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 "infinite bodies", the possibility of using body and region names instead of numbers and getting rid of alignment rules, the availability of parentheses, 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 name based format. For backward compatibility there are also other formats. Name based format is not the default!
- One advantage of name based 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 advanced geometry 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 advanced geometry 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.

Name based format

Name based 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.

Name based 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 name based 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

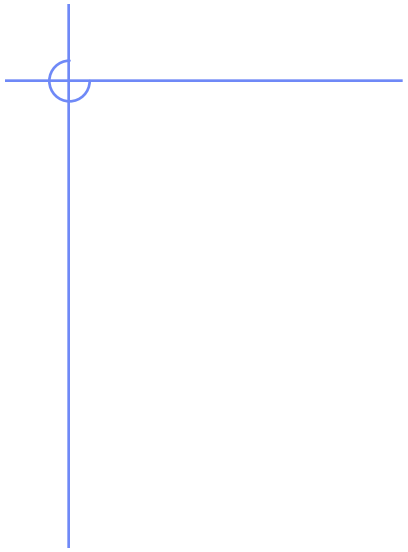
IVOPT = 0 → no region volumes will be input before GEOEND

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.

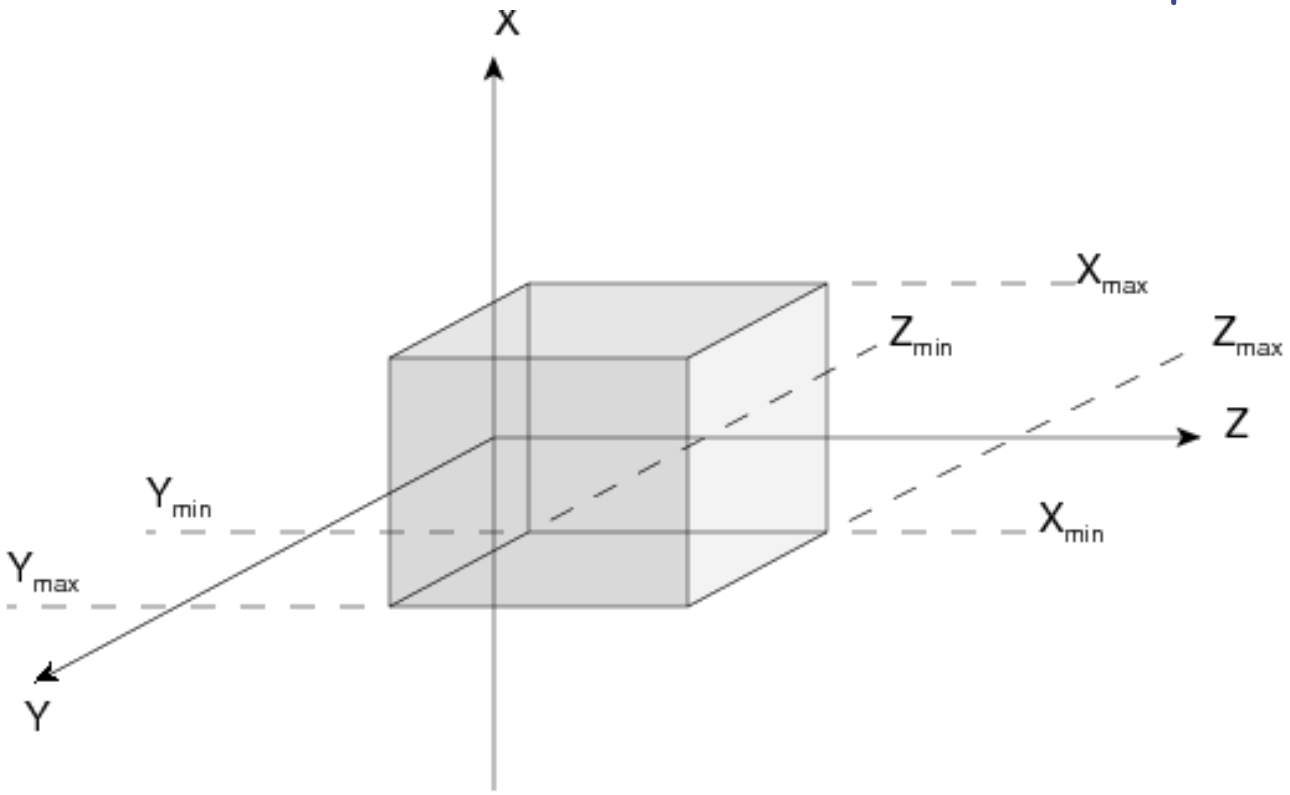
All values are in cm!

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

Name based format

```
RPP SmlBrick -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)
```

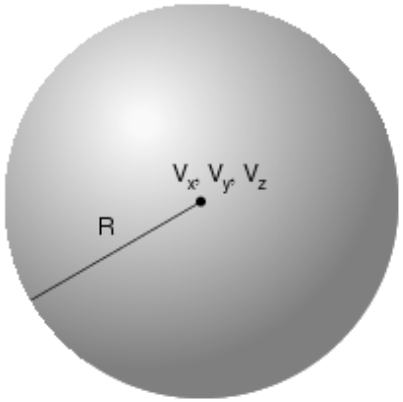
High-accuracy fixed format

```
*...+....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: Vx, Vy, Vz (coordinates of the centre), R (radius)

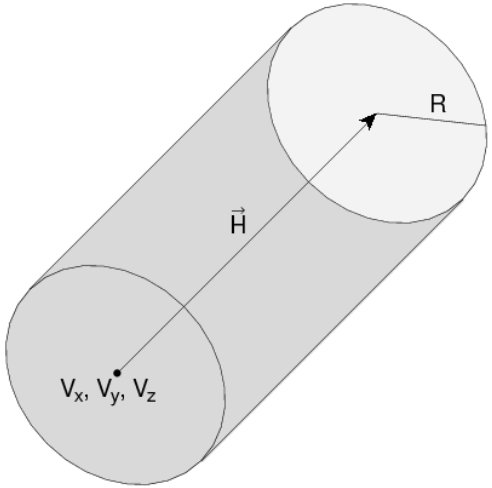


Right circular cylinder: RCC

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

Vx, Vy, Vz (centre of one face);

Hx, Hy, Hz (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.

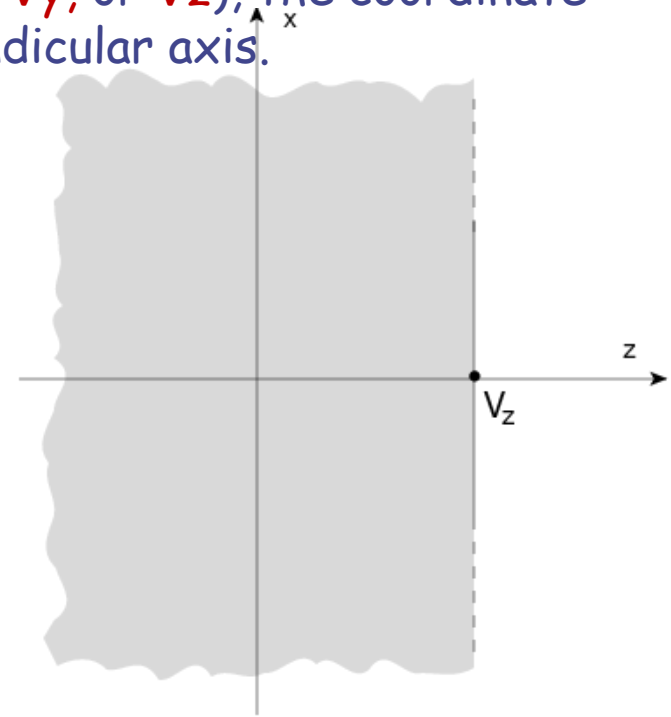
Three are delimited by planes perpendicular to the coordinate axes:

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

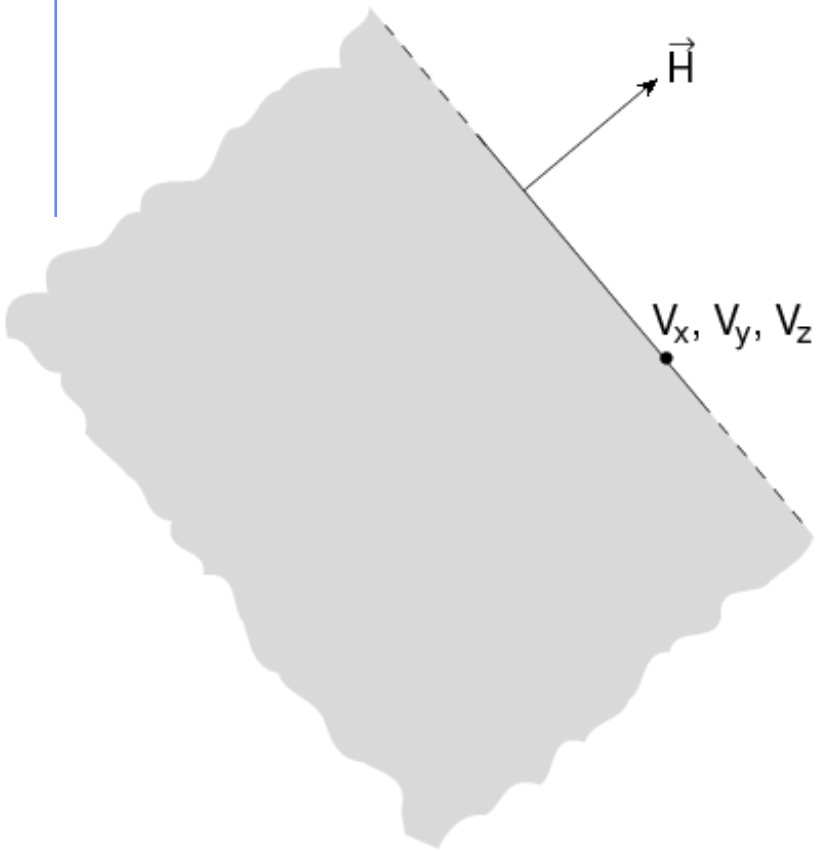
   All defined by a single number: Vx (resp. Vy, or Vz), the coordinate of the plane on the corresponding perpendicular axis.

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

$x < Vx$ (resp. $y < Vy$, or $z < Vz$)



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:

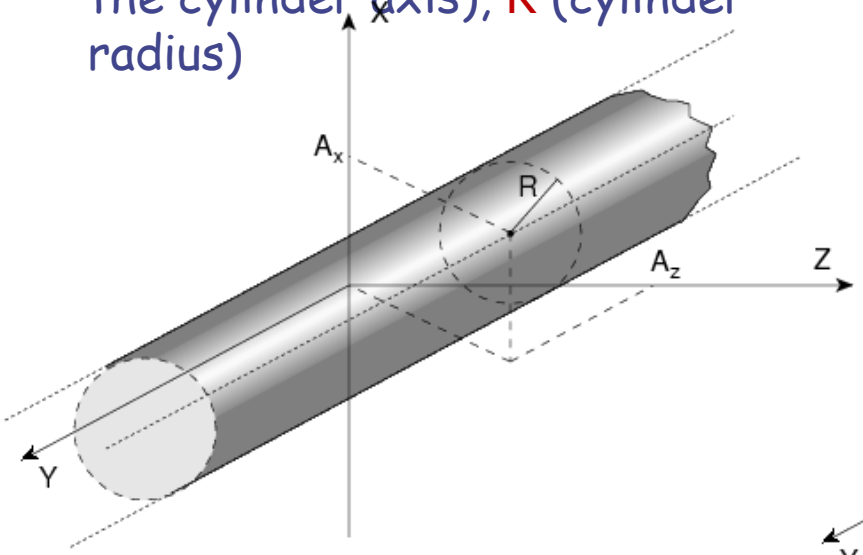
Hx, Hy, Hz (vector of arbitrary length ⊥ to the plane);

Vx, Vy, Vz (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

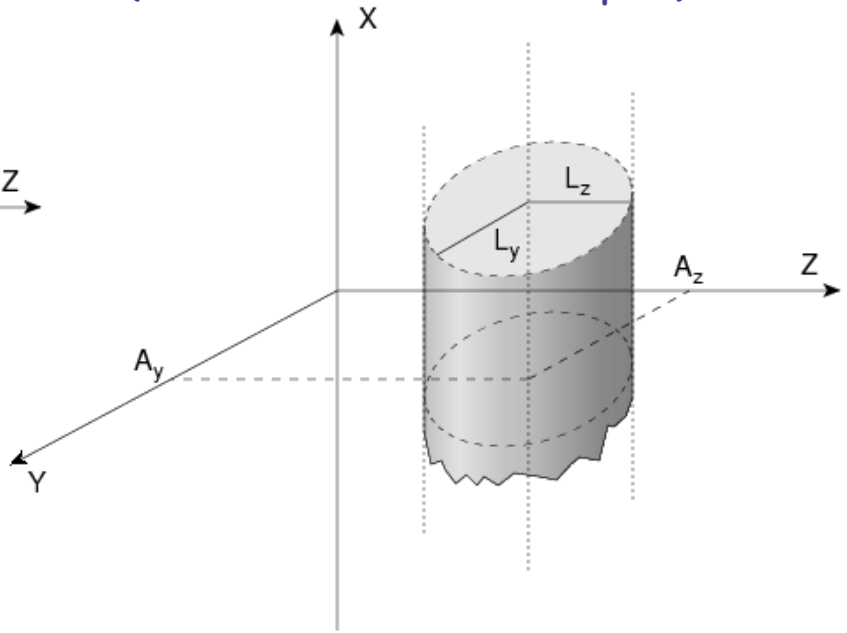
Infinite Circular Cylinder || to coordinate axis: **XCC, YCC, ZCC**

Each XCC (YCC, ZCC) is defined by 3 numbers: Ay, Az for XCC (Az, Ax for YCC, Ax, Ay 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: Ay, Az for XEC (Az, Ax for YEC, Ax, Ay for ZEC) (coordinates of the cylinder axis); Ly, Lz for XEC (Lz, Lx for YEC, Lx, Ly 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:

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

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
Name based format	|	–	+
Fixed format	OR	–	+
Mathematically	∪	−	∩

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 Normally you do not need to alter its default value (5). See next slide
- "boolean-zone-expression" is a sequence of one or more body names preceded by the operators + (intersection) or – (subtraction, or intersection with the complement). Several zone expressions can be combined by the union operator | . **Parentheses** are available in name based format (use with care!)

Regions

- *NAZ* is a rough guess for the number of zones which can be entered leaving the current region zones (5 by default). What in fact matters is the NAZ sum over all regions, defining the size of the *contiguity list*.

  At the beginning, to find the neighboring zones, the code searches over the whole geometry, but as the tracking proceeds, it learns the neighbors of each zone: if one is not yet in the contiguity list, it is added, making the calculation more and more efficient. When the above size limit is reached, the code prints a warning: GEOMETRY SEARCH ARRAY FULL. This is not lethal: the calculation continues but with a reduced efficiency. If the neighboring zone is not found in the contiguity list, the code will scan ALL zones.

If you have more than 1000 regions, you must issue a GLOBAL card putting in WHAT(1) a higher limit (not beyond 10000)

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.

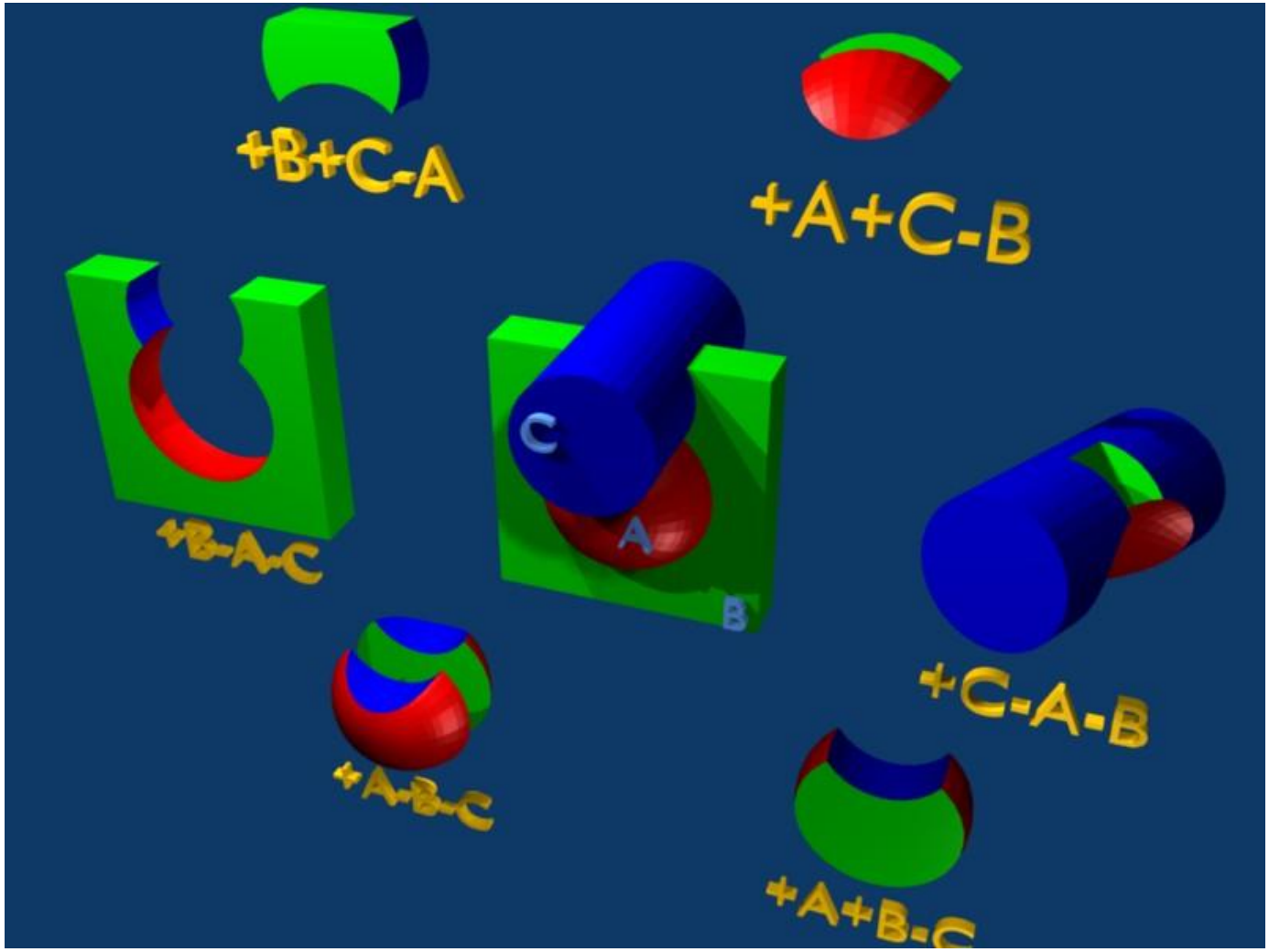
Meaning of + and - operators

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: normally 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 + and – operators

```
Target  5 +Outer +CutPlane
* (the above region is the part of space common to bodies Outer and
* CutPlane)

Regex  5  +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)
```

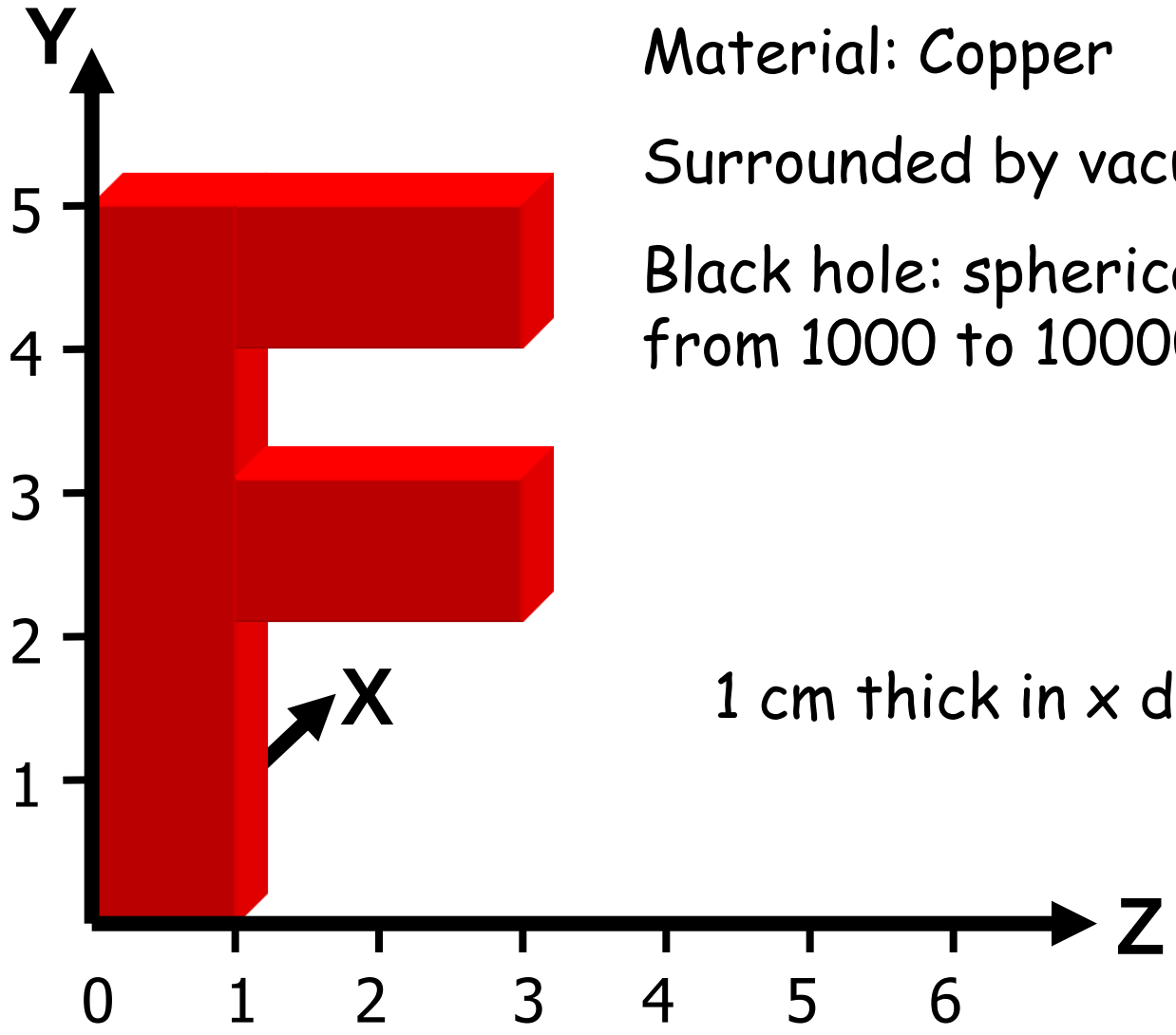
Meaning of the | (OR) operator

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 body intersections or subtractions as previously explained, and then the region is formed by the union of these zones.

Examples:

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

Geometry Example "F" shaped target



Material: Copper

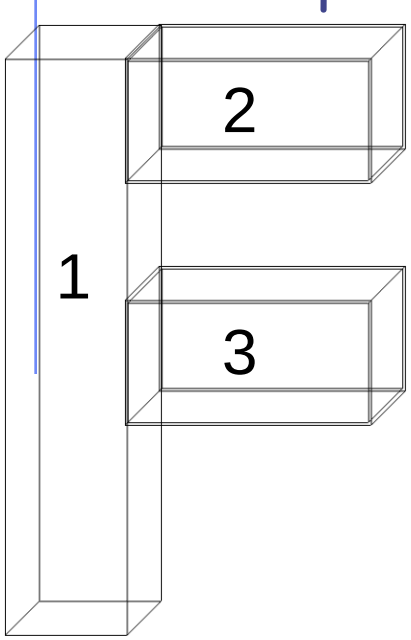
Surrounded by vacuum

Black hole: spherical shell
from 1000 to 10000 cm radius

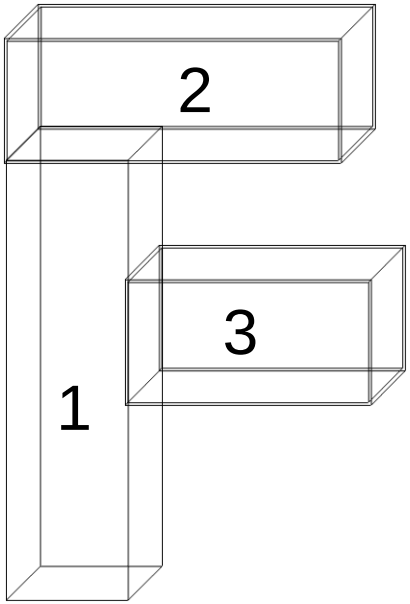
1 cm thick in x direction

Geometry example "F": Bodies

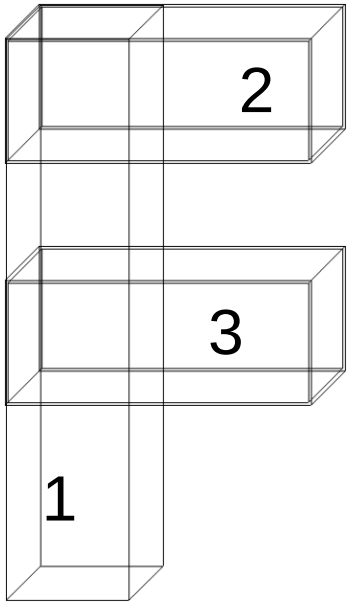
Several possibilities for bodies:



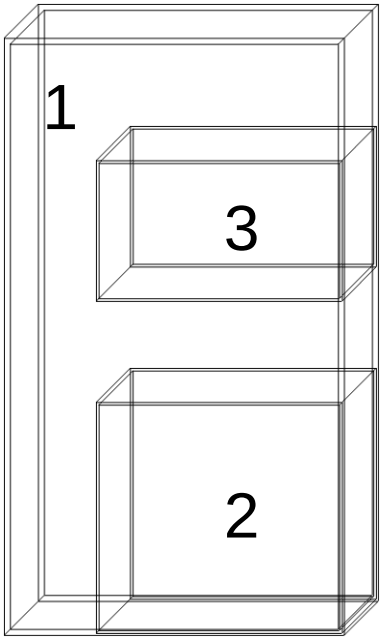
(A) 3 bodies



(B) 3 bodies



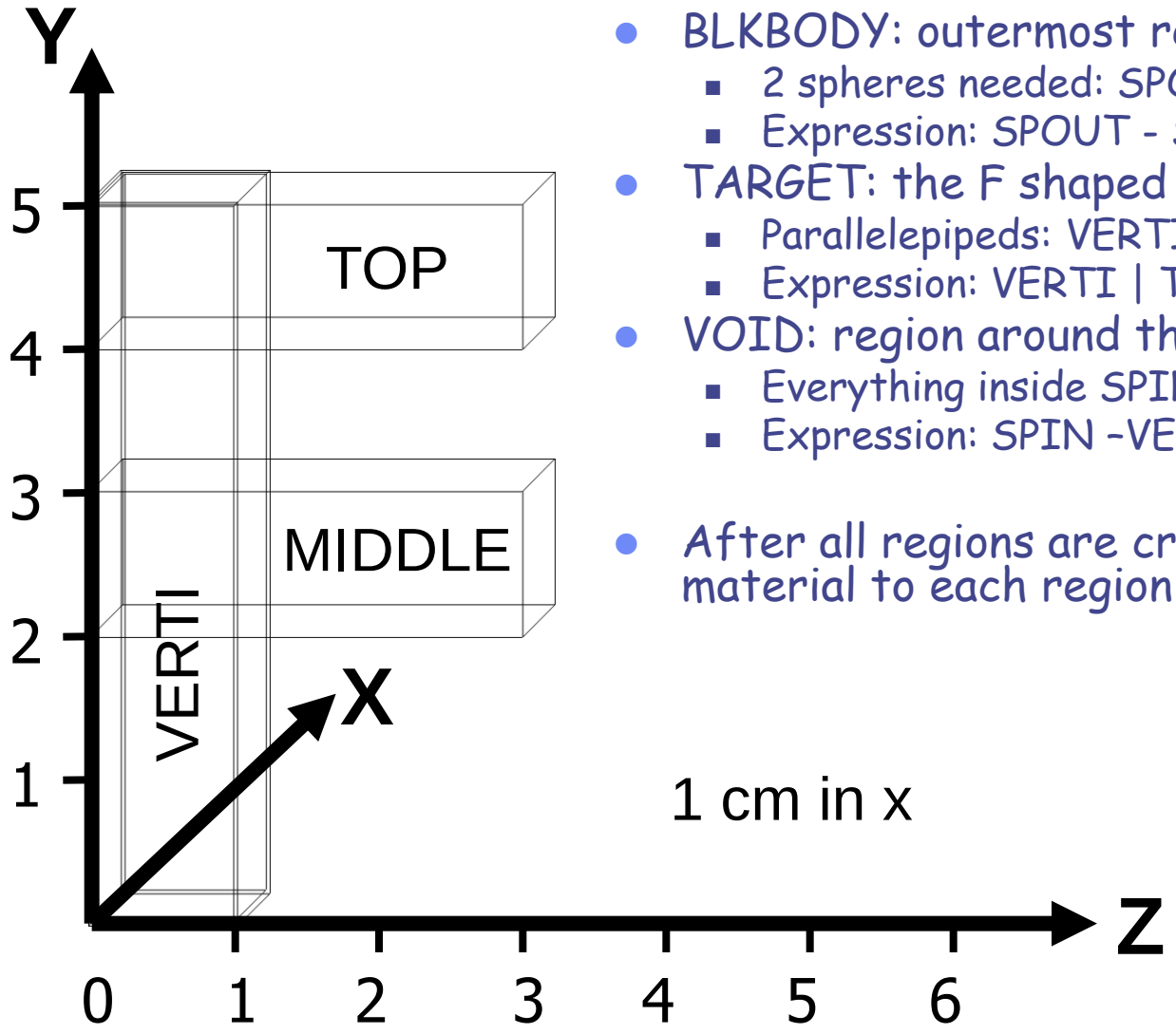
(C) overlapping



(D) subtraction

The 4 options are equivalent. We will use C.

Geometry example "F": REGIONS



- BLKBODY: 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
```

Help

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 ▼

..+....1....+....2....+....3....+....4....+....5....+....6....+....7..

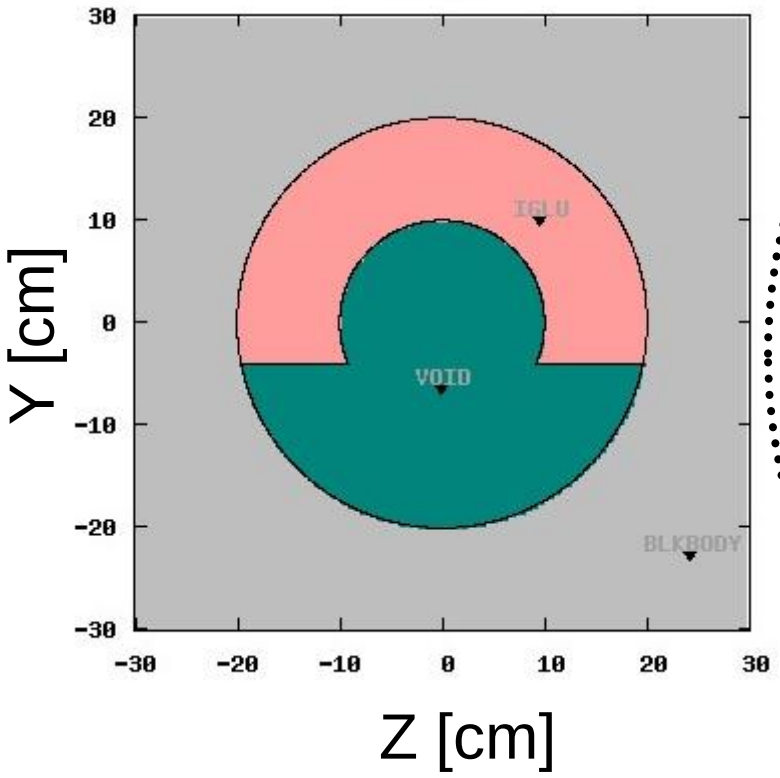
ASSIGNMA Mat: BLCKHOLE ▼ Reg: BLKBODY ▼ to Reg: ▼
Step: Field: ▼

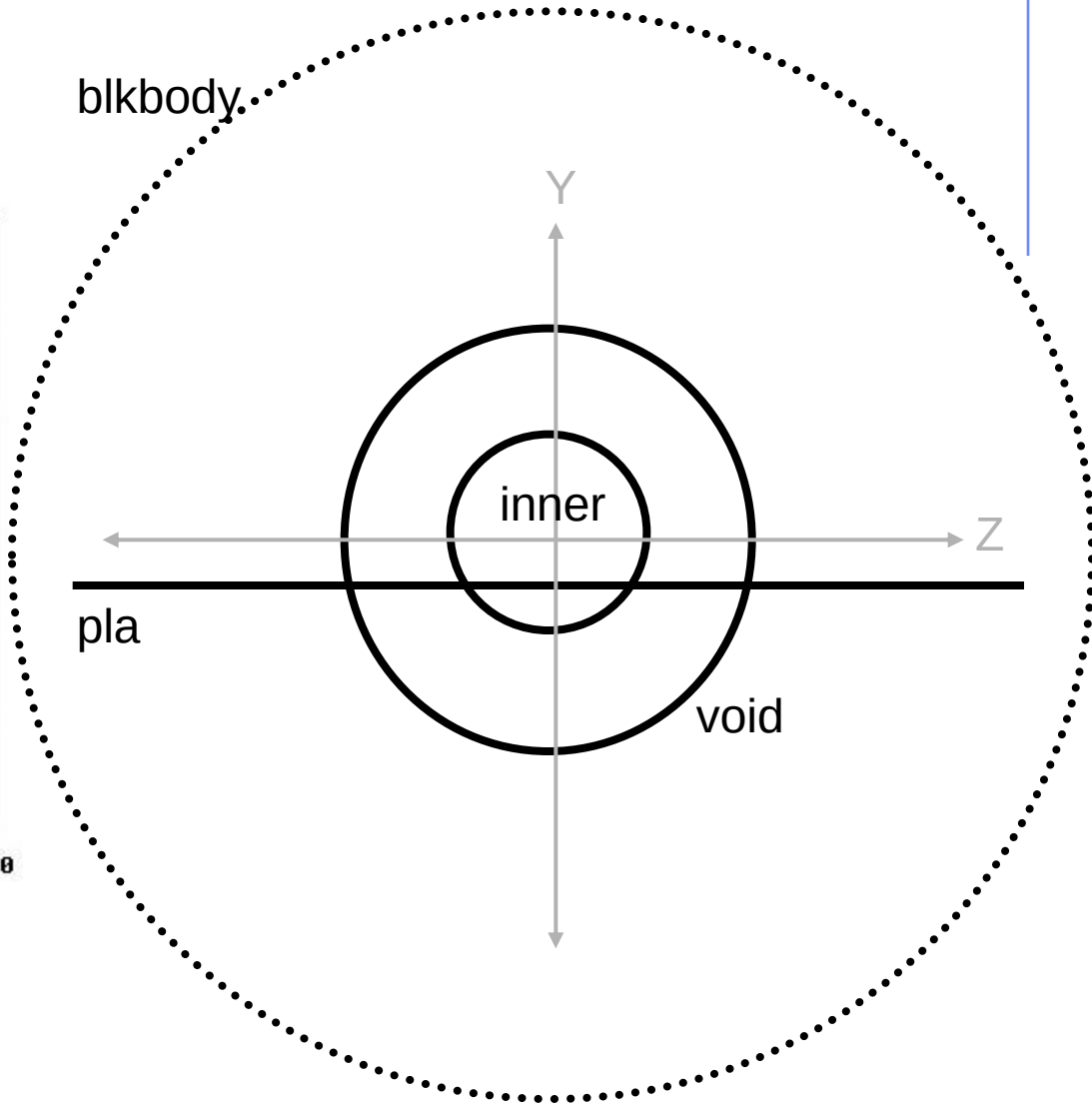
ASSIGNMA Mat: VACUUM ▼ Reg: VOID ▼ to Reg: ▼
Step: Field: ▼

ASSIGNMA Mat: COPPER ▼ Reg: TARGET ▼ to Reg: ▼
Step: Field: ▼

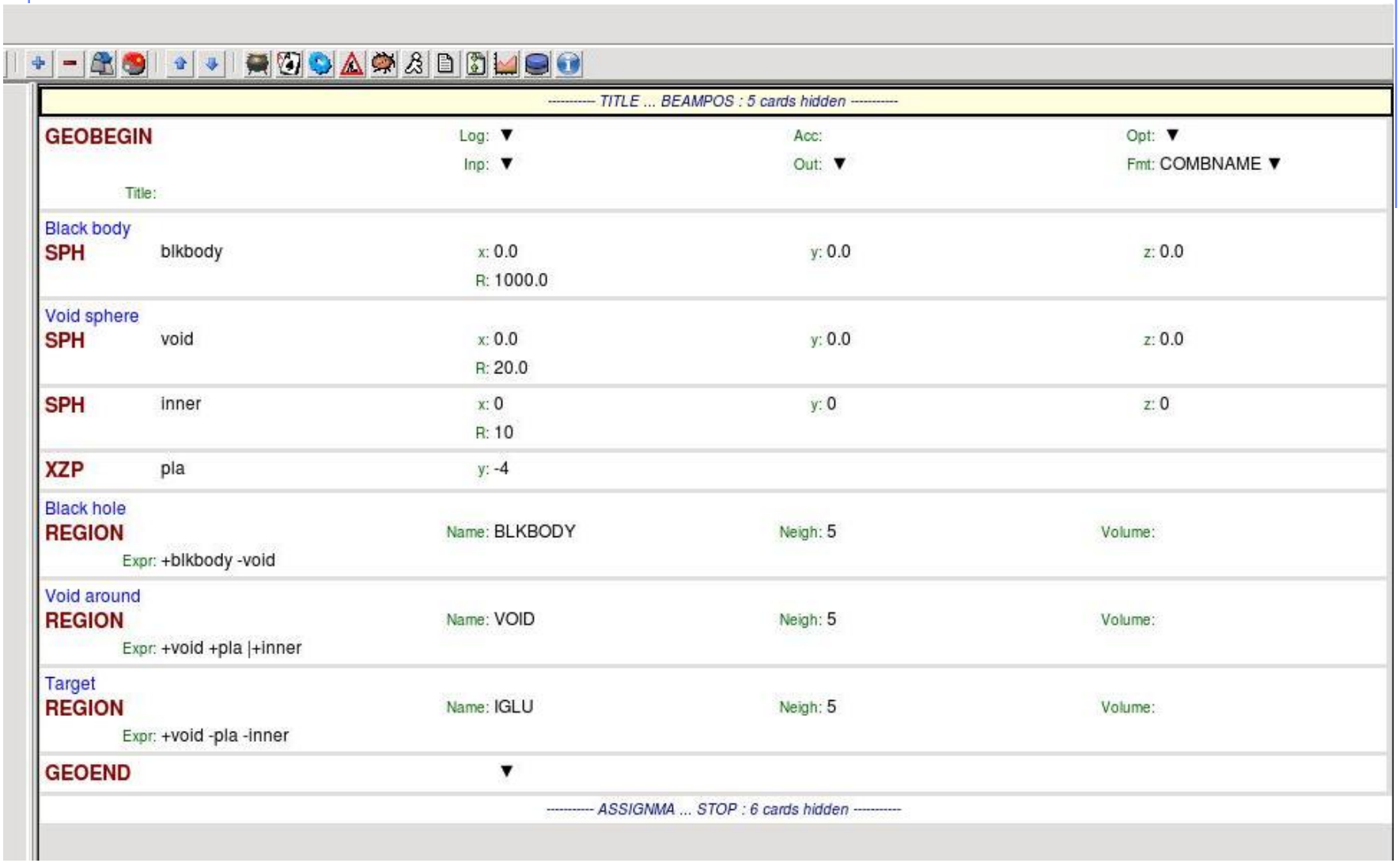
USRBIN Unit: 21 BIN ▼ Name: edep
Type: X-Y-Z ▼ Xmin: -10 Xmax: 10 NX: 100

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 y: 0.0 z: 0.0
R: 1000.0

Void sphere
SPH void x: 0.0 y: 0.0 z: 0.0
R: 20.0

SPH inner x: 0 y: 0 z: 0
R: 10

XZP pla y: -4

Black hole
REGION Name: BLKBODY Neigh: 5 Volume:
Expr: +blkbody -void

Void around
REGION Name: VOID Neigh: 5 Volume:
Expr: +void +pla |+inner

Target
REGION Name: IGLU Neigh: 5 Volume:
Expr: +void -pla -inner

GEOEND ▼

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

Important Notes

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

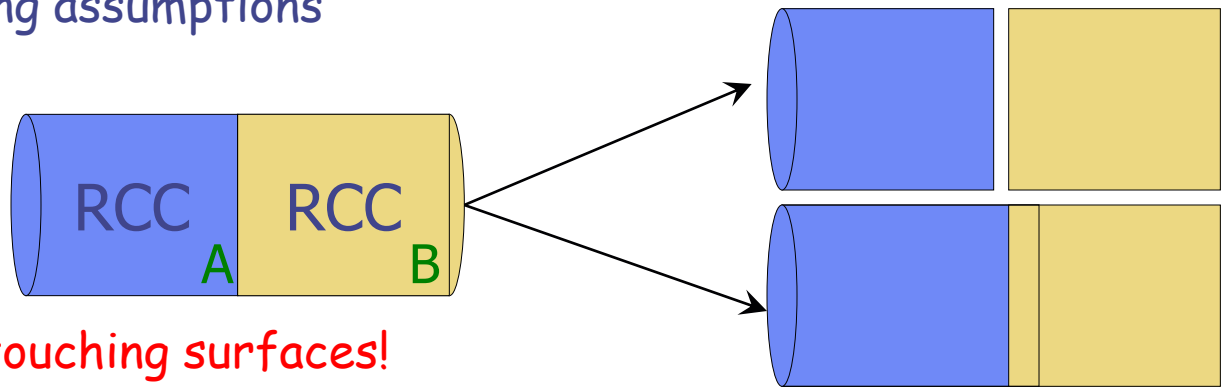
  PLA, RPP, SPH, XCC, XEC, XYP
  XZP, YCC, YEC, YZP, ZCC, ZEC, QUA

  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 name based format or the high-accuracy fixed format should always be used in these cases.

Precision Errors

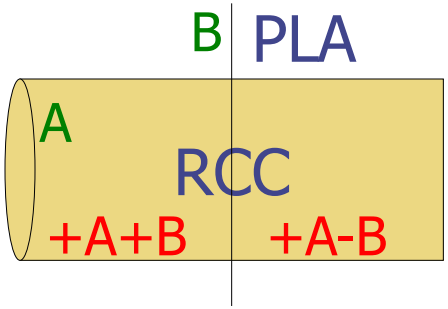
Modeling assumptions



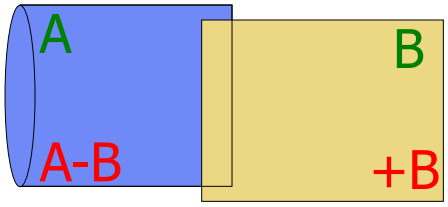
Avoid touching surfaces!

When floating point operations are involved.

- Use cutting surface B instead



- Or force partial overlap of bodies



Tracking accuracy

FLUKA uses systematically double precision arithmetic (i.e. 16 significant digits)

GEOBEGIN's WHAT(2)*10^{-6}cm is the *absolute accuracy (AA)* requested for tracking, applying to boundary identification.

The *relative accuracy (RA)* achievable in double precision is of the order of 10^{-14}-10^{-15}.

So *AA* should be larger than *RA*L*, being *L* the largest coordinate value in the problem world (excluding the outer blackhole shell containing it), i.e. the whole geometry size.

For very large (e.g., Earth) and very small geometries, you may need to increase or decrease, respectively, the WHAT(2) default value of 0.0001.

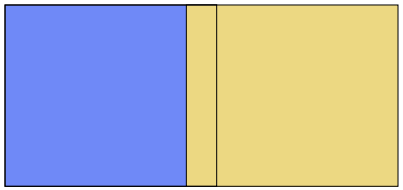
Geometry Errors and Debugging

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!!



Geometry Errors

Further types of errors

- Problem space not enclosed by a black body region
- Never eject a primary particle along a surface. You could get a geometry error even if the geometry is correct because FLUKA cannot determine the region
- Precision errors
- Lattice replica←→basic cell mismatch (see advanced geometry lecture)

Debugging Tools

- GEOEND card with the DEBUG option (*handled through a dedicated Flair frame*)
- Error messages during simulation in the .err file
- Geometry plotting by Flair (automatically invoking the PLOTGEOM card)
- **FLAIR geoviewer** (very powerful!)
- 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(2)=Y_{max} WHAT(3)=Z_{max}
  WHAT(4)=X_{min} WHAT(5)=Y_{min} WHAT(6)=Z_{min}
  SDUM = DEBUG

- Second Card

  WHAT(1)=Nx WHAT(2)=Ny WHAT(3)=Nz
  SDUM = &

```
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

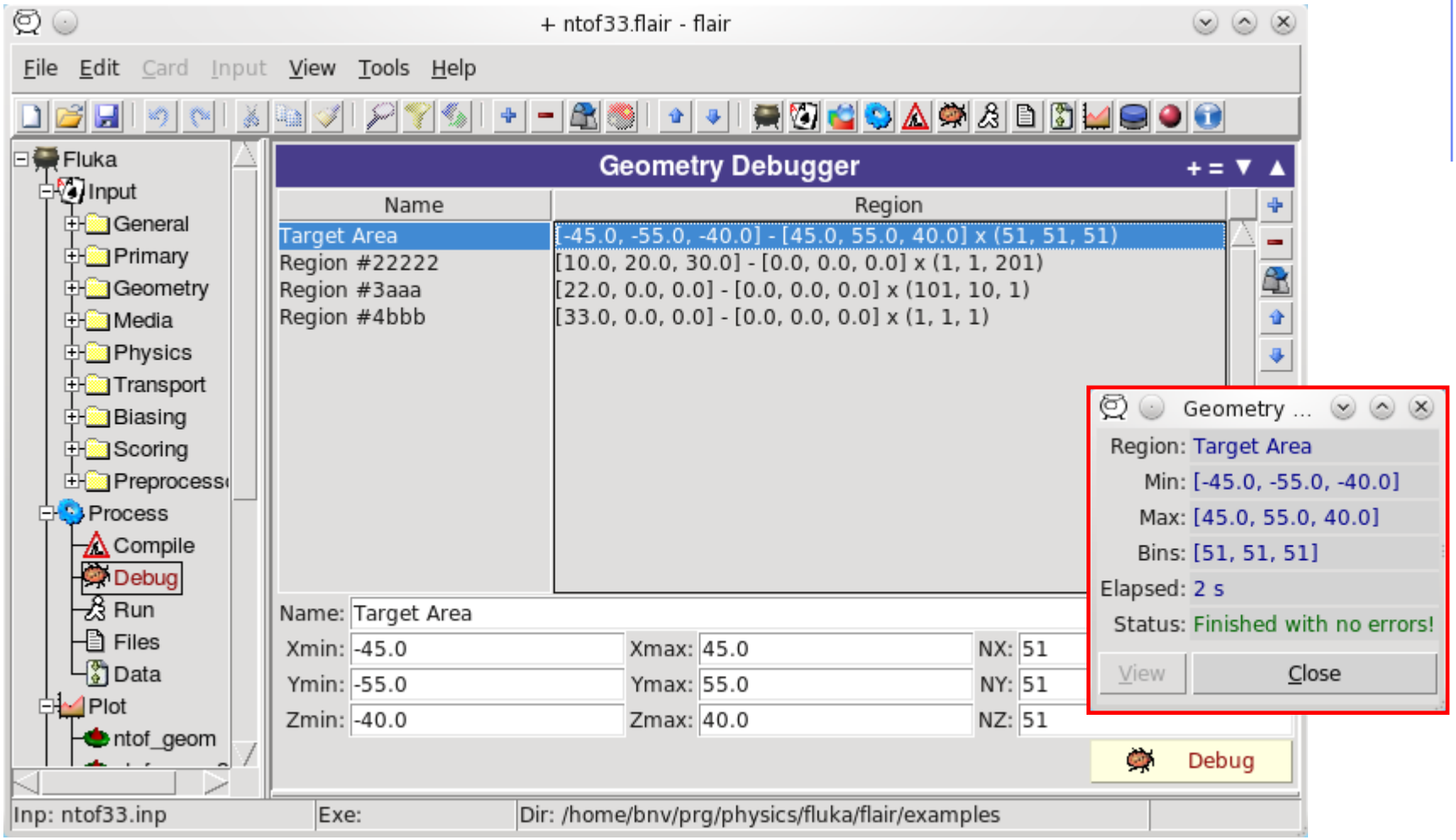
    **** is contained in more than 1 region ****
    **** (regions: 6 7) ****

  - Not contained in any region

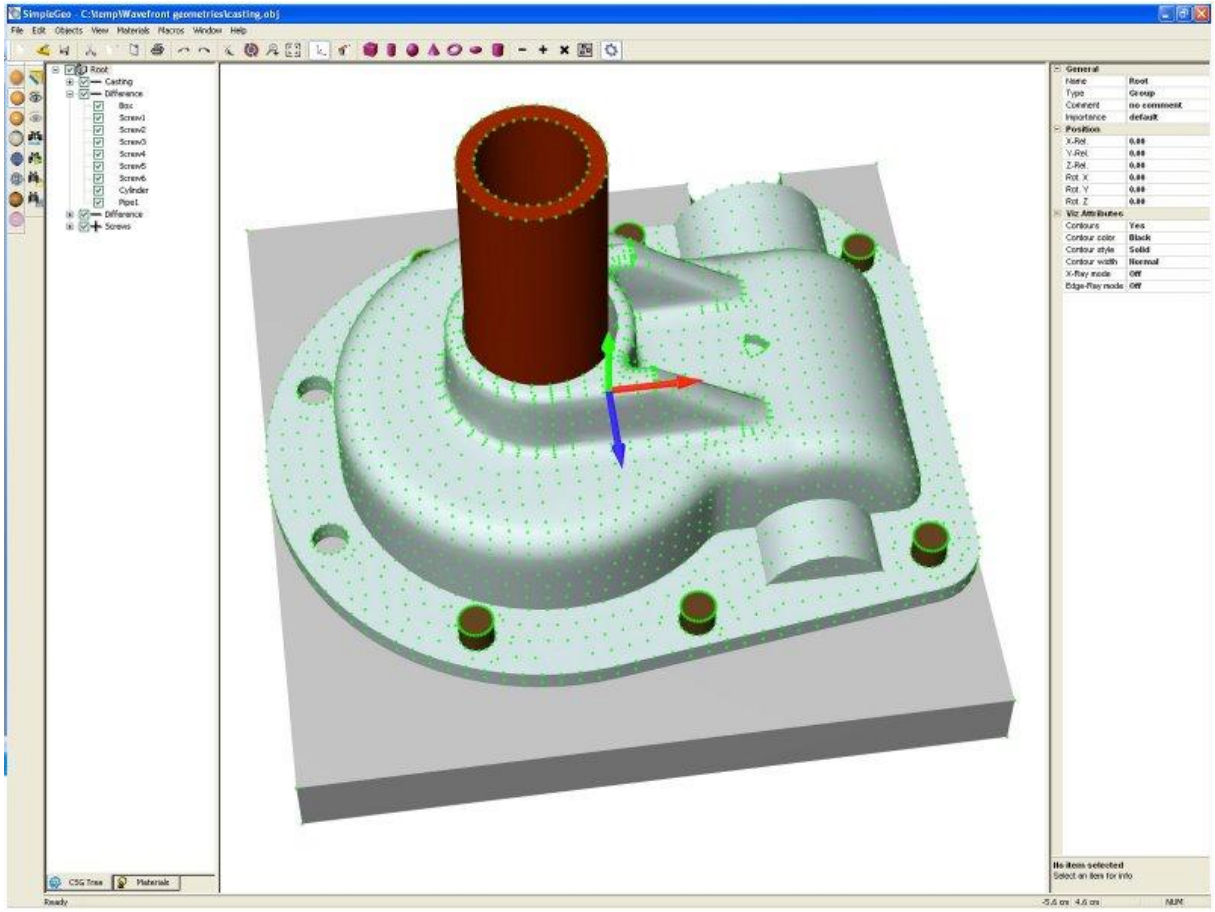
    **** is not contained in any region

Debugging

It can be easily run through Flair



Auxiliary program: *SimpleGeo*



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