FLUKA

蘭州大學
LANZHOU UNIVERSITY

FLUKA combinatorial geometry

23rd FLUKA Beginner's Course
Lanzhou University
Lanzhou, China

June 2–7, 2024

Introduction

- the Geometry is the description of the part of space where particles should be transported
- In real world, the geometry it is made of objects, occupying positions
  - a detector at 1m from a target
- In FLUKA, objects are called **Regions**
- Each region is identified by a number and a name[1]
- Like objects, each region must have a material assigned to it
- Other simulation parameters can be assigned "by region", for instance electron transport thresholds.
- Positions are defined in a cartesian system with distances in units of **cm**
- Regions are constructed from building blocks called **Bodies**
- through Boolean operations

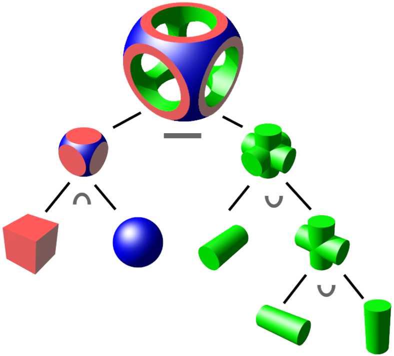
[1] In old versions, only the number

Principle of Combinatorial Geometry

Basic objects called bodies (such as cylinders, spheres, parallelepipeds, etc.) are combined to form more complex objects called regions

This combination is done using Boolean operations

Math	Operation	FLUKA	Meaning (A and B bodies)
∪	Union	A | B	All what is in A and all what is in B
∩	Intersection	A + B	What is in both A and B
—	Subtraction	A - B	What is in A and not in B

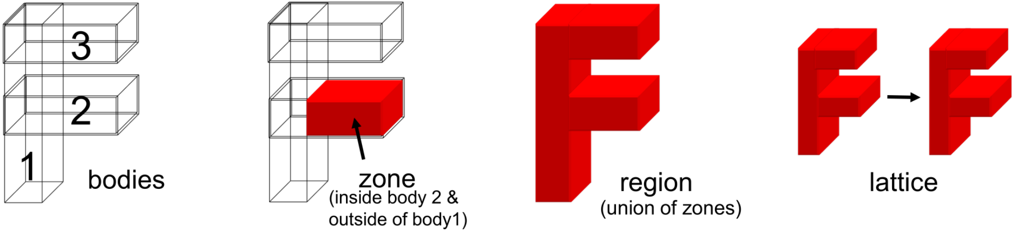


Fluka Combinatorial Geometry

Zones

Zones: sub-regions defined only via bodies intersection and subtraction

- The Fluka transport works with **Zones**, because they ensure that a trajectory can have only one entrance and one exit point. Would not be true for Unions, that can even be disjointed
- A region can, instead, be the Union of many zones
- Moreover, the use of Zones adds flexibility to the construction of regions



Fluka Combinatorial Geometry

Requirements

- The geometry must be contained within an outer border
- And surrounded by a region made by Blackhole : an all-absorbing material
- Every point within the geometry must belong to one region

  Not only objects, but also space in between objects must be assigned to one (or more) regions.

  No holes can be left.
- Every point within the geometry must belong to only one region:

  region overlaps are not allowed
- However, Zones belonging to the same region can overlap.

Combinatorial Geometry Input

CG input must respect the following structure:

GEOBEGIN card
name of geometry input file, optional, see later
name of geometry log file, optional, see later
VOXELS card (optional, see medical lecture)
Geometry title and options line (formatted, see later)
Body data
END card
Region data
END card
LATTICE card (optional, for repeated structures)
Region volumes (optional, see later)
GEOEND card

An asterisk (*) in the 1st column comments the line

No blank line are allowed No tabs are allowed

Geometry input format

The input file format for the geometry is different from the one adopted anywhere else in FLUKA (i.e. the number and length of the input fields is different)

Different formats can be used (due to backward compatibility

- **Fixed Format** (old format, not used nor described in this lecture)
  - Alignment is mandatory
  - Bodies and regions are identified by numbers (rather than names)
- **Name based Format (recommended)**
  - no alignment
  - Bodies and regions are identified by names
  - parentheses can be used to perform complex Boolean operations (advanced)
  - **It is not the default**, it is activated through the GLOBAL or GEOBEGIN card

Internally regions are identified by numbers

- In some of the output/error messages numbers are used
- Numbers and Names are printed in the .out file
- Name ←→ Number functions are available for user routine coding (advanced)

GEOBEGIN card

WHAT(1) >0 switches on online parenthesis expansion (see FLUKA manual)

WHAT(2) set accuracy parameter – use only if necessary ! See later

WHAT(3) logical unit for the geometry input. Default is from input file
Geometry input can be read from a separate file. If set should be >20.0

WHAT(4) logical unit for the geometry output; Default is standard output
Where the "interpreted" geometry echo is written. If set should be >20.0

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

WHAT(6) if >0 debugging printout (dangerous!! can be huge!)

SDUM COMBNAME or COMBINAT
COMBNAME selects name based format, COMBINAT fixed format
Default: COMBINAT (!)
Is overwritten by WHAT(5) of a possible GLOBAL card

GEOmetry Title card

This card has no keyword, it must follow the GEOBEGIN card (unless voxels are used) and its format is **(2I5, 10X, A60)**

The card gets three inputs: **IVLFLG, IDBG, TITLE**

IVLFLG (Input VoLumes FLAG) enables the normalization of the quantities scored in regions by the option SCORE

IVLFLG= 0 → no normalization applied (default)

IVLFLG= 3 → results divided by region volumes to be input by the user before GEOEND. See manual, seldom used.

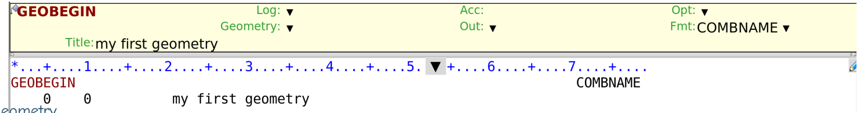
IDBG selects different kinds of geometry fixed format input: (not used in COMBNAME)

IDBG = 0 : default fixed format

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

TITLE Custom, optional. Must start after column 20

Flair merges this line with the GEOBEGIN card



GEOBEGIN Log: ▾ Acc: Opt: ▾
Geometry: ▾ Out: ▾ Fmt: COMBNAME ▾
Title: my first geometry

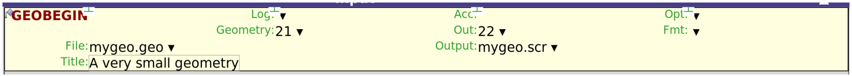
```
*...+....1....+....2....+....3....+....4....+....5. ▼ +....6....+....7....+....
GEOBEGIN                                                            COMBNAME
    0    0          my first geometry
```

GEOmetry from external file

The geometry input can be prepared in a file separated from the input file
Useful in case of shared geometries and/or common to different applications

- with the #include preprocessor directive or
- from the GEOBEGIN card:
  - set what(3) and what(4) to non zero (21 < what < 100, logical units)
  - Add two lines after the GEOBEGIN card with
    name of the geometry file
    name of the geometry log file

```
 GEOBEGIN          21   22          COMBNAME
mygeo.geo
mygeo.scr
 GEOEND
```



GEOBEGIN
Log: ▾
Geometry: 21 ▾
Acc:
Out: 22 ▾
Opt: ▾
Fmt: ▾
File: mygeo.geo ▾
Output: mygeo.scr ▾
Title: A very small geometry

Bodies

Bodies:

- basic convex objects (like a sphere or a cube)
- plus infinite planes (half-spaces)
- infinite cylinders (circular and elliptical)
- and generic quadric surfaces (surfaces described by 2nd degree equations)

Each body divides the space into two domains: inside and outside.
In the Fluka boolean language, inside is +, outside is -

Each body is identified by a three charachters code

Available Bodies

RPP:	**R**ectangular **P**arallele**P**iped
SPH:	SPHere
XYP, XZP, YZP:	Infinite half space delimited by a **P**lane $\perp$ to the z, y, x axis
PLA:	Generic infinite half-space, delimited by a **PLA**ne
XCC, YCC, ZCC:	Infinite **C**ircular **C**ylinder, parallel to coordinate axis
XEC, YEC, ZEC:	Infinite **E**lliptical **C**ylinder, parallel to coordinate axis
RCC:	**R**ight **C**ircular **C**ylinder
REC:	**R**ight **E**lliptical **C**ylinder
TRC:	**T**runcated **R**ight angle **C**one
ELL:	**ELL**ipsoid of revolution
QUA:	**QUA**dric
PYX, PYY, PYZ:	Pyramid parallel to a coordinate axis.

Deprecated bodies: ARB, RAW, WED, BOX : do not use!!!

Bodies: syntax

- The input for each body consists of:
  - the 3-letter code indicating the body type (RPP, ZCC...)
  - a unique "body name" (alphanumeric identifier, 8 character maximum, case sensitive)
  - a set of geometrical quantities defining the body (the number depends on the body type, see next slides)
- geometrical quantities can extend over as many lines as needed (Maximum 132 characters per line accepted)
- geometrical quantities have to be separated by one or more blanks, or by one of the separators , / ; :
- Each body divides the space in two. Inside and Outside
- The normal vector points towards Outside

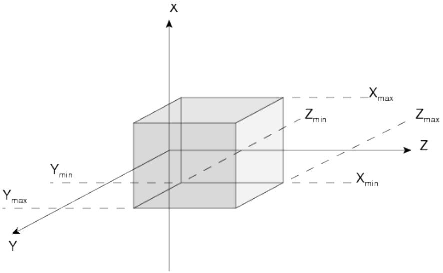
All values are in cm!

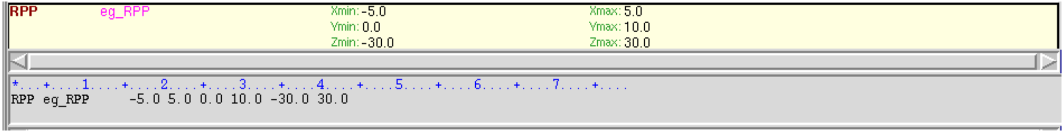
Rectangular Parallelepiped: RPP

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}





```
RPP        eg_RPP              Xmin: -5.0              Xmax: 5.0
                               Ymin: 0.0               Ymax: 10.0
                               Zmin: -30.0             Zmax: 30.0
```

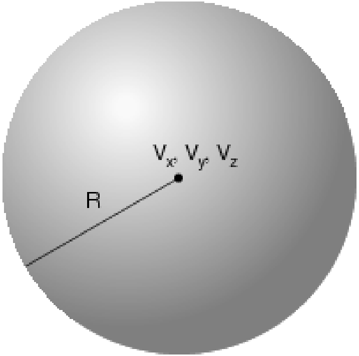
```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
RPP eg_RPP    -5.0 5.0 0.0 10.0 -30.0 30.0
```

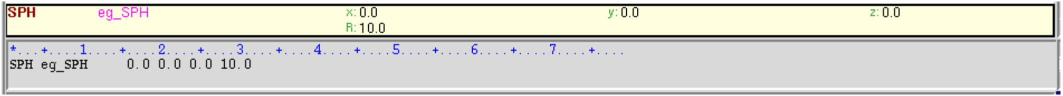
Sphere: SPH

A SPH is defined by 4 numbers:

Vx, Vy, Vz :coordinates of the centre

R: radius





```
SPH          eg_SPH               x:0.0                    y:0.0                    z:0.0
                                  R:10.0
```

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
SPH eg_SPH     0.0 0.0 0.0 10.0
```

Infinite half-space parallel to coordinate axis

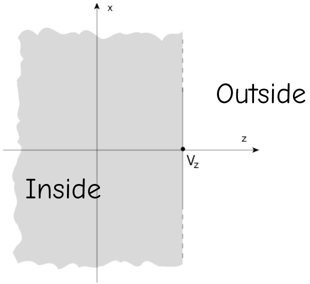
There are 4 kinds of infinite half-spaces

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

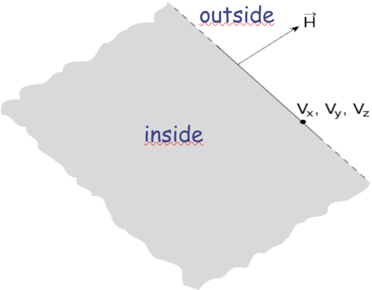
All defined by a single number:
Vx (resp. Vy, or Vz),
coordinate of the plane on the corresponding axis.
Points for which: **$x < V_x$** (resp. $y < V_y$, or $z < V_z$)
are **inside the body**

```
    XYP     MyXYP    10.0
```



Arbitrarily oriented infinite half-space: PLA

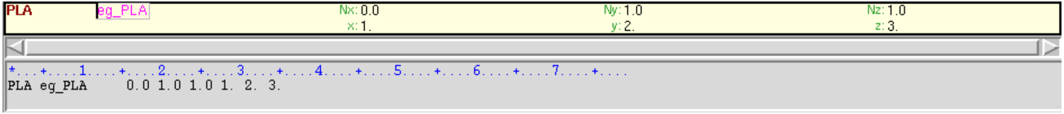
A PLA defines the infinite half space delimited by a generic plane



A PLA is defined by 6 numbers:

Hx, Hy, Hz :vector ⊥ to the plane, arbitrary length;
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 normal points outside**).



PLA eg_PLA Nx: 0.0 Ny: 1.0 Nz: 1.0

x: 1. y: 2. z: 3.

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
PLA eg_PLA     0.0 1.0 1.0 1.  2.  3.
```

Right circular cylinder: RCC

An RCC can have any orientation in space

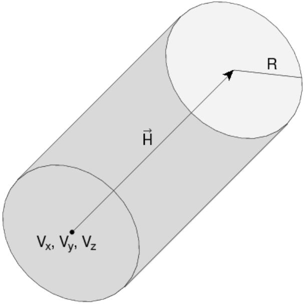
It is limited by a cylindrical surface and two plane faces ⊥ to its axis.

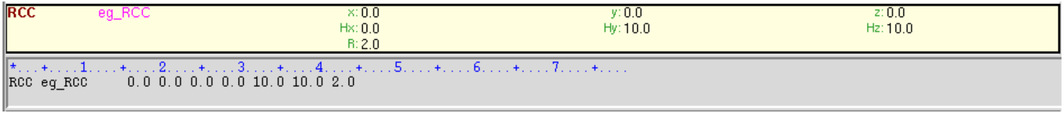
Each RCC is defined by 7 numbers:

Vx, Vy, Vz : centre of one face

Hx, Hy, Hz : vector corresponding to the cylinder height, pointing toward the other face

R : cylinder radius.



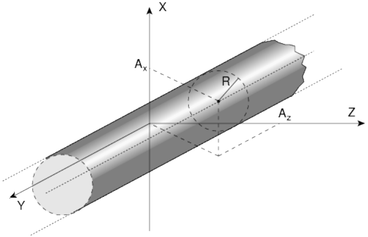


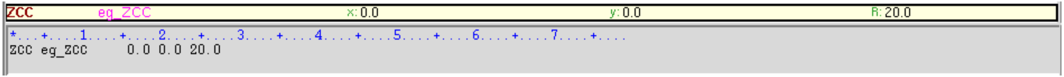
```
RCC          eg_RCC          x: 0.0          y: 0.0          z: 0.0
                            Hx: 0.0         Hy: 10.0        Hz: 10.0
                             R: 2.0
```

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
RCC eg_RCC     0.0 0.0 0.0 0.0 10.0 10.0 2.0
```

Infinite cylinders parallel to one 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.




ZCC eg_ZCC x: 0.0 y: 0.0 R: 20.0

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
ZCC eg_ZCC       0.0 0.0 20.0
```

Other bodies

Are less frequently used, their description is in the manual

Regions

Reminder:

Regions are composed by one or more Zones

Joined together by Union sign: (|)

A Zone is obtained by Intersection(+) or Subtraction (-) of bodies

Regions but must be of homogeneous material composition.

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

Region format

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
  - alphanumeric identifier,
  - 8 character maximum,
  - case sensitive.
  - Must start by an alphabetical character
- NAZ See next slide
- boolean-zone-expression See next slides

Region
NAZ

NAZ stands for Number of Adjacent Zones:

- When tracking, the code discovers and keeps memory of the zones a particle can enter when leaving the current one,
- So that it saves time by checking first the zones in this "neighbor list" , before scanning the others.
- this list has a global dimension for the whole set of regions, cannot be ∞
- Dimension is calculated at initialization as sum of the NAZs for all the regions
- When the the list is full the code prints a warning: GEOMETRY SEARCH ARRAY FULL. This is not lethal: the calculation continues but with a reduced efficiency

NAZ

NAZ is a rough estimate of the number of zones a particle can enter when leaving the current region

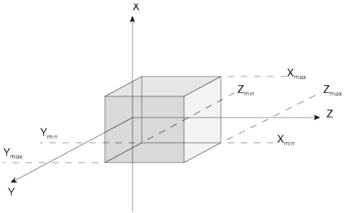
It is only an "efficiency" parameter

If you do not know.. 5 is usually a good candidate

Zone expressions

- a REGION is the union (|) of one or more ZONES
- a ZONE is defined by intersections (+) and/or subtraction (-) of bodies

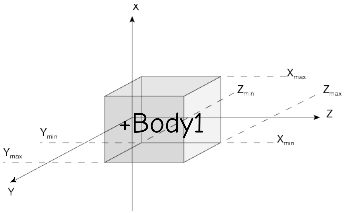
```
RPP     Body1   -5.0    5.0    0.    10.    -30.    30.
 END
```



Zone expressions

- a REGION is the union (|) of one or more ZONES
- a ZONE is defined by intersections (+) and/or subtraction (-) of bodies
  - **+body**: only the inner part of the body can belong to the zone (means that the zone being described is fully contained inside this body)

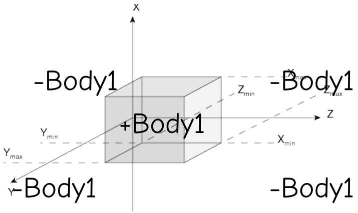
```
RPP     Body1   -5.0    5.0    0.    10.    -30.    30.
 END
InRPP    5    +Body1    a single-zone region
```



Zone expressions

- a REGION is the union (|) of one or more ZONES
- a ZONE is defined by intersections (+) and/or subtraction (-) of bodies
  - **+body**: only the inner part of the body can belong to the zone (means that the zone being described is fully contained inside this body)
  - **-body** only the outside of the body can belong to the zone (means that the zone being described is fully outside this body)

```
RPP     Body1   -5.0    5.0    0.    10.    -30.    30.
 END
InRPP     5    +Body1   a single-zone region
OuRPP     5    -Body1                is this ok?
```



Zone expressions

- a REGION is the union (|) of one or more ZONES
- a ZONE is defined by intersections (+) and/or subtraction (-) of bodies
  - **+body**: only the inner part of the body can belong to the zone (means that the zone being described is fully contained inside this body)
  - **-body** only the outside of the body can belong to the zone (means that the zone being described is fully outside this body)
  - Zones must be finite: there must always be at least a + sign. (otherwise the tracking would never end…)

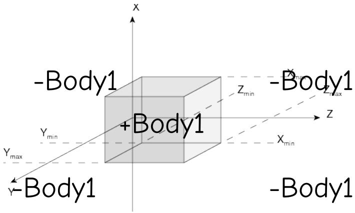
RPP   Body1   -5.0   5.0   0.   10.   -30.   30.

END

InRPP   5   +Body1   *a single-zone region*

OuRPP   5   -Body1   *is this ok?*

~~OuRPP   5   -Body1~~   no, missing +:

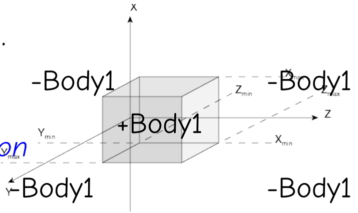


Zone expressions

- a REGION is the union (|) of one or more ZONES
- a ZONE is defined by intersections (+) and/or subtraction (-) of bodies
  - **+body**: only the inner part of the body can belong to the zone (means that the zone being described is fully contained inside this body)
  - **-body** only the outside of the body can belong to the zone (means that the zone being described is fully outside this body)

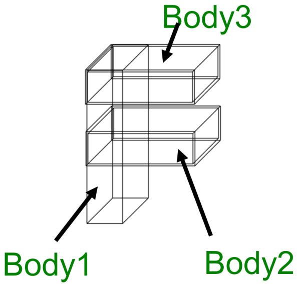
```
RPP       Body1   -5.0    5.0        0.    10.    -30.    30.
SPH     Outside       0.      0.   1000.
END
InRPP       5              +Body1
  OuRPP     5     -Body1 +Outside
```

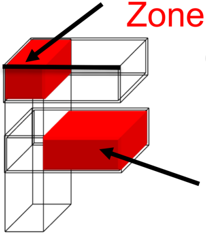
a single-zone region



Zone expressions

playing with three RPP's





Zone1 = +Body1 +Body3

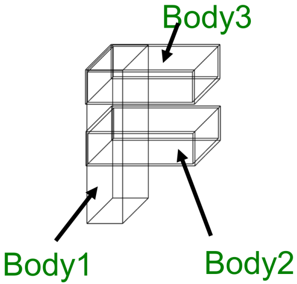
(Zone1 is the space common to Body1 and Body3)

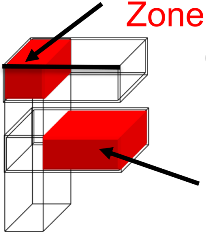
Zone2 = +Body2 –Body1

(Zone2 is the space inside of Body2 but outside of Body1)

Zone expressions

playing with three RPP's define the red and the white region





Zone1 = +Body1 +Body3

(Zone1 is the space common to Body1 and Body3)

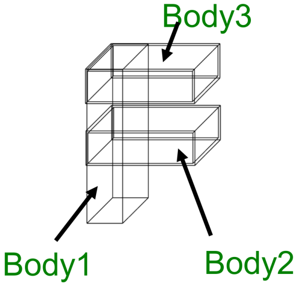
Zone2 = +Body2 –Body1

(Zone2 is the space inside of Body2 but outside of Body1)

```
RedF    5  | +Body1 +Body3  | +Body2 -Body1
WhiteF  5  | +Body3 -Body1  | +Body1 -Body3
```

Zone expressions

playing with three RPP's All as a single region



ALLF 5 | +Body1 | +Body2 | +Body3

Zones of the same region can overlap

Parentheses

In name based format one can also use parentheses to form more complex Boolean operations

Advanced topic, Not described in this course

Special Region: The Black Hole

All particles entering a black-hole are absorbed (they vanish)

FLUKA geometry MUST be embedded into a BLCKHOLE region

to avoid tracking particles to infinity

The outer surface of the BLCKHOLE region must be closed.

Further black-hole regions can be defined by the user if desired

BLCKHOLE region: has material BLCKHOLE assigned to it

Black Hole and empty space

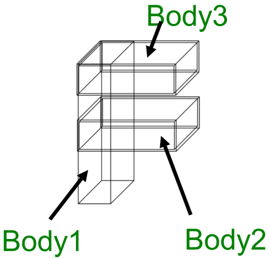
How to define the Black Hole? Remember, it must be a REGION, not a body

Usually as the region contained between two large spheres, enclosing your geometry and a bit more

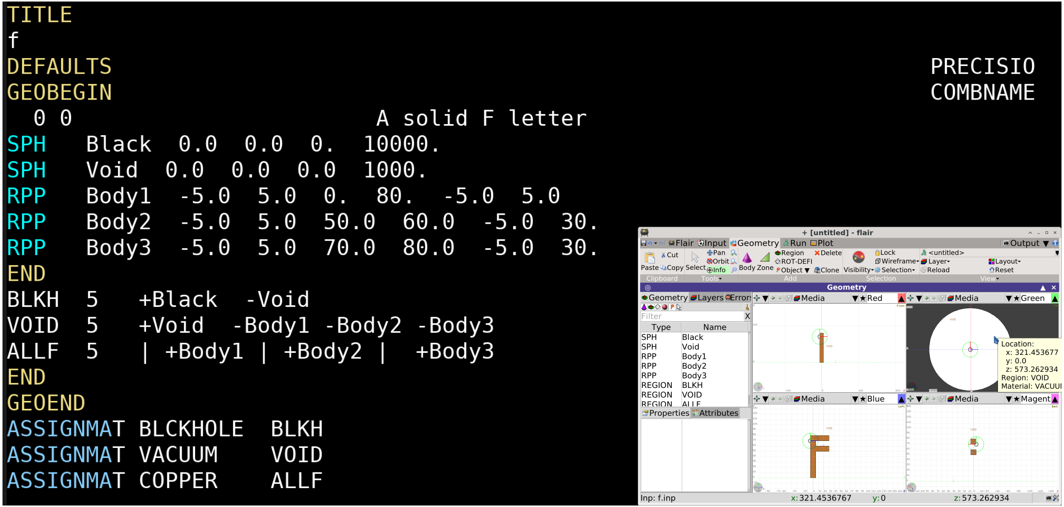
All the space points at the interior of the Black Hole must belong to some region

For really empty space, one can assign material VACUUM to it

```
SPH     Black    0.0    0.0      0.   10000.
SPH      Void    0.0    0.0     0.0    1000.
RPP     Body1   -5.0    5.0      0.     100.   -5.0    5.0
RPP     Body2   -5.0    5.0    70.0     80.0   -5.0    30.
RPP     Body3   -5.0    5.0    90.0    100.0   -5.0    30.
END
BLKH    5                        +Black -Void
VOID    5    +Void -Body1 -Body2 -Body3
ALLF    5        | +Body1 | +Body2 | +Body3
END
```



f.inp



```
TITLE
f
DEFAULTS                                                                PRECISIO
GEOBEGIN                                                                COMBNAME
  0 0                           A solid F letter
SPH    Black   0.0   0.0   0.   10000.
SPH    Void   0.0   0.0   0.0   1000.
RPP    Body1  -5.0   5.0   0.   80.   -5.0   5.0
RPP    Body2  -5.0   5.0   50.0   60.0   -5.0   30.
RPP    Body3  -5.0   5.0   70.0   80.0   -5.0   30.
END
BLKH   5    +Black  -Void
VOID   5    +Void  -Body1 -Body2 -Body3
ALLF   5    | +Body1 | +Body2 |  +Body3
END
GEOEND
ASSIGNMAT BLCKHOLE  BLKH
ASSIGNMAT VACUUM    VOID
ASSIGNMAT COPPER    ALLF
```

+ [untitled] - flair

Type	Name
SPH	Black
SPH	Void
RPP	Body1
RPP	Body2
RPP	Body3
REGION	BLKH
REGION	VOID
REGION	ALLF

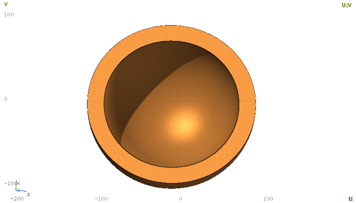
Location:
x: 321.453677
y: 0.0
z: 573.262934
Region: VOID
Material: VACUUI

Inp: f.inp x: 321.4536767 y: 0 z: 573.262934

Example with a plane

“Infinite” bodies, planes and cylinders, are really useful and allow for fast tracking. They are strongly recommended. Here a simple exemple of use.

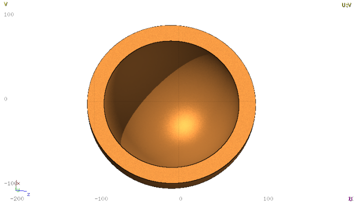
Imagine you want to model an hemispherical cup



Example with a plane

“Infinite” bodies, planes and cylinders, are really useful and allow for fast tracking. They are strongly recommended. Here a simple exemple of use.

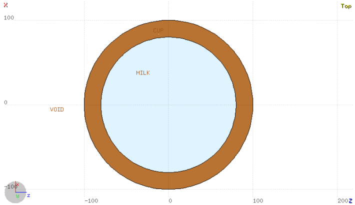
Imagine you want to model an hemispherical cup
That is, one half of the space in between two spheres.



Example with a plane

“Infinite” bodies, planes and cylinders, are really useful and allow for fast tracking. They are strongly recommended. Here a simple exemple of use.

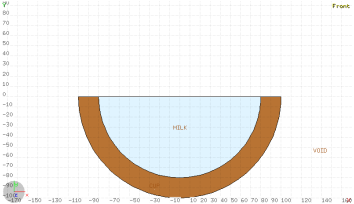
Imagine you want to model an hemispherical cup
That is, one half of the space in between two spheres.
So, we take the space between two spheres



Example with a plane

“Infinite” bodies, planes and cylinders, are really useful and allow for fast tracking. They are strongly recommended. Here a simple exemple of use.

Imagine you want to model an hemispherical cup
That is, one half of the space in between two spheres.
So, we take the space between two spheres
and we “cut” in the middle.
With? A plane.



Example with a plane

Writing the geo

The black hole must be there

```
SPH      Black   0.0   0.0    0.   10000.
SPH       Void   0.0   0.0   0.0    1000.
  END
BLKH      5   +Black -Void
   END
```

Example with a plane

Writing the geo

Add two spheres

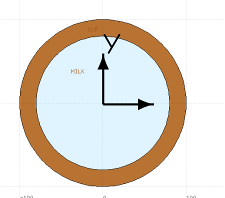
```
SPH       Black   0.0   0.0    0.  10000.
SPH        Void   0.0   0.0   0.0   1000.
SPH      OutSph    0.    0.    0.    100.  sphere at origin, R=100 cm
SPH       InSph    0.    0.    0.     80.   sphere at origin, R=80 cm
 END
BLKH      5   +Black -Void
   END
```

Example with a plane

Writing the geo

And build a region with the space between spheres. Do not forget the space around!
And fill it with something to drink

```
SPH      Black   0.0   0.0    0.   10000.
SPH       Void   0.0   0.0   0.0    1000.
SPH     OutSph    0.    0.    0.     100.   sphere at origin, R=100 cm
SPH      InSph    0.    0.    0.      80.    sphere at origin, R=80 cm
 END
BLKH      5    +Black -Void
 CUP      5    +OutSph -InSph
VOID      5    +Void -OutSph
MILK      5    +InSph
   END
```

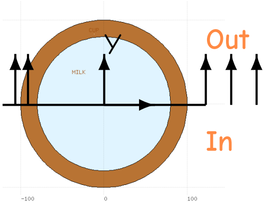


Example with a plane

Writing the geo

Add a plane to cut in two parts

```
SPH      Black   0.0    0.0      0.   10000.
SPH       Void   0.0    0.0    0.0     1000.
SPH     OutSph    0.     0.     0.      100.   sphere at origin, R=100 cm
SPH      InSph    0.     0.     0.       80.    sphere at origin, R=80 cm
XZP       Zero    0.                                    Plane ⊥ y, at y=0
END
BLKH      5   +Black -Void
 CUP      5   +OutSph -InSph
VOID      5   +Void -OutSph
MILK      5   +InSph
   END
```

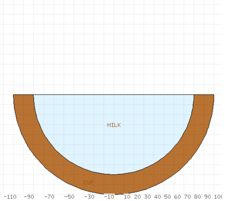


Example with a plane

Writing the geo

cut the region

```
SPH      Black   0.0   0.0    0.  10000.
SPH       Void   0.0   0.0   0.0   1000.
SPH     OutSph    0.    0.    0.    100.  sphere at origin, R=100 cm
SPH      InSph    0.    0.    0.     80.   sphere at origin, R=80 cm
XZP       Zero    0.                               Plane ⊥ y, at y=0
END
BLKH    5    +Black -Void
  CUP   5    +OutSph -InSph +Zero
  END
```

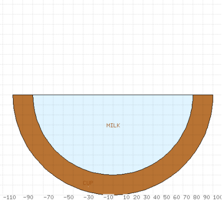


Example with a plane

Writing the geo

take care of surroundings and content

```
SPH      Black  0.0  0.0   0.  10000.
SPH       Void  0.0  0.0  0.0   1000.
SPH     OutSph   0.   0.   0.    100.   sphere at origin, R=100 cm
SPH      InSph   0.   0.   0.     80.    sphere at origin, R=80 cm
XZP       Zero   0.                               Plane ⊥ y, at y=0
END
BLKH    5   +Black -Void
 CUP    5   +OutSph -InSph +Zero
VOID    5   | +Void -Zero | +Void +Zero -OutSph
 END
```

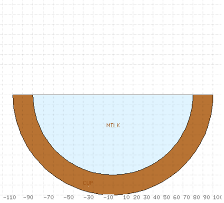


Example with a plane

Writing the geo

take care of surroundings and content

```
SPH      Black   0.0    0.0     0.   10000.
SPH       Void   0.0    0.0    0.0    1000.
SPH     OutSph    0.     0.     0.     100.   sphere at origin, R=100 cm
SPH      InSph    0.     0.     0.      80.    sphere at origin, R=80 cm
XZP       Zero    0.                                 Plane ⊥ y, at y=0
END
BLKH     5    +Black -Void
  CUP    5    +OutSph -InSph +Zero
VOID     5    | +Void -Zero | +Void +Zero -OutSph
MILK     5    +InSph +Zero
END
```



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 bodies make the tracking faster thanks to special extra coding

Precision

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.

Tracking accuracy

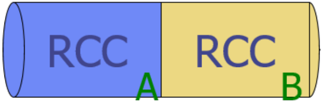
- FLUKA uses systematically double precision mathematic (i.e. 16 significant digits)
- The relative accuracy (RA) achievable in double precision is of the order of 10^{-14}-10^{-15}
- A particle can never be exactly on a boundary : can be within a given precision
- FLUKA uses by default 10^{-10} cm as absolute accuracy (AA) during tracking
- This has to be compared with the geometry size:
  - Ex: atmospheric showers: at the earth radius, R $\approx 6{\cdot}10^{8}$ cm, the rounding is of the order of $10^{-14} \times 6^{8}$ cm = 10^{-6} cm. The condition " I am on a boundary if the distance is < 10^{-10} cm" has a random outcome and makes the tracking crazy
  - Conversely, for really small geometries, one could wish better accuracy
- **GEOBEGIN**'s WHAT(2) can be used to change the AA: AA=WHAT(2)*10^{-6}cm
- AA should be larger than RA*L , being L the largest coordinate value in the considered problem, (except black hole)
- The default (WHAT(2)=10^{-4}) is OK for most geometries (L<100 m)

Precision Errors

Imagine to have two cylindrical regions next to each other. The most natural solution would be to use two adjacent RCC (cylinders delimited by faces)

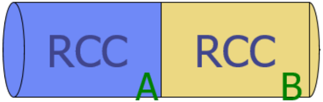
Precision Errors

Imagine to have two cylindrical regions next to each other. The most natural solution would be to use two adjacent RCC (cylinders delimited by faces)



Precision Errors

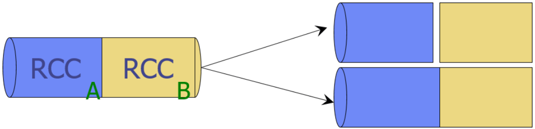
Imagine to have two cylindrical regions next to each other. The most natural solution would be to use two adjacent RCC (cylinders delimited by faces)



this means that the same surface is defined in two different ways, in A and in B. Rounding may be different, especially for complex surfaces and/or non-optimized AA

Precision Errors

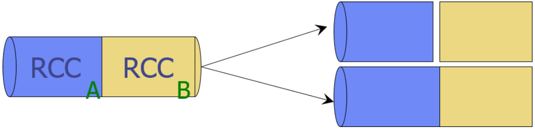
Imagine to have two cylindrical regions next to each other. The most natural solution would be to use two adjacent RCC (cylinders delimited by faces)



Rounding may be different on A and B, resulting in tracking errors

Precision Errors

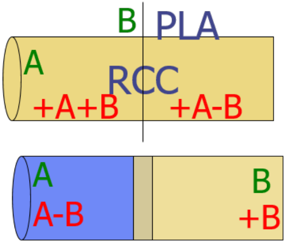
Imagine to have two cylindrical regions next to each other. The most natural solution would be to use two adjacent RCC (cylinders delimited by faces)



Rounding may be different on A and B, resulting in tracking errors

If possible, avoid touching surfaces

- Use a single RCC cut in two parts by a plane
- Or force partial overlap of RCCs



Geometry Errors

During execution the code needs to know the region where a particle is located at every step

- The program will stop if a particle position does not belong to any region
- An error message will be printed in the .err file with the particle position and direction
  - The problem might be at the present position (E.g. rounding on a boundary)
  - Or far away: next region not found
- IMPORTANT! Tracking will not stop if a particle position belongs to more than one region. It will accept the first region it finds but the results will be completely unreliable!!
- WARNING flair will NOT detect and signal geometry errors if they are not contained in at least one of the views: always inspect the whole geometry

More Geometry Errors

- Problem space not enclosed by a black body region
- Never start a primary particle along a surface. You could get a geometry error even if the geometry is correct because FLUKA cannot determine the starting region
- Precision errors: always give as many digits as possible
- Lattice replica $\longleftrightarrow$ basic cell mismatch (Lattices not covered in this course)

Debugging tools

- GEOEND card with the DEBUG SDUM
- Error messages during simulation in the .err file
- Geometry plot by Flair (it automatically invokes the PLOTGEOM card)
- FLAIR geoviewer
  - extremely powerful, it saves you!. However:
  - Be sure to cover the whole geometry
  - flair-geoviewer uses a mathematical representation different from the one in fluka. Rarely, inconsistecies show up. Should it happen, please report to the developers.

Debugging with FLUKA

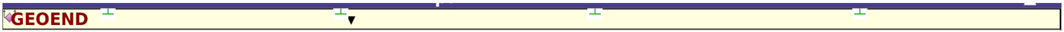
GEOEND card can activate the geometry debugger.

For every node of a cartesian mesh selected by the user, it checks whether the point is assigned to one and only one region.

Two cards are needed

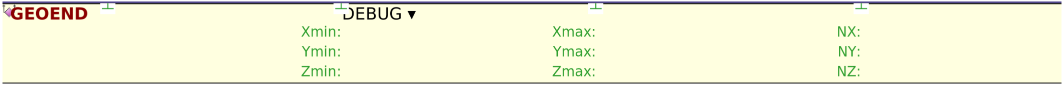
GEOEND	Xmax	Ymax	Zmax	Xmin	Ymin	Zmin	**DEBUG**
GEOEND	Nx	Ny	Nz				**&**

In flair, add the GEOEND card



GEOEND ▾

click on the small arrow to activate the DEBUG fields



GEOEND DEBUG ▾

Xmin:	Xmax:	NX:
Ymin:	Ymax:	NY:
Zmin:	Zmax:	NZ:

Debugging with FLUKA
II

- If no error is found, no .err file will be created
- If the debugger does not find any error it does not mean that the geometry is error free!
- One has to test, changing the GEOEND settings especially for critical and complicated regions
- If too many errors are found, the program stops
- Errors will be listed in the .err file in the form:

```
**** Lookdb: Geometry error found ****
**** The point: -637.623762 -244.554455 -96.039604 ****
```

Overlap of regions:

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

Undefined space:

```
**** is not contained in any region
```

Not the end of the story

The FLUKA geometry has more capabilities, not covered in this course. You can explore them through the manual, or by looking at the Geometry lecture in the last advanced course https://agenda.infn.it/event/20624/timetable/

Briefly

- Parentheses group boolean operations like in algebra
- Translation, Rotation, Expansion of bodies. For example, $Start_translat......$End_translat allow to move one or more bodies by given amounts along the three axis
- ROT-DEFI card defines rototranslation, to be used on bodies, lattices, binnings
- LATTICE card defines replicas of parts of the geometry
- VOXELS Defines a part of the geometry as composed by voxels (see lecture on medical applications)

Summary
I

Body

Basic objects to build the geomety: bodies

Each body divides the space into Inside and outside

Zone

a Zone is a part of space defined as intersection (+) and/or subtraction (-) of bodies

can be interpreted as a sub-region

Region

a Region is the union (|) of one or more Zones.

This is the real “object”, it will have materials and properties assigned to it

Summary
II

Names

Both bodies and regions are identified by names chosen by the user

name = alphanumeric identifier, 8 character maximum, case sensitive

Black Hole

The geometry must be enclosed in a region filled with BLCKHOLE

Particles entering in a BLCKHOLE region are immediately killed

completeness and unicity

Every point of the space enclosed by the black hole must belong to one and only one region

What next?

Questions?

Next: geometry exercise

Have fun!