Simple sources and preprocessor

Outline

1. Required cards and defaults
2. Definition of a simple beam
   - Beam parameters
   - Beam visualization
   - Beam rotation
3. Volumetric beams
4. Further possibilities

5. FLUKA preprocessor

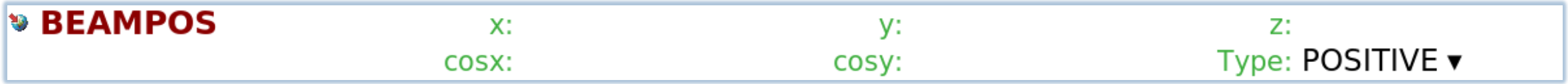
1. Required cards and defaults

Required cards

- In the “Basic input & Flair introduction” lecture we already saw to cards related to defining a beam in FLUKA:

  - **BEAM**

  - **BEAMPOS**



- What happens if they are not filled out or they are missing?

  Nothing, FLUKA will use the built in defaults.

Default beam

- Then what is the default beam in FLUKA?
  - Protons at 200 GeV/c momentum
  - Pencil beam: No divergence, zero radius
  - Starting from the origin of the coordinate system
  - Directed along the positive z axis
- This is almost never what you want!
- Always complete the relevant information on the **`BEAM`** and **`BEAMPOS`** card
- It is a good practice to check the FLUKA output (See the *Standard output* lecture)
- **Note**: This behavior may change in the future

2. Definition of a simple beam

Beam parameters

Simple beam definition

Input card: **BEAM**

- The **BEAM** card allows to specify the following parameters:
  - Particle type
  - Momentum or kinetic energy
  - Momentum distribution
  - Angular distribution
  - Shape in the X-Y plane

Simple beam definition

Input card: **BEAM**

- Particle:

Select particle type from the dropdown menu

Non standard particles:

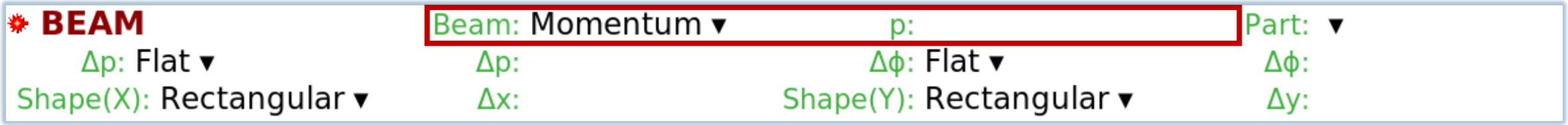
- **HEAVYION**: Ion beams heavier than ^{4}He – Requiring a **HI-PROPE** card.
  See the *Physics* lecture
- **ISOTOPE**: Radioactive isotope sources – Requiring the **HI-PROPE** and **RADDECAY** cards.
  See the *Activation* lecture

Default particle: **PROTON**

Simple beam definition

Input card: **BEAM**

- Momentum or Kinetic energy:



Select **Momentum** or **Energy** from the dropdown menu

Give **value** in the input field next to it

Default value: 200 [GeV/c]

In case of advanced sources, setting the momentum slightly higher than the maximum momentum used in those sources is **crucial**. As this value is used to initialize the cross section data tables.

(See the *Source routine* and *Advanced sources* lectures)

Simple beam definition

Input card: **BEAM**

- Momentum and angular distributions:

Momentum distribution types:

- **Flat**: Full width of a rectangular **momentum** distribution centred at beam momentum [GeV/c]
- **Gaussian**: FWHM of a Gaussian **momentum** distribution [GeV/c]

IMPORTANT: This is always momentum distribution, even **Energy** was selected

Angular distribution types:

- **Flat**: Full width if a rectangular angular distribution centred around the beam axis [mrad]
- **Isotropic**: Isotropic distribution (Input as **Flat** distribution with angle larger than 2π [rad])
- **Gaussian**: FWHM of a Gaussian angular distribution [mrad]

Simple beam definition

Input card: **BEAM**

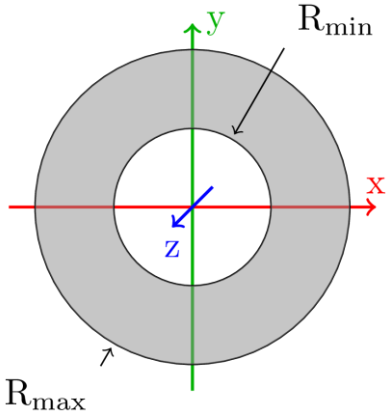
- Shape in the X-Y plane:

BEAM	Beam: Momentum ▾	p:	Part: ▾
Δp: Flat ▾	Δp:	Δϕ: Flat ▾	Δϕ:
Shape(X): Rectangular ▾	Δx:	Shape(Y): Rectangular ▾	Δy:

Distribution type:

- **Rectangular**: Full beam width in **x/y** direction centred at the beam axis [cm]
- **Gaussian**: FWHM of a Gaussian distribution in x/y direction centred at the beam axis [cm]

  Default: **Δx** = 0.0, **Δy** = **Δx**
- **Annular** distribution can be selected in the dropdown menu of **Shape(X)**
  - **Rmin** and **Rmax** are the radii of the distribution
  - The beam particle position is uniformly sampled on the **x**-**y** plane between **Rmin** and **Rmax**
  - For circular beam use **Rmin = 0**



Beam position and direction

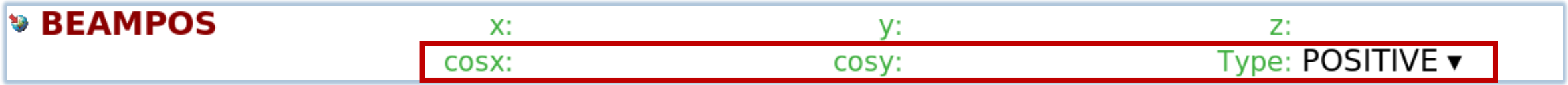
Input card: **BEAMPOS**

- *Position*:

The beam position is defined with its **x**, **y** and **z** coordinates [cm]

Default: Origin of the coordinate system

- *Direction*:



The beam axis is defined with direction cosines with regard to the x and y axes

The third direction cosine (**cosz**) are automatically calculated by FLUKA

But this is not enough for an unequivocal beam direction, the sign of the **cosz** has to be provided as well. Select **POSITIVE** or **NEGATIVE** from the *Type* dropdown

Default: Positive z direction

2. Definition of a simple beam

Beam visualization

Beam visualization

- The easiest way to check whether the beam parameters are set correctly is to visualize the beam.

- There are two options:
  - Use the Geoviewer's BEAM object
  - Use standard FLUKA scorings (See the *Scoring* lectures) with **BEAMPART** particle type
    - USRBIN for particle location and direction
    - USRBDX for energy spectrum (with a closed surface surrounding the source location)

- Example (1):

  1 GeV Gaussian beam,
  0.1 GeV/c FWHM momentum distribution,
  0.4 rad flat angular distribution,
  rotated around the y axis by -30 degrees

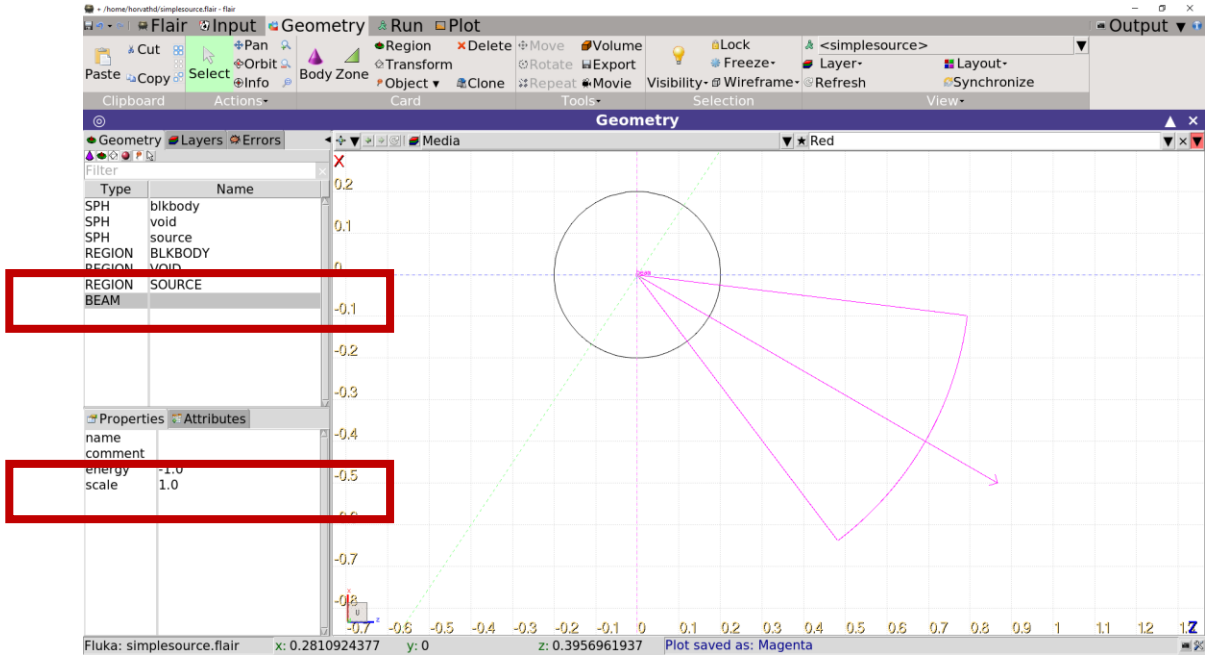
Beam visualization

- FLUKA input:

```
BEAM        Beam: Energy ▾    E: 1.0          Part: ▾
   Δp: Gauss ▾   Δp(FWHM): 0.1   Δϕ: Flat ▾    Δϕ: 400.0
Shape(X): Rectangular ▾   Δx:   Shape(Y): Rectangular ▾   Δy:
BEAMPOS     x: 0.0          y: 0.0          z: 0.0
            cosx: -0.5      cosy: 0.0       Type: POSITIVE ▾
```

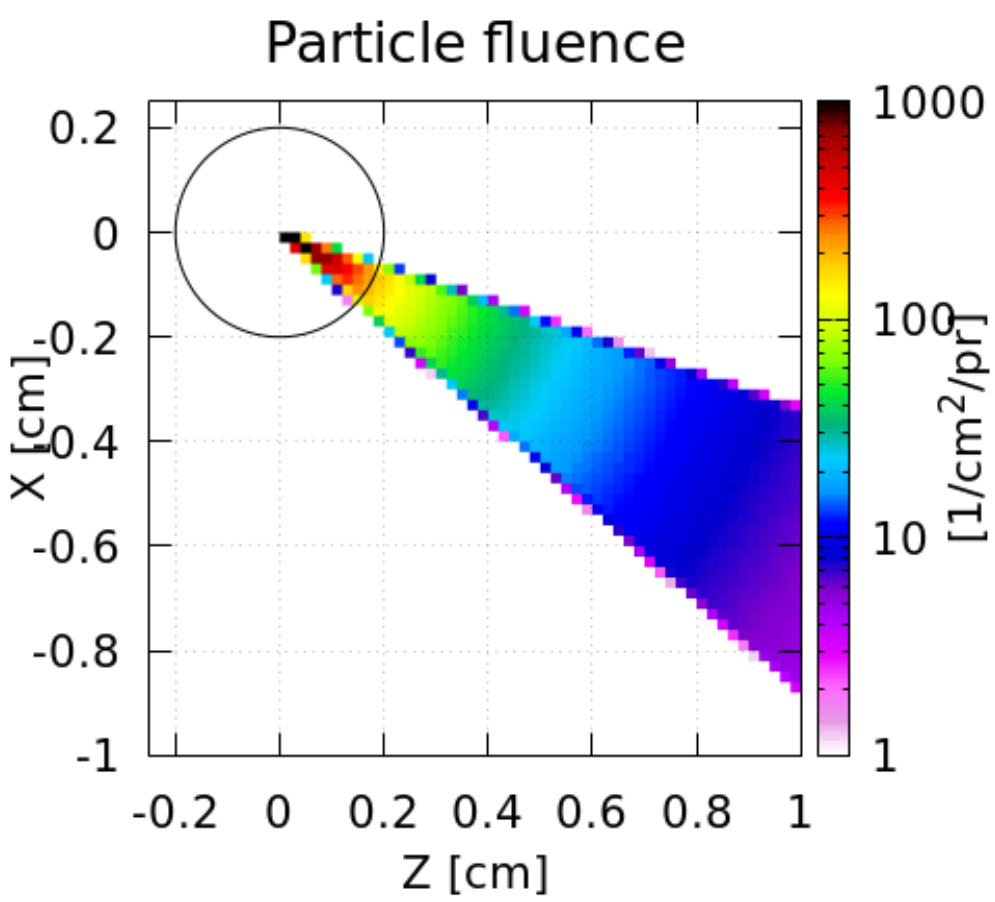
- Geoviewer:

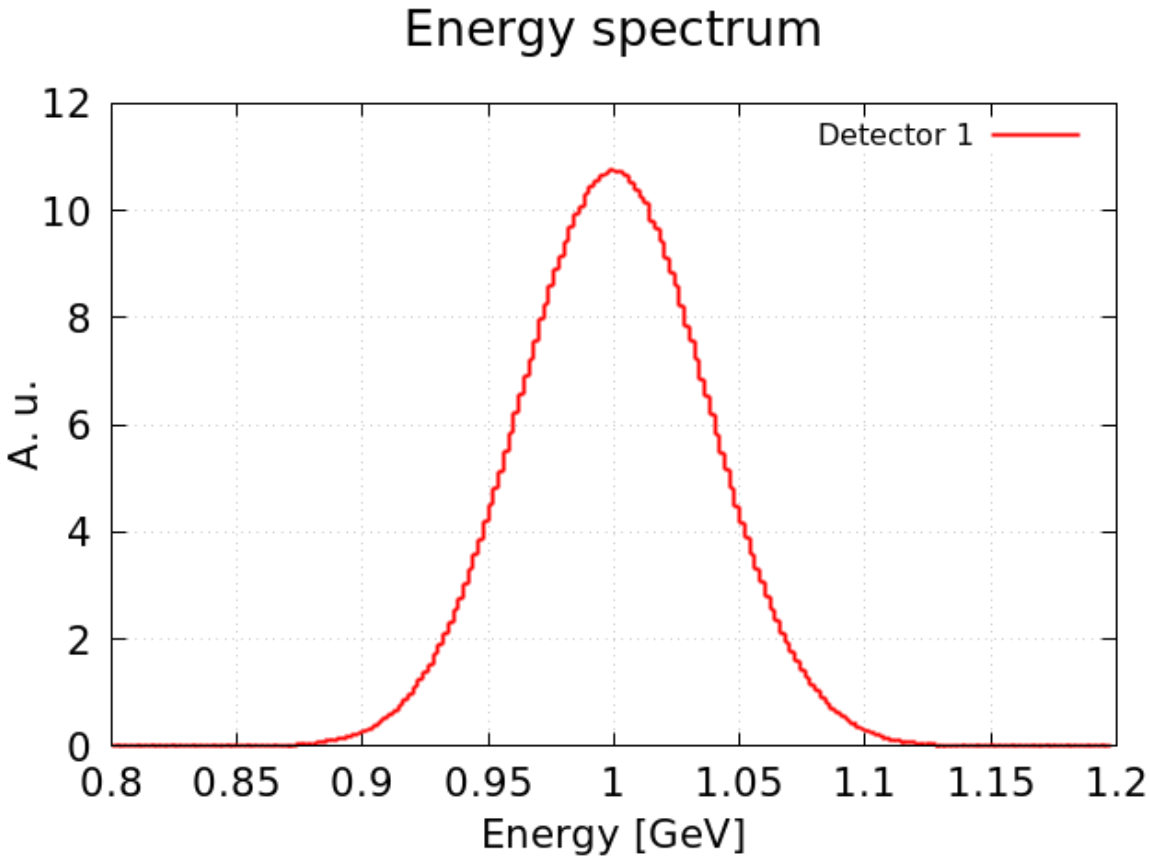
  - Starting point
  - Direction
  - Angular distribution
  - Beam mean energy
  - Default scale: 1 GeV(/c) = 1 cm
    Can changed with
    the scale parameter



Beam visualization

- FLUKA scoring:

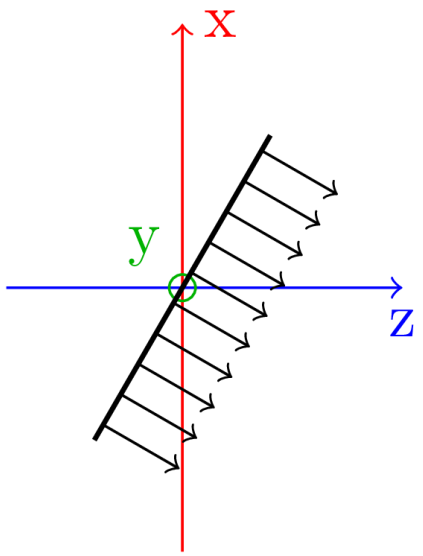




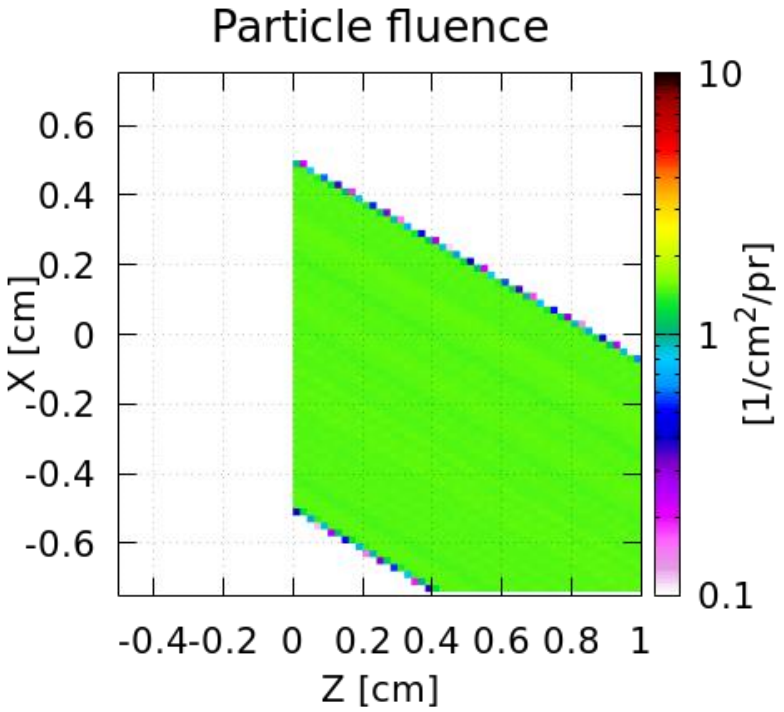
Beam visualization

- Example (2):
  - R = 0.5 cm cylindrical beam with zero angular distribution, rotated around the y axis by -30 degrees

Expectation:



Reality:



- *Remember*: **BEAM** card sets the X-Y shape of the beam, which is not influenced by the beam direction set in the **BEAMPOS** card.

2. Definition of a simple beam

Beam rotation

Beam rotation

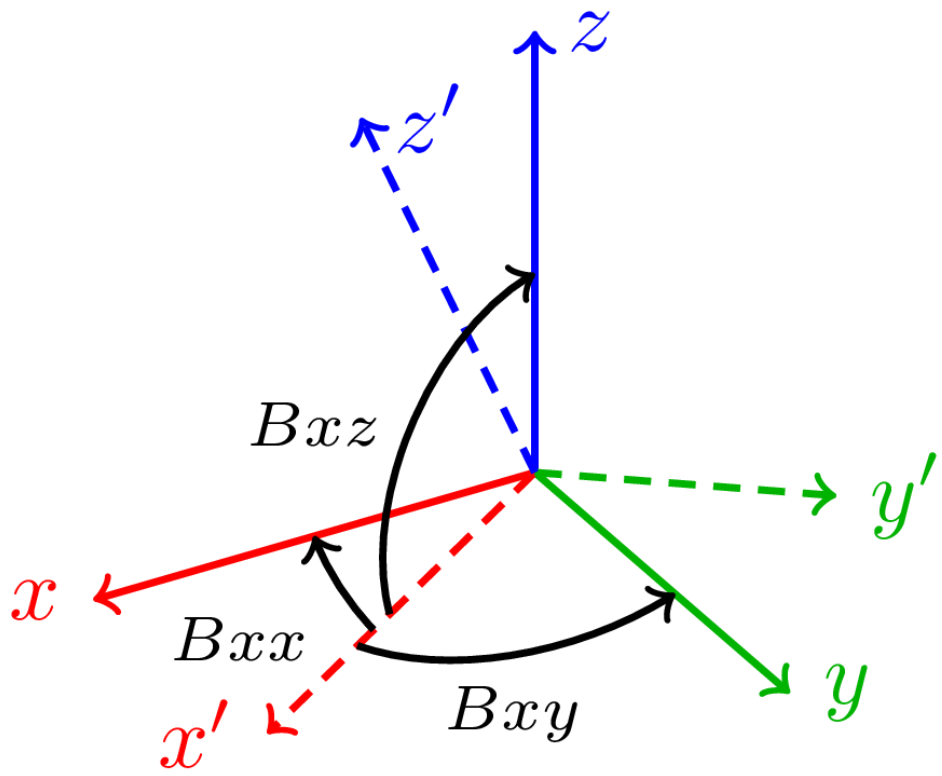
- Input card: **BEAMAXES**

BEAMAXES	cosBxx:	cosBxy:	cosBxz:
	cosBzx:	cosBzy:	cosBzz:

Defines the beam coordinate system (**x′, y′, z′**) with regards to the geometry one (**x, y, z**)

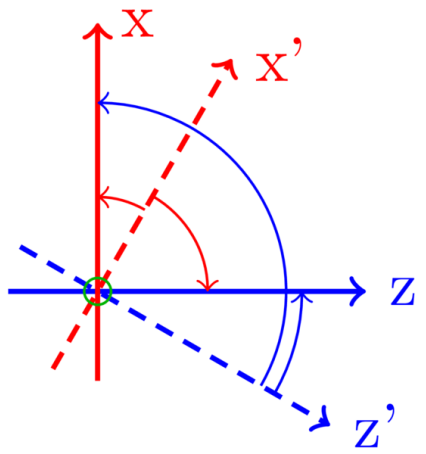
Input fields:

- **cosBxx**: cosine of the angle between **x′** and **x**
- **cosBxy**: cosine of the angle between **x′** and **y**
- **cosBxz**: cosine of the angle between **x′** and **z**
- **cosBzx**: cosine of the angle between **z′** and **x**
- **cosBzy**: cosine of the angle between **z′** and **y**
- **cosBzz**: cosine of the angle between **z′** and **z**

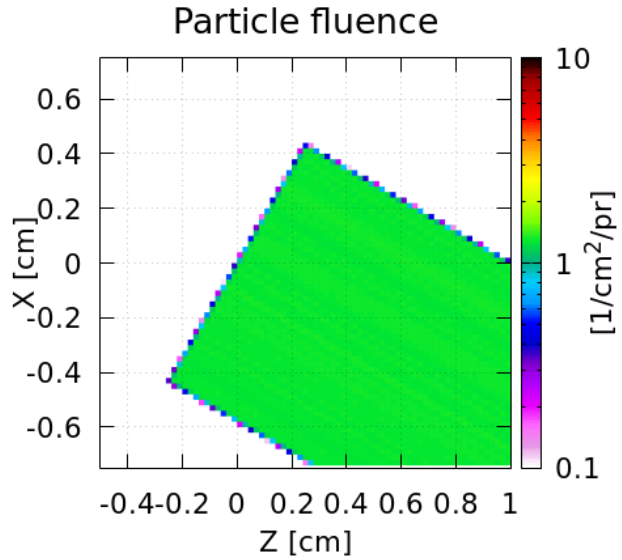


Beam rotation

- Example (2):
  - 1 GeV Gaussian beam, zero angular distribution, 0.5 cm radius, rotated around the y axis by -30 degrees



WHAT	Beam axis	Geo axis	Angle [°]	Cos(Angle)
cosBxx	x'	x	30	~0.866
cosBxy	x'	y	90	0
cosBxz	x'	z	-60	0.5
cosBzx	z'	x	120	-0.5
cosBzy	z'	y	90	0
cosBzz	z'	z	30	~0.866



BEAMAXES cosBxx: 0.86602540378 cosBxy: 0.0 cosBxz: 0.5
cosBzx: -0.5 cosBzy: 0.0 cosBzz: 0.86602540378

3. Volumetric sources

Volumetric sources

With a second **BEAMPOS** card volumetric sources can be defined:

- Available types:
  - Spherical shell (**SPHE-VOL**)
  - Cylindrical shell (**CYLI-VOL**)
  - Cartesian shell (**CART-VOL**)
  - Spherical surface (**FLOOD**)

- The volumetric sources are centred around the position defined with the first **BEAMPOS** card
- The location in the volume sampled uniformly
- The particle direction and angular distribution set on the first **BEAMPOS** and the **BEAM** card are still applied
- **Warning:** The spatial distributions specified on the **BEAM** card will be disregarded

Volumetric sources – *Spherical shell*

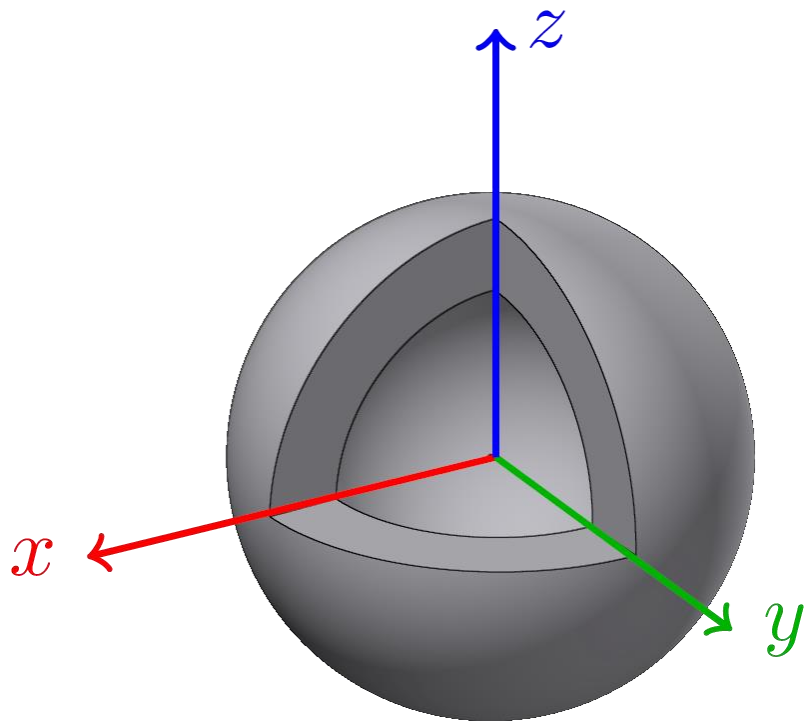
Input card: **BEAMPOS**

BEAMPOS	Rin:	Rout:	Type: SPHE-VOL ▾

Specifies a spherical shell shaped source

Input fields:

- **Rin**: Inner radius [cm]
- **Rout**: Outer radius [cm]



Volumetric sources – *Cylindrical shell*

Input card: **BEAMPOS**

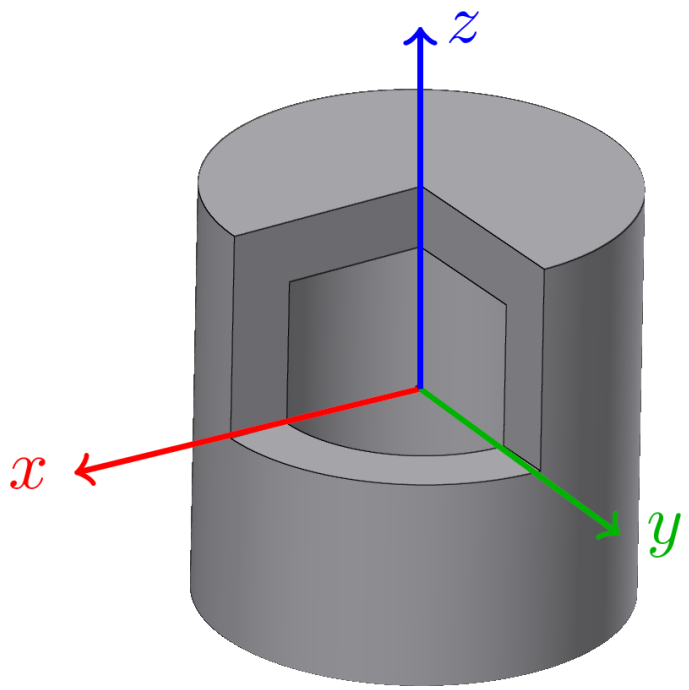
BEAMPOS	Rin:	Rout:	Type: CYLI-VOL ▾
	Hin:	Hout:	

Specifies a cylindrical shell shaped source around the **z** axis of the geometry

Input fields:

- **Rin**: Inner radius [cm]
- **Rout**: Outer radius [cm]
- **Hin**: Inner height [cm]
- **Hout**: Outer height [cm]

Note: The reference coordinate system can be changed with the **BEAMAXES** card



Volumetric sources – *Cartesian shell*

Input card: **BEAMPOS**

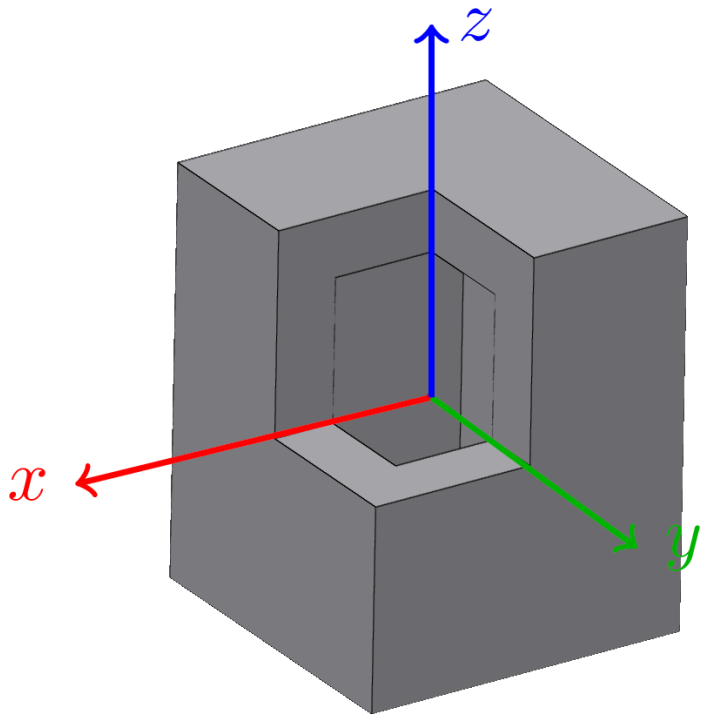
BEAMPOS	Xin:	Xout:	Type: CART-VOL ▾
	Yin:	Yout:	
	Zin:	Zout:	

Specifies a Cartesian shell shaped source along the axes of the geometry

Input fields:

- **Xin & Xout**: Inner & outer length of the **x**-sides
- **Yin & Yout**: Inner & outer length of the **y**-sides
- **Zin & Zout**: Inner & outer length of the **z**-sides

Note: The reference coordinate system can be changed with the **BEAMAXES** card



Volumetric sources – *Spherical surface source*

Input card: **BEAMPOS**

BEAMPOS	R:	Type: FLOOD ▾

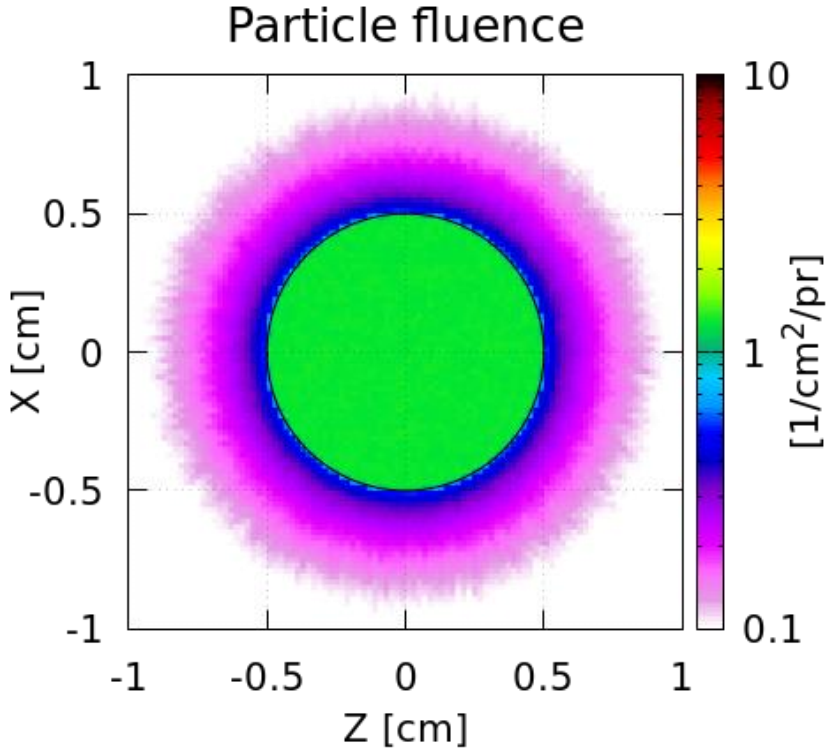
Specifies a spherical surface source in a way that the fluence inside the sphere is **uniform** and **isotropic**

The value of the produced fluence is: $\frac{1}{\pi R^2}$ cm^{-2}

Input fields:

- **R**: Radius of the sphere [cm]

Warning: The particle direction and angular distribution set on the first **BEAMPOS** and the **BEAM** card are disregarded



4. Further possibilities

Further possibilities

Sometimes the **BEAM**, **BEAMPOS**, and **BEAMAXES** cards are not enough

- Special sources available in FLUKA
  - Colliding beams
  - Synchrotron radiation
  - Cosmic rays
  - Multiple beam spots
  - USRBIN source

    (See the *Advanced sources* lecture)

- Program your own sources from scratch

  (See the *Source routine* lecture)

5. FLUKA preprocessor

FLUKA preprocessor

- Limited, “C” like preprocessor
- Manipulates the input before execution using directives

- Directives start with: **#** (number / hash / pound sign)

- 3 type of directives:
  - Definition:
    #define, #undef
  - Conditional:
    #if, #elif, #else, #endif
  - Include:
    #include

FLUKA preprocessor - *Definition*

Directive: `#define`

- Identifiers without numerical or character value:

```
#define  identifier_name      :
```

  - Used in conjunction with conditional directive
  - ***identifier_name*** can be up to 40 character long

- Identifiers with numerical or character value:

```
#define  identifier_name      : value
```

  - Can be used in conjunction with conditional directives
  - The ***value*** can be used on any other input card by referencing **$identifier_name**
  - ***value*** can be up to 40 character long

FLUKA preprocessor - *Definition*

- Identifiers can be *defined* and *referenced* anywhere in the input file
- Example:

```
#define  Momentum          : 0.1
#define  Particle          : PROTON
BEAM                 Beam: Momentum ▾     p: $Momentum          Part: $Particle ▾
     Δp: Flat ▾          Δp:              Δϕ: Flat ▾            Δϕ:
Shape(X): Rectangular ▾   Δx:          Shape(Y): Rectangular ▾   Δy:
```

- It is possible to redefine an identifier with a second `#define` directive
  - This is **NOT RECOMMENDED**
  - If an identifier is redefined, the new value only applied to cards below
  - The output and error files will contain warning messages

Directive: `#undef`

```
#undef  identifier_name ▾
```

  - Deletes a previously defined identifier

FLUKA preprocessor - *Conditional*

Directives: `#if, #elif, #else, #endif`

- To include or exclude parts of the input from execution
- Used in conjunction with the `#define` directives

```
#define identifier_A        :
#define identifier_B        :
...
#if    identifier_A ▾
#elif  identifier_B ▾
#else
#endif
```

- They work similar to any programming language

  Limitation: Can't compare value, only tests if an identifier is defined or not

- Can be nested

  Limitation: Maximum of 10 nesting levels can be used

FLUKA preprocessor - *Conditional*

```
#define NeutronBeam        :
...
#if    NeutronBeam ▾
    BEAM                Beam: Energy ▾          E: 0.1              Part: NEUTRON ▾
     Δp: Flat ▾           Δp:                   Δϕ: Flat ▾           Δϕ:
     Shape(X): Rectangular ▾  Δx:               Shape(Y): Rectangular ▾  Δy:
#else
    BEAM                Beam: Energy ▾          E: 0.05             Part: PHOTON ▾
     Δp: Flat ▾           Δp:                   Δϕ: Flat ▾           Δϕ:
     Shape(X): Rectangular ▾  Δx:               Shape(Y): Rectangular ▾  Δy:
#endif
...
#if    NeutronBeam ▾
    ASSIGNMA            Mat: CONCRETE ▾         Reg: SHIELD ▾       to Reg: ▾
                   Mat(Decay): ▾                Step:               Field: ▾
#else
    ASSIGNMA            Mat: LEAD ▾             Reg: SHIELD ▾       to Reg: ▾
                   Mat(Decay): ▾                Step:               Field: ▾
#endif
```

FLUKA preprocessor - *Conditional*

```
#define NeutronBeam      :
...
#if   NeutronBeam ▾
   #define Energy        : -0.1
   #define Particle      : NEUTRON
   #define ShieldMa      : CONCRETE
#else
   #define Energy        : -0.05
   #define Particle      : PHOTON
   #define ShieldMa      : LEAD
#endif
...
BEAM          Beam: Energy ▾     E: $Energy        Part: $Particle ▾
    Δp: Flat ▾          Δp:       Δϕ: Flat ▾          Δϕ:
Shape(X): Rectangular ▾  Δx:  Shape(Y): Rectangular ▾  Δy:
...
ASSIGNMA      Mat: $ShieldMa ▾   Reg: SHIELD ▾   to Reg: ▾
              Mat(Decay): ▾      Step:           Field: ▾
```

FLUKA preprocessor - *Include*

Directive: `#include`

```
#include <path>/filename.inp
```

- Includes the specified file to the input
- Can be nested at multiple levels

- Path can be:
  - Relative to the **main input** file
  - Absolute

- Use cases:
  - Split large input files into multiple smaller ones
  - Reuse same input section (beam definition, scoring, etc) in multiple input files


FLUKA