Advanced topics

A brief overview of some advanced features

Lecture overview

- Introduction
- Defining complex geometries (Lattice)
- Accessing detailed particle information
- Defining complex magnetic fields
- Other advanced features and conclusion

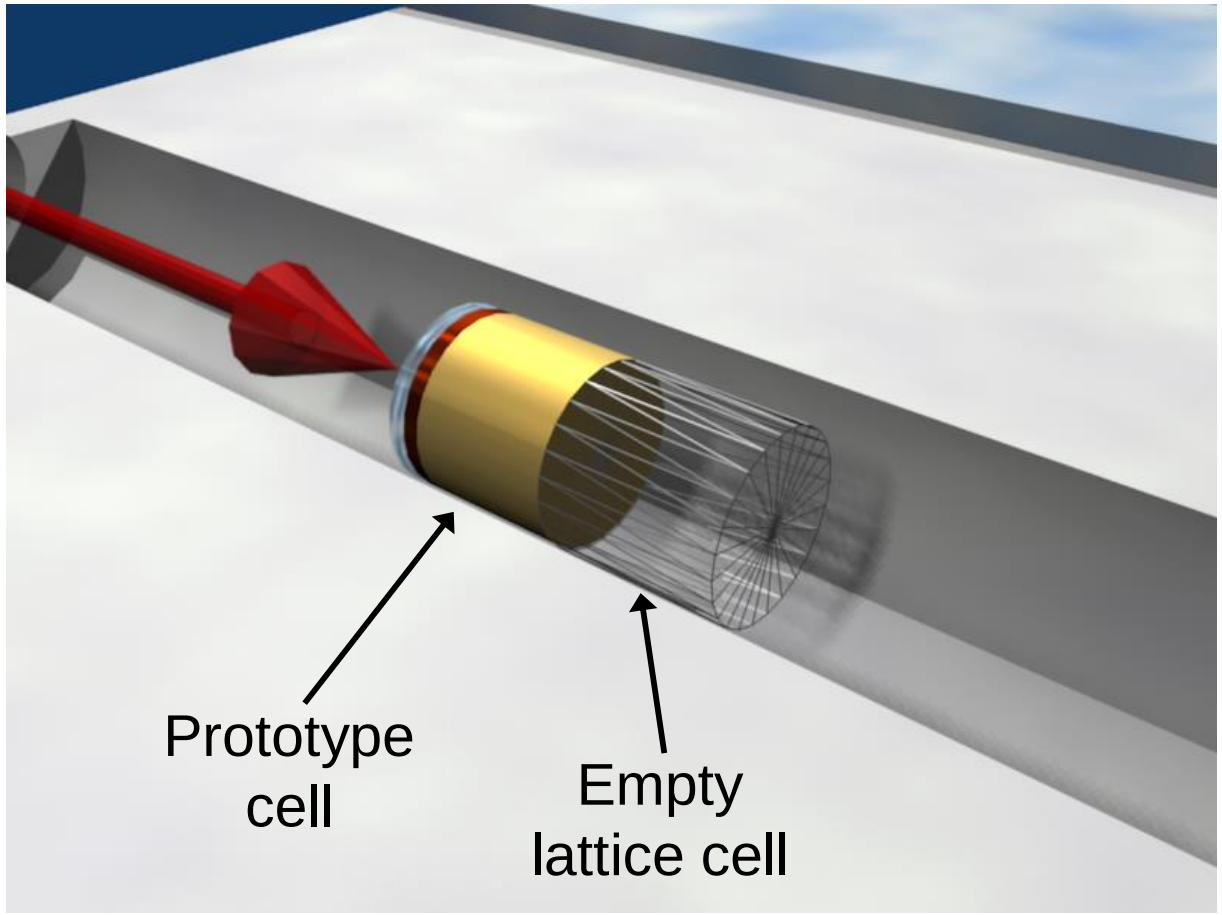
Advanced topics

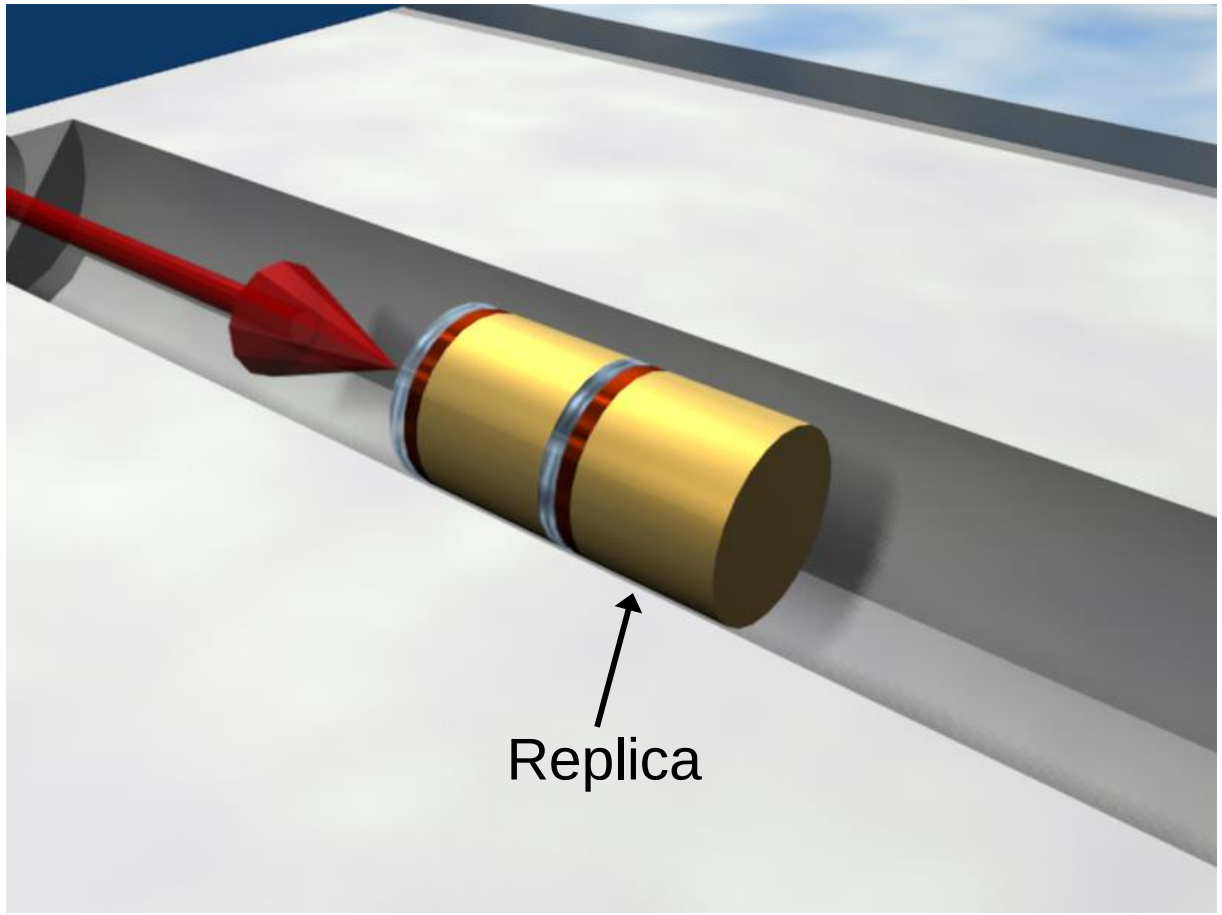
- We have covered the fundamental capabilities of FLUKA during this course
- These features are controlled through the default options and can be combined to cover a wide variety of problems, including complex geometries, scoring needs etc.
- Still, more complicated requirements may arise, such as:
  - Replicating complex geometries (Lattice)
  - Complex particle sources
  - Custom scoring, extraction of detailed particle information (on an event-by-event basis)
  - Tracking in complex magnetic fields
  - Region-independent importance biasing, and more…
- These more advanced capabilities generally require modified user routines and compilation of custom FLUKA executables
  - Default versions of the user routines can be found in the **`pathtofluka/src/user/`** directory
- We will cover three examples in this lecture

1: Lattices

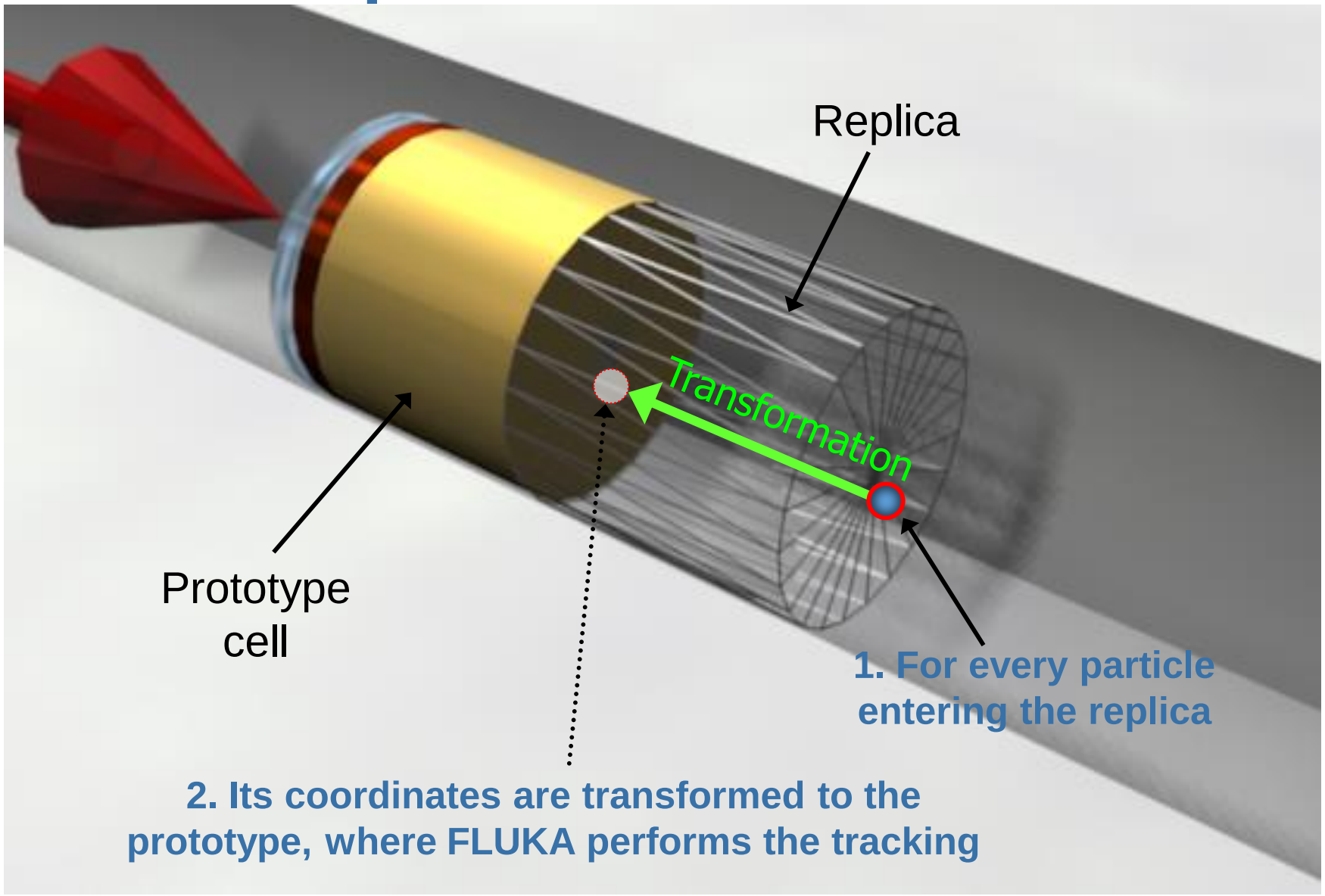
Lattice: the idea

FLUKA geometry has replication capabilities with the **LATTICE** card





Lattice: the concept



Lattice: how it works

- The regions, which constitute the elementary cell (prototype) to be replicated, can be defined in detail
- Only one level is implemented (no nested lattices are allowed)
- The lattices (replicas/containers) have to be defined as “empty” regions in their correct location
  - WARNING: The lattice region should map exactly the outer surface definition of the elementary cell
  - Materials, thresholds, etc., must be assigned **ONLY** to the regions contained in the prototype
- The lattice regions are declared as such with a **LATTICE** card
- In the **LATTICE** card, the user also assigns lattice names/numbers to the lattices. These names/numbers will identify the replicas in all FLUKA routines and scorings
- Several basic cells and associated lattices can be defined within the same geometry, one **LATTICE** card will be needed for each set
- Non-replicas carry the lattice number 0
- Lattices and plain regions can coexist in the same problem

How to define a lattice

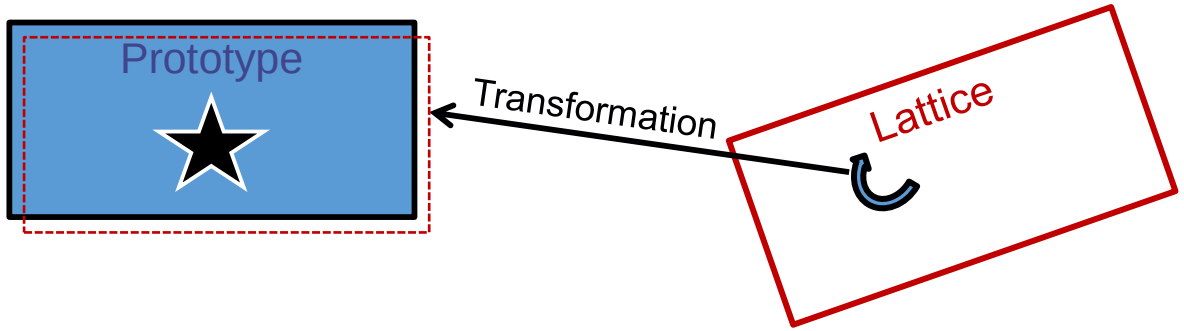
- The user defines lattice positions in the geometry and provides transformation rules from the lattice to the prototype region
  - in the input with the **`ROT-DEFI`** card
  - in a subroutine (**`lattic.f`**)
- Transformations include Translation, Rotation and Mirroring (the latter only through the subroutine **`lattic.f`**)

Reg	container region
Lat	name/number of the lattice
Trans	transformation name (if empty, **lattic.f** will be called)

Remember that matrix multiplication is not commutative: the order of the Rotation/Translation in 3D is important!

Lattice: Numerical Precision

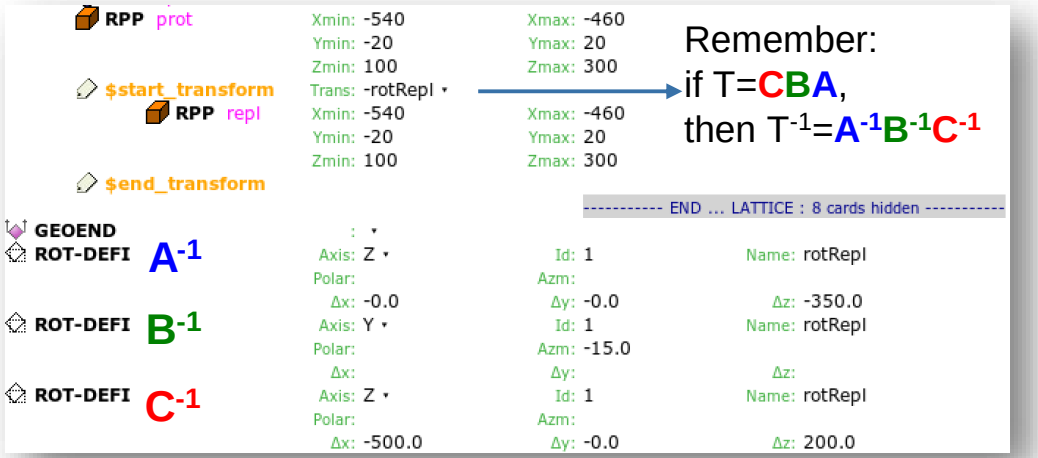
- Due to the nature of the floating point operations in CPU, even if the transformation looks correct the end result could be problematic



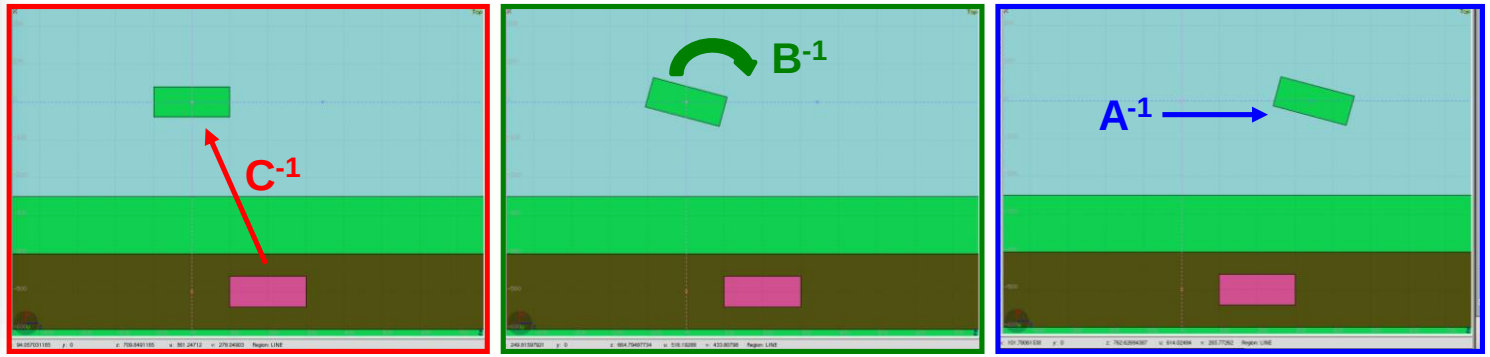
- This small misalignment between lattice/transformation/prototype could lead to geometry errors
- Use as many digits as possible to describe correctly the prototype and lattice cells as well as the transformation.
  It is mandatory that the transformation applied to the container makes the latter **exactly** corresponding to the prototype
- One can use a **`FREE`** and **`FIXED`** card before and after the **`ROT-DEFI`** to input more than 9 digits
- **`GEOBEGIN`** WHAT(2) allows to relax the accuracy in boundary identification (**Use with caution**)

Lattice: Important remarks – 1

- Remember that the **transformation must bring the container onto the prototype** and not vice versa!
- Rotations are always around the origin of the geometry, and not the center of the object. To rotate an object
  1. Translate the object to the origin of the axes
  2. Perform the rotation
  3. Move it by a final translation to the requested position
     Of course with the inverse order since everything should apply to the replica
- A transformation can be divided into many **`ROT-DEFI`** cards for easier manipulation



Remember:
if $T=CBA$,
then $T^{-1}=A^{-1}B^{-1}C^{-1}$



Lattice: Important remarks – 2

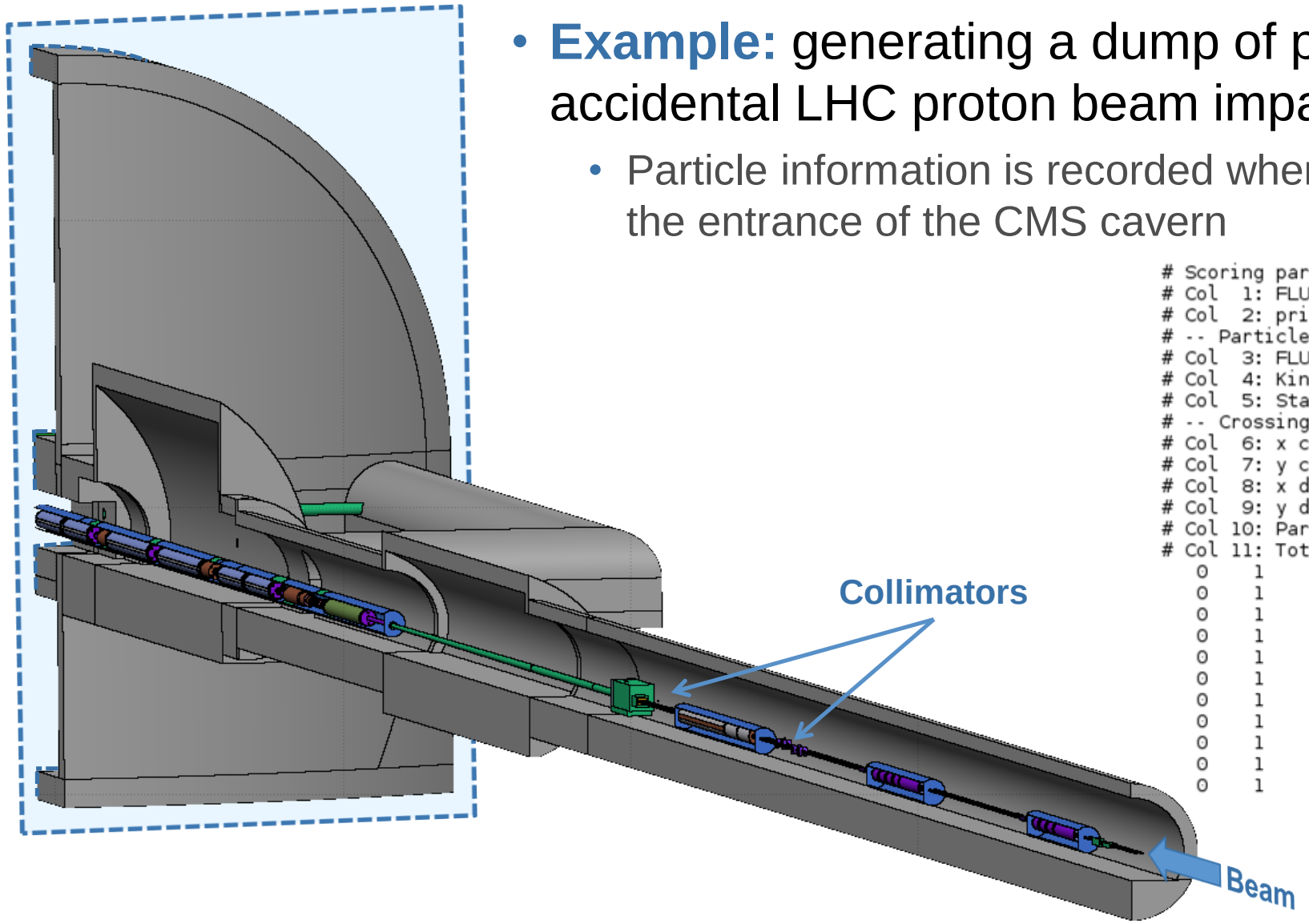
- Materials and other properties have to be assigned only to the regions constituting the prototype
- In all (user) routines the **region number** refers to the corresponding one **in the prototype**
- The **SCORE** summary in the .out file and the scoring by regions add together the contributions of the prototype region as well as of all its replicas!
- The lattice identity can be recovered runtime by the lattice number, as set in the **LATTICE** card, or available through the **GEON2L** routine if it is defined by name
- In particular, the **LUSRBL** user routine allows to manage the scoring on lattices in the special **USRBIN/EVENTBIN** structure

2: Accessing particle information

Accessing particle information

- Access to many different particles properties on an event-by-event basis, such as:
  - Coordinates
  - Direction
  - Energy
  - Age
  - Generation (primary, secondary etc.)
  - parent particle and more...
- This can be (not exclusively) accomplished with the `mgdraw.f` routine, activated via a `USERDUMP` card
- Particle information can be accessed at:
  - source particle generation
  - each step
  - boundary crossings
  - energy deposition events
  - interactions etc.

Accessing particle information

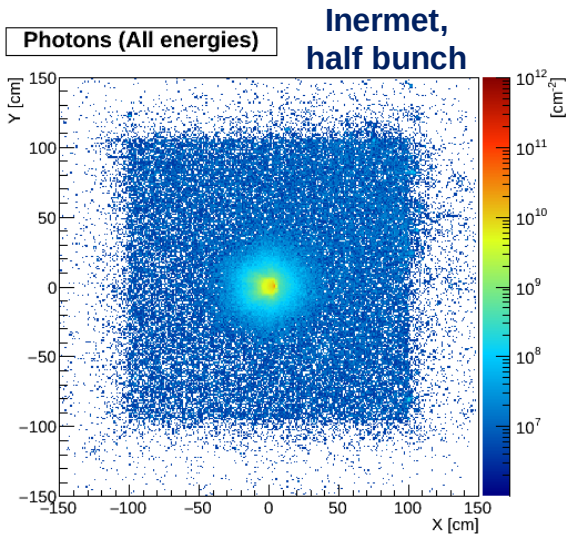


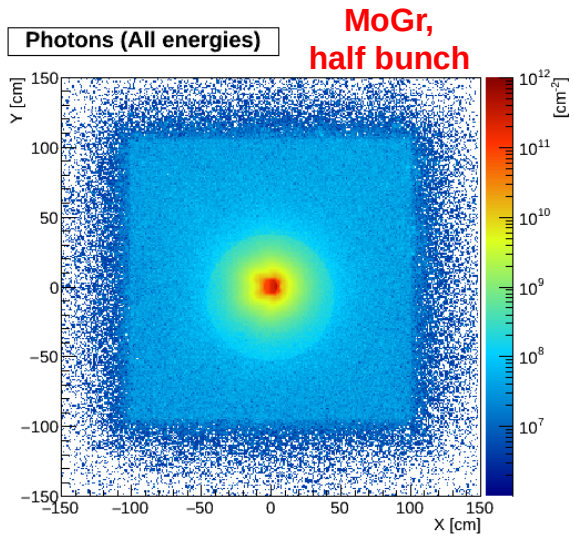
- **Example:** generating a dump of particles produced from an accidental LHC proton beam impact on collimators
  - Particle information is recorded when crossing an interface plane at the entrance of the CMS cavern

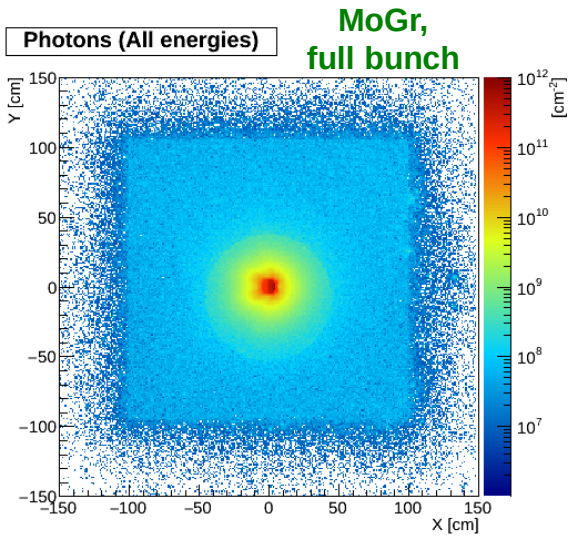
```
# Scoring particles entering Region No  2126
# Col  1: FLUKA run number
# Col  2: primary event number
# -- Particle information --
# Col  3: FLUKA particle type ID
# Col  4: Kinetic energy (GeV)
# Col  5: Statistical weight
# -- Crossing at scoring plane --
# Col  6: x coord (cm)
# Col  7: y coord (cm)
# Col  8: x dir cosine
# Col  9: y dir cosine
# Col 10: Particle age since primary event (sec)
# Col 11: Total energy (GeV)
   0    1    7  1.0685640710268577E-03  1.0000000000000000E+00  5.80402
   0    1    7  2.0869470497192035E-04  1.0000000000000000E+00  6.14101
   0    1    8  3.4885316857469206E-11  1.0000000000000000E+00 -8.73614
   0    1    7  7.0087267686422025E-02  1.0000000000000000E+00 -4.77469
   0    1    8  5.0713988564154988E-11  1.0000000000000000E+00 -7.06454
   0    1    7  3.1391604740674895E-03  1.0000000000000000E+00 -3.73255
   0    1    7  8.2376767285229514E-03  1.0000000000000000E+00 -4.63773
   0    1    4  2.3790418550118337E-02  1.0000000000000000E+00  5.64980
   0    1    7  2.0485104425603303E-02  1.0000000000000000E+00 -4.90033
   0    1    7  1.1054655441846896E-03  1.0000000000000000E+00  2.43893
   0    1    3  2.2395286454039216E-02  1.0000000000000000E+00  3.63118
```

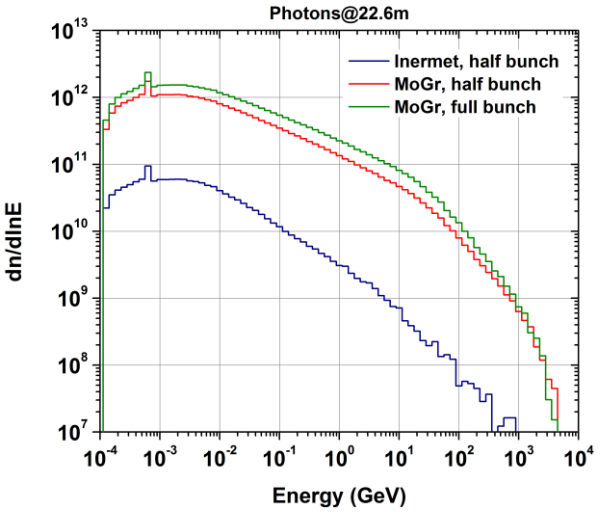
Accessing particle information

- This information can be processed to study the properties of the particle population
  - e.g. the spatial distribution near the beam-line and the energy distribution of particles for different choice of collimator materials and beam impacts:









- **Note:** The great flexibility of this type of scoring comes with the penalty of ad-hoc post-processing with custom tools!

3: Complex magnetic fields

Complex magnetic fields

- As previously discussed, the **MGNFIELD** card can be used to define uniform magnetic fields with fixed (B_x, B_y, B_z) components
- For all other cases, you need to call a user-modified **magfld.f** routine by setting $B_x = B_y = B_z = 0.0$ in the **MGNFIELD** card

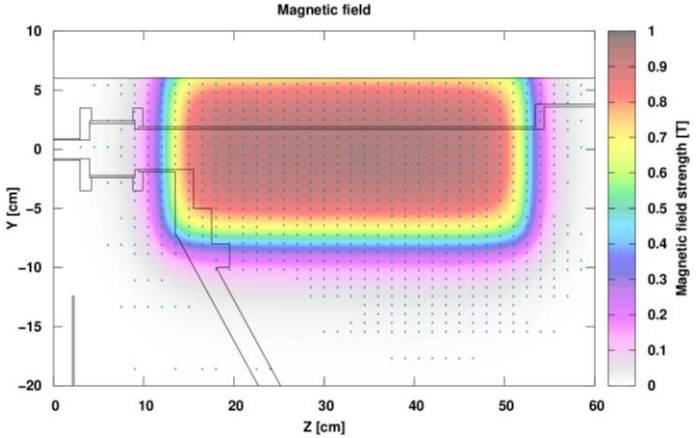
- Through the **magfld.f** routine you can implement arbitrarily complex magnetic fields, describing them analytically or interpolating an external field map, e.g.:

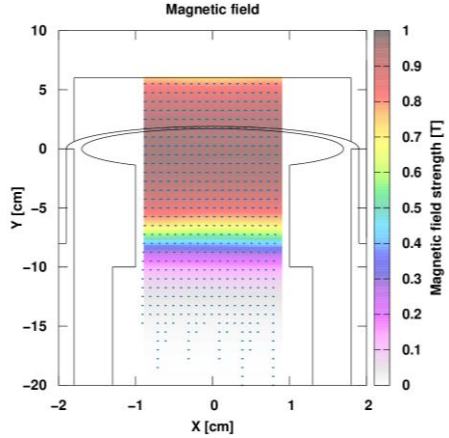
```
BTX = QUAGRAD * Y/1.0D+02  ! Quadrupole gradient in T/m
BTY = QUAGRAD * X/1.0D+02
BTZ = ZERZER
B   = SQRT(BTX**2 + BTY**2 + BTZ**2)
```

- **Note:** The **magfld.f** routine is called only in regions declared as magnetic via the relevant **ASSIGNMAT** card! (see Materials lecture)

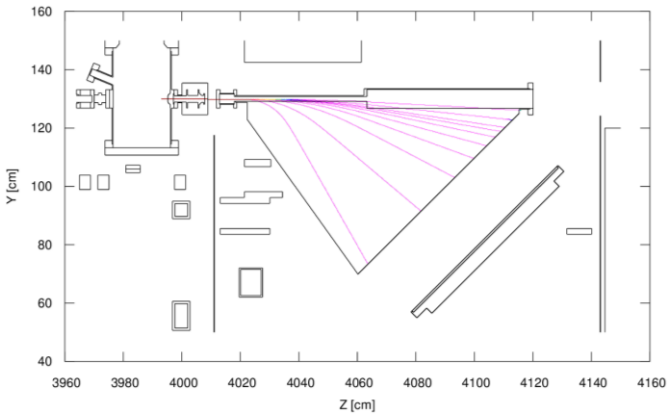
Complex magnetic fields

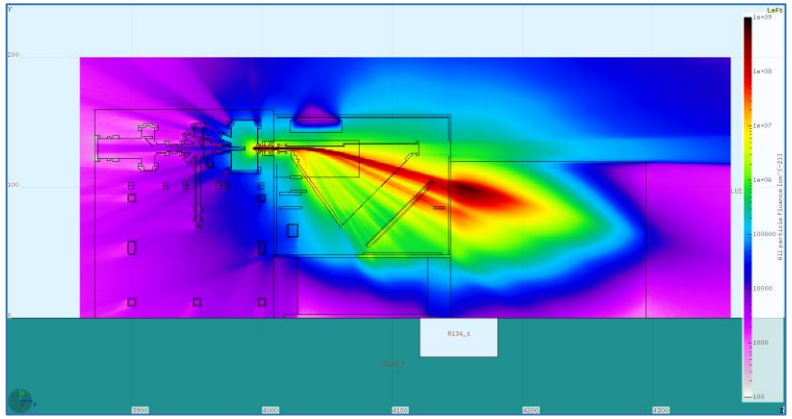
- You can (and should) check that the field has been correctly implemented by:
  - Plotting it with Flair (example: dipole field loaded from external field map)





  - Generating (single) charged particles and observing their behaviour (example: electrons correctly bent downwards by the dipole described above)





Conclusion

All good things…

- Many other advanced/specialised topics can be studied with FLUKA…
  - Medical applications (DICOM import, treatment planning etc.)
  - Cosmic rays
  - Neutrino interactions
  - Optical photons
  - Crystal channeling (upcoming release)
  - …and more!
- …and a lot more flexibility can be achieved via user routines

- This online training was meant to get you started on FLUKA (building geometries, defining materials and simple sources, scoring), while also introducing more advanced concepts and techniques (biasing, advanced geometry features, advanced sources)
  - We would like to hear your feedback, positive and negative!

- **Consider following a FLUKA intermediate school! ☺**


FLUKA