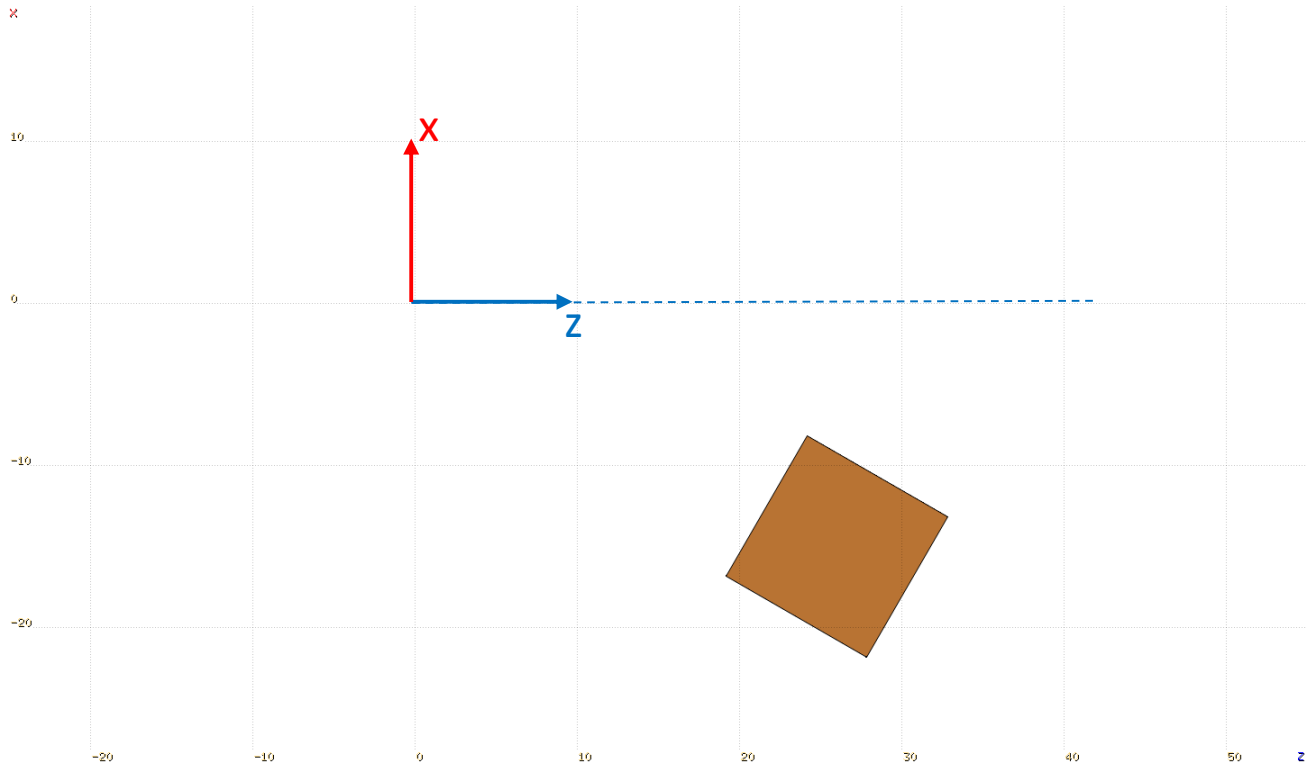
- Example:

  Rotating a body in place away from the origin of the coordinate system



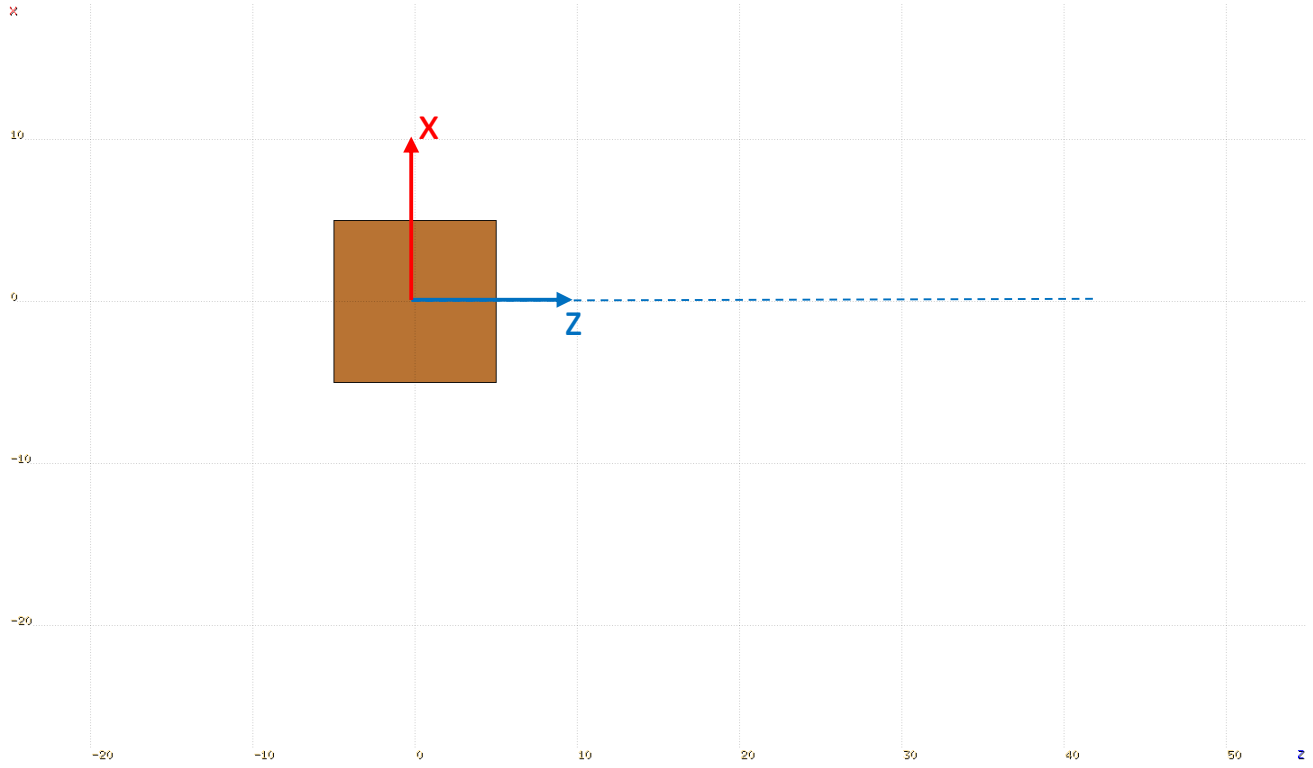
ROT-DEFIni card – Example

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



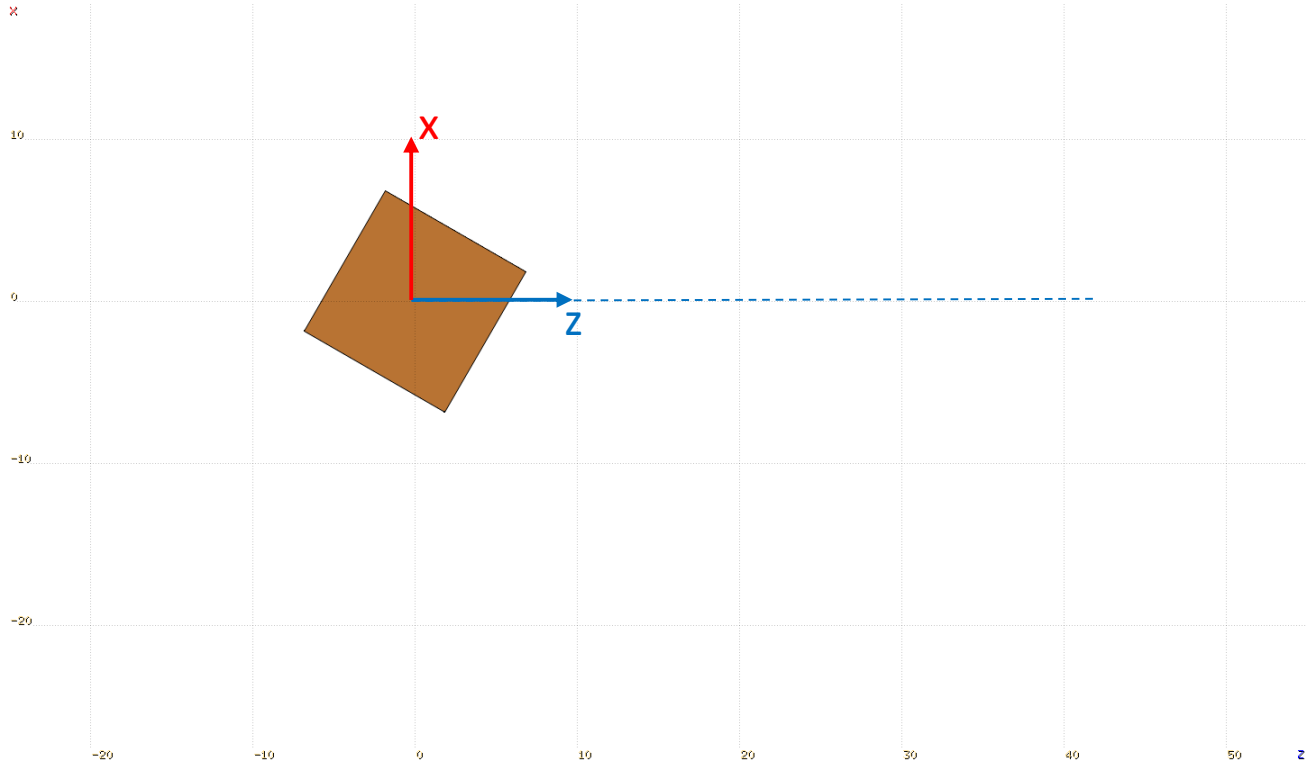
ROT-DEFIni card – Example

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



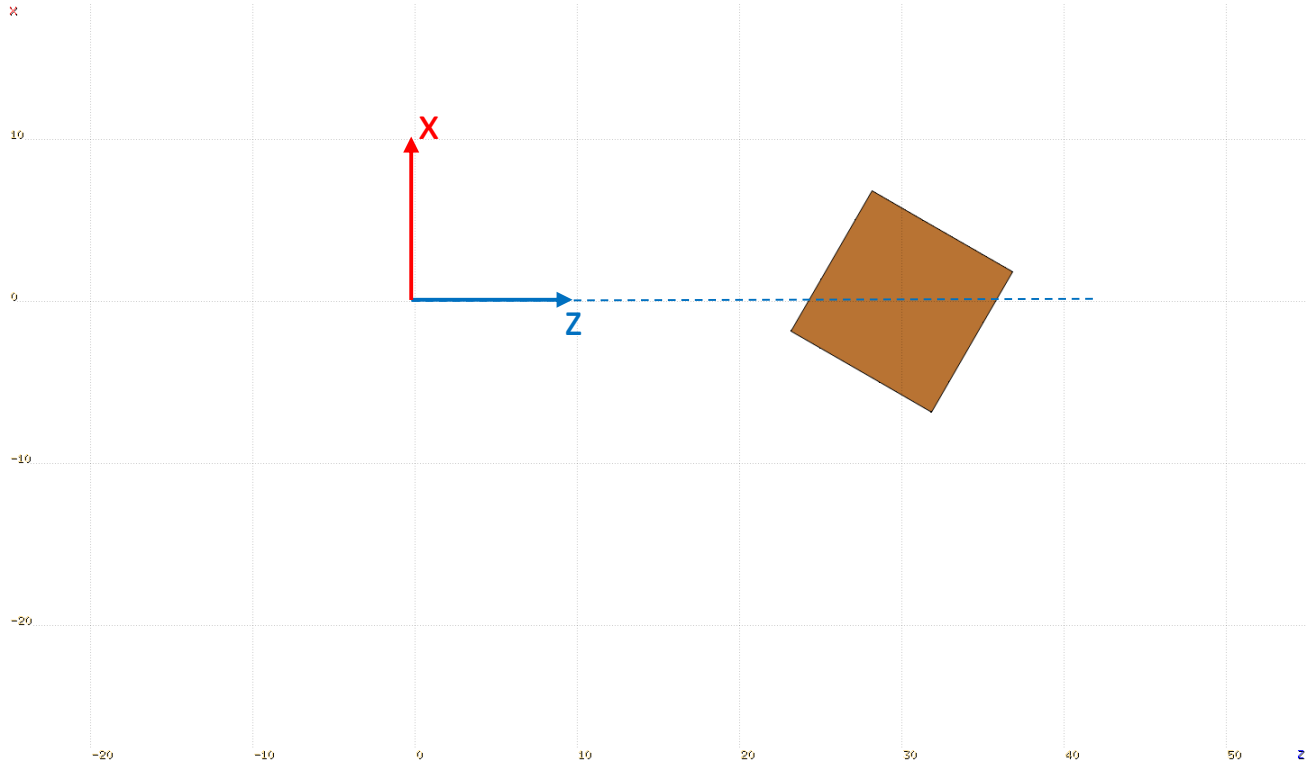
ROT-DEFIni card – Example

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



ROT-DEFIni card – Example

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



ROT-DEFIni card – “Chaining”

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

- It is possible to “chain” multiple **ROT-DEFIni** cards as a single transformation
  - The Name (or Id) on the “chained” **ROT-DEFIni** cards has to be the same
  - The **ROT-DEFIni** cards applied from top to bottom
- The inverse transformation is also accessible with a minus sign (“**-**”) before the name or Id number

Contents

- Roto-translation transformations
  - **`ROT-DEFIni`** card
- Geometry directives
  - **`expansion`**
  - **`translat`**
  - **`transform`**
- Additional card related to a transformation
  - **`ROTPRBIN`** card
- Tips for building a modular geometry with bounding boxes

Geometry directives

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

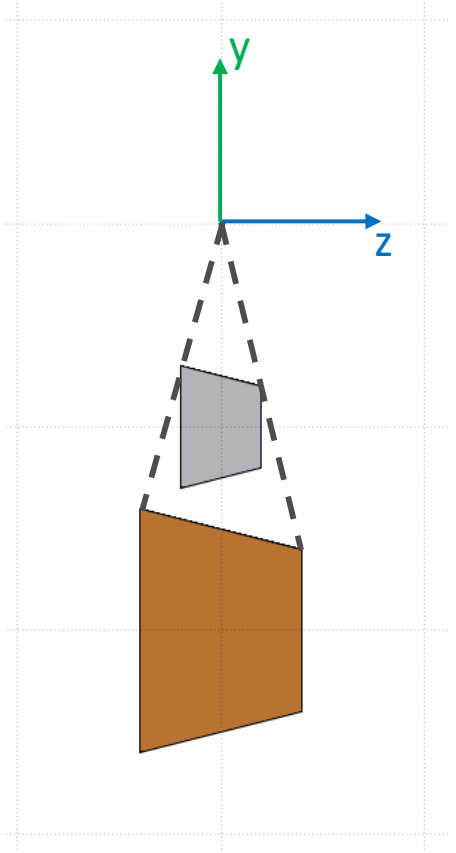
```
$start_xxx
...
$end_xxx
```

- Where "**xxx**" stands for "**expansion**", "**translat**" or "**transform**"

- 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
```

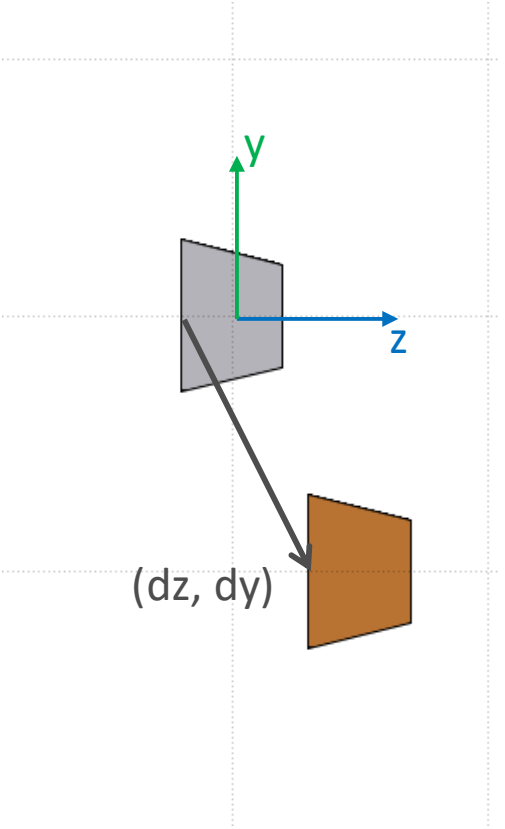


provides a coordinate expansion (or reduction) of the body dimensions by a defined scaling factor (f), for all bodies embedded between the two lines

```
$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
```

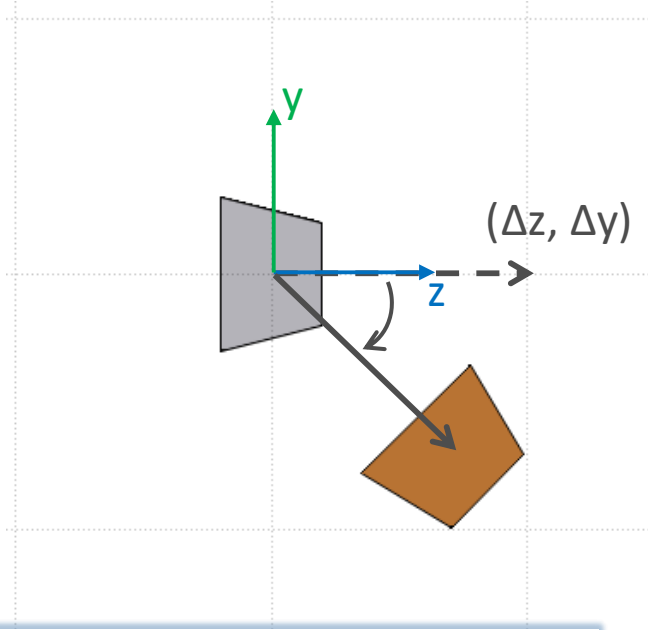


provides a coordinate translation (dx, dy, dz) for all bodies embedded within the directive

$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 pre-defined (via **ROT-DEFI**) roto-translation 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 when reading the geometry
  → 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
```

Contents

- Roto-translation transformations
  - **`ROT-DEFIni`** card
- Geometry directives
  - **`expansion`**
  - **`translat`**
  - **`transform`**
- Additional card related to a transformation
  - **`ROTPRBIN`** card
- Tips for building a modular geometry with bounding boxes

ROTPRBIN card

- Let's think about the following problem:
  - Pencil beam impinging on a cylindrical target
  - Using the R-Phi-Z USRBIN scoring, for symmetry
  - The beam is rotated by 30 around the **y** axis

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

  - It is important to note, that on the ROTPRBIN card the "inverse" transformation must be used, i.e., not the scoring mesh is transformed, but the transformation is applied to the scoring location, bringing it to the described location of the mesh

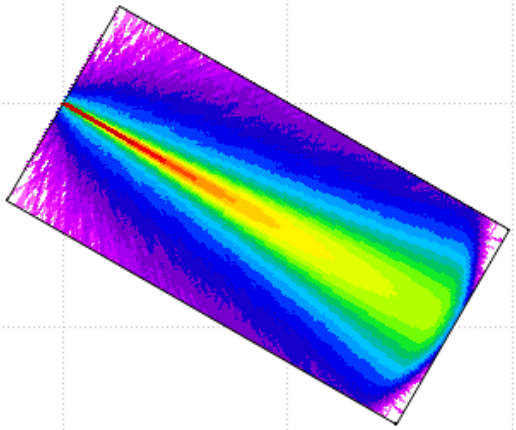
ROTPRBIN card

- Example:

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			

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:



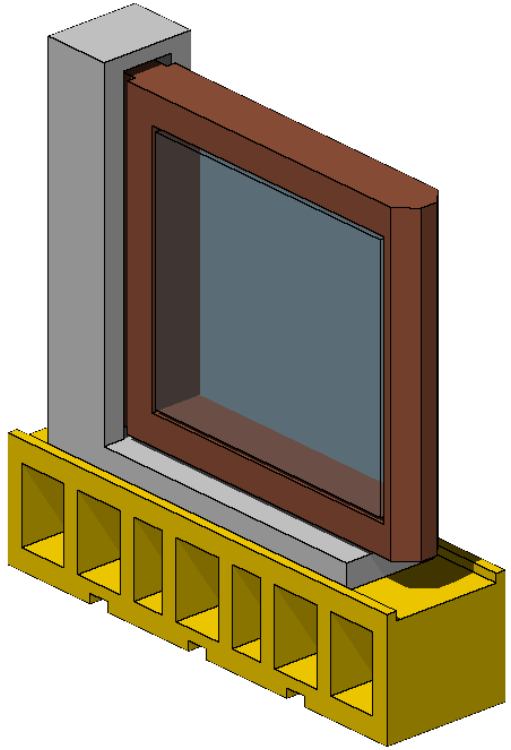
Contents

- Roto-translation transformations
  - **`ROT-DEFIni`** card
- Geometry directives
  - **`expansion`**
  - **`translat`**
  - **`transform`**
- Additional card related to a transformation
  - **`ROTPRBIN`** card
- Tips for building a modular geometry with bounding boxes

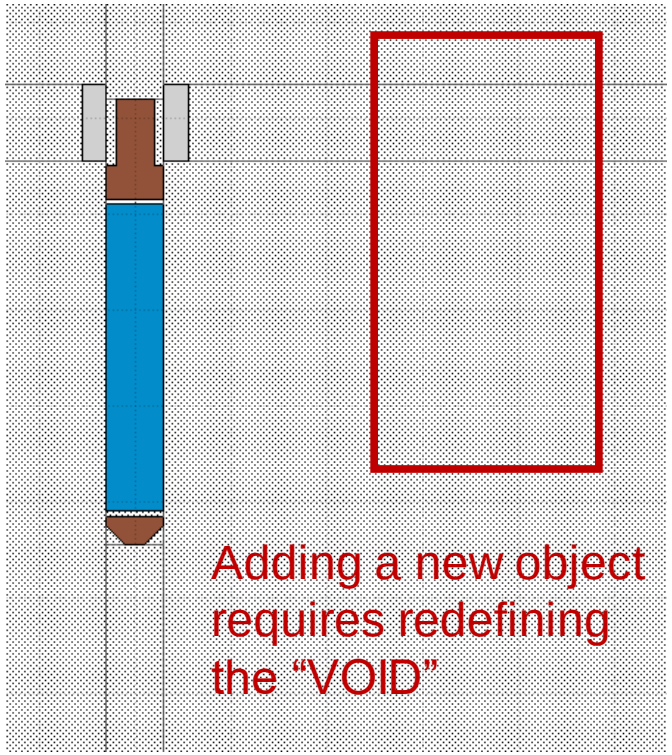
Bounding Box

Last week we learned, that defining the “VOID” around objects can be quite difficult

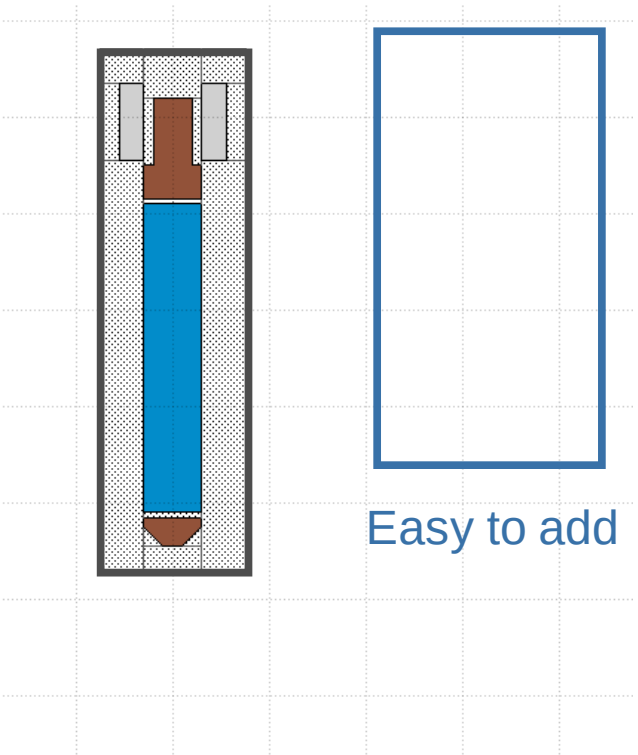
Complex object:



Complex “VOID”:

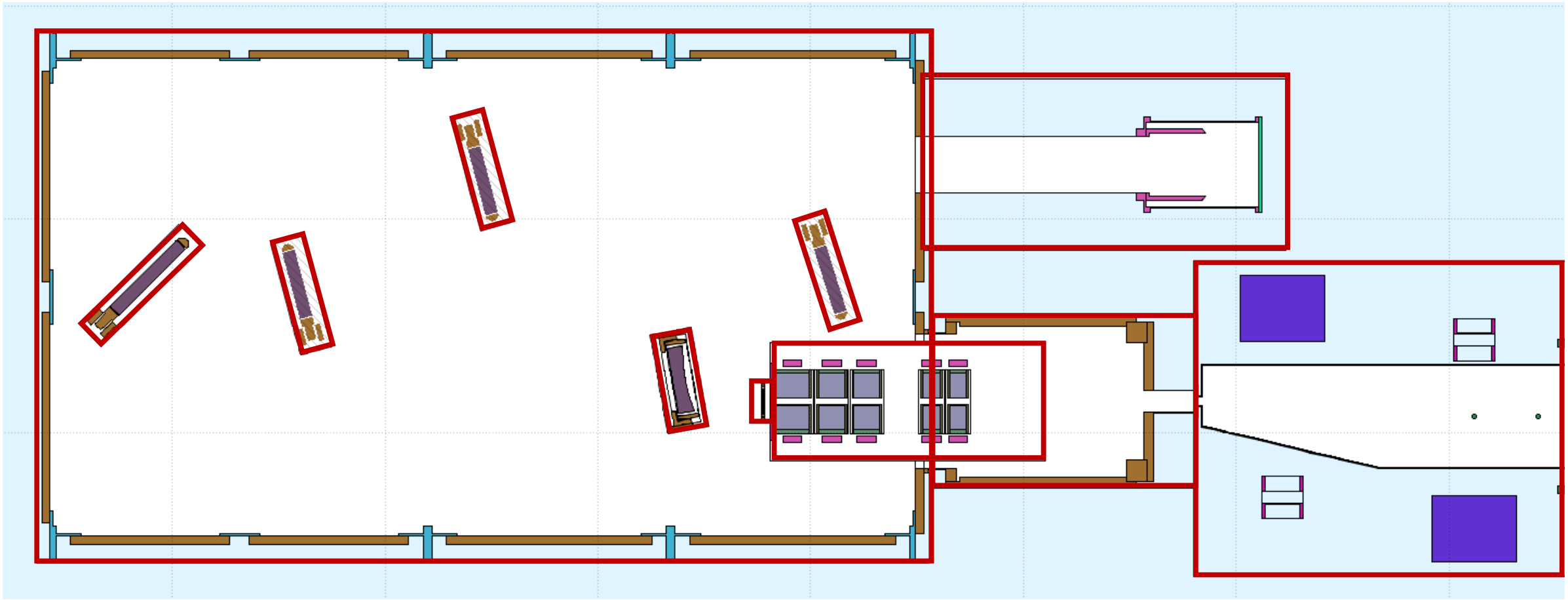


Solution the Bounding Box:



Use a finite body (**RPP**, **RCC**, etc.) to encapsulate the whole object

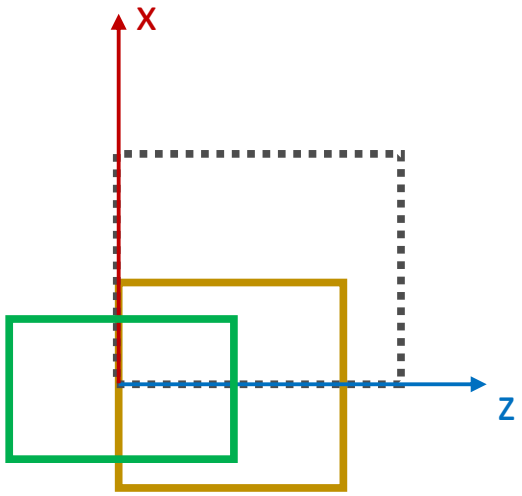
Bounding Box

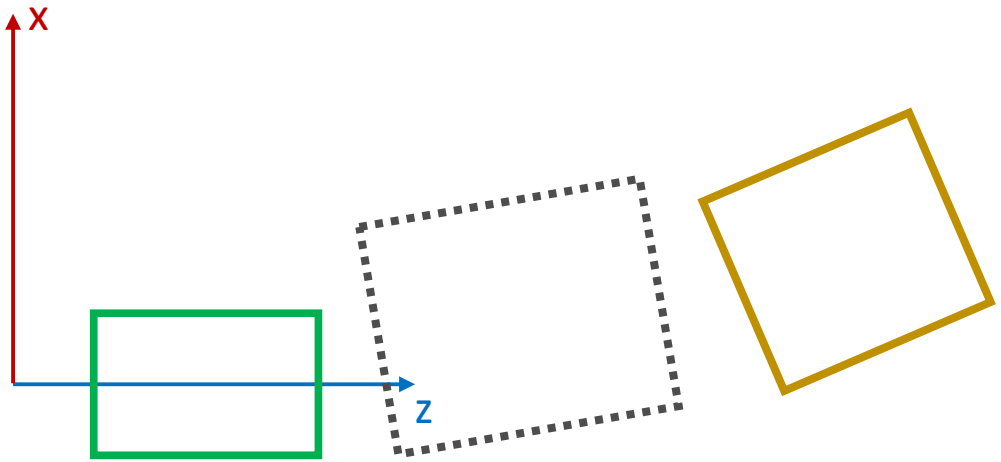


Only the Bounding Boxes have to be subtracted from the surrounding regions

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 it's final position with the geometry directives





Naming convention

- 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:

  Agreeing on a naming convention: Set prefixes for each object

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

Summary

- **ROT-DEFI** to define roto-translations

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

- **ROTPRBIN** to set the correspondence between a roto-translation transformation and selected **USRBIN** and **EVENTBIN** scorings

- Tips how to build complex geometries easier


FLUKA

Naming convention

- 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:

  Agreeing on a naming convention: Set prefixes for each object

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