



FLUKA

Combinatorial Geometry

Beginners' FLUKA Course

Introduction

Principle of Combinatorial Geometry: Basic convex shapes (**bodies**) such as cylinders, spheres, parallelepipeds, etc. are combined to more complex shapes called **regions**. This combination is done by the boolean operations **union**, **intersection** and **subtraction**.

The **Combinatorial Geometry** of FLUKA was initially similar to the package developed at ORNL for the neutron and gamma-ray transport program Morse (M.B. Emmett ORNL-4972 1975) which was based on the original combinatorial geometry by MAGI (Mathematical Applications Group, Inc., W. Guber et al, MAGI-6701 1967).

The present FLUKA CG package has been strongly improved, with the addition of **open surfaces**, **free format**, **parentheses**, the possibility of using **body and region names** instead of numbers, **expansion** and **roto-translation** of bodies, **comments**, **repetition of patterns** (lattices), **voxels**

Input

- The input format for the geometry is different from that adopted elsewhere in FLUKA, i.e. the number and length of the input fields is different.
- The **recommended** format is called **free format**. For backward compatibility there are also other formats. **Free format is not the default!**
- One advantage of free format is that alignment of the input parameters is not necessary. Bodies and regions are identified by **names**.
- In fixed format alignment is mandatory. Bodies and regions are identified by numbers and not by names which makes creation and updating of the geometry difficult.

Basic Concepts

Four concepts are fundamental in the FLUKA *CG*:

- **Bodies** - basic **convex objects**, plus **infinite planes**, **infinite cylinders** and **generic quadric surfaces**
- **Zones** - sub-regions defined only with intersection and subtraction of bodies
- **Regions** - defined as boolean operations of bodies (union of zones)
- **Lattices** - duplication of existing objects (translated & rotated), will be explained in a separate lecture

In the original description (Morse) **bodies were convex solid bodies** (finite portions of space completely delimited by surfaces of first or second degree, i.e. planes or quadrics). In FLUKA, the definition has been extended to include **infinite cylinders** (circular and elliptical), **planes** (half-spaces), and generic **quadrics** (surfaces described by 2nd degree equations)

Use of such "**infinite bodies**" is encouraged since it makes input less error-prone. They also provide a more accurate and faster tracking.

Bodies

- Each body divides the space into two domains **inside** and **outside**. The **outside** part is pointed to by the **normal** to the surface.
- 3-character code of available bodies:
 - **RPP**: Rectangular Parallelepiped
 - **SPH**: SPHere
 - **XYP, XZP, YZP**: Infinite half space delimited by a coordinate plane
 - **PLA**: Generic infinite half-space, delimited by a PLANE
 - **XCC, YCC, ZCC**: Infinite Circular Cylinder, parallel to coordinate axis
 - **XEC, YEC, ZEC**: Infinite Elliptical Cylinder, parallel to coordinate axis
 - **RCC**: Right Circular Cylinder
 - **REC**: Right Elliptical Cylinder
 - **TRC**: Truncated Right angle Cone
 - **ELL**: ELLipsoid of revolution
 - **QUA**: QUAdric
- Other bodies **ARB, RAW, WED, BOX**
 - **don't use them**, they cause sometimes rounding problems

The Black Hole

To avoid infinite tracking the particles must be stopped somewhere. This has to be insured by the user by defining a region surrounding the geometry and assigning the material **BLCKHOLE** to it.

The outer surface of this region must be defined by a single closed body (generally an RPP or a Sphere).

All particles that enter the blackhole are absorbed (they disappear). Further blackhole regions can be defined by the user if necessary.

The blackhole is the outermost boundary of the geometry. Inside its outer surface:

Each point of space must belong to one and only one region!

Combinatorial Geometry Input

CG input must respect the following sequential order:

GEOBEGIN card

VOXELS card (optional, see Voxel lecture)

Geometry title (and reading format options)

Body data

END card (not needed in flair)

Region data

END card (not needed in flair)

LATTICE cards (optional, see Lattice lecture)

Region volumes

(optionally requested by a flag in the *Geometry title*, used together with the **SCORE** command)

GEOEND card

Cards having a * in column 1 are treated as comments.

Free format

Free format is recommended with respect to the other formats, which will be kept however for reasons of back compatibility. Its main advantages, in addition to the freedom from strict alignment rules, are the possibility to modify the input sequence without affecting the region description (for instance, by inserting a new body) and the availability of parentheses to perform complex boolean operations in the description of regions.

Free format input is used for both body and region if requested by `SDUM = COMBNAME` in the `GEOBEGIN` card, or by a `GLOBAL` command at the beginning of the input file.

GEOBEGIN card

The meanings of the **WHAT** and **SDUM** parameters are:

WHAT(1) flag for switching off geometry error messages: **don't touch!!**

Default = 0.0 (all geometry error messages are printed)

WHAT(2) used to set the accuracy parameter - **use with care !**

WHAT(3) = logical unit for the geometry input.

If > 0.0 and different from 5, the name of the corresponding file must be input on the next card. Otherwise, the geometry input follows.

WHAT(4) = logical unit for the geometry output. If > 0.0 and different from 11, the name of the corresponding file must be input on the next card. Otherwise, geometry output is printed on the standard output.

WHAT(5) Parenthesis optimization level (see FLUKA manual)

WHAT(6) not used

SDUM = **COMBNAME** or **COMBINAT**

COMBNAME selects free format, **COMBINAT** fixed format

Default: **COMBINAT (!)**

Can be overwritten by **WHAT(5)** of a possible **GLOBAL** card

The geometry title card

This card has no keyword, it is the line that follows the *GEOBEGIN* card (unless voxels are used).

Three variables are input in the *CG* Title card: **IVOPT**, **IDBG**, **TITLE**.

The format is (2I5, 10X, A60).

The first integer value (**IVOPT** = Input Volumes OPTion) indicates how to normalize the quantities scored in regions by the option **SCORE**

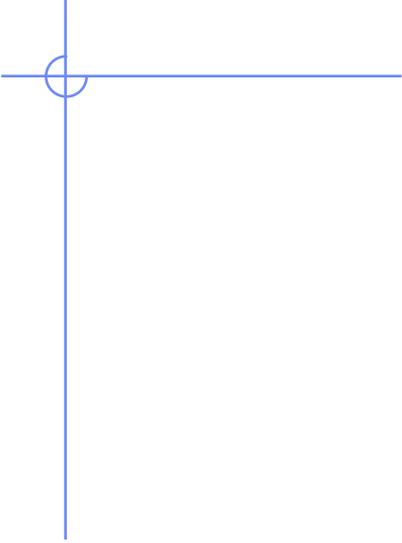
IVOPT = 0 → no region volumes will be input before *GEOEND*

IVOPT = 3 → region volumes will be input before *GEOEND*. As many volume values must be input as there are geometry regions, in this format: (7E10.5).

IDBG is a flag used to request different kinds of geometry **fixed** format input.

IDBG = 0 : default fixed format

IDBG = -10 or -100 : high accuracy fixed format



Bodies

Bodies

The input for each **body** consists of

- the 3-letter code indicating the body type (RPP, ZCC...)
- a unique "**body name**" (8 character maximum, alphanumeric identifier, **case sensitive**)
- a set of geometrical quantities defining the body (their number depends on the body type as explained below).

A maximum of 132 characters per line are accepted, use extra lines if required.

The different items, separated by **one or more blanks**, or by one of the separators **, / ; :** can extend over as many lines as needed.

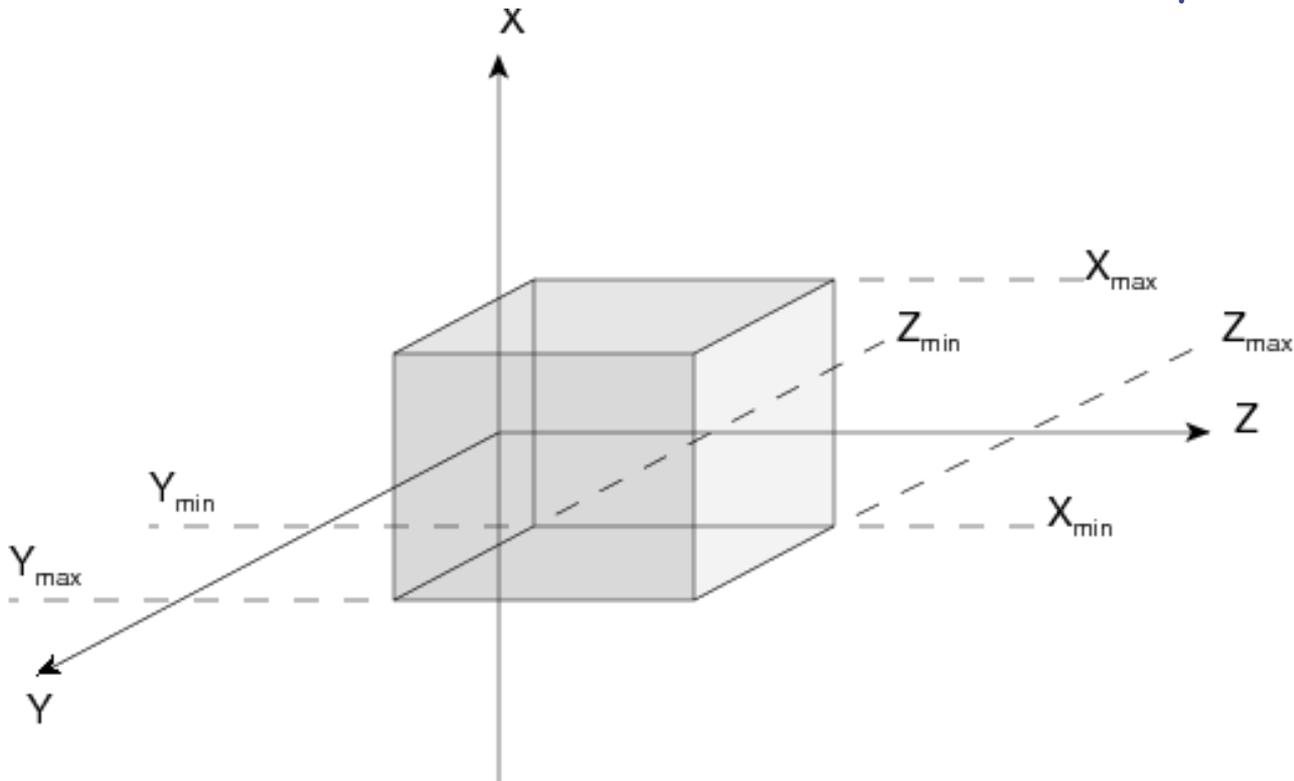
All values are in cm!

Rectangular Parallelepiped: **RPP**

An **RPP** has its edges parallel to the coordinate axes. It is defined by 6 numbers in the following order:

$$X_{\min}, X_{\max}, Y_{\min}, Y_{\max}, Z_{\min}, Z_{\max}$$

(minimum and maximum coordinates which bound the parallelepiped).



Rectangular Parallelepiped: **RPP**

Free format

```
RPP Sm1Brick -20.0 +20.0 -50.0 +50.0 -38.5 +38.5
```

An **RPP** definition extends over one single card in default fixed format, or over two cards in high-accuracy body fixed format

Example in **fixed format** (the comment lines shown are allowed input lines):

```
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+.  
RPP      4      -20.0      +20.0      -50.0      +50.0      -38.5      +38.5  
* (a parallelepiped centered on the origin)
```

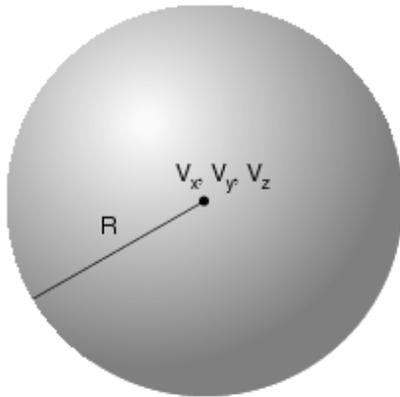
High-accuracy fixed format

```
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+.  
RPP      4              -20.0              +20.0              -50.0  
              +50.0              -38.5              +38.5
```

Sphere and circular cylinder

Sphere: SPH

A SPH is defined by 4 numbers: V_x, V_y, V_z (coordinates of the centre), R (radius)

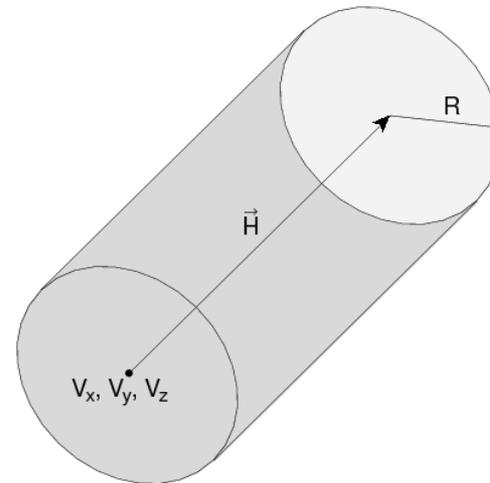


Right circular cylinder: RCC

An RCC can have any orientation in space. It is limited by a cylindrical surface and by two plane faces \perp to its axis. Each RCC is defined by 7 numbers:

V_x, V_y, V_z (centre of one face);

H_x, H_y, H_z (vector corresponding to the cylinder height, pointing to the other face); R (cylinder radius).



Infinite half-space parallel to coordinate axis

There are 4 kinds of infinite half-spaces.

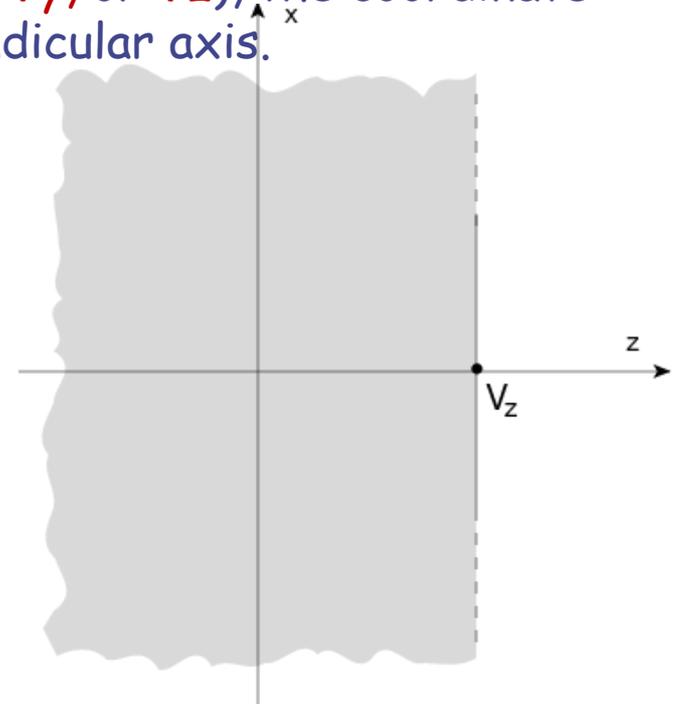
Three are delimited by planes perpendicular to the coordinate axes:

1. Delimited by a plane \perp to the x -axis. Code: **YZP**
2. Delimited by a plane \perp to the y -axis. Code: **XZP**
3. Delimited by a plane \perp to the z -axis. Code: **XYP**

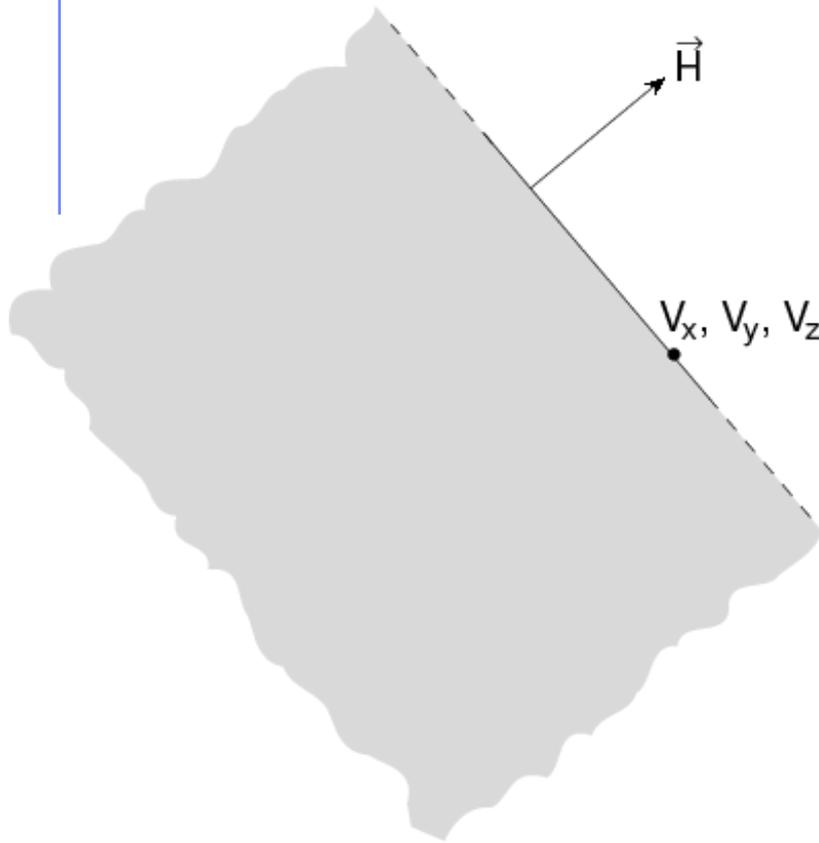
All defined by a single number: V_x (resp. V_y , or V_z), the coordinate of the plane on the corresponding perpendicular axis.

The half-space "inside the body" is the position of points for which:

$$x < V_x \quad (\text{resp. } y < V_y, \text{ or } z < V_z)$$



Arbitrarily orientated infinite half-space: **PLA**



A **PLA** defines the infinite half space delimited by a generic plane.

A **PLA** is defined by 6 numbers:

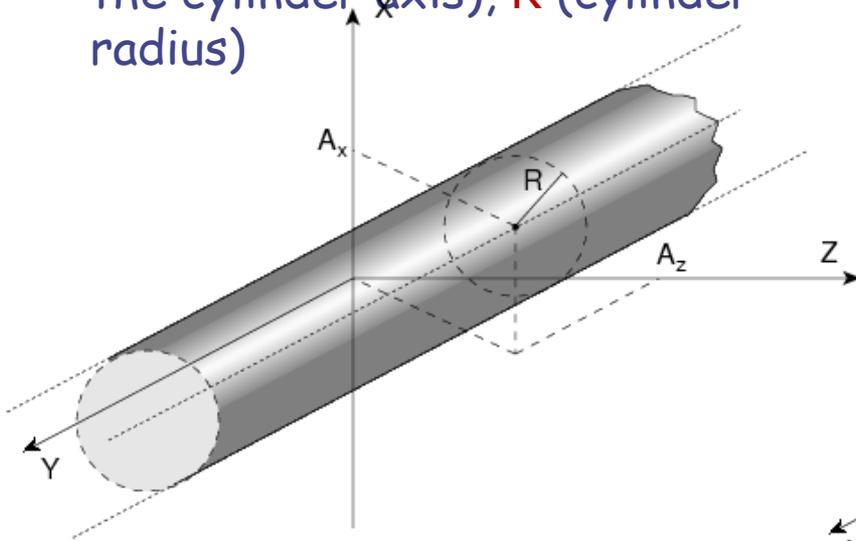
H_x, H_y, H_z (vector of arbitrary length \perp to the plane);

V_x, V_y, V_z (any point lying on the plane). The half-space "inside the body" is that from which the vector is pointing (i.e. the vector points "outside").

Infinite cylinders

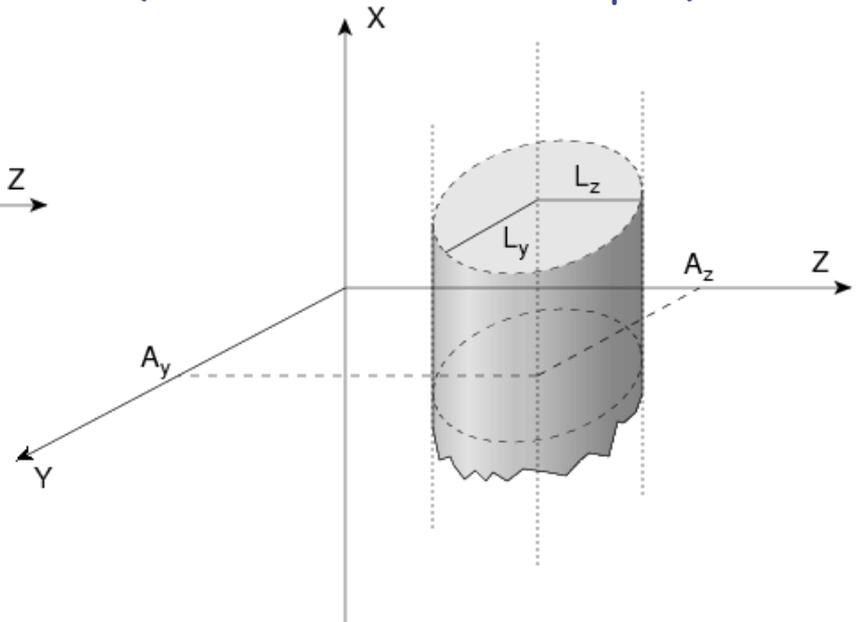
Infinite Circular Cylinder || to coordinate axis: **XCC**, **YCC**, **ZCC**

Each **XCC** (**YCC**, **ZCC**) is defined by 3 numbers: A_y , A_z for **XCC** (A_z , A_x for **YCC**, A_x , A_y for **ZCC**) (coordinates of the cylinder axis), R (cylinder radius)



Infinite Elliptical Cylinder || to coordinate axis: **XEC**, **YEC**, **ZEC**

Each **XEC** (**YEC**, **ZEC**) is defined by 4 numbers: A_y , A_z for **XEC** (A_z , A_x for **YEC**, A_x , A_y for **ZEC**) (coordinates of the cylinder axis); L_y , L_z for **XEC** (L_z , L_x for **YEC**, L_x , L_y for **ZEC**) (semi-axes of the ellipse)



Arbitrary generic quadric: **QUA**

A **QUA** defines a quadric surface defined by a 2nd degree equation $F(x,y,z) = 0$.

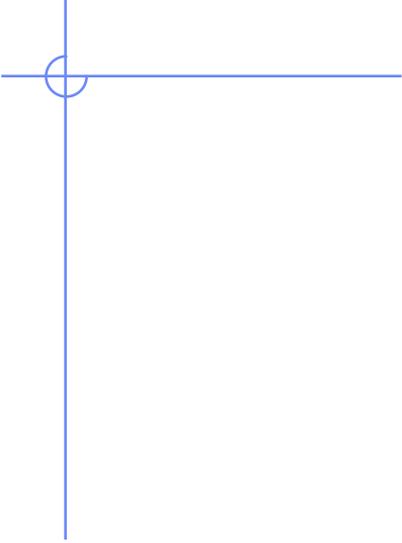
Each **QUA** is defined by 10 numbers:

$$A_{xx}, A_{yy}, A_{zz}, A_{xy}, A_{xz}, A_{yz}, A_x, A_y, A_z, A_0$$

corresponding to the equation:

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

A **QUA** definition extends over two cards in default fixed format, and over 4 cards in high-accuracy body fixed format.



Regions

Concept

Regions are defined as combinations of bodies obtained by boolean operations:

	Union	Subtraction	Intersection
Free Format		-	+
Fixed format	OR	-	+
Mathematically	\cup	-	\cap

Regions are not necessarily simply connected (they can be made as the union of two or more non contiguous or partially overlapping zones) but must be of homogeneous material composition.

Regions

Input for each region starts on a new line and extends on as many continuation lines as are needed. It is of the form:

REGNAME NAZ boolean-zone-expression

or

REGNAME NAZ boolean-zone-expression | boolean-zone-expression | ...

- **REGNAME** is the **region "name"** (8 character maximum, alphanumeric identifier, **case sensitive**). Must begin by an alphabetical character
- **NAZ** is a rough guess for the number of regions which can be entered leaving the bodies of the current region, normally 5
- "**boolean-zone-expression**" is a sequence of one or more **body** names preceded by the operators + (intersection) or - (complement or subtraction). A **zone** expression can be contained inside one set of left and right parentheses. Several **zone** expressions can be combined by the union operator | .

Regions

- | (OR) operators are used to join **zones**
- **zones** are defined by intersections and/or subtraction of **bodies**, each body being preceded by its + or - sign.
- In evaluating the expressions, the highest operator precedence is given to parentheses, followed by +, - and the | operator.
- If one line is not sufficient, any number of continuation lines can be added.
- Blanks are ignored.

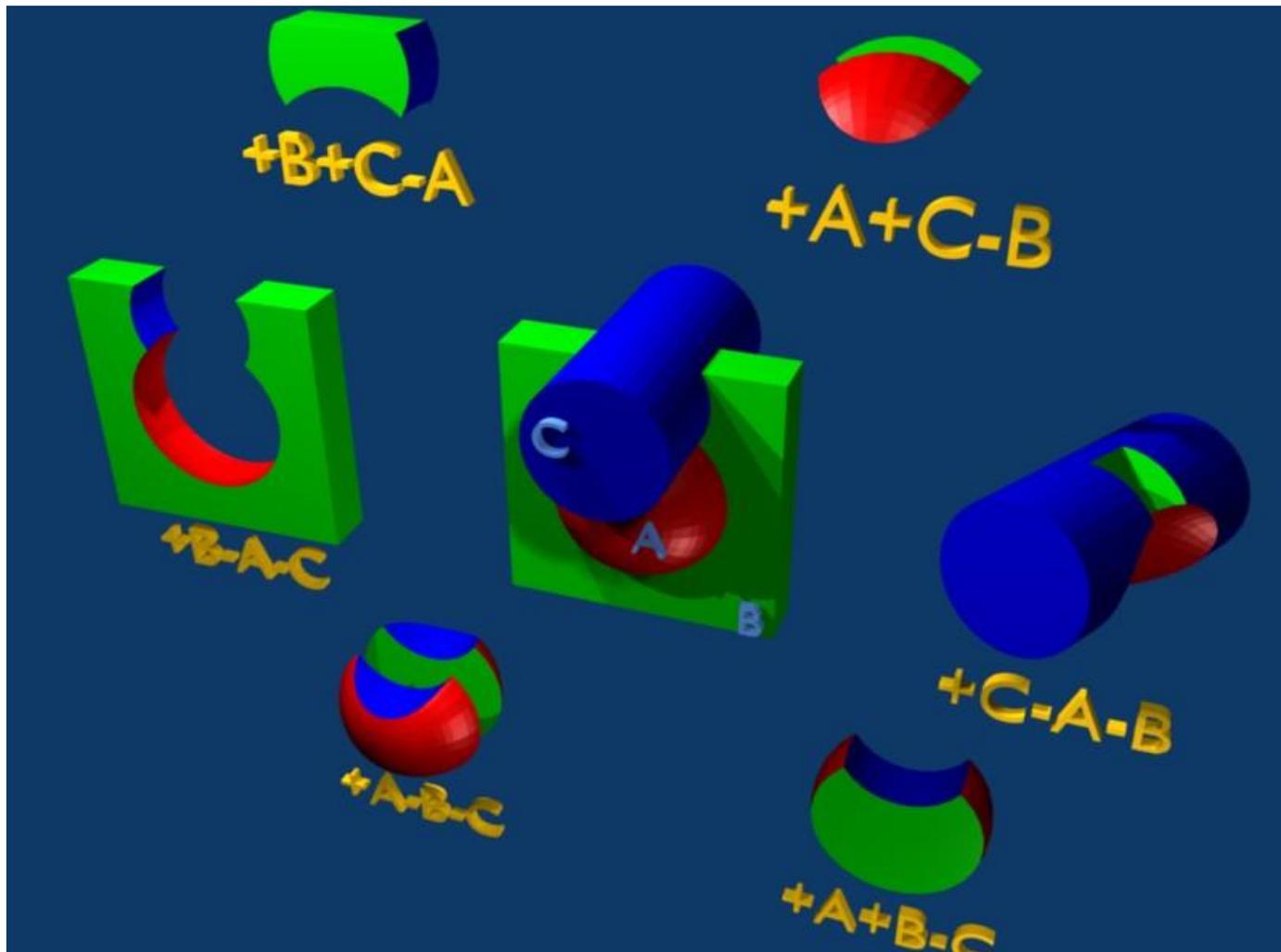
Meaning of + and - operators

If a **body** number appears in a **zone** description preceded by a + operator, it means that the **zone** being described is **wholly contained inside** the **body**.

If a **body** number appears in a **zone** description preceded by a - operator, it means that the **zone** being described is **wholly outside** the **body**.

Zones must be finite: obviously, in the description of each **zone** and hence of each **region** the symbol + must appear at least once.

Illustration of the $+$ and $-$ operators



Meaning of the + - operators

Target 4 +Outer +CutPlane

- * (the above region is the part of space common to bodies Outer and
- * CutPlane)

Regex 7 +Red -Blue -Green +Yellow

- * (the above region is the part of space common to body Red and Yellow,
- * excluding however that which is inside body Blue and that which is
- * inside Green)

Air 5 +Room

- * (the latter region coincides entirely with body Room)

Meaning of the | (OR) operator

The | (or OR) operator is used as a boolean **union** operator in order to combine **zones** (**sub regions** partially overlapping or not). **Zones** are formed as intersections or differences as explained above, and then the **region** is formed by a union of these **zones**.

Examples:

```
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
  SA7  11OR  +4    +6    -7    -8OR  +6    +3    -21
*           <---- first zone          -----><- second zone  ->
```

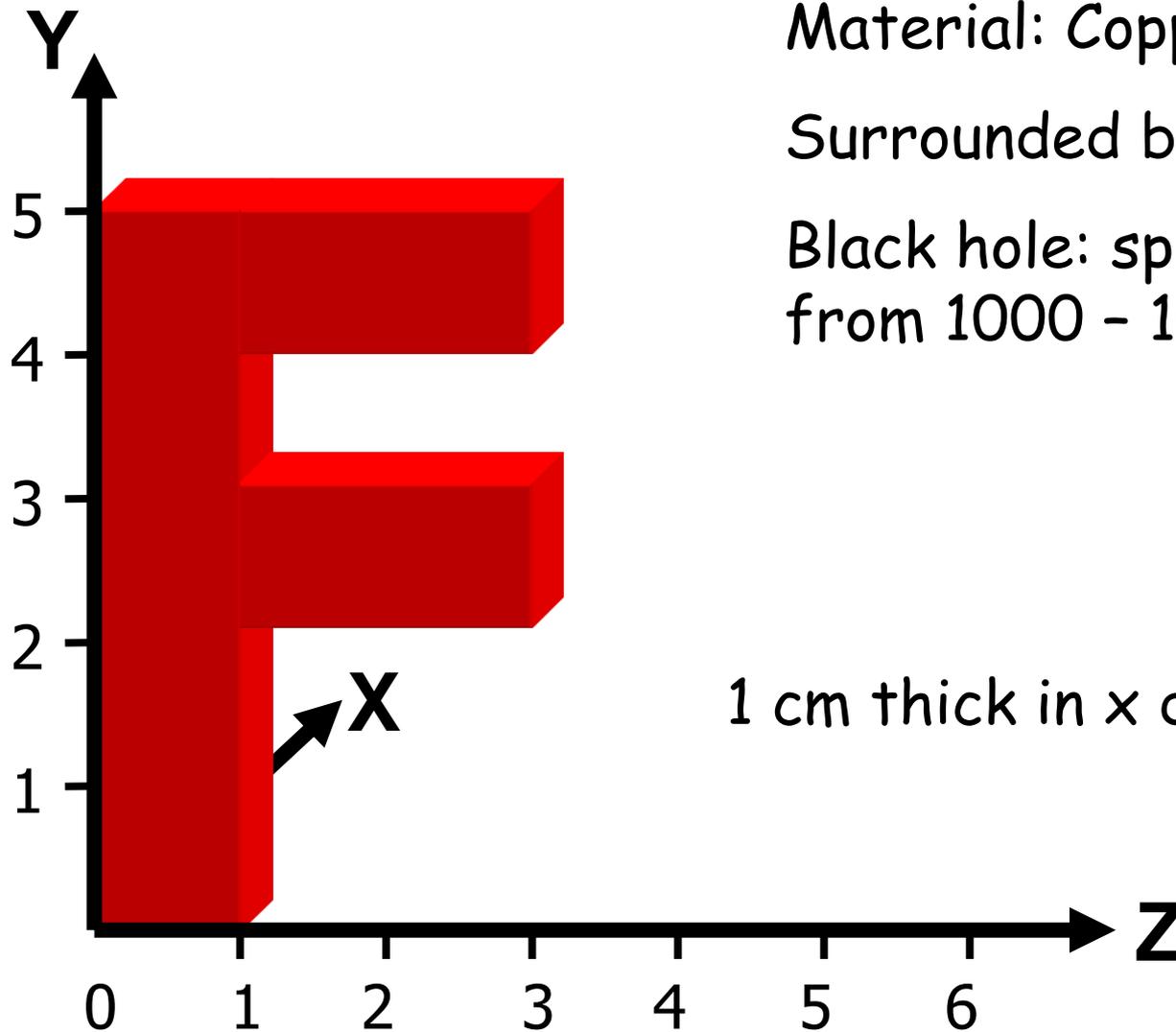
```
Ground 5 | +Body9 | +Body15 | +Body1 | +Body8 -Body2 | +Body8 -Body3 |
*           <- 1st -><- 2nd -><- 3rd -><----- 4th -----><----- 5th ----->
           | +Body8 +Body18 | +Body12 -Body10 -Body11 -Body13 -Body14
*           <----- 6th -----><----- 7th and last zone ----->
* (continuation line)
```

Geometry Example "F" shaped target

Material: Copper

Surrounded by vacuum

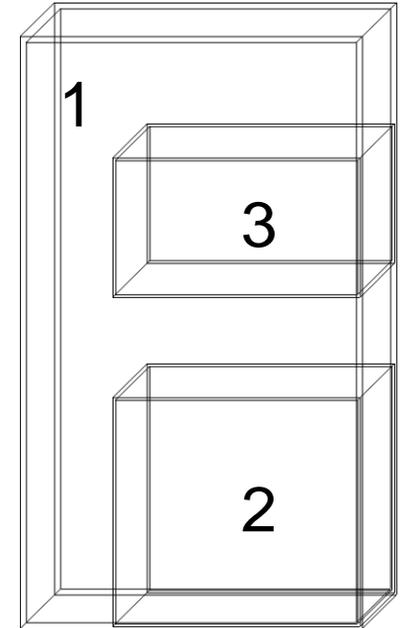
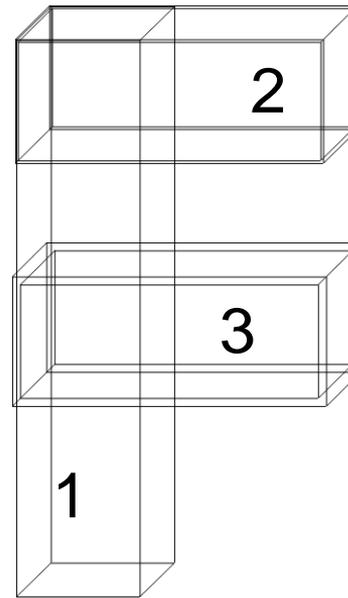
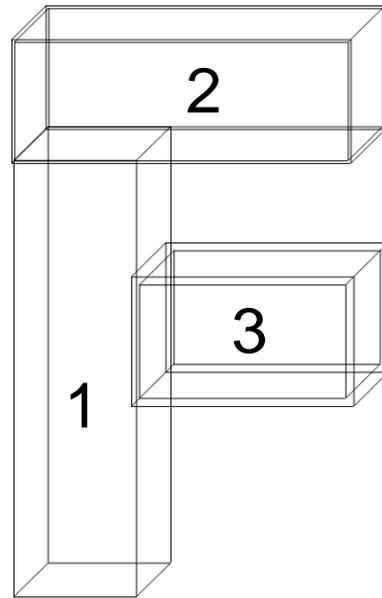
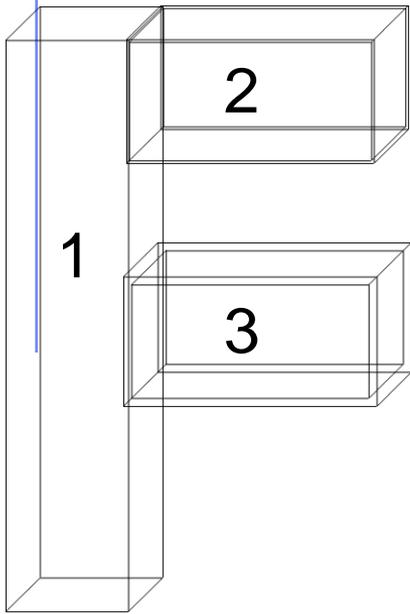
Black hole: spherical shell
from 1000 - 10000 cm



1 cm thick in x direction

Geometry example "F": Bodies

Several possibilities for bodies:



(A) 3 bodies

(B) 3 bodies

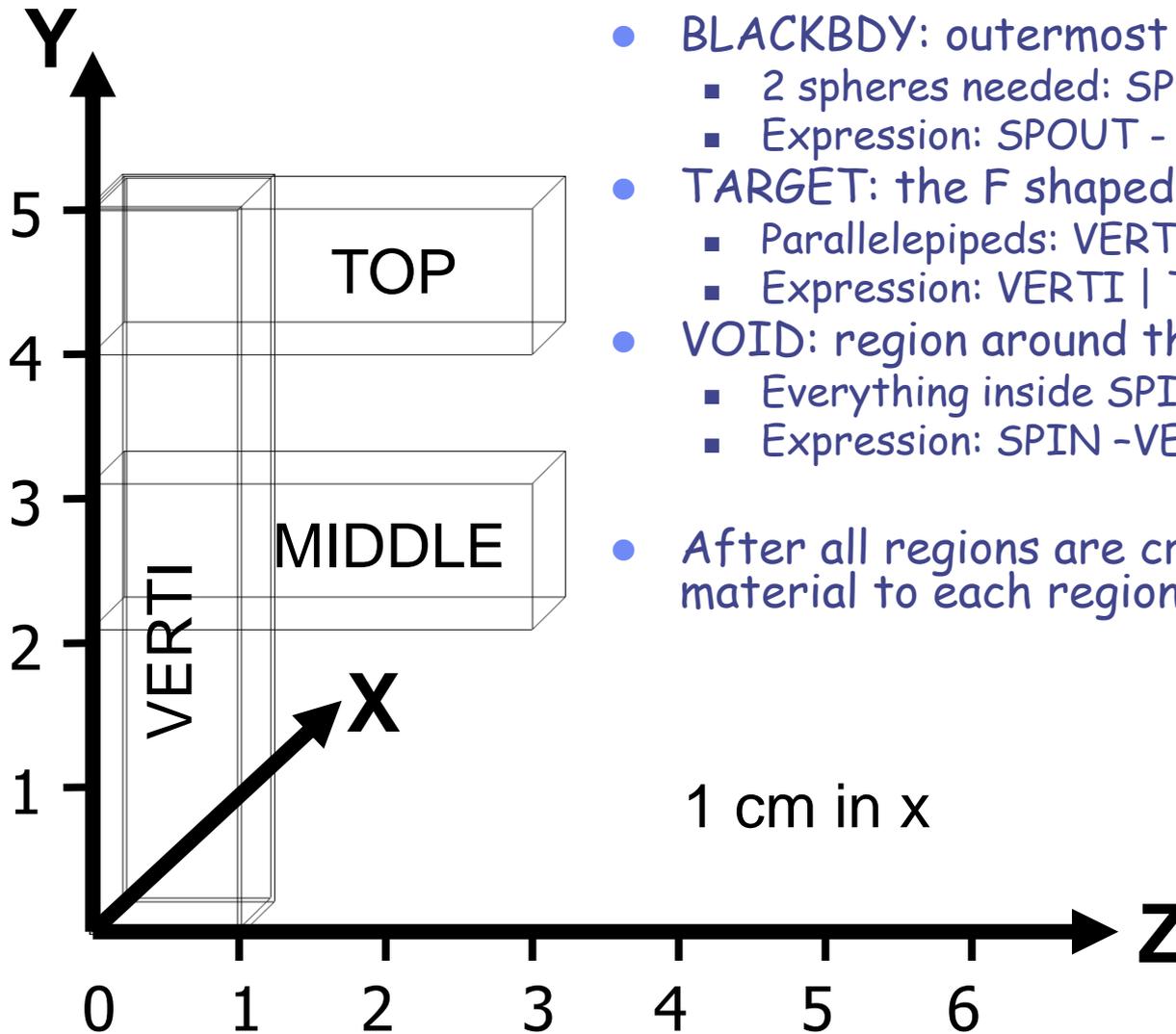
(C) overlapping

(D) subtraction

If we use the 3 bodies to make a single region, the 4 options are equivalent. We will use C.

If 1, 2 and 3 are separate regions, avoid A and B: there are surfaces shared by different regions

Geometry example "F": REGIONS



- BLACKBDY: outermost region like a shell
 - 2 spheres needed: SPOUT, SPIN
 - Expression: SPOUT - SPIN
- TARGET: the F shaped object
 - Parallelepipeds: VERTI, TOP, MIDDLE
 - Expression: VERTI | TOP | MIDDLE
- VOID: region around the target
 - Everything inside SPIN that is not target
 - Expression: SPIN - VERTI - TOP - MIDDLE
- After all regions are created: Assign material to each region with ASSIGNMA

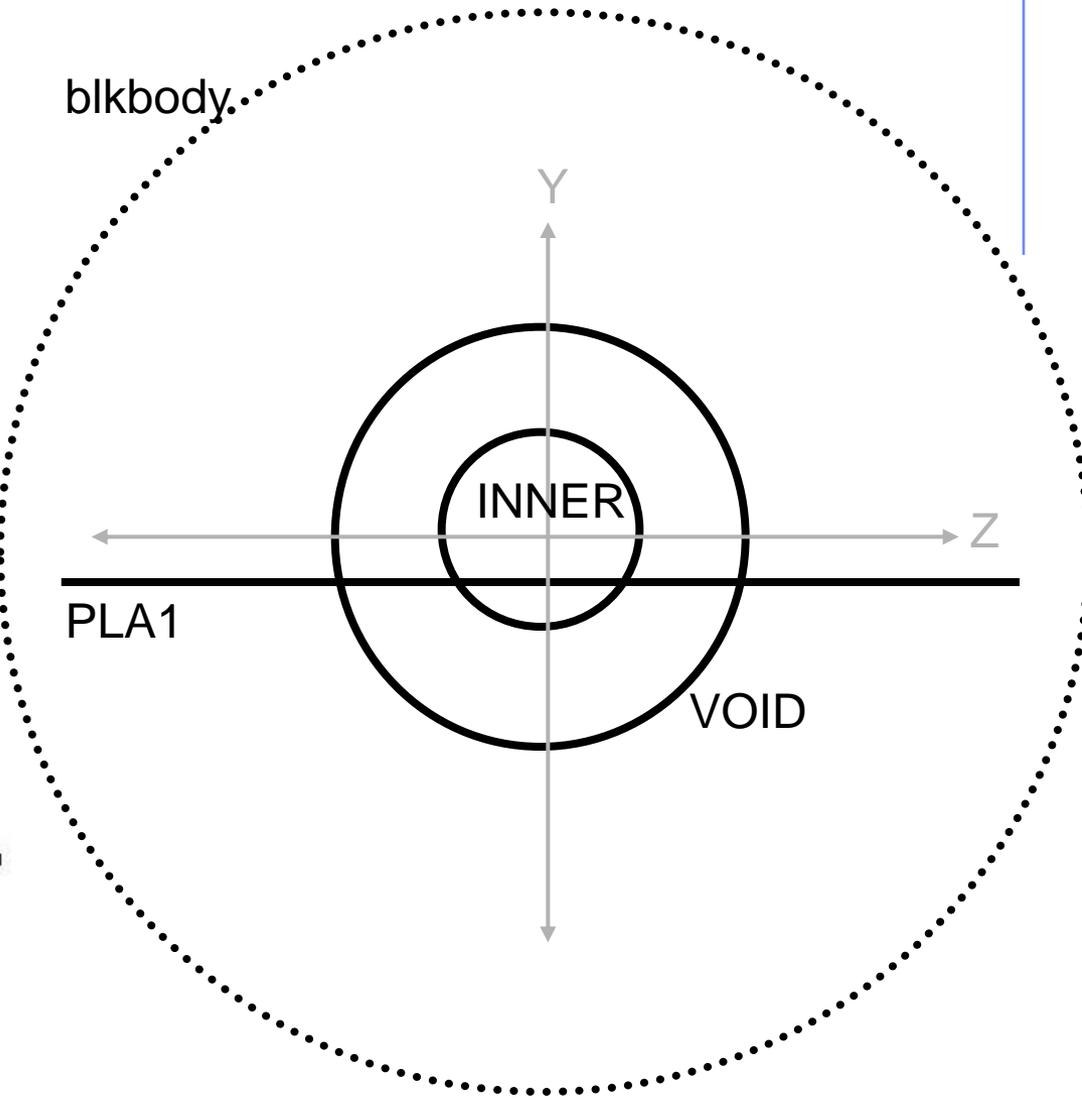
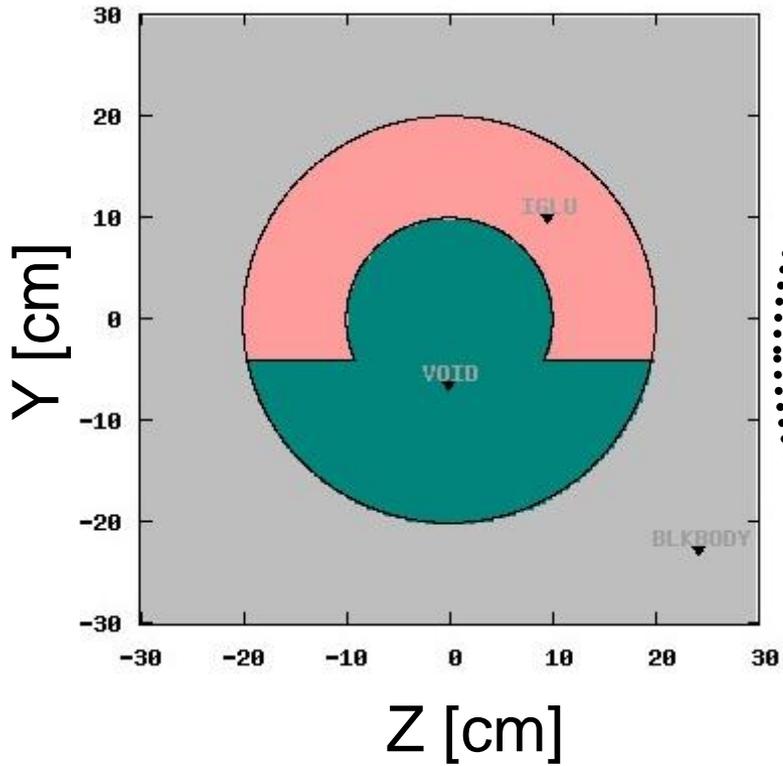
Geometry example "F": input file

```
●  GEOBEGIN COMBNAME
      0      0 The copper F
SPH SPIN      0.0 0.0 0.0 1000.
SPH SPOUT     0.0 0.0 0.0 10000.
RPP VERTI     0.0 1. 0.0 5. 0.0 1.
RPP TOP       0.0 1. 4. 5. 0.0 3.
RPP MIDDLE    0.0 1. 2. 3. 0.0 3.
END
* Black hole
BLKBODY      5  +SPOUT -SPIN
* Void around
VOID         5  +SPIN -TOP -VERTI -MIDDLE
* Target
TARGET      5  TOP | VERTI | MIDDLE
END
GEOEND
ASSIGNMA    BLCKHOLE    BLKBODY
ASSIGNMA    VACUUM      VOID
ASSIGNMA    COPPER      TARGET
```



BEAMPOS		x:	y:	z:
		cosx:	cosy:	Dirz: POSITIVE ▼
GEOBEGIN		Log: ▼	Acc:	Opt: ▼
		Inp: ▼	Out: ▼	Fmt: COMBNAME ▼
Title: The copper F				
SPH	SPOUT	x: 0	y: 0	z: 0
		R: 10000		
SPH	SPIN	x: 0	y: 0	z: 0
		R: 1000		
RPP	VERTI	Xmin: 0	Xmax: 1	
		Ymin: 0	Ymax: 5	
		Zmin: 0	Zmax: 1	
RPP	TOP	Xmin: 0	Xmax: 1	
		Ymin: 4	Ymax: 5	
		Zmin: 0	Zmax: 3	
RPP	MIDDLE	Xmin: 0	Xmax: 1	
		Ymin: 2	Ymax: 3	
		Zmin: 0	Zmax: 3	
Black hole				
REGION		Name: BLKBODY	Neigh: 5	Volume:
Expr: +SPOUT -SPIN				
Void around				
REGION		Name: VOID	Neigh: 5	Volume:
Expr: +SPIN -TOP-VERTI -MIDDLE				
Target				
REGION		Name: TARGET	Neigh: 5	Volume:
Expr: TOP VERTI MIDDLE				
GEOEND		▼		
...t...1...t...2...t...3...t...4...t...5...t...6...t...7..				
ASSIGNMA		Mat: BLCKHOLE ▼	Reg: BLKBODY ▼	to Reg: ▼
			Step:	Field: ▼
ASSIGNMA		Mat: VACUUM ▼	Reg: VOID ▼	to Reg: ▼
			Step:	Field: ▼
ASSIGNMA		Mat: COPPER ▼	Reg: TARGET ▼	to Reg: ▼
			Step:	Field: ▼
USRBIN			Unit: 21 BIN ▼	Name: edep
	Type: X-Y-Z ▼	Xmin: -10	Xmax: 10	NX: 100

Example: Igloo



Example: Igloo

----- TITLE ... BEAMPOS : 5 cards hidden -----

GEOBEGIN	Log: ▼	Acc:	Opt: ▼
	Inp: ▼	Out: ▼	Fmt: COMBNAME ▼
Title:			
Black body SPH	blkbody	x: 0.0 R: 1000.0	y: 0.0 z: 0.0
Void sphere SPH	void	x: 0.0 R: 20.0	y: 0.0 z: 0.0
SPH	inner	x: 0 R: 10	y: 0 z: 0
XZP	pla	y: -4	
Black hole REGION	Expr: +blkbody -void	Name: BLKBODY	Neigh: 5 Volume:
Void around REGION	Expr: +void +pla +inner	Name: VOID	Neigh: 5 Volume:
Target REGION	Expr: +void -pla -inner	Name: IGLU	Neigh: 5 Volume:
GEOEND	▼		

----- ASSIGNMA ... STOP : 6 cards hidden -----

Parentheses

Parentheses are grouping together combinations of bodies.
Parentheses can be used in **free format** only.

Examples:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
* Subtract from body2 regions regR03, regR04, regR05
regV02 5 +body2 - (+body4 -body3)
           - (+body6 -body5 | +body8 -body7) - body9 -body10
regR03 5 +body4 -body3
regR04 5 +body6 -body5 | +body8 -body7
regR05 5 +body9 | +body10
```

Nested parentheses are supported, however:

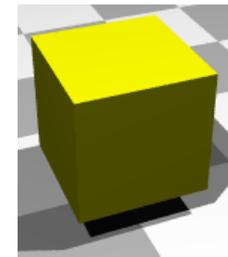
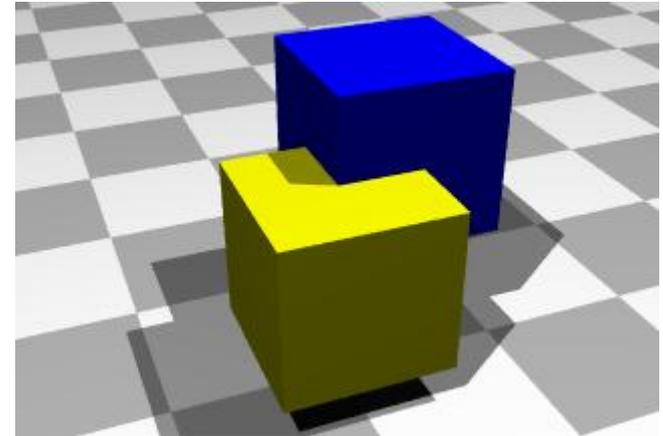
parentheses should be used with care since their expansion can generate a quickly diverging amount of terms. A partial optimization is performed on planes (aligned with the axes) and bounding boxes only

Parentheses Expansion

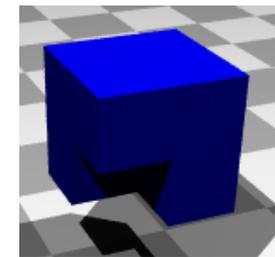
```

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 EXTUVOID 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 +EXTUVOID
    -(+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 enclosed between two lines:

\$Start_xxx

.....

\$End_xxx

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

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

Directives can be nested.

Geometry directives: expansion

\$Start_expansion 10.

QUA Elipsoid 0.0025 0.04 0.0064 0.0 0.0 0.0 0.055 0.64
0.0256 1.8881

\$End_expansion

transforms an ellipsoid centred at $(11, 8, -2)$,
with semiaxes $20, 5$ and 12.5 cm long parallel to the
coordinate axes,

to one centered at $(110, 80, -20)$
with semiaxes $200, 50$ and 125 cm long.

Geometry directives: translation

```
$Start_translat -5., -7., +9.
```

```
QUA Elipsoid 0.0025 0.04 0.0064 0.0 0.0 0.0 0.055 0.64  
0.0256 1.8881
```

```
$End_translat
```

transforms an ellipsoid centred at $(11, 8, -2)$,

with semiaxes $20, 5$ and 12.5 cm long parallel to the coordinate axes,

to an identical one centred at $(6, 1, 7)$

Geometry directives: transformation

- * ROT-DEFI transformation: shift of (0,-2,-30) then rotation of -21 deg.
- * around the x axis

```
ROT-DEFI      0.0                -2.0  -30.0Rotdefi1
ROT-DEFI      100.              -21.                Rotdefi1
```

- * A cylindrical target is shifted and rotated with transformation "Rotdefi1"

```
$Start_transform Rotdefi1
RCC targRepl  0.0  0.0  -5.0  0.0  0.0  10.0  5.0
$End_transform
```

The ROT-DEFIni cards are used by the geometry directive to transform the coordinates of the bodies (not of the particles!).

If a "-" sign is placed in front of the ROT-DEFIni name of the "\$Start_transform" directive, the **inverse transformation** is used instead

Directives can be nested. Priority order:

1) Expansion 2) Translation 3) Transformation

Important Notes

- Whenever it is possible, the following bodies should be preferred:

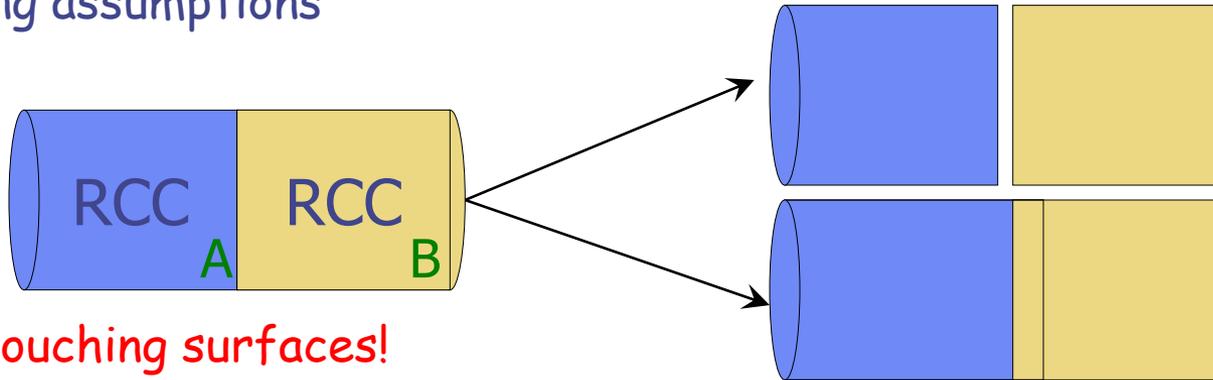
PLA, RPP, SPH, XCC, XEC, XYP
XZP, YCC, YEC, YZP, ZCC, ZEC, QUA

These make tracking faster, since for them extra coding ensures that unnecessary boundary intersection calculations are avoided when the length of the next step is smaller than the distance to any boundary of the current **region**.

- Always **use as many digits as possible** in the definition of the body parameters, particularly for body heights (RCC, REC, TRC), and for direction cosines of bodies with slant surfaces. The free format or the high-accuracy fixed format should always be used in these cases.
- **Avoid** as much as possible building separate regions with **bodies which share the same surface** (e.g. two RPP's with a common face). Rather make one or both bigger and cut both with a common plane.

Precision Errors

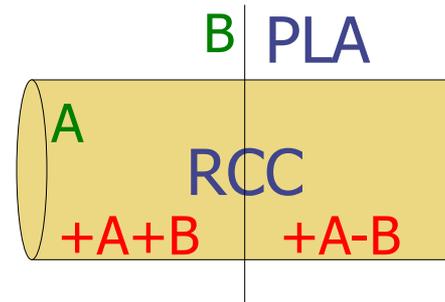
Modeling assumptions



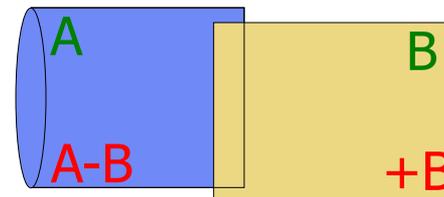
Avoid touching surfaces!

When floating point operations are involved.

- Use cutting surfaces **B** instead.



- Or force partial overlap of objects

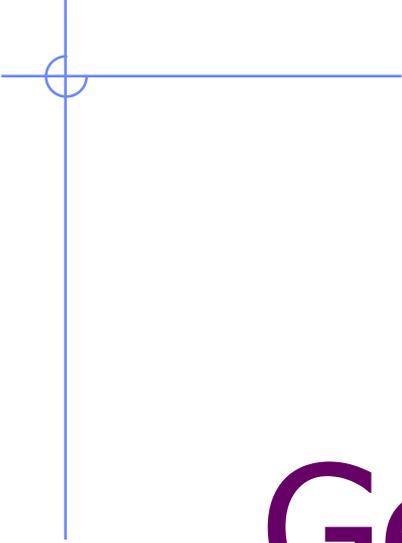


Precision Errors

FLUKA uses systematically double precision arithmetic:

- Double precision: **16 significant digits**
- Due to rounding errors: **~14 significant digits**
- Precision is **relative** to the position.
- Avoid objects like:
 - **BOX**: arbitrary oriented in space. It is difficult to ensure perpendicularity of the surfaces. Use instead RPP, and rotate it
 - **ARB, WED**: use planes instead
- If and only if there is no other solution, one can relax the geometry precision with the **WHAT(2)** on the **GEOBEGIN** card.

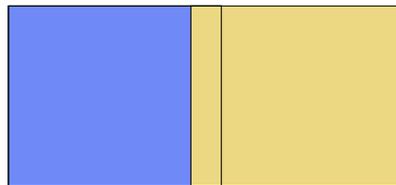
USE WITH CAUTION!



Geometry Errors and Debugging

Geometry Errors

- During execution the code always needs to know the region where a particle is located at every step.
- The program will **stop** only if a particle's position **doesn't not belong to any region**. It will issue an error message on the **.err** file with the particle position.
- **IMPORTANT!** It will **not stop** if a particle's position **belongs to more than one region**. It will accept the first region it finds but the results will be meaningless!!



Some hints

- **Never start a particle from a surface.** You could get a geometry error even if the geometry is correct because FLUKA cannot determine the region.
- FLUKA tries to correct this by moving the particle a bit.
- However if it happens too often the run will stop.
- To avoid this the starting point of the beam particle must be slightly moved
- **Never eject a particle along a surface** (for the same reason)

Geometry Debugging

Possible types of errors

- Point belonging to more than one region
 - Program will not stop, no error message!
- Point not belonging to any region
 - The program will stop with an error message when a particle will enter to this region
- Problem space not enclosed by a black body region (Geometry = Problem space + Black hole)
- Precision errors
- Lattice replica \leftrightarrow basic cell mismatch (see lattice lecture)

Debugging Tools

- **GEOEND** card with the **DEBUG** option
- Error messages during simulation
- **PLOTGEOM** card (see manual, used automatically by Flair)
- FLAIR geoviewer
- Simplegeo

Debugging

GEOEND card activates the geometry debugger. Detects both undefined or multiple defined points in a selected X,Y,Z mesh

- Two cards are needed

First card

WHAT(1)=X_{max}
WHAT(4)=X_{min}
SDUM = DEBUG

WHAT(2)=Y_{max}
WHAT(5)=Y_{min}

WHAT(3)=Z_{max}
WHAT(6)=Z_{min}

- Second Card

WHAT(1)=Nx
SDUM = &

WHAT(2)=Ny

WHAT(3)=Nz

```
GEOEND Xmax Ymax Zmax Xmin Ymin Zmin DEBUG
GEOEND Nx Ny Nz &
```

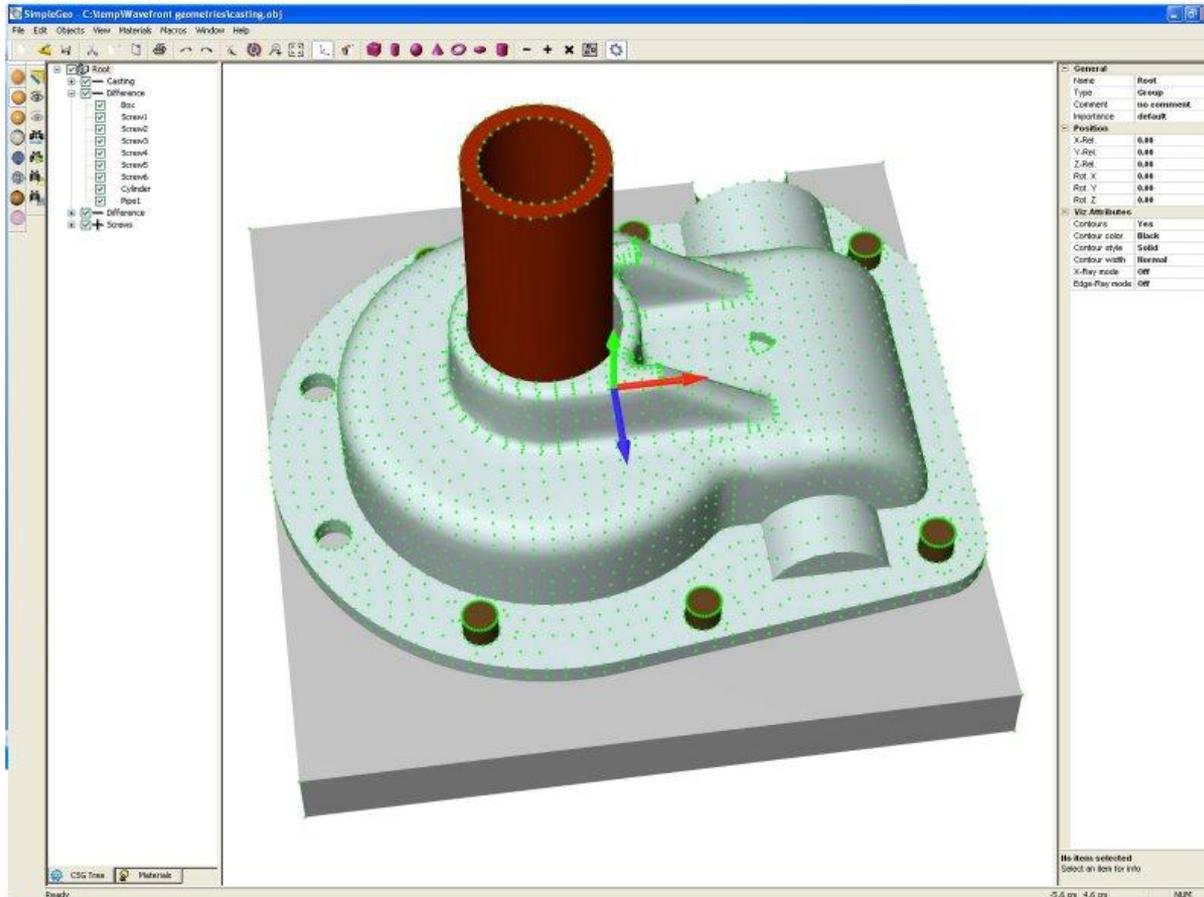
WARNING!

The program stops if too many errors are found.
A message will be issued on the output unit.

Debugging

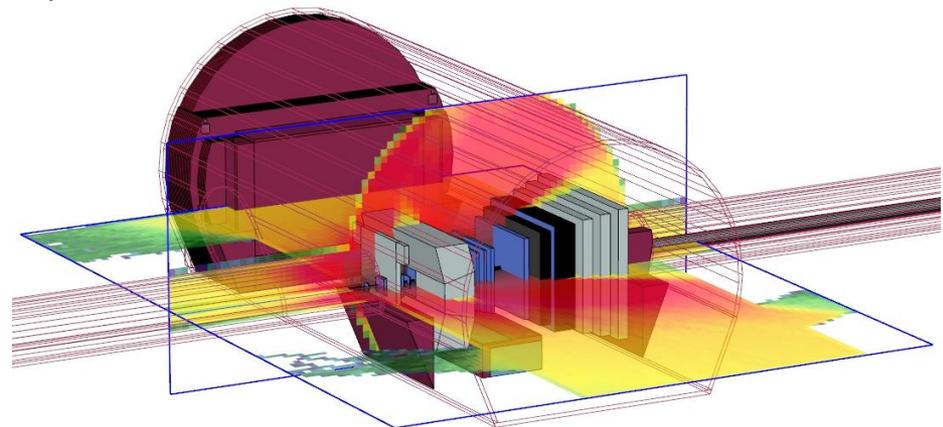
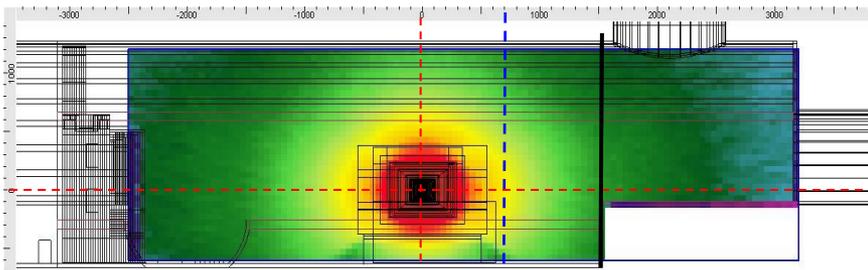
- If **no error** is found, no **.err** file will be created.
- **REMINDER**: If the debugger doesn't find any error it doesn't mean that the geometry is error free!
- One has to experiment, changing the **GEOEND** settings especially for the critical/complicated regions.
- Errors will be listed in the **.err** file in the form:
 - **** Lookdb: Geometry error found ****
**** The point: -637.623762 -244.554455 -96.039604 ****
 - **Point is contained in more than one region**
**** is contained in more than 1 region ****
**** (regions: 6 7) ****
 - **Not contained in any region**
**** is not contained in any region

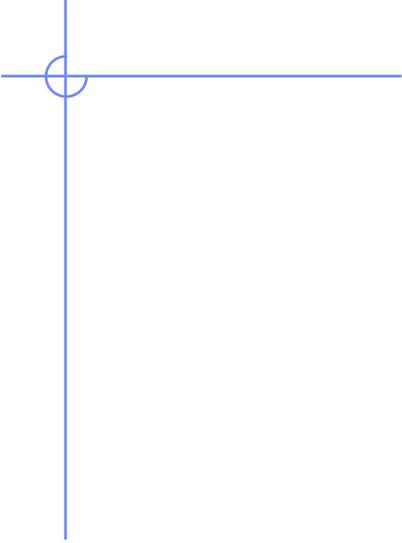
Auxiliary program: Simple Geo



Auxiliary program: *SimpleGeo*

- SimpleGeo is an interactive solid modeler which allows for flexible and easy creation of the models via drag & drop, as well as on-the-fly inspection
- Imports existing geometries for viewing
- Creating new geometries from scratch
- Export to various formats (FLUKA, MCNP, MCNPX)
- Download, Tutorials, etc.:
<http://theis.web.cern.ch/theis/simplegeo>
- Operating system: Windows only





Extra Slides

The geometry title card: IDBG

The second integer value **IDBG** can be used to modify the format with which body and region data are read:

IDBG

- 0 or 10** default fixed format for both bodies and regions
- 10** high-accuracy fixed format for bodies; default fixed region
- 100** high-accuracy fixed format for bodies; region fixed format
- allowing more than 10000 regions
- 100** default fixed format for bodies; region fixed format
- allowing more than 10000 regions

Any other value should be avoided.

The value of **IDBG** is irrelevant if free format has been requested (see the **GLOBAL** and **GEOBEGIN** commands)

The fixed geometry input format: regions

Each **body** is referred to by its sequential number in the body description table.

Input for each region extends on one or more cards, in a format which depends on the value of **IDBG** on the Geometry Title card.

If **IDBG = 0, 10** or **-10**, region input format is

(2X, **A3**, **I5**, **9(A2,I5)**)

If **IDBG = 100** or **-100**, region input format is

(2X, **A3**, **I5**, **7(A2,I7)**)

The fixed geometry input format: regions

The 3 characters in columns 3-5 are:

- on the first card of a given **region**, an **arbitrary non-blank string** chosen by the user (it can be used, together with optional comment cards, to help identifying the region or the region material).

Note: that **regions** are identified in FLUKA by an integer number corresponding to the order of their input definition. Therefore it can be useful (but not mandatory) to have that number appearing in the string.

For instance, if the 5th region is air, it could be labeled **AI5**.

On all **continuation cards**, columns 3-5 must be **blank**.

Parentheses Expansion

- 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:
 $A + A = A$
 $A - A = \emptyset$
 $\text{exp}A | \text{exp}A = \text{exp}A$

Use parentheses with care!!!

It can easily create an expansion with thousands of terms!

Parentheses Expansion

BLACK Region (36/108)

OR +Bottom1 +Bottom2 +EXTVOID
OR +Bottom1 +EXTVOID +Front2
OR +Bottom1 +EXTVOID +Right2
OR +Bottom1 +EXTVOID -Back2
OR +Bottom1 +EXTVOID -Left2
OR +Bottom1 +EXTVOID -Top2
OR +Bottom2 +EXTVOID +Front1
OR +Bottom2 +EXTVOID +Right1
OR +Bottom2 +EXTVOID -Back1
OR +Bottom2 +EXTVOID -Left1
OR +Bottom2 +EXTVOID -Top1
OR +EXTVOID +Front1 +Front2
OR +EXTVOID +Front1 +Right2
OR +EXTVOID +Front1 -Back2
OR +EXTVOID +Front1 -Left2
OR +EXTVOID +Front1 -Top2
OR +EXTVOID +Front2 +Right1
OR +EXTVOID +Front2 -Back1
OR +EXTVOID +Front2 -Left1
OR +EXTVOID +Front2 -Top1
OR +EXTVOID +Right1 +Right2
OR +EXTVOID +Right1 -Back2
OR +EXTVOID +Right1 -Left2
OR +EXTVOID +Right1 -Top2
OR +EXTVOID +Right2 -Back1
OR +EXTVOID +Right2 -Left1

OR +EXTVOID +Right2 -Top1
OR +EXTVOID -Back1 -Back2
OR +EXTVOID -Back1 -Left2
OR +EXTVOID -Back1 -Top2
OR +EXTVOID -Back2 -Left1
OR +EXTVOID -Back2 -Top1
OR +EXTVOID -Left1 -Left2
OR +EXTVOID -Left1 -Top2
OR +EXTVOID -Left2 -Top1
OR +EXTVOID -Top1 -Top2

CUBE1 Region (1/6)

+Back1 +Left1 +Top1 -Bottom1 -Front1 -Right1

CUBE2 Region (6/42)

OR +Back2 +Bottom1 +Left2 +Top2 -Bottom2 -Front2 -Right2
OR +Back2 +Front1 +Left2 +Top2 -Bottom2 -Front2 -Right2
OR +Back2 +Left2 +Right1 +Top2 -Bottom2 -Front2 -Right2
OR +Back2 +Left2 +Top2 -Back1 -Bottom2 -Front2 -Right2
OR +Back2 +Left2 +Top2 -Bottom2 -Front2 -Left1 -Right2
OR +Back2 +Left2 +Top2 -Bottom2 -Front2 -Right2 -Top1

Region Optimization

Geometrical optimization can drastically reduce the number of sub-zones.

Currently work is going on for 2 types of optimization to be implemented in FLUKA

1. Elimination of same type of planes (XYP, XZP, YZP) inside a sub-zone (product).
2. Optimization of sub-zones based on the **bounding boxes** of the bodies.
 - **Infinite objects** have an **infinite bounding** box on some of the dimensions ie. **XYP, ZCC** etc.
 - **PLANes** do not have a bounding box
 - For each sub-zone after the expansion, if the intersection of the bounding boxes is impossible the sub-zone is discarded

Expansion Optimization

BLACK Region (30/90)

OR +Bottom1 +EXTVOID
OR +Bottom1 +EXTVOID +Front2
OR +Bottom1 +EXTVOID +Right2
OR +Bottom1 +EXTVOID -Back2
OR +Bottom1 +EXTVOID -Left2
OR +Bottom2 +EXTVOID +Front1
OR +Bottom2 +EXTVOID +Right1
OR +Bottom2 +EXTVOID -Back1
OR +Bottom2 +EXTVOID -Left1
OR +EXTVOID +Front2
OR +EXTVOID +Front1 +Right2
OR +EXTVOID +Front1 -Left2
OR +EXTVOID +Front1 -Top2
OR +EXTVOID +Front2 +Right1
OR +EXTVOID +Front2 -Left1
OR +EXTVOID +Front2 -Top1
OR +EXTVOID +Right1
OR +EXTVOID +Right1 -Back2
OR +EXTVOID +Right1 -Top2
OR +EXTVOID +Right2 -Back1
OR +EXTVOID +Right2 -Top1
OR +EXTVOID -Back1
OR +EXTVOID -Back1 -Left2
OR +EXTVOID -Back1 -Top2
OR +EXTVOID -Back2 -Left1

OR +EXTVOID -Back2 -Top1
OR +EXTVOID -Left2
OR +EXTVOID -Left1 -Top2
OR +EXTVOID -Left2 -Top1
OR +EXTVOID -Top2

CUBE1 Region (1)

+Back1 +Left1 +Top1 -Bottom1 -Front1 -Right1

CUBE2 Region (3/18)

OR +Front1 +Left2 +Top2 -Bottom2 -Front2 -Right2
OR +Back2 +Left2 +Top2 -Bottom2 -Front2 -Left1
OR +Back2 +Left2 +Top2 -Front2 -Right2 -Top1

The optimization process can correct only for some of the "obvious" things