Experiment Simulation

CERN School of Computing 2006
Helsinki

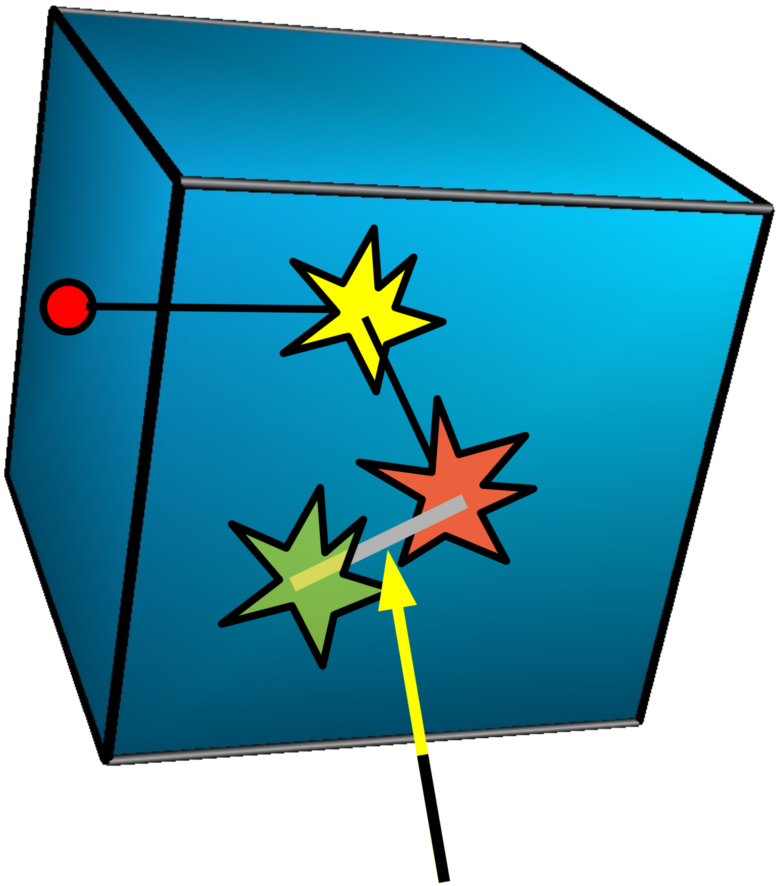
Lecture 3

SWISS EXCHANGE
Martin Liendl
SWX Swiss Exchange

Overview Lecture ~3~

- A – Physics Model in GEANT4
  - Stepping: moving in free path lengths
  - Physics Processes
- B - Detector Description in GEANT4
  - Solids/Shape Model
  - Volumes
  - Hierarchy of Volumes
- Combining A + B
  - Stepping through a detector description

Remember ...



$p_i(L) = 1 - \exp(-L/\lambda_i)$

Probability of having an interaction within L due to **process i**

Monte Carlo Algorithm:

- Sample the free path length from the distributions of all participating processes
- Select the smallest path length
- Move the particle by this step
- Simulate the interaction

Woah! A G4Step

is determined by a G4VProcess

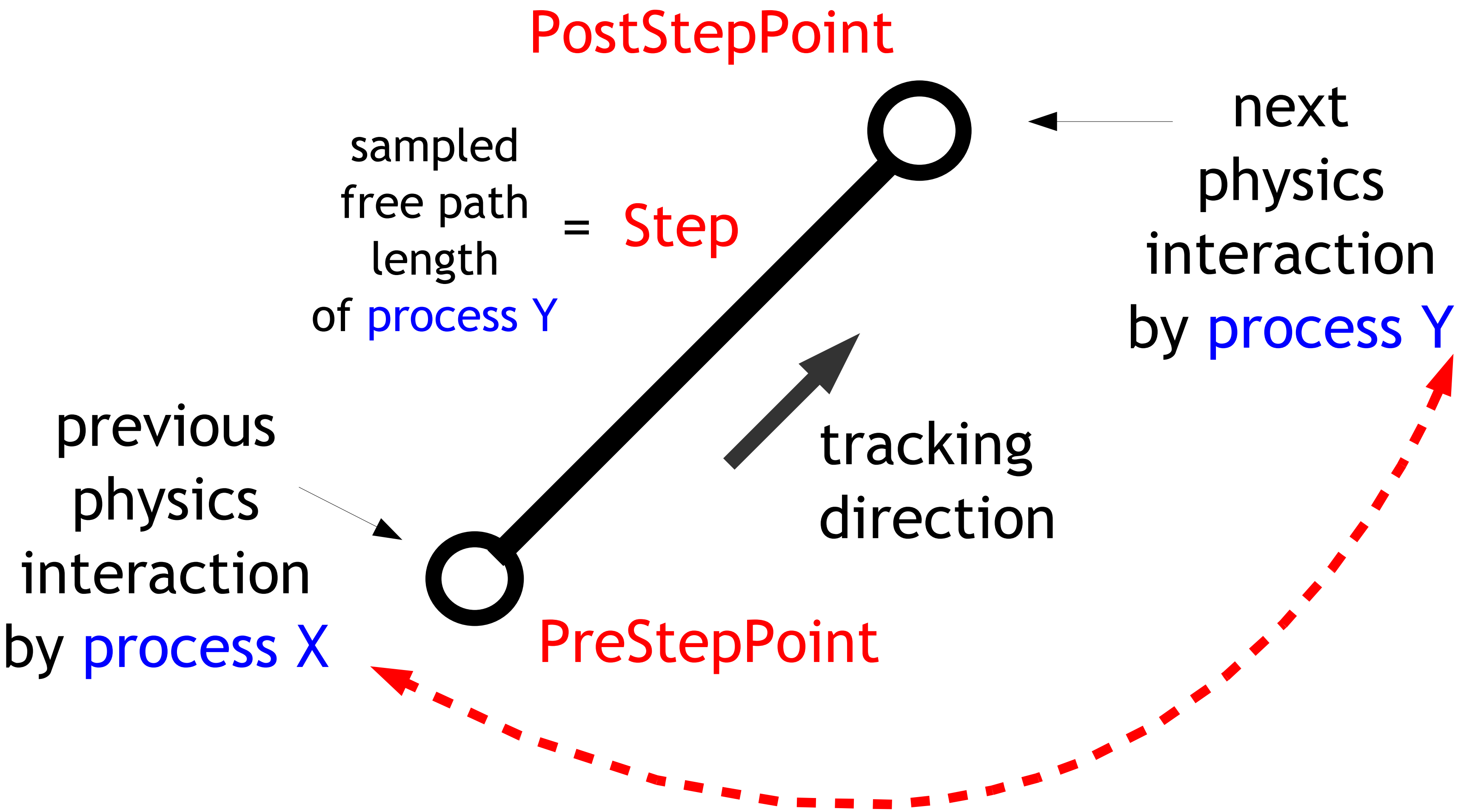
a G4VProcess subclass basically implements σ(E,..), dσ(E, ..)

G4Step ← G4VProcess

- Will talk more detailed about G4Step and step related classes
  - steps result from free path lengths sampling
  - instances of G4Step are **exposed to the simulation user**
    - entry point to extract simulation information
- Only an overview of G4VProcess will be given
  - G4VProcess determines G4Step by implementing the total and differential cross sections
  - G4 has a VAST areal of physics processes
  - plenty of very specific physics knowledge required
  - processes are **not directly exposed to the simulation user**

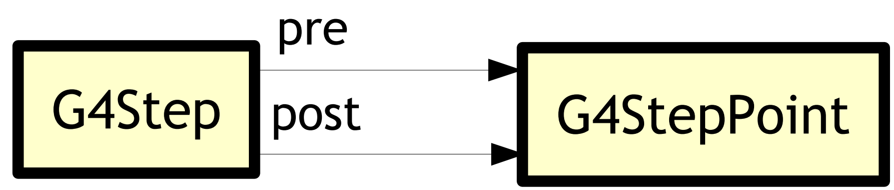
“That´s one small Step for G4 ..”

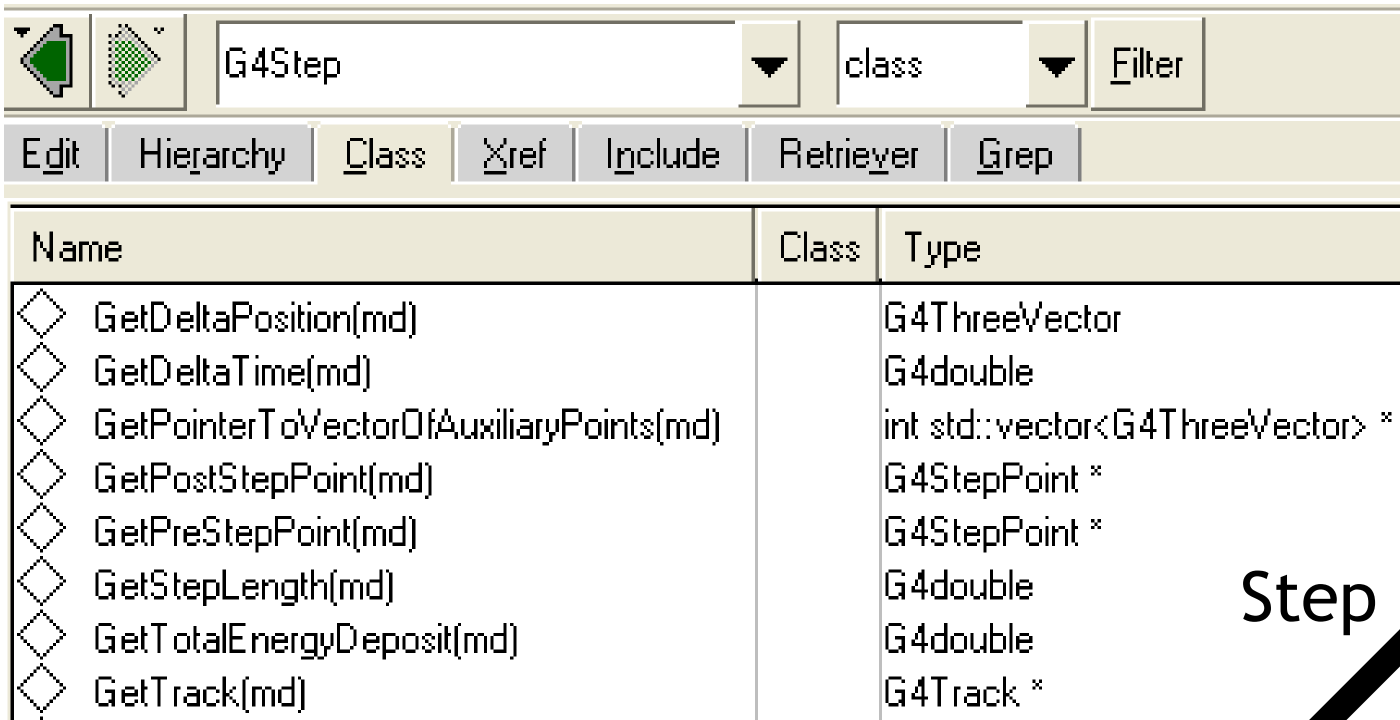
class **G4Step** and class **G4StepPoint**

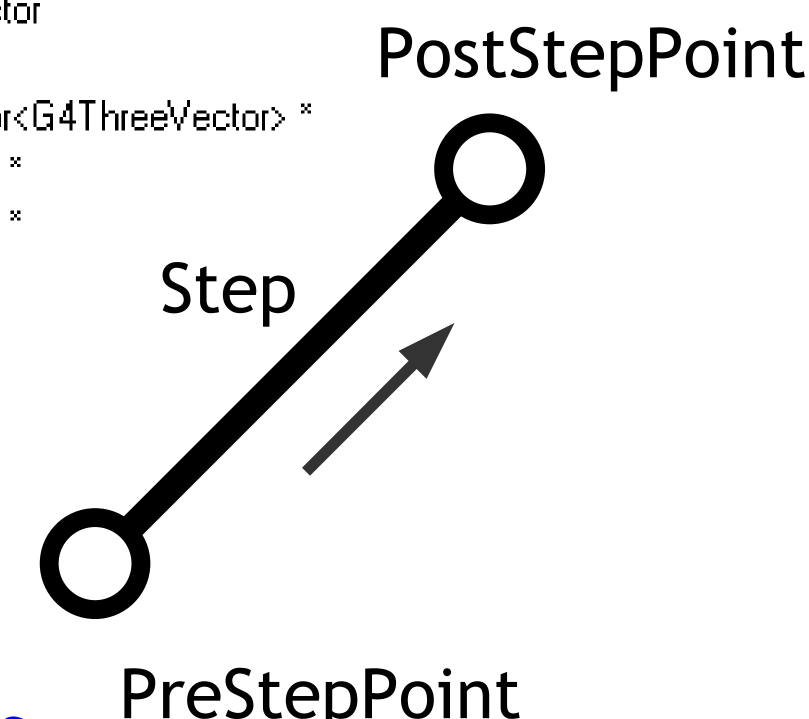


X not necessarily Y!

class G4Step

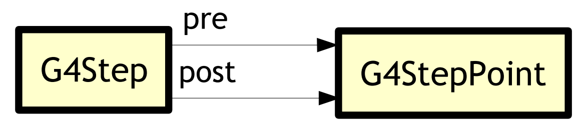


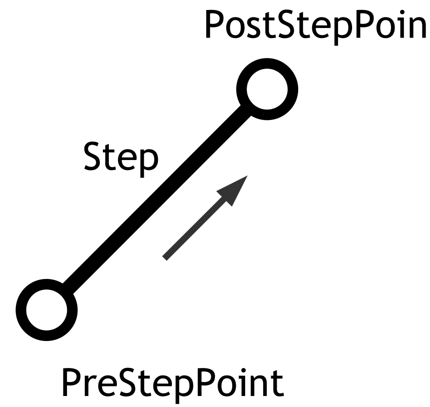


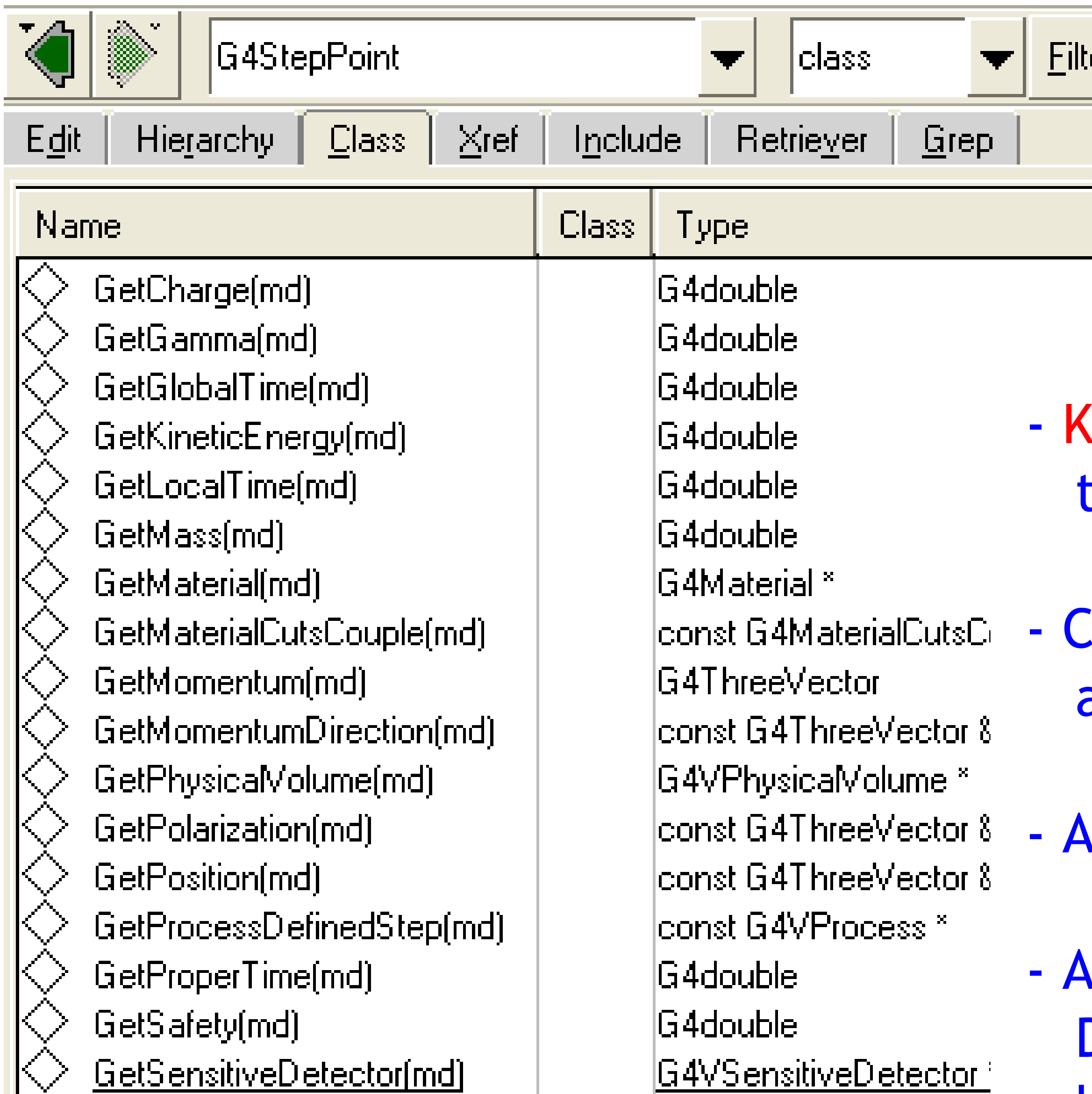


Access to "Delta"-Information, i.e. kinematic differences of the particle at the two StepPoints

class G4StepPoint





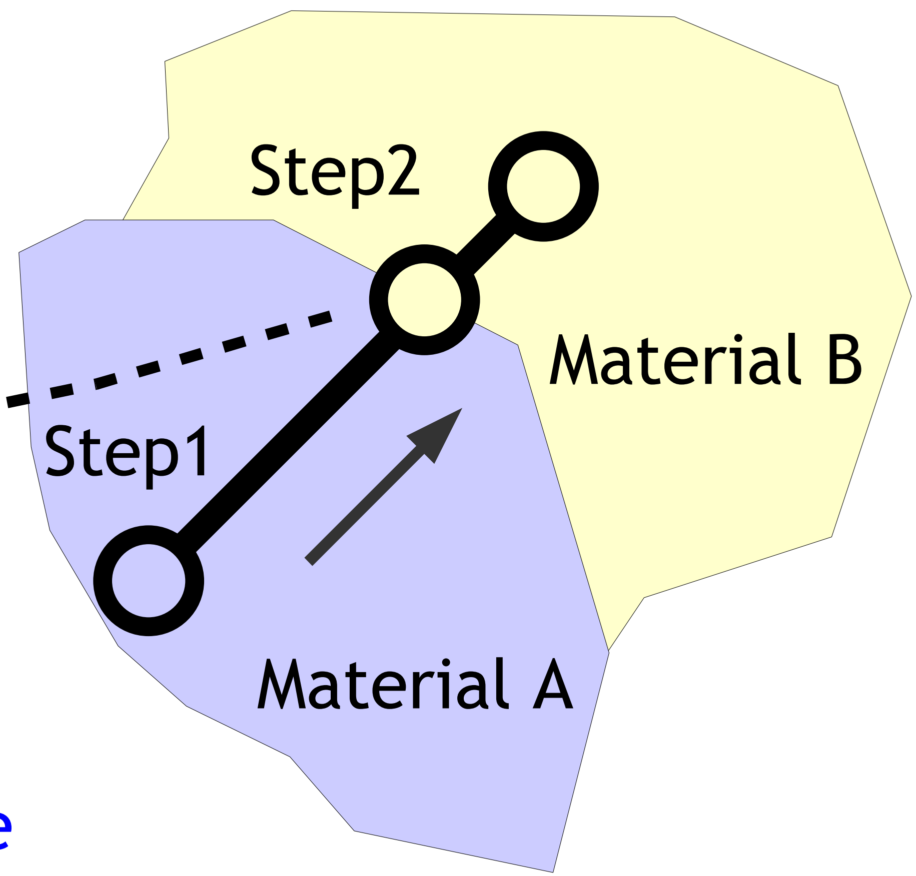


- Kinematic information of the particle at this point
- Convenience methods for accessing G4ParticleDefinition
- Access to Physics Process
- Access to Detector Description Data: G4Material & volume hierarchy (see later)

Material boundaries

G4 will always terminate a step at a material boundary!
The PostStepPoint on the boundary logically belongs
to the next volume being entered.

PostStepPoint1
=
PreStepPoint2



It's good for us to know,
when particles are going
from one region in the
detector to another,
e.g. from a support structure
into a sensitive element!

Recap: Monte Carlo Algorithm

Step 1:

(1) set properties for incident particle (momentum, ..)
(2) get values for λ_i for all relevant processes i=1,2,..,m
(3) for each process i (i=1,2,..,m):
sample L_i from $p_i(x)$
(4) L_c = min(L_i) from all sampled L_i , c in (1,2,..,m)
(5) transport incident particle by L_c
(6) simulate interaction
(7) if particle still exists: goto (1)

λ_1 Proc.1
λ_2 P2

Drawing the random free path length from both processes:

$$p_i(L) = 1 - \exp(-L/\lambda_i)$$

the shortest wins

L_1
L_2

3 GeV

Pre-StepPoint:
initial values for the step

Recap: Monte Carlo Algorithm

(1) set properties for incident particle (momentum, ..)

(2) get values for λ_i for all relevant processes i=1,2,..,m

(3) for each process i (i=1,2,..,m):

    sample L_i from $p_i(x)$

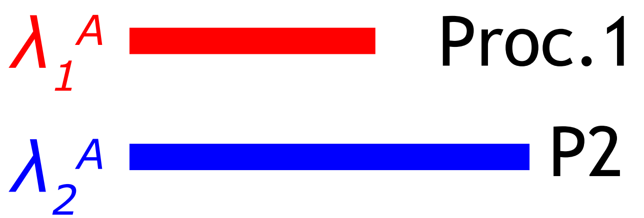
(4) L_c = min(L_i) from all sampled L_i , c in (1,2,..,m)

(5) transport incident particle by L_c

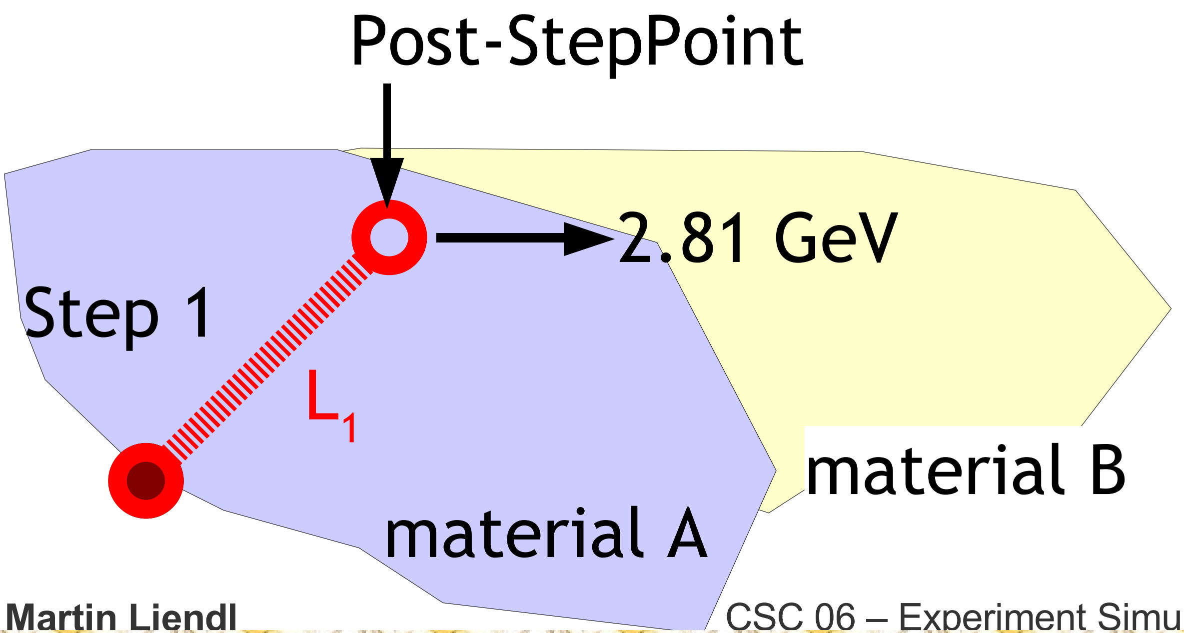
(6) simulate interaction

(7) if particle still exists: goto (1)

Step 1:



values for material A



Recap: Monte Carlo Algorithm

(1) set properties for incident particle (momentum, ..)

(2) get values for λ_i for all relevant processes i=1,2,..,m

(3) for each process i (i=1,2,..,m):

sample L_i from $p_i(x)$

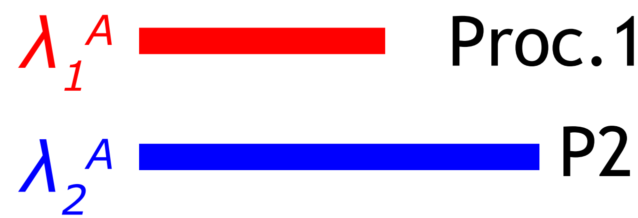
(4) L_c = min(L_i) from all sampled L_i , c in (1,2,..,m)

(5) transport incident particle by L_c

(6) simulate interaction

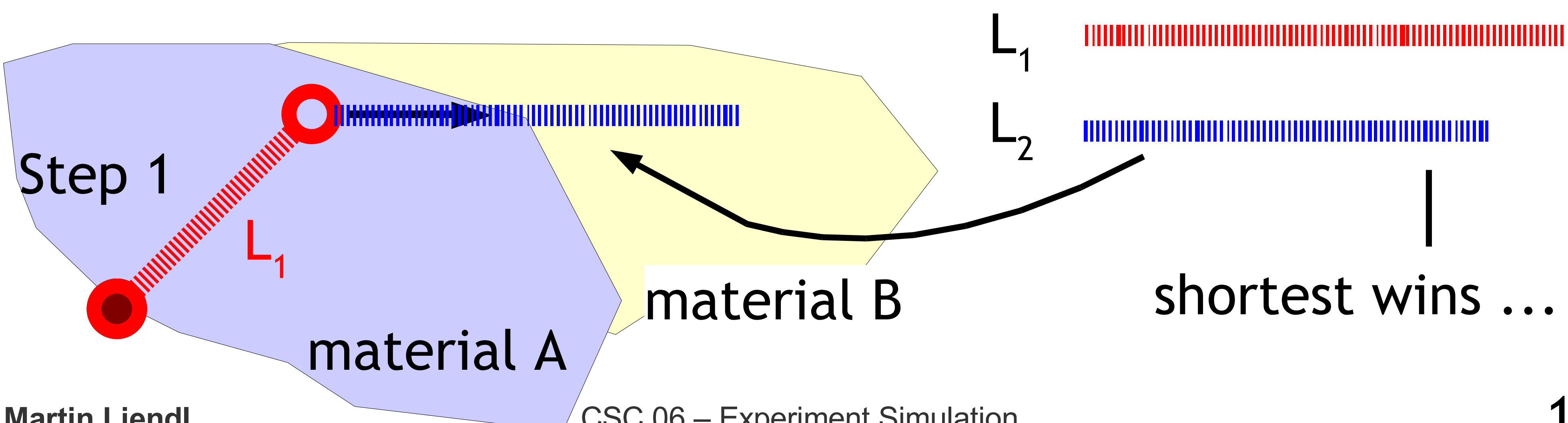
(7) if particle still exists: goto (1)

Step 2:



values for material A

Sampling the step-length:



Recap: Monte Carlo Algorithm

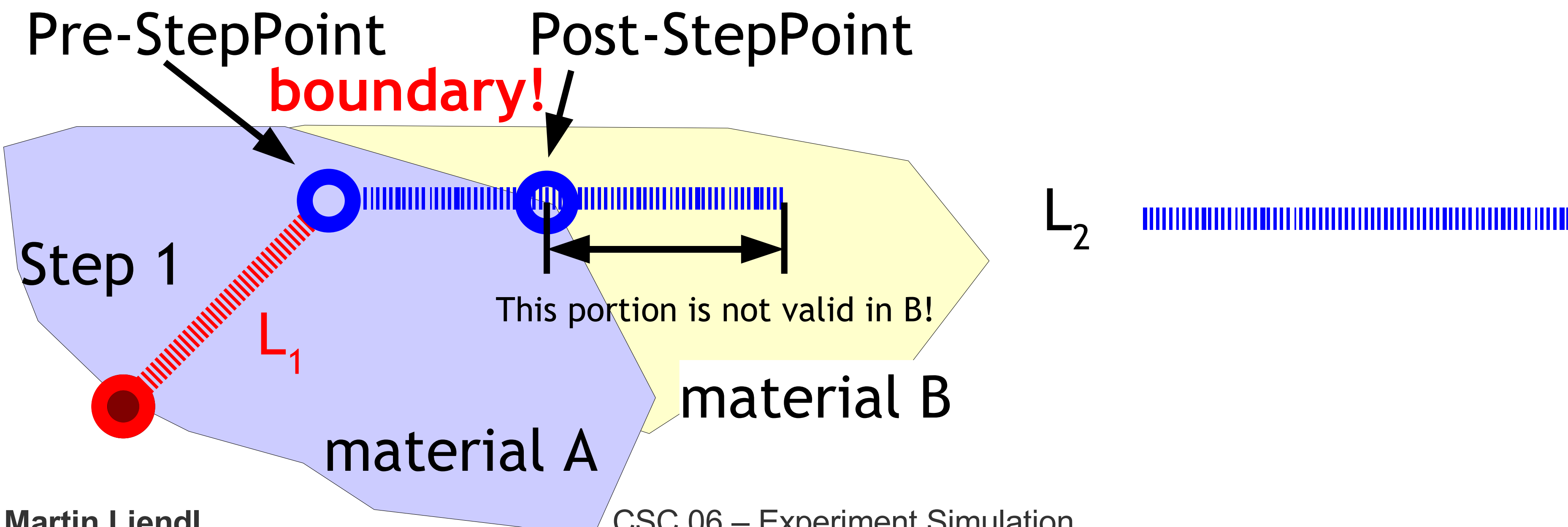
(1) set properties for incident particle (momentum, ..)
(2) get values for λ_i for all relevant processes i=1,2,..,m
(3) for each process i (i=1,2,..,m):
    sample L_i from $p_i(x)$
(4) L_c = min(L_i) from all sampled L_i , c in (1,2,..,m)
(5) transport incident particle by L_c
(6) simulate interaction
(7) if particle still exists: goto (1)

Step 2:

λ_1^A Proc.1

λ_2^A P2

values for material A



Recap: Monte Carlo Algorithm

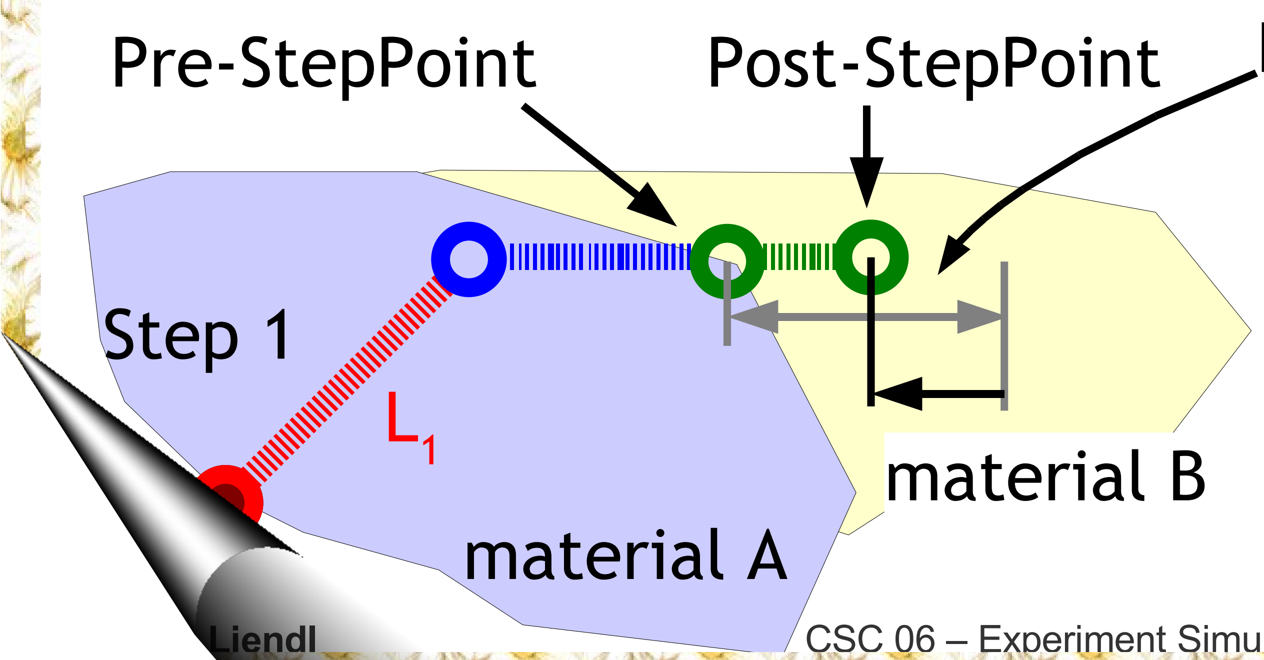
(1) set properties for incident particle (momentum, ..)
(2) get values for λ_i for all relevant processes i=1,2,..,m
(3) for each process i (i=1,2,..,m):
sample L_i from $p_i(x)$
(4) L_c = min(L_i) from all sampled L_i , c in (1,2,..,m)
(5) transport incident particle by L_c
(6) simulate interaction
(7) if particle still exists: goto (1)

Step 3:

λ_1^B Proc.1

λ_2^B P2

values for material B



Rescale step-length:

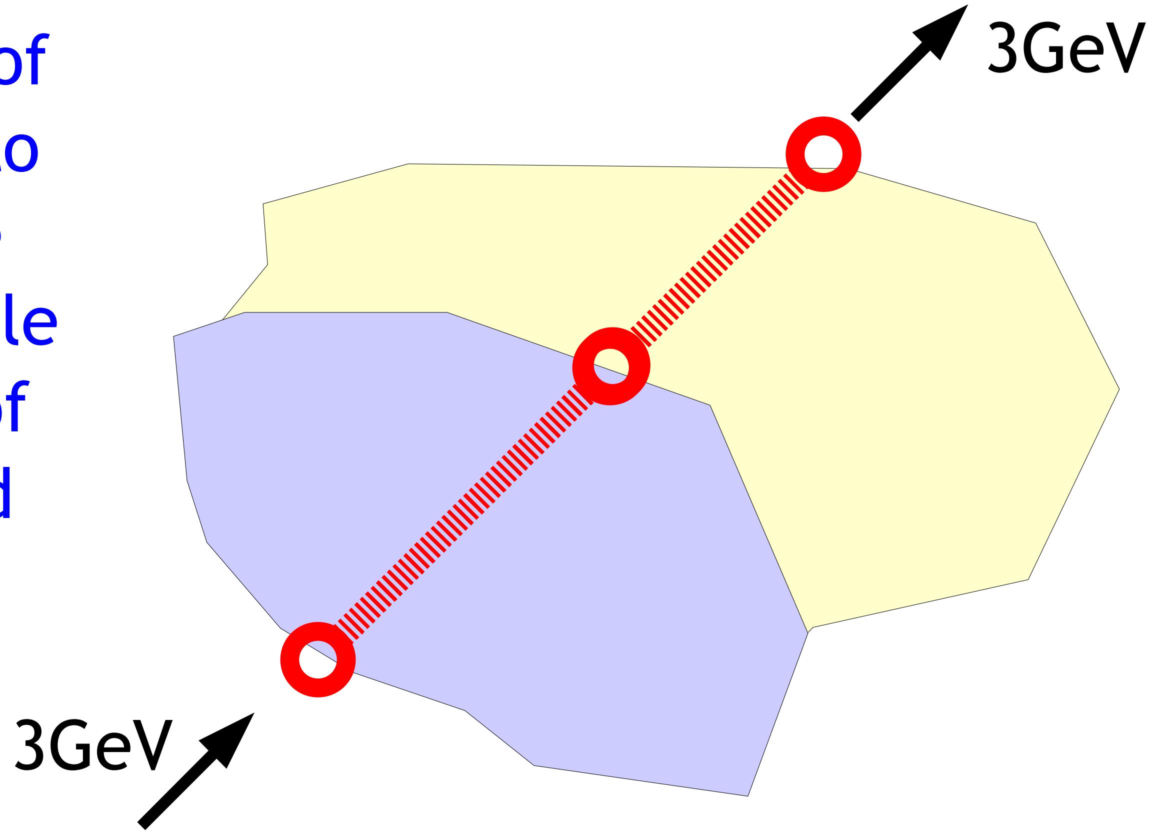
L_2 in material A

in material B

scaling factor: $\lambda_2^B / \lambda_2^A$

True trajectory & G4 steps

Usually, an instance of G4Step corresponds to a fraction of the true trajectory of a particle only in the absence of physics processes and external fields, e.g. magnetic field.



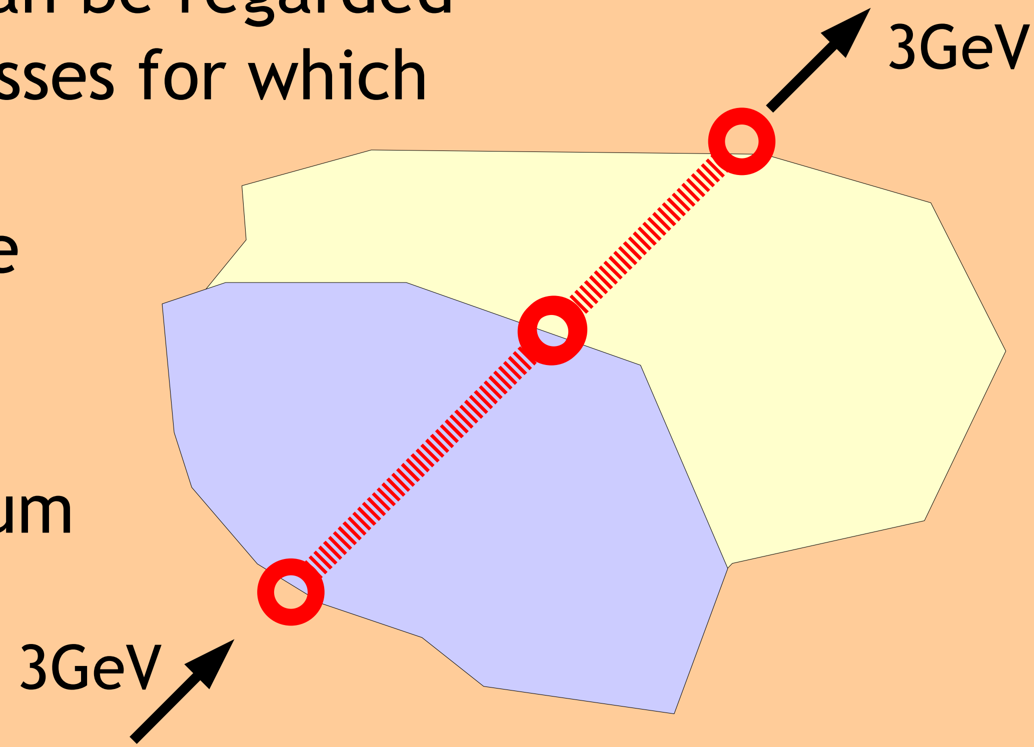
The stepping mechanism of G4 is then comparable to a ray-tracer intersecting geometrical boundaries.

True trajectory & G4 steps

Stepping without physics can be regarded as stepping with one processes for which

p(L) = 1, for L = distance to the next boundary along momentum direction

p(L) = 0, else

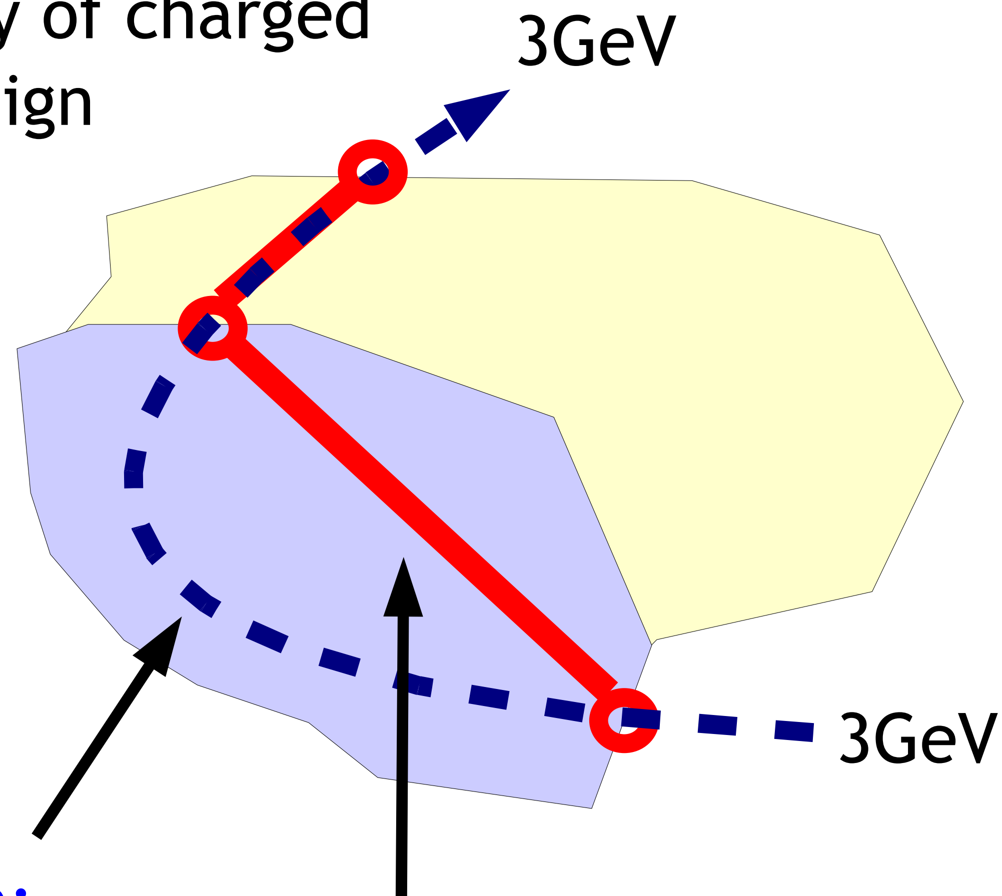


The stepping mechanism of G4 is then comparable to a ray-tracer intersecting geometrical boundaries.

Magnetostatic Fields

Used to bend the trajectory of charged particles according to the sign of their charge.

In G4: static B-fields don´t change the energy of particles (all physics processes assumed to be switched off, i.e. no bemsstrahlung)



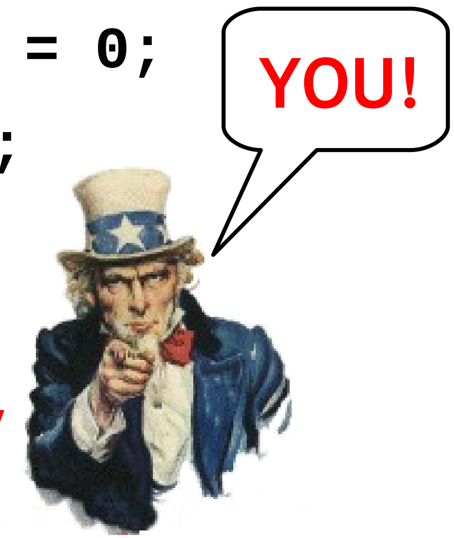
true particle path; exact trajectory is not available for a G4 user!

≠ G4Step

Implementing a G4 B-field

C++ interface:

```
class G4Field {
   virtual void GetFieldValue(
     const double point[4], double * field ) = 0;
  virtual G4bool DoesFieldChangeEnergy() = 0;
};
```



Input, provided by G4 tracking:

`point[0]`, `point[1]`, `point[2]` — global space point (x,y,z)

`point[3]` — global, laboratory time

Output, provided by

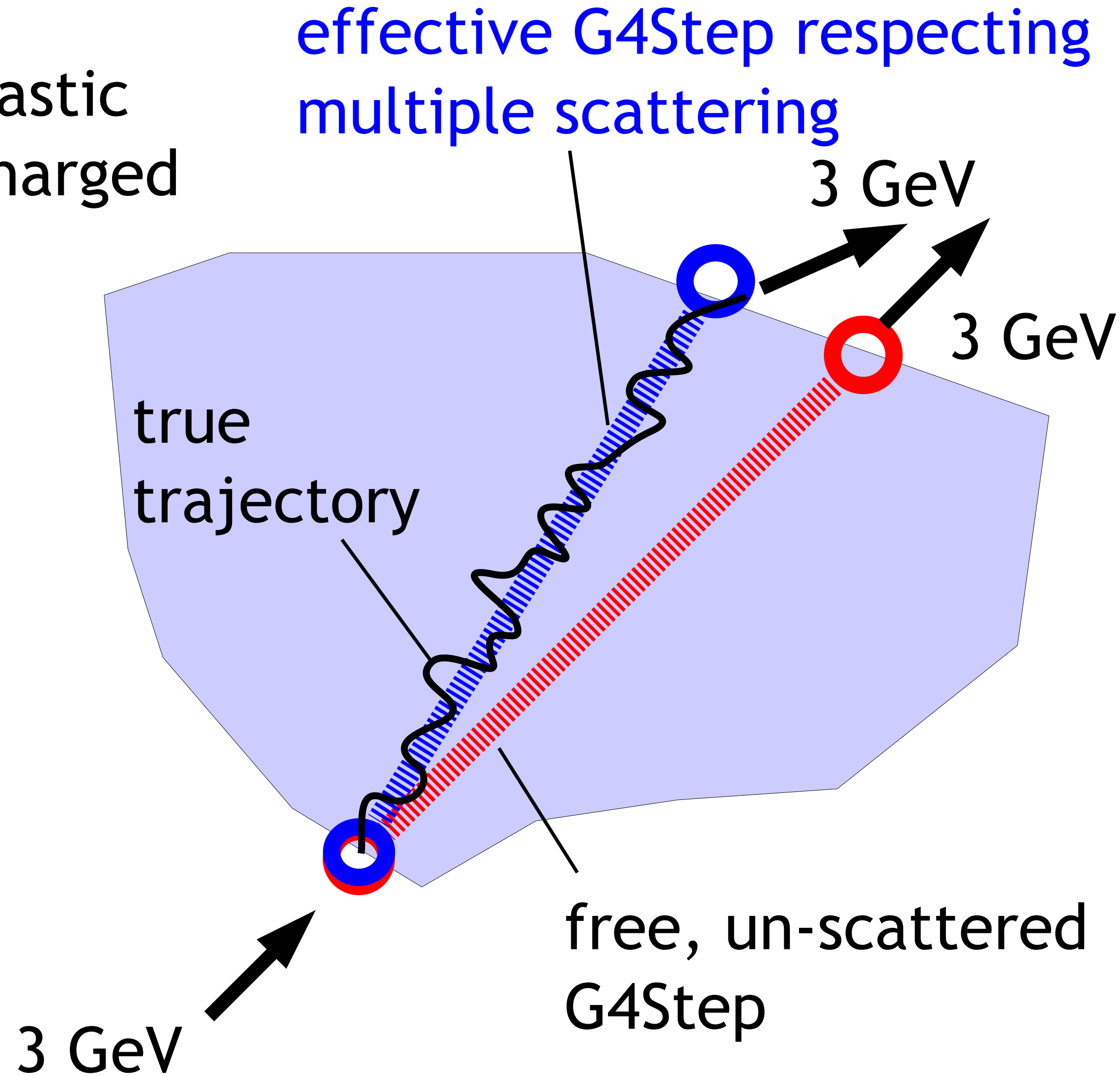
`*(field+0)`, `*(field+1)`, `*(field+2)` — magnetic component

`*(field+3)`, `*(field+4)`, `*(field+5)` — electric component

Multiple (elastic) Coulomb scattering

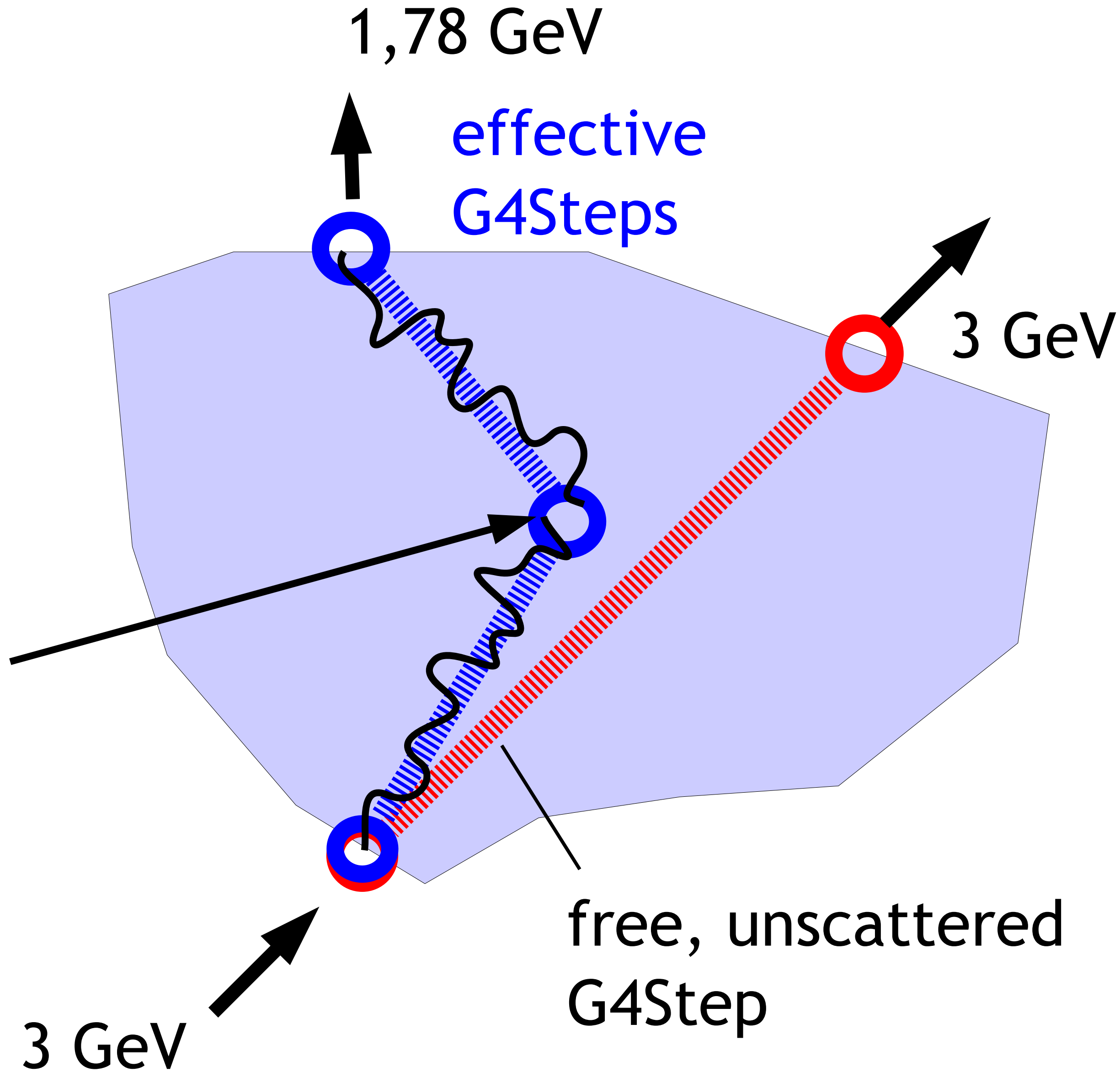
Zick-zack path due to elastic Coulomb scattering of charged particles

It's respected by G4 - lateral displacement, direction change - but not visible to the user!

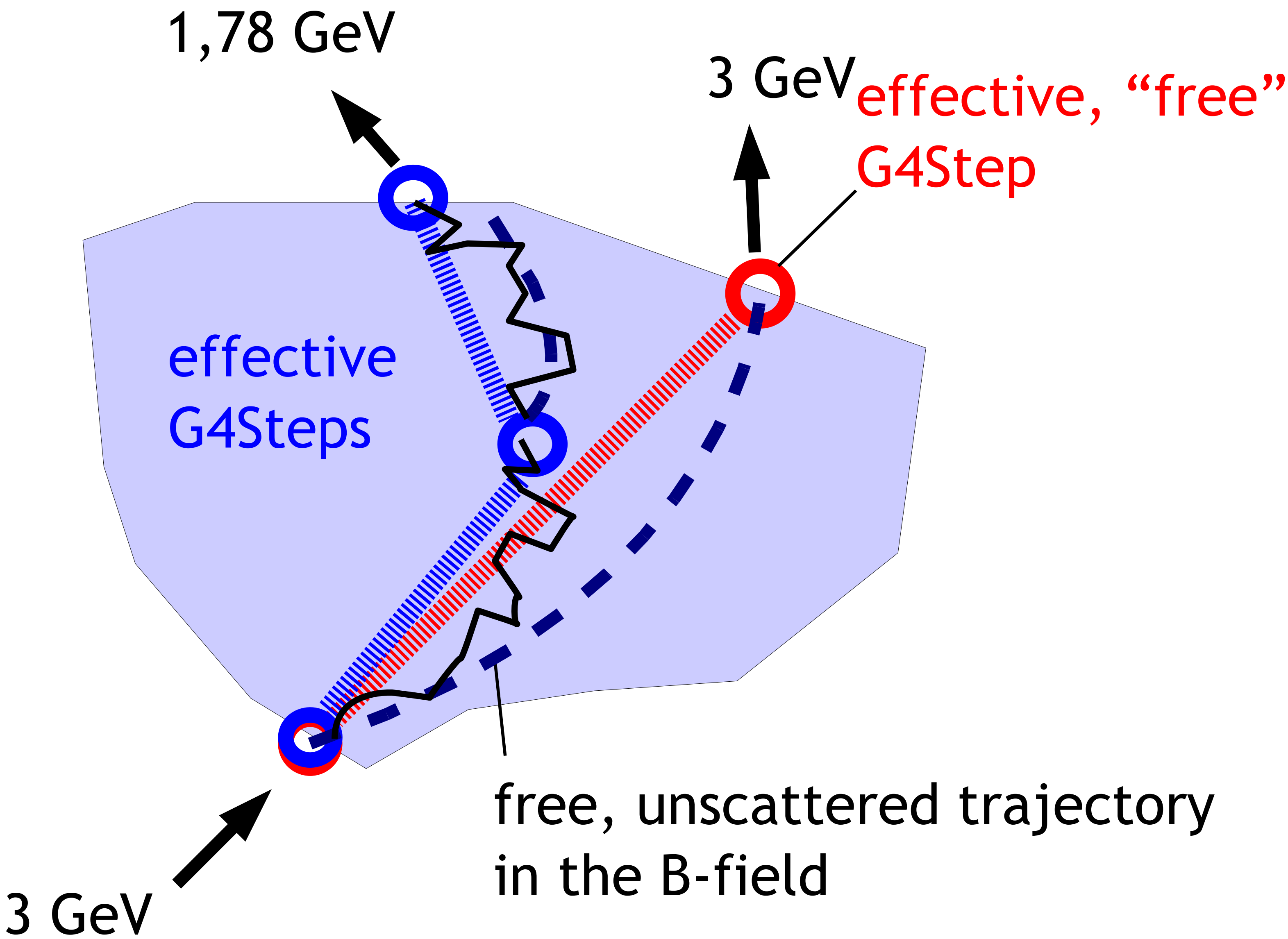


Physics processes included

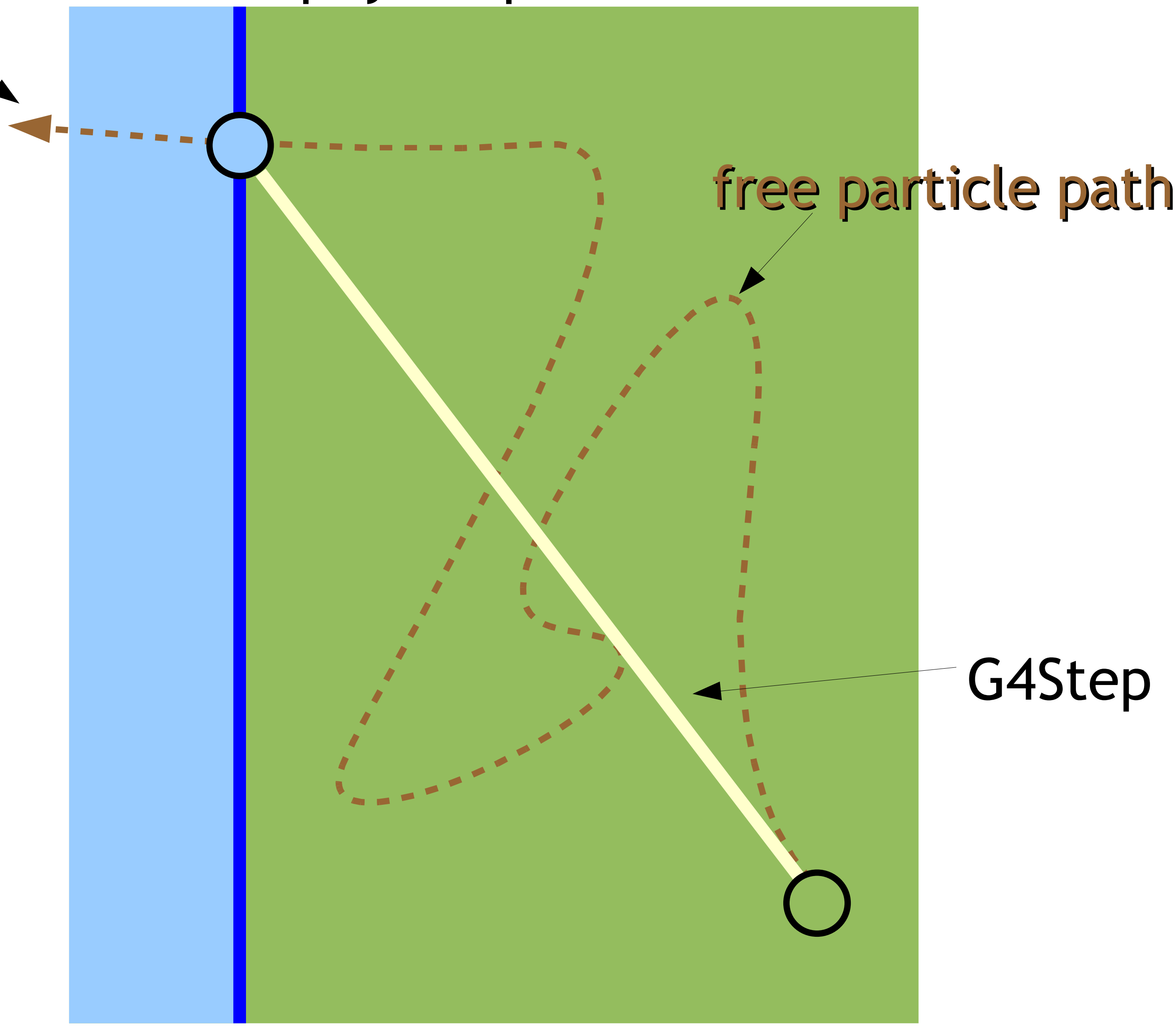
Physics processes limit the step size according to the samples free path length (Monte Carlo algorithm!).



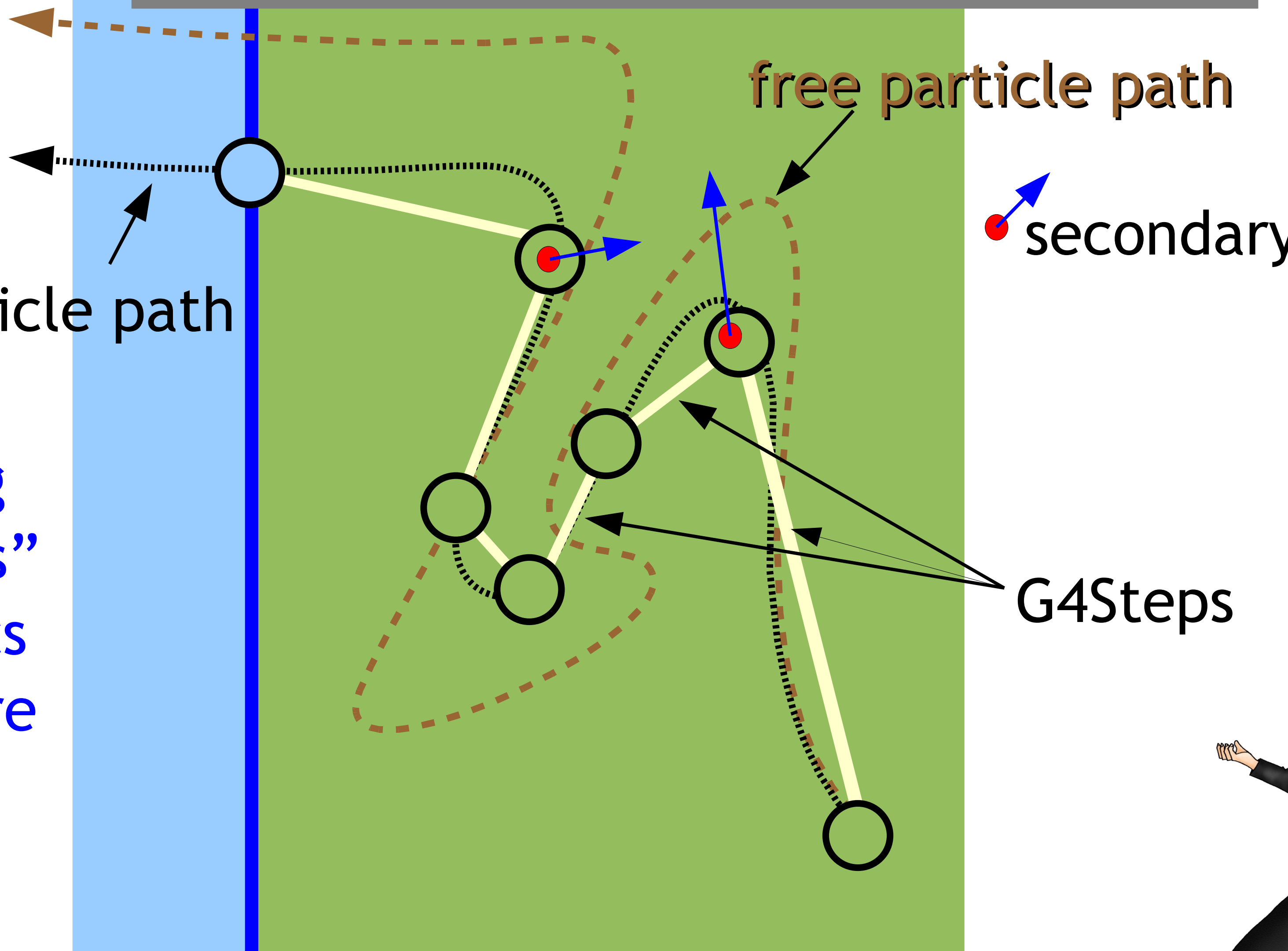
... and the magnetic field ...



Elegant “Fred Astaire - Steps” with B-field and no G4-physics processes



Physics: also the trajectory will be different because the tracked particle looses energy, changes momentum, produces secondaries ...



vs. “fighting Matrix-Steps” when physics processes are enabled

Step related information

- For each step a G4DynamicParticle is being tracked, a new G4Step instance is created representing the step
  - G4Steps only live for the duration of the step
  - they are exposed to the simulation user via various callback actions: user actions (will be discussed soon)
  - a step provides access to the track it belongs to
- For each G4DynamicParticle which is tracked by G4, an instance of G4Track is created
  - this instance lives as long as the particle lives
  - it keeps track of the latest state of the particle
  - provides access to the current step, but does not save previous steps
  - it is exposed to the simulation user via the current step

Step related information

For each step a G4DynamicParticle is beiing tracked,
a new G4Step instance is created representing the step.

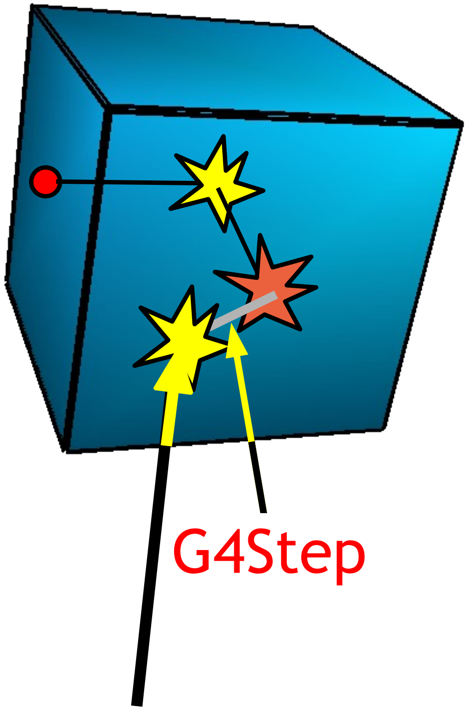
For each G4DynamicParticle which is tracked by G4,
an instance of G4Track is created. This instance lives as long as the particle lives, and it keeps track of the latest state of the particle.

The GEANT4 Physics Model

or

how to cast the last ~100 years of research of particle interaction with matter into G4VProcess

The GEANT4 Physics Model



$p_i(L) = 1 - \exp(-L/\lambda_i)$

Probability of having an interaction due to process i within the path length L

Macroscopic distance G4Step until a microscopic interaction G4VProcess takes place

G4VProcess: microscopic description of the particle interaction with another particle of the material or of the external field

ideally ..

microscopic: according to quantum theory;
→ *Monte Carlo method!*

Trade offs ...

Microscopic processes are particle processes described by quantum theory:

- **creation** and **destruction** of **particles**
- in GEANT4: creation / destruction in the PostStepPoint

To treat every interaction as a particle creation / destruction process is not possible! Takes longer and longer the lower the energies of the secondary particles get!

In G4 an approximation is used:

if (energy of secondary particle > **cut off energy E_c**):

- use Monte Carlo method as described

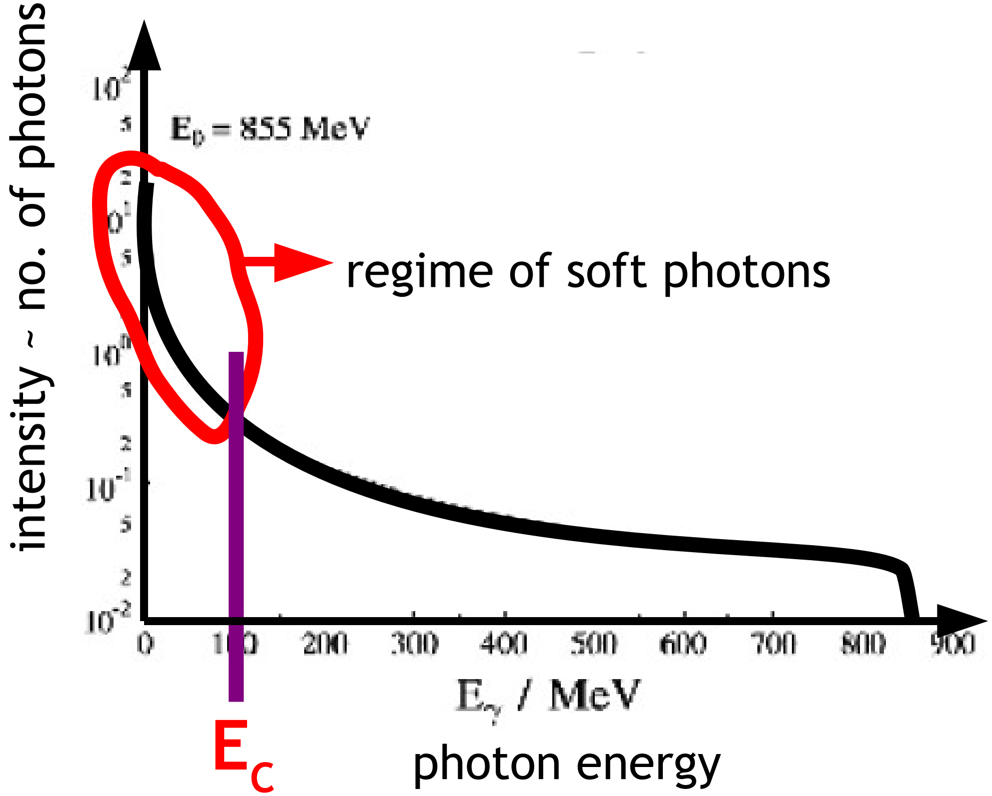
if (energy of secondary particle ≤ **cut off energy E_c**):

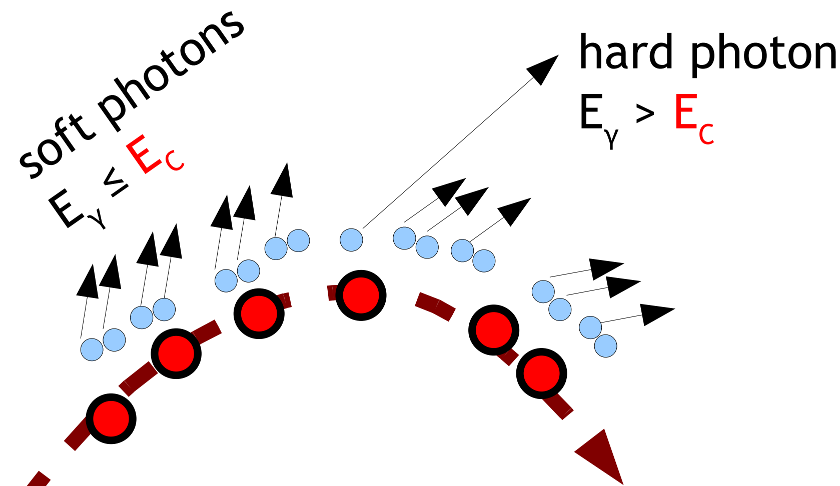
- don't produce a secondary, subtract the average energy loss from the incident particle **along the step**

Trade offs: example

lots of soft (=very low energetic) photons are emitted in the "brems-strahlung" process. They don't change the path of the electron too much.

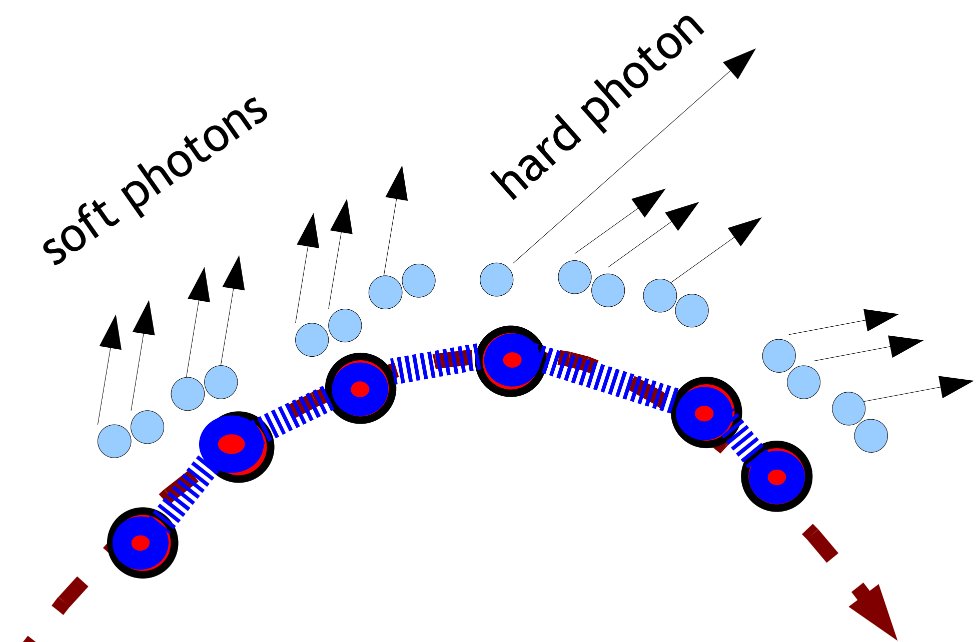
Bremsstrahlung spectrum:





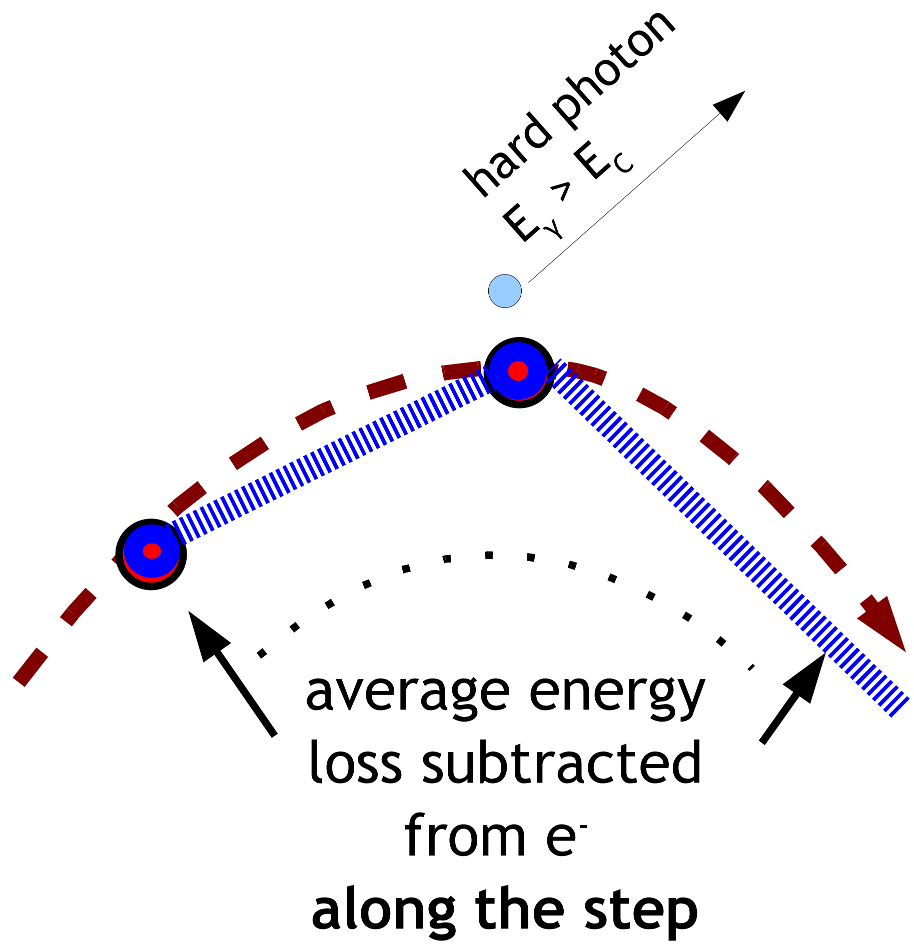
E_C is a cut off energy, a tunable parameter in the physics model to distinguish soft from hard photons ..

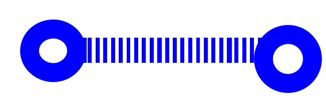
Smaller E_c:



Many more steps,
many more additional
photons to be
simulated for a small E_c

Larger E_c:

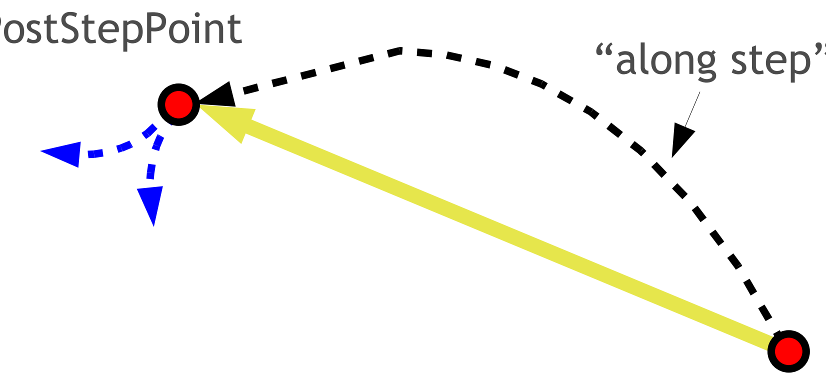




Tunable parameters for G4VProcesses

- E_c makes sense for electromagnetic processes (QED)
  - suffer from "infra-red divergence", i.e. lots of very low energetic secondaries are produced
  - for a particular process, E_c is not directly supplied by the simulation user
    - user sets a lower boundary for the free path length of the secondary particles λ_c
    - if a secondary has a lower free path length than the cut off one, it's not being tracked (except it could reach another material region ..)
    - specifying λ_c should give "more coherent" simulation results, because it corresponds to different E_c in different materials
- Other processes can have different tuning parameters, which we won't cover here

G4VProcess



class G4VProcess:

- mother of all processes in Geant4
- 3 sets of **abstract methods**

A particular implementation of G4VProcess has to implement all of them!

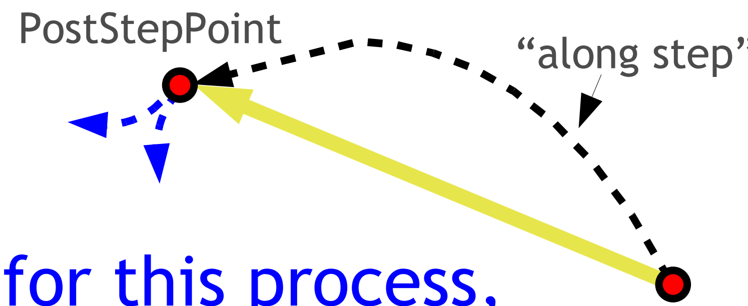
double PostStep
double AlongStep
double AtRest
} GetPhysicalInteractionLength(..)

G4VParticleChange * PostStep
G4VParticleChange * AlongStep
G4VParticleChange * AtRest
} DoIt(..)

won't look at AtRest..
(lack of time)

G4VProcess::PostStep

$p_i(L) = 1 - \exp(-L/\lambda_i)$



returns the mean free path for this process,
used to sample the free path from the exp.distribution

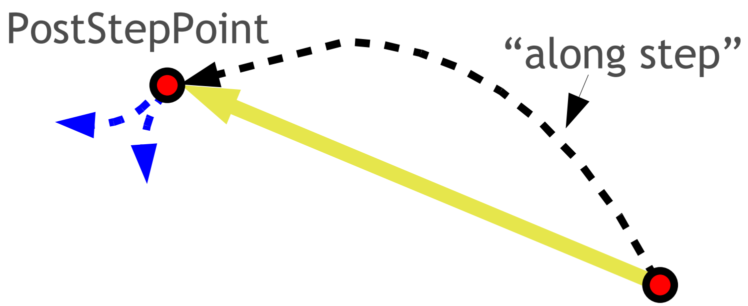
double PostStepGetPhysicalInteractionLength(..) = L_i

Monte Carlo Algorithm

(1) set properties for incident particle (momentum, ..)
(2) get values for λ_i for all relevant processes i=1,2,..,m
(3) for each process i (i=1,2,..,m):
    sample L_i from $p_i(x)$
(4) L_c = min(L_i) from all sampled L_i , c in (1,2,..,m)
(5) transport incident particle by L_c
(6) simulate interaction
(7) if particle still exists: goto (1)

G4VProcess::PostStep

$p_i(L) = 1 - \exp(-L/\lambda_i)$



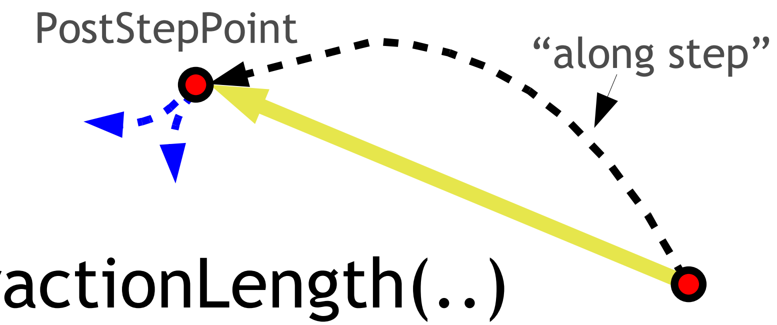
G4VParticleChange * PostStepDoIt(..)

PostStepDoit(..) is called, if this process has returned the shortest λ and thus is responsible for the interaction.

From the **differential cross section**, *the new state of the incident particle is sampled, as well as the states of any secondary particles.*

All instances related to G4Step are updated accordingly (G4Track, ...)

G4VProcess::AlongStep

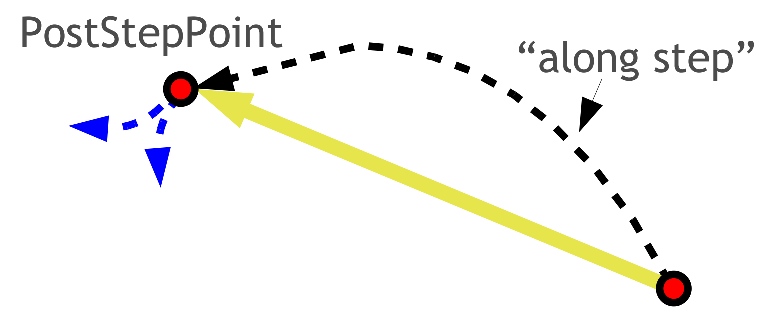


double
G4VProcess::AlongStepGetPhysicalInteractionLength(..)

This method also returns a path length: it's a reasonable limit over which the average energy loss of the particle is still consistent with the cross sections.

If the along step – interaction length is shorter than all post step – free path length, the step is limited by the along step interaction length. No PostStepDoIt() will be triggered in this case.

G4VProcess::AlongStep



The AlongStepDoIts() of **all processes** are always called before the PostStepDoIt() to account for the average description of the processes, i.e. average energy loss along the step.
This is, because the particle is always loosing the average energy along its path / step.

Extensibility

- G4VProcess is an abstract base class
  - G4 provides already a broad spectrum of specific implementations covering most of today's high energy physics range of processes and several low energetic processes
  - G4VProcess is the only interface to the tracking algorithm in G4
    - G4 doesn't “know” which specific process was limiting the step length, ...
- If the provided processes are insufficient:
  - simply derive class YourProcess : public G4VProcess
  - implement the xxGetPhysicalInteractionsLength(), xxDoIt() according to your physics model
  - add the process to the process manager in G4 and it will be respected as any other process!!

Excerpt: subclasses of G4VProcess

- G4VContinuousDiscreteProcess
  - G4hLowEnergyLoss
    - G4hLowEnergyIonisation
      - G4ionLowEnergyIonisation
  - G4IonisationByLogicalVolume
  - G4MultipleScattering52
  - G4VeLowEnergyLoss
    - G4eLowEnergyLoss
      - G4LowEnergyBremsstrahlung
      - G4LowEnergyIonisation
      - G4PenelopeBremsstrahlung
      - G4PenelopeIonisation
  - G4VEnergyLoss
    - G4VeEnergyLoss
      - G4eBremsstrahlung52
      - G4eIonisation52
    - G4VhEnergyLoss
      - G4hIonisation52
    - G4VMuEnergyLoss
      - G4MuBremsstrahlung52
      - G4MuIonisation52
      - G4MuPairProduction52
    - G4VPAIenergyLoss
      - G4PAIonisation
    - HpdSiEnergyLoss
  - G4VEnergyLossProcess
    - G4eBremsstrahlung
      - G4eBremsstrahlungCMS
    - G4eIonisation
    - G4hIonisation
    - G4ionIonisation
    - G4MuBremsstrahlung
    - G4MuIonisation
    - G4MuPairProduction
  - G4VMultipleScattering
    - G4MultipleScattering

Excerpt: subclasses of G4VProcess

- G4VContinuousDiscreteProcess
  - G4hLowEnergyLoss
    - G4hLowEnergyIonisation
      - G4ionLowEnergyIonisation
- lume
- G4eLowEnergyLoss
  - G4LowEnergyBremsstrahlung
  - G4LowEnergyIonisation
  - G4PenelopeBremsstrahlung
  - G4PenelopeIonisation
- VeEnergyLoss
  - G4eBremsstrahlung52
  - G4eIonisation52
- VhEnergyLoss
  - G4hIonisation52
- VMuEnergyLoss
  - G4MuBremsstrahlung52
  - G4MuIonisation52
  - G4MuPairProduction52
- VPAIenergyLoss
  - G4PAIonisation
- dSiEnergyLoss
  - G4eBremsstrahlung
    - G4eBremsstrahlungCMS
  - G4eIonisation
  - G4hIonisation
  - G4ionIonisation
  - G4MuBremsstrahlung
  - G4MuIonisation
  - G4MuPairProduction
- G4MultipleScattering

Categories

- Electromagnetic
  - standard
  - low energy
- Hadronic
  - pure hadronic
  - radioactive decay
  - photo- and electro-nuclear
- Decay
- Optical photon
- Parameterization
- Transportation

Physics initialization

- The simulation user is free to choose any of the available processes to participate in the simulation
  - this is done in a user initialisation class (see later)
  - great flexibility to build optimized physics lists
- Physics processes are coupled to the particles they influence
  - the particle types which should participate in the simulation are also declared by the simulation user in the same initialisation class
- G4 comes with a set of pre-defined physics lists
  - good starting point for nowadays HEP experiments
  - fine tuned by collaborations to satisfy their needs

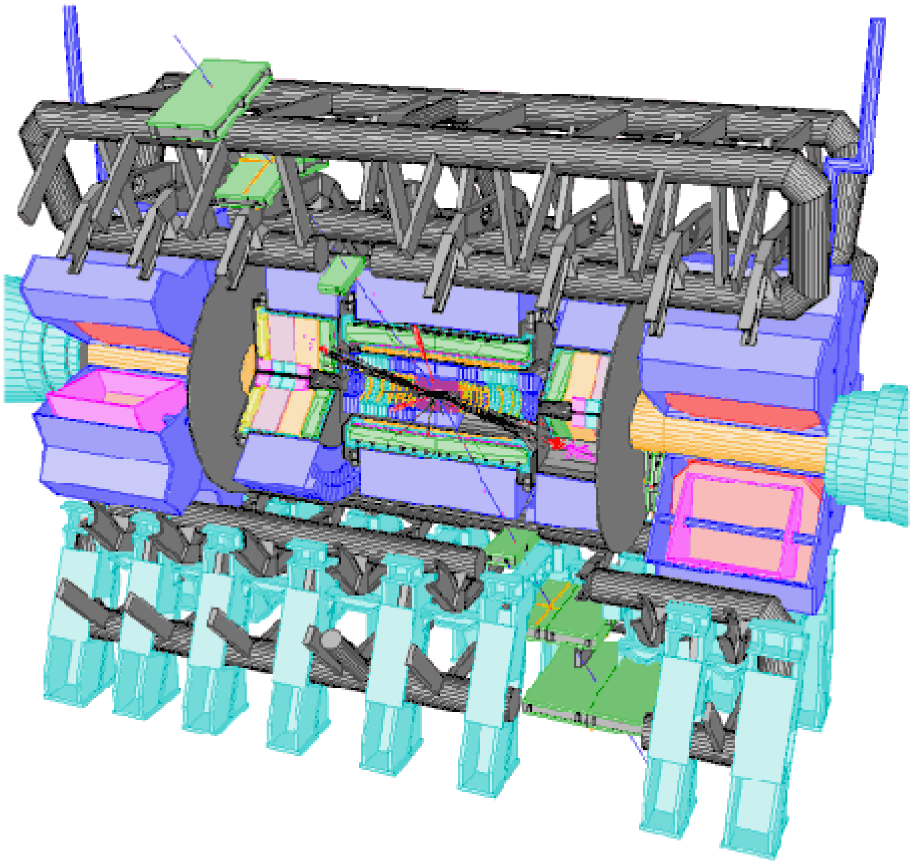
Declaration of processes

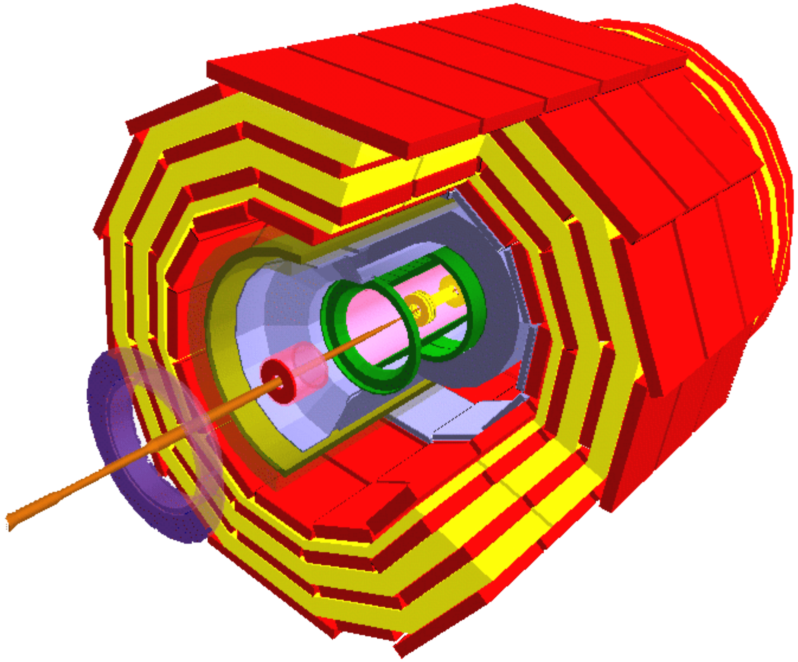
- The G4 user must provide an implementation of the **`G4VUserPhysicsList`** interface. There he declares:
  - which particles he wants to have included in the simulation
  - and for each particle, which physics process it should be subjected
  - in case a process of particle A could produce a particle B, but you have not declared to use particle B, your simulation might crash!!
- Transportation, i.e. the geometrical tracking of any particle in an (optional) external field, is a process and must be declared **for each particle**!
- No more details in this lectures - physics is for experienced users
  - code examples will be provided in our exercises

- A - Physics Model in GEANT4
  - Stepping: moving in free path lengths
  - Physics Processes
- B - Detector Description in GEANT4
  - Solids/Shape Model
  - Volumes
  - Hierarchy of Volumes
- Combining A + B
  - Stepping through a detector description

Detector Description

Have you ever wondered where such pictures come from?



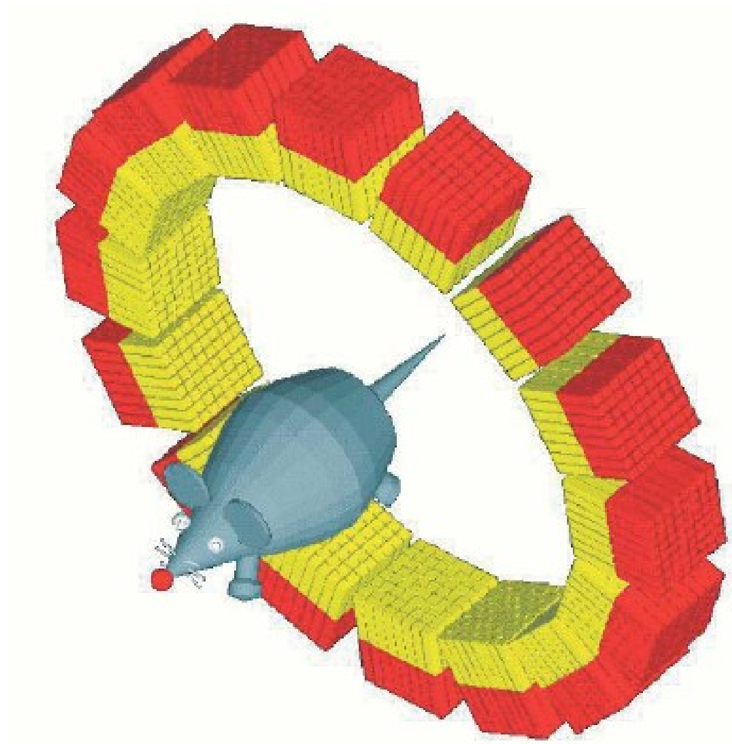


In many cases these are visual representations of the detector representations underlying the detector simulation program!

... or this one:

Geant4 development and physics generators

GEANT4 development. The major new achievement by the HIP team in 2005 was the con-



Geant4 is a showcase example of technology transfer from particle physics to other fields such as medical imaging. Simulation of a small PET scanner using Geant4. (Courtesy of the OpenGate Collaboration.)

(from the “Annual Report 2005”, Helsinki Institute of Physics)

General Requirements

- Must be "realistic" for the physicist / physics to be studied
  - Particles tracked through the detector must "see" / "feel" a realistic environment in order to deliver realistic simulations results
  - One has to know which details of the detector are essential and which can be safely ignored with respect to physics
    - e.g., one often can safely ignore glass fiber cables, while inter crystal gaps (of the same order of magnitude as the cables!!) in the electromagnetic calorimeter can severely influence the prediction of the energy resolution ...

Needs lots of experience and expertise!

General Requirements

- Description must be expressed in the GEANT4 model imposing several constraints!
  - For efficiency reasons the tracking algorithm assumes several properties of / constraints on the geometry
  - Violations of these rules lead to unpredictable simulation results if not program crashes!
  - Example: Geometry must fit into memory ...
    - Simulation speed, coherency of simulation
  - Unfortunately, Geant4 does not help you very much to enforce these constraints ...

Often, we cannot model the detector 1:1 – need additional effort to fulfil these constraints, e.g. introduction of artificial envelope volumes

General Requirements

- Must be compatible with detector models employed by other experiment software:
  - **<u>Engineering data / CADs:</u>** describe the same detector as it is actually planned to be built!
  - **<u>Event reconstruction:</u>** In your simulation, if you save the information that a track has passed silicon wafer No. 4032, the reconstruction SW should also know about No. 4032 (especially where in 3D space it is ...)
  - **<u>Field calculations:</u>** very often a specialized SW suite is necessary to perform complex magnetic field calculations using its own representation of “the detector”

Not all compatibilities / dependencies can be treated automatically!

Focus on the G4 geometry model

Realistic description

GEANT4 conforming description

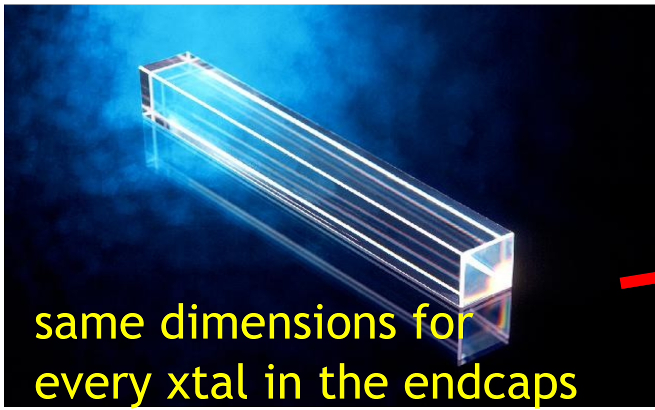
Compatibility between different descriptions of the same detector

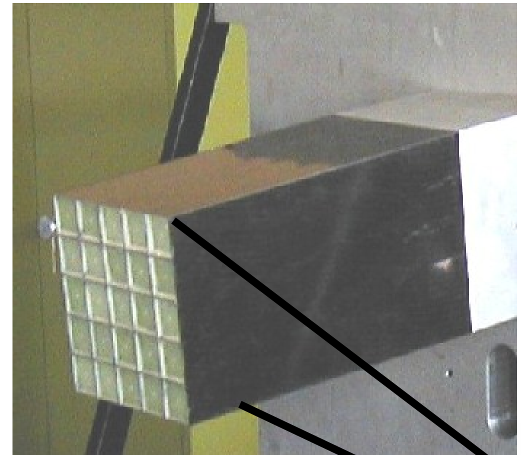
G4 Geometry Model

- Volume model
  - volume = shape / solid filled with a material
- Hierarchical model
  - hierarchies of volumes
  - graph character: a volume in the model can represent multiple “real” volumes
- Not only geometrical aspects; anchor points for
  - sensitive detector / hit collectors
  - digitization modules
  - visualization aspects (visibility, colour, ..)
- Extensible model
  - define custom shapes / solids

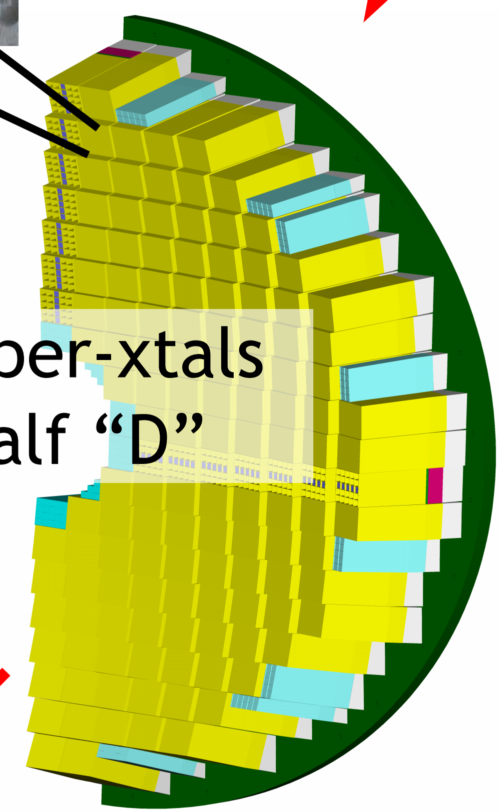
Example: CMS Endcap ECal

Lead-tungstate crystal
~2 x 2 x 20 cm

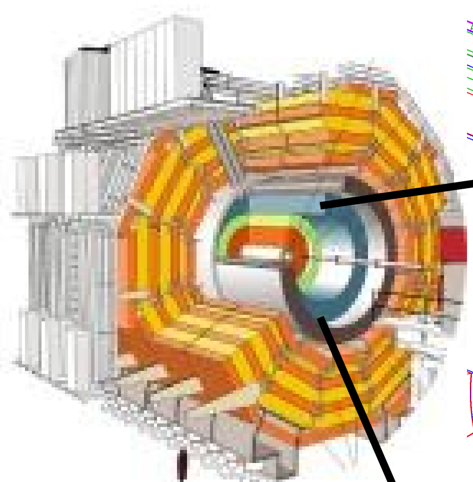


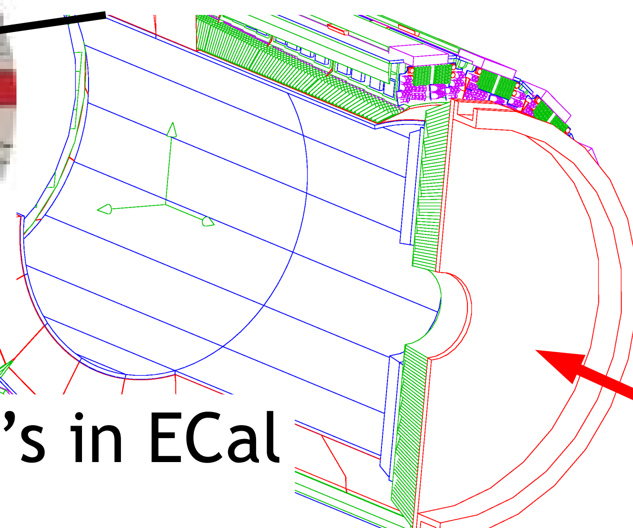


5 x 5 crystals
in a super-xtal



CMS ECal endcaps consist of
~5x5x140x4 = 14000 crystals!





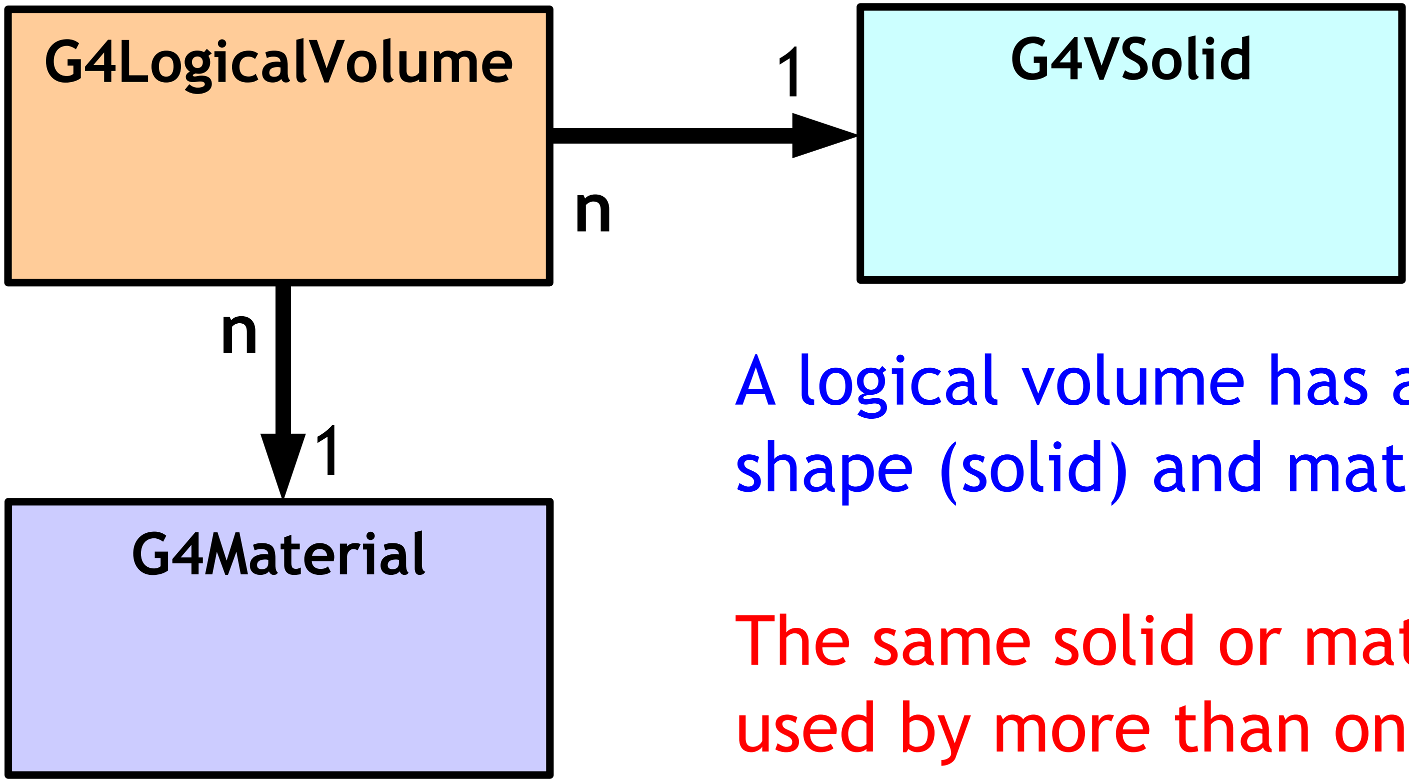
~140 super-xtals
in a half "D"

4 half "D"s in ECal

Logical volume

- Basic ingredient of the G4 geometry model is the **G4LogicalVolume**
- Mandatory and optional features
  - **Mandatory** properties of a volume:
    - Has a **shape**, i.e. there's an inside and an outside of each volume
    - Has a **material**, i.e. the material that fills the inside of the volume homogeneously
  - **Optional** properties:
    - Can contain **children volumes** ("daughter volumes" in G4 lingo) placed in the inside of the parent volume
    - **External field description** attached to the volume
    - Data extraction facilities: sensitivity & digitization modules

Logical Volume



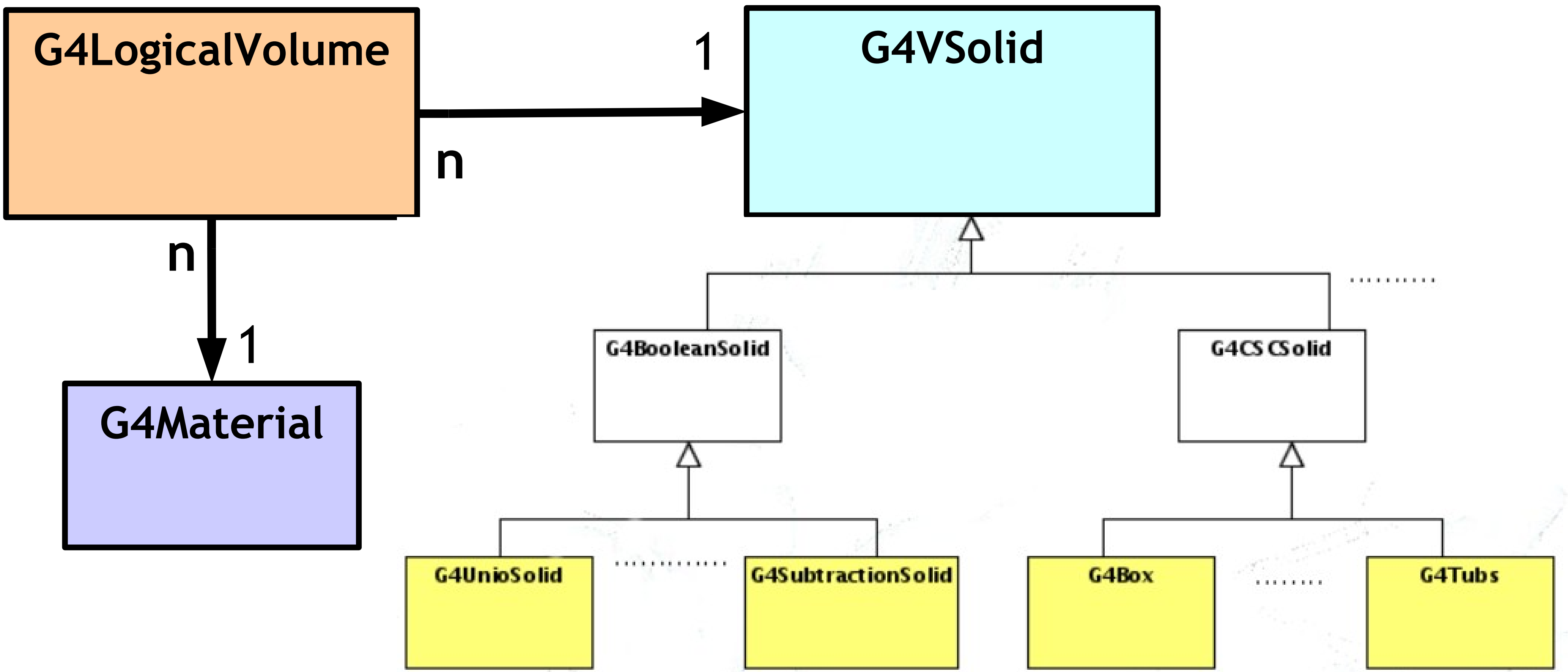
A logical volume has access to its shape (solid) and material.

The same solid or material can be used by more than one logical volume.

A solid or material does not know to which logical volumes it belongs.

Remember:
G4Material – we know already about it!

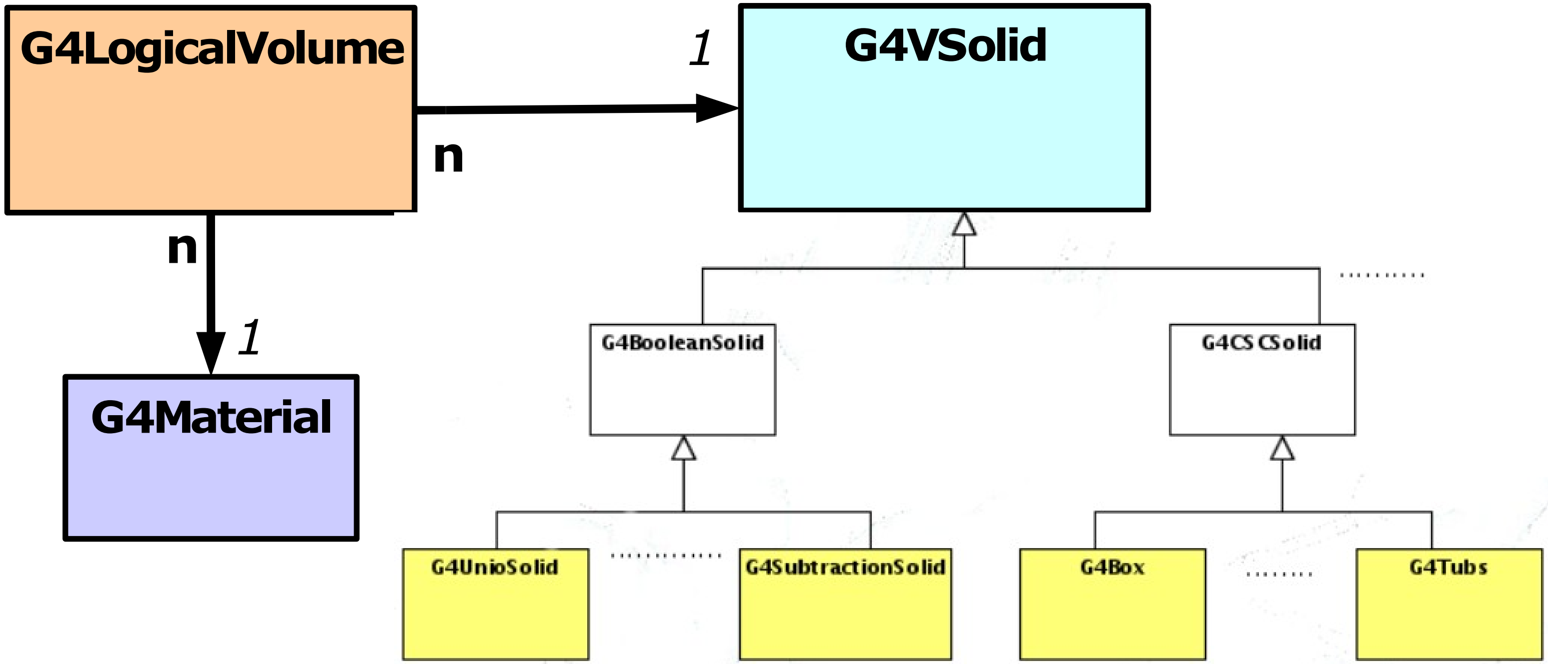
Solids



GEANT4 comes with a wide variety of solids

The G4 Kernel uses solids only via their common G4VSolid base class!

Solids

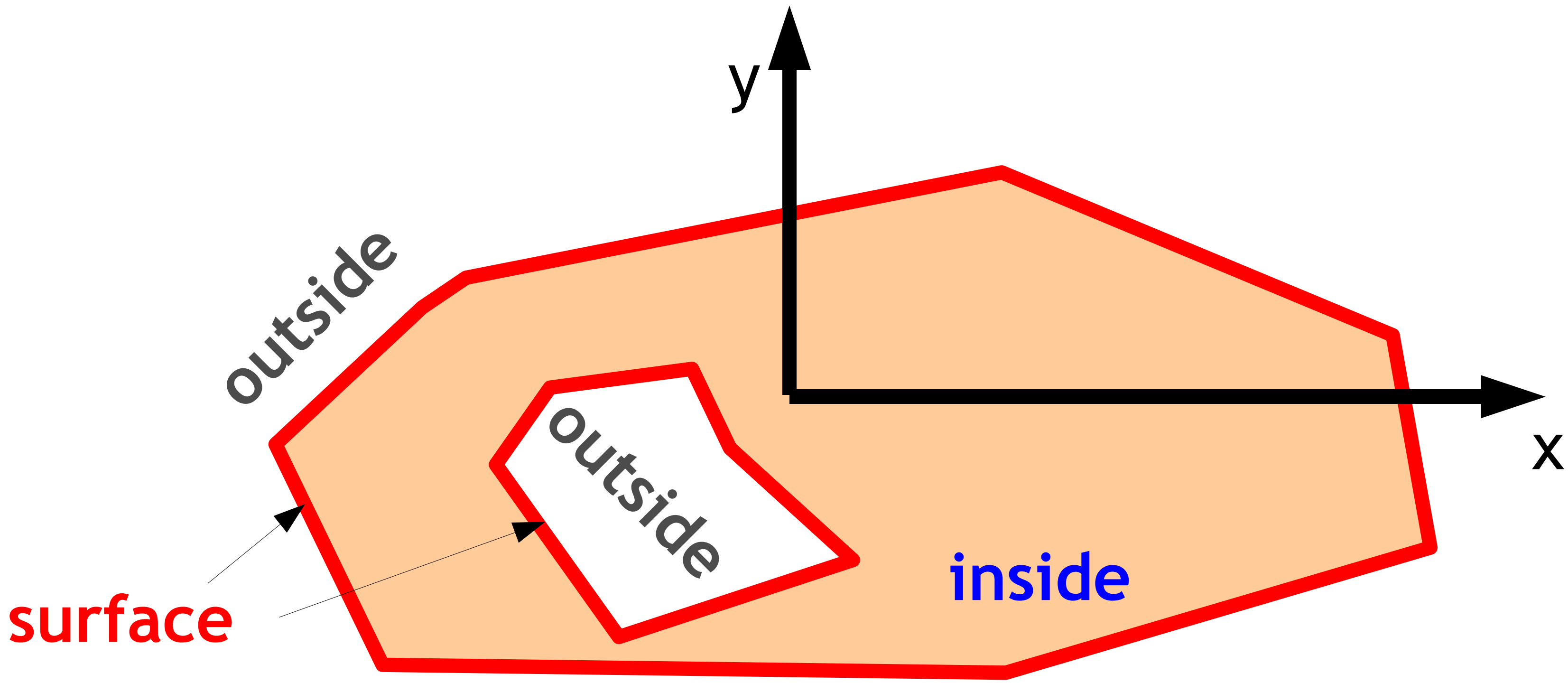


We can extend GEANT4 by adding own classes of solids!!

The G4 Kernel uses solids only via their common G4VSolid base class!

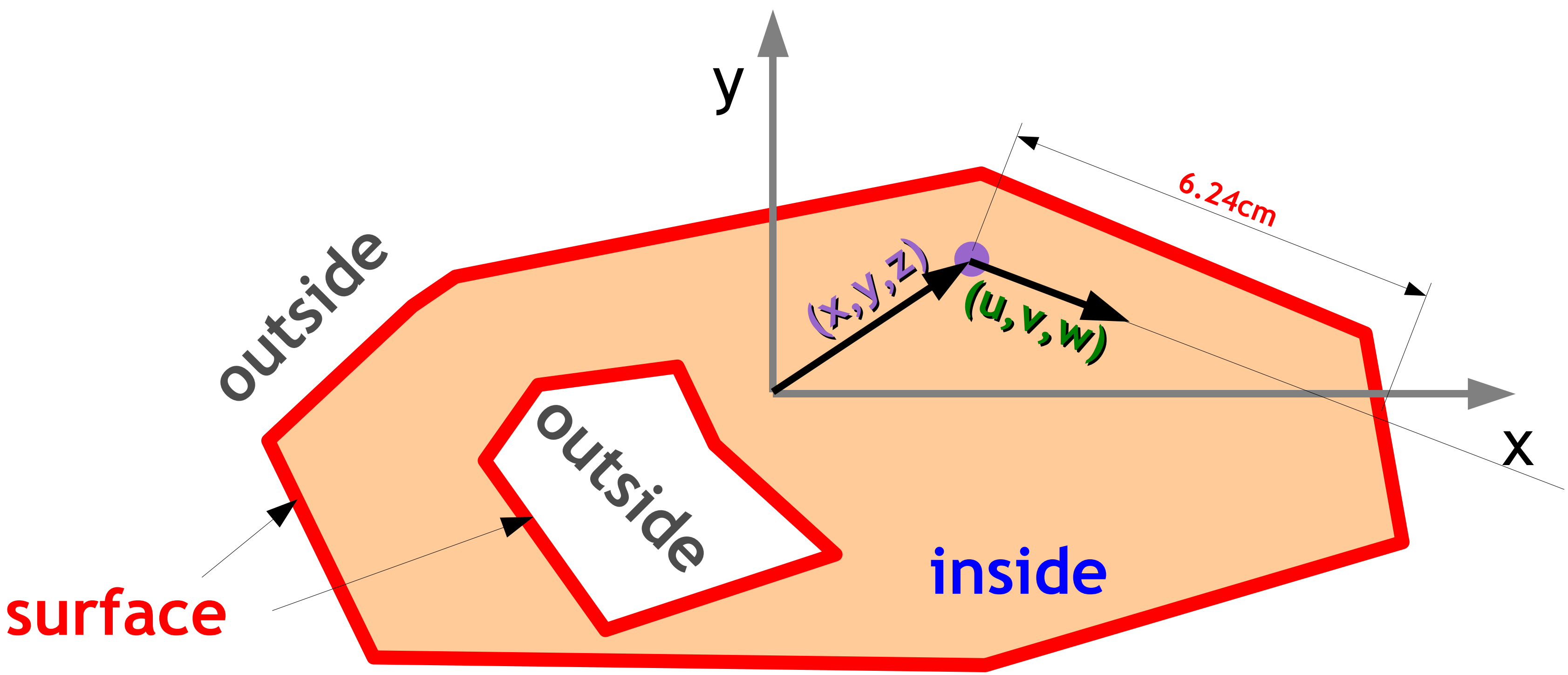
Properties of a G4VSolid

- A G4VSolid has
  - a well defined inside, outside, and boundary/surface within certain numerical tolerances
  - a Cartesian system of reference



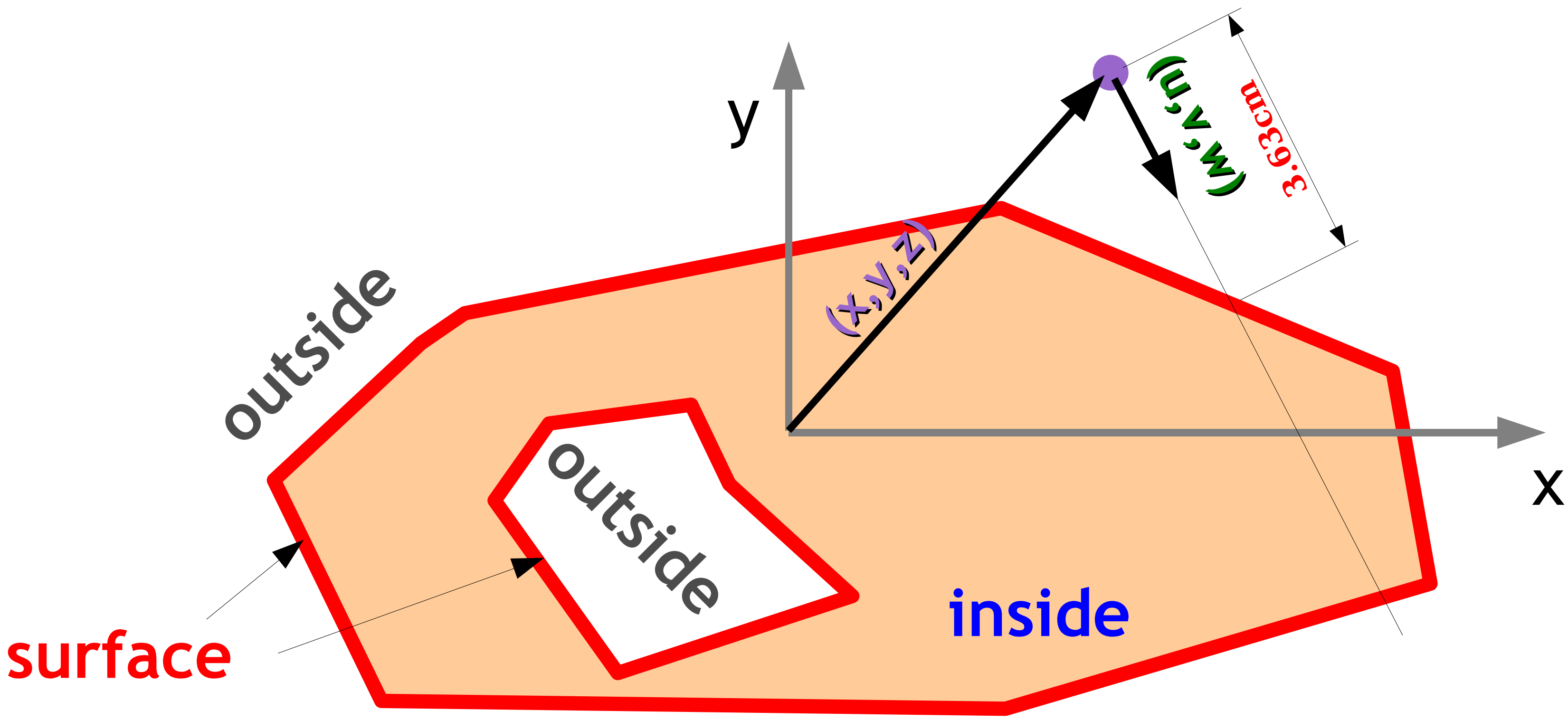
Properties of a G4VSolid

- A G4VSolid knows, given any point (x, y, z)
  - inside the solid: the distance to its boundary to the outside
  - outside the solid: the distance to its boundary to the inside



Properties of a G4VSolid

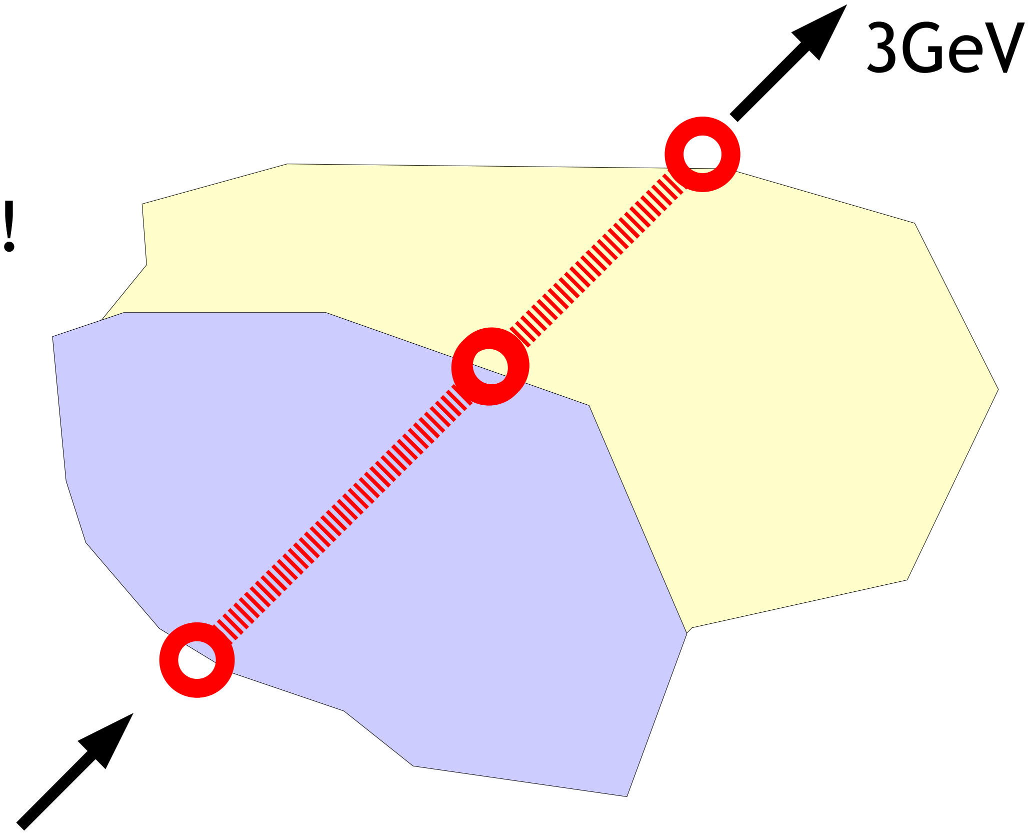
- A G4VSolid knows, given any point (x, y, z)
  - inside the solid: the distance to its boundary to the outside
  - outside the solid: the distance to its boundary to the inside



Remember?

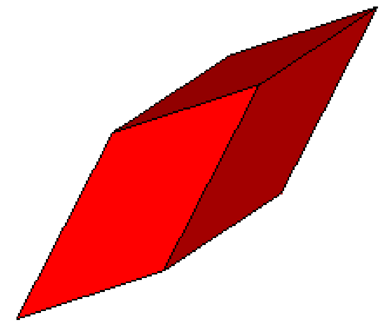
The **stepping algorithm** needs to interrupt a trajectory on boundaries!

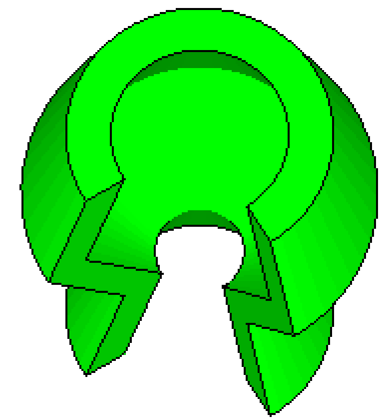
The solid of the volume is asked for distance information towards its boundary!

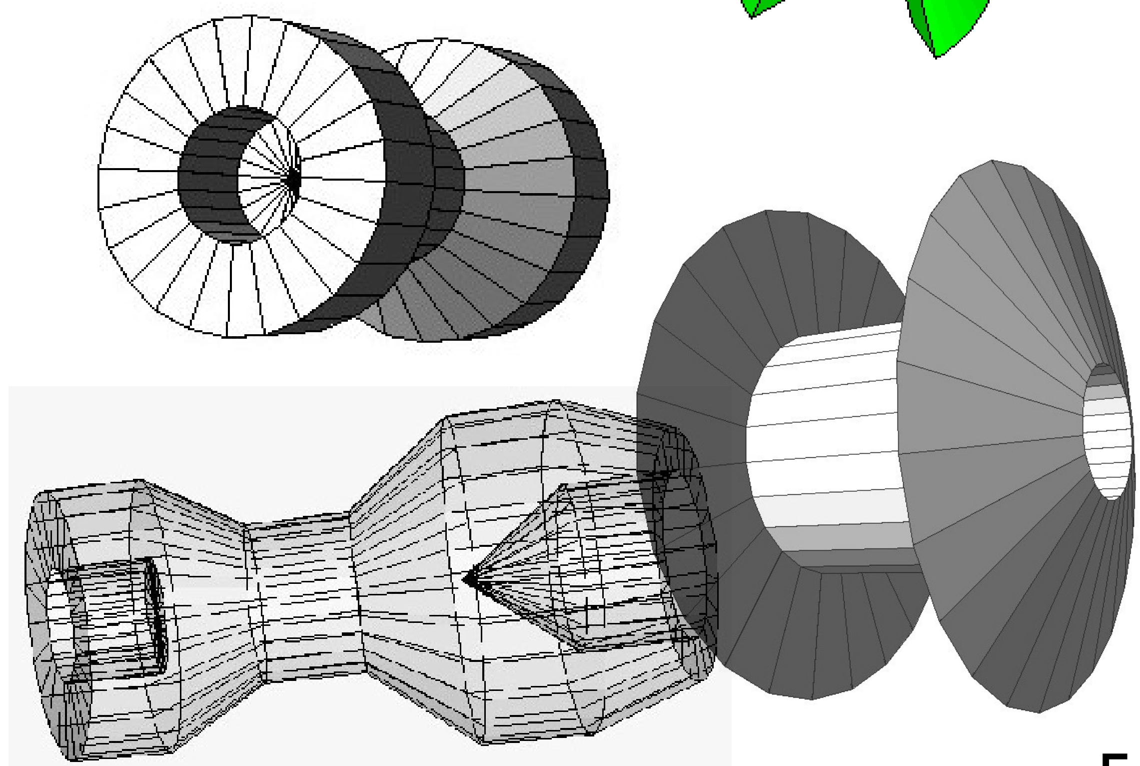


"Batteries included"

- G4VSolid
  - G4BooleanSolid
    - G4IntersectionSolid
    - G4SubtractionSolid
    - G4UnionSolid
  - G4BREPSolid
    - G4BREPSolidBox
    - G4BREPSolidCone
    - G4BREPSolidCylinder
    - G4BREPSolidPCone
    - G4BREPSolidPolyhedra
    - G4BREPSolidSphere
    - G4BREPSolidTorus
  - G4CSGSolid
    - G4Box
    - G4Cons
    - G4Orb
    - G4Para
    - G4Sphere
    - G4Torus
    - G4Trap
    - G4Trd
    - G4Tubs
  - G4DisplacedSolid
  - G4EllipticalTube
  - G4Hype
  - G4ReflectedSolid
  - G4VCSGfaceted
    - G4Polycone
    - G4Polyhedra



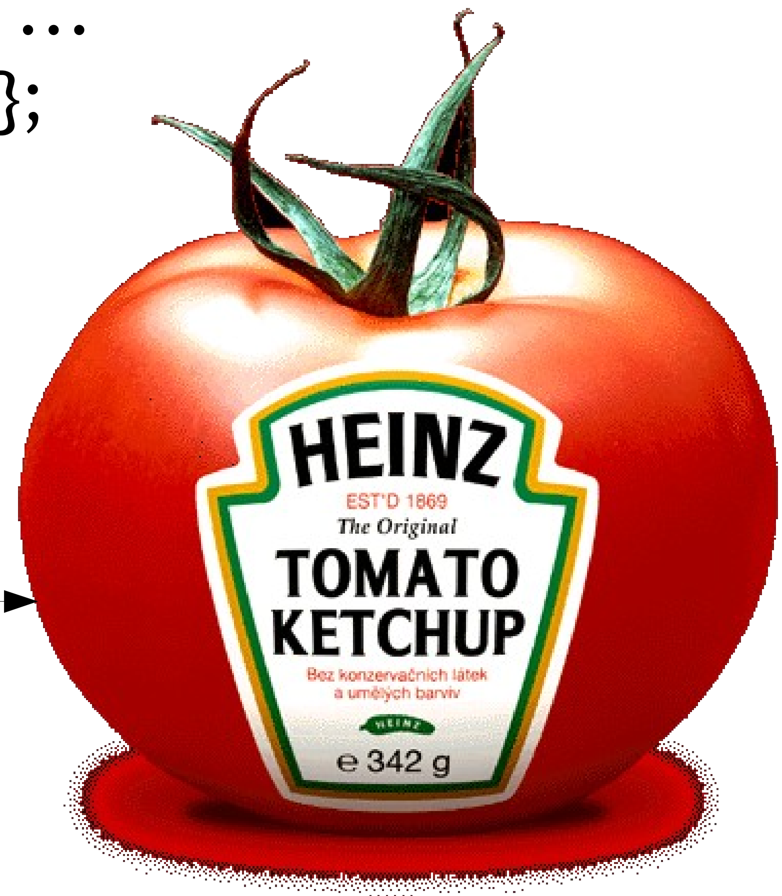




“Batteries included”

- G4VSolid
  - G4BooleanSolid
    - G4IntersectionSolid
    - G4SubtractionSolid
    - G4UnionSolid
  - G4BREPSolid
    - G4BREPSolidBox
    - G4BREPSolidCone
    - G4BREPSolidCylinder
    - G4BREPSolidPCone
    - G4BREPSolidPolyhedra
    - G4BREPSolidSphere
    - G4BREPSolidTorus
  - G4CSGSolid
    - G4Box
    - G4Cons
    - G4Orb
    - G4Para
    - G4Sphere
    - G4Torus
    - G4Trap
    - G4Trd
    - G4Tubs
  - G4DisplacedSolid
  - G4EllipticalTube
  - G4Hype
  - G4ReflectedSolid
  - G4VCSGfaceted
    - G4Polycone
    - G4Polyhedra

```
class MyTomatoSolid :
public G4VSolid
{
 ...
};
```



Recap:

What we have up to now:

- **Material**
  - simple, composites
- **Solid**
  - frame of reference, inside, outside, surface
- **Logical Volume**
  - defined by
    - a mandatory solid
    - a mandatory material
  - optionally:
    - field, hit-collector, ..

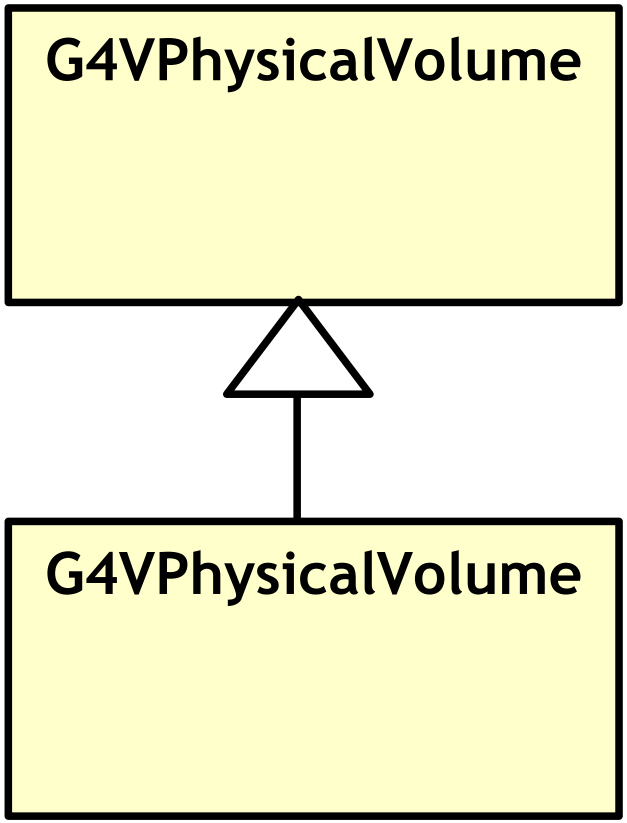
Example:

We still need tools to build geometrical hierarchies from these ingredients

Volume Hierarchies

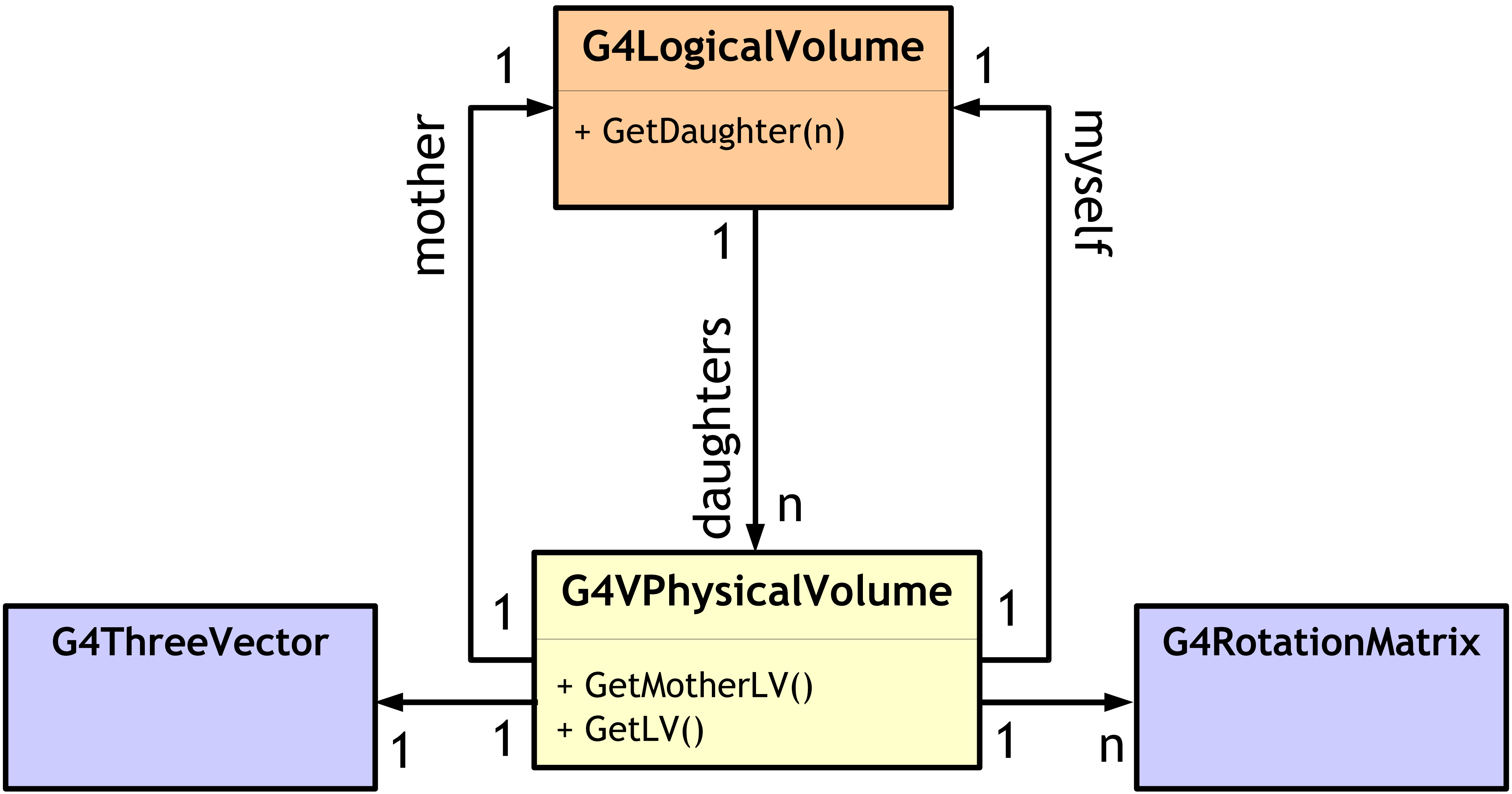
- A logical volume can contain children volumes, **recursively!**
- The position of a child within its parent is defined by
  - a translation vector and a rotation matrix
  - specifying the relative orientation
    - of the reference frame of the child's solid
    - with respect to the reference frame of the parent's solid
- There are constraints on the parent-child relationship!
- Several possibilities to define a parent-child relationship:
  - single placement of a child in a parent
  - dividing the parent into several children
  - parametrized multiple placement

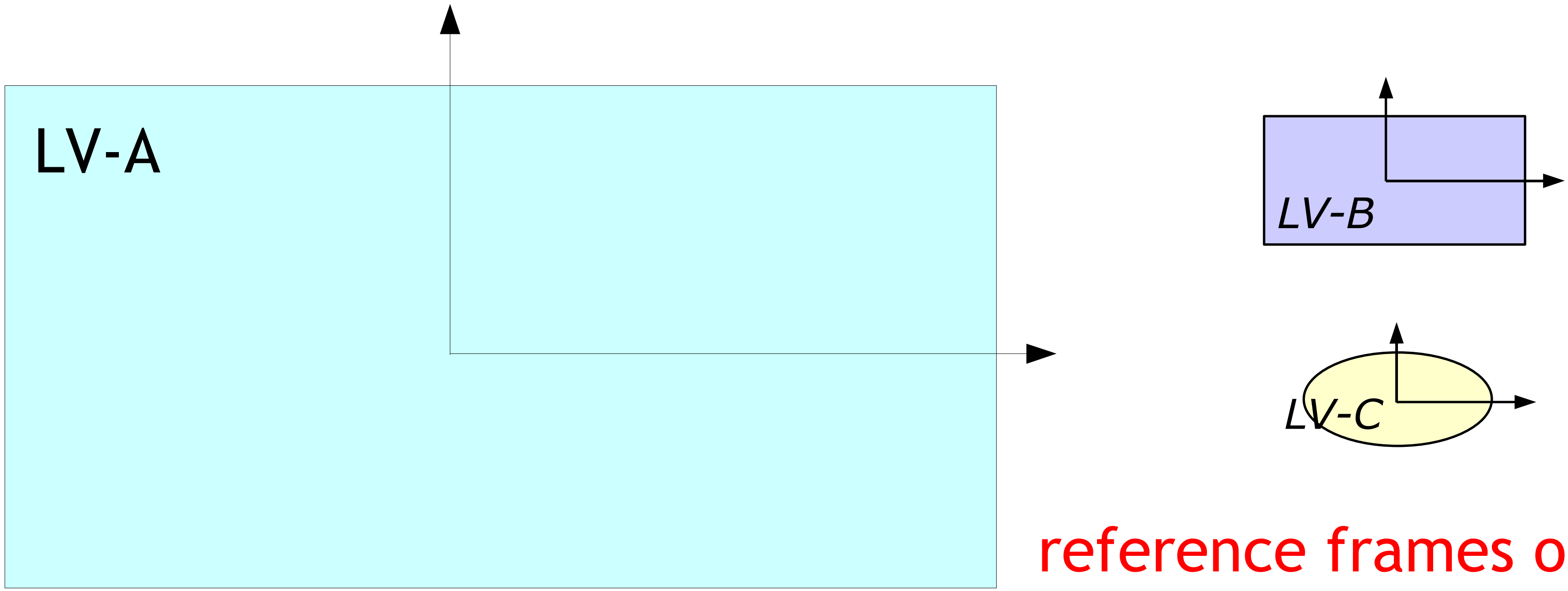
G4VPhysicalVolume



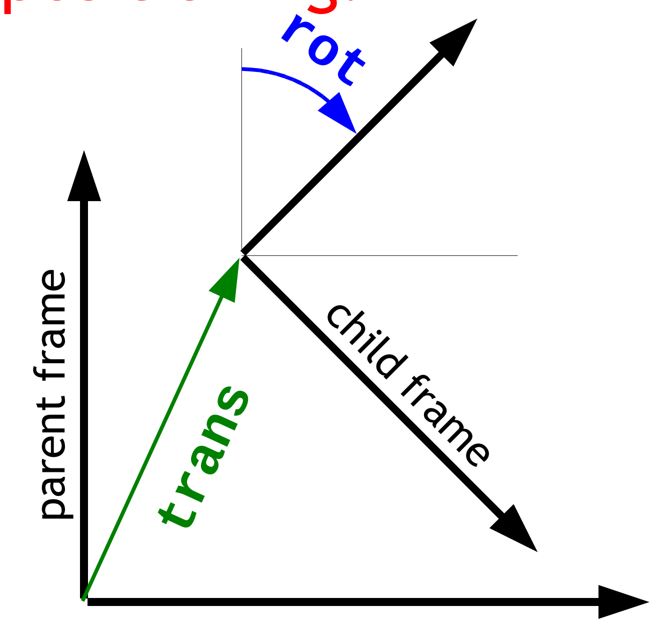
abstract base class

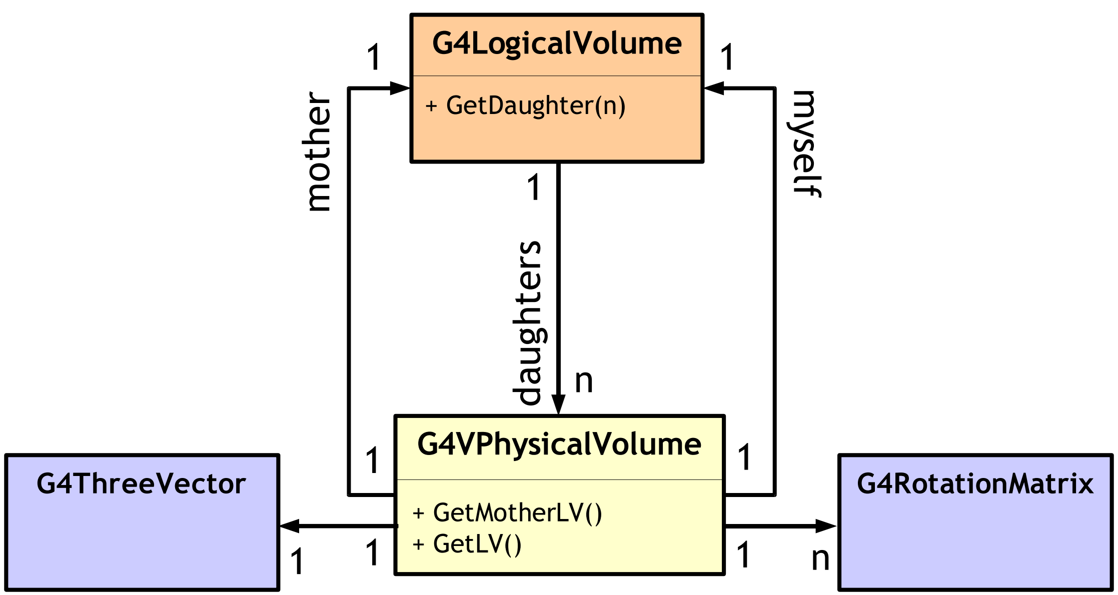
concrete implementation - implements a single parent-child relation of two volumes.

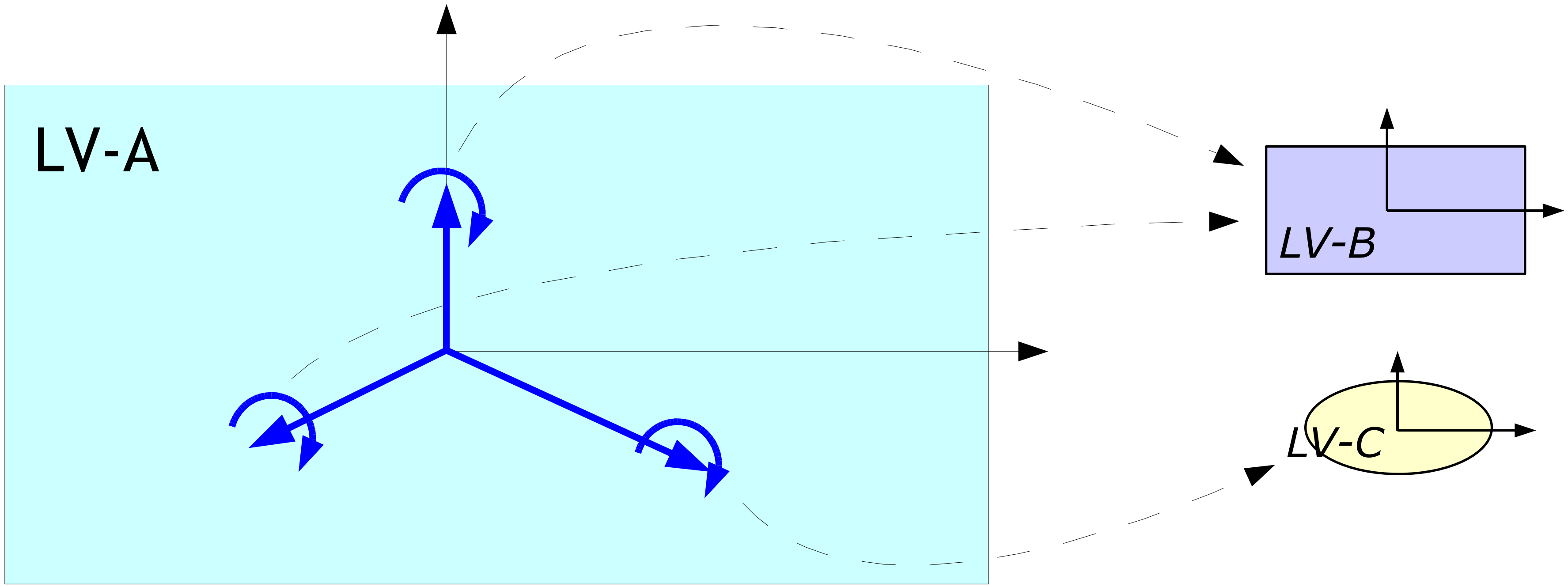

G4LogicalVolume
+ GetDaughter(n)
mother
myself
daughters
1
1
1
n
G4VPhysicalVolume
+ GetMotherLV()
+ GetLV()
1
1
G4ThreeVector
1
1
G4RotationMatrix
1
n

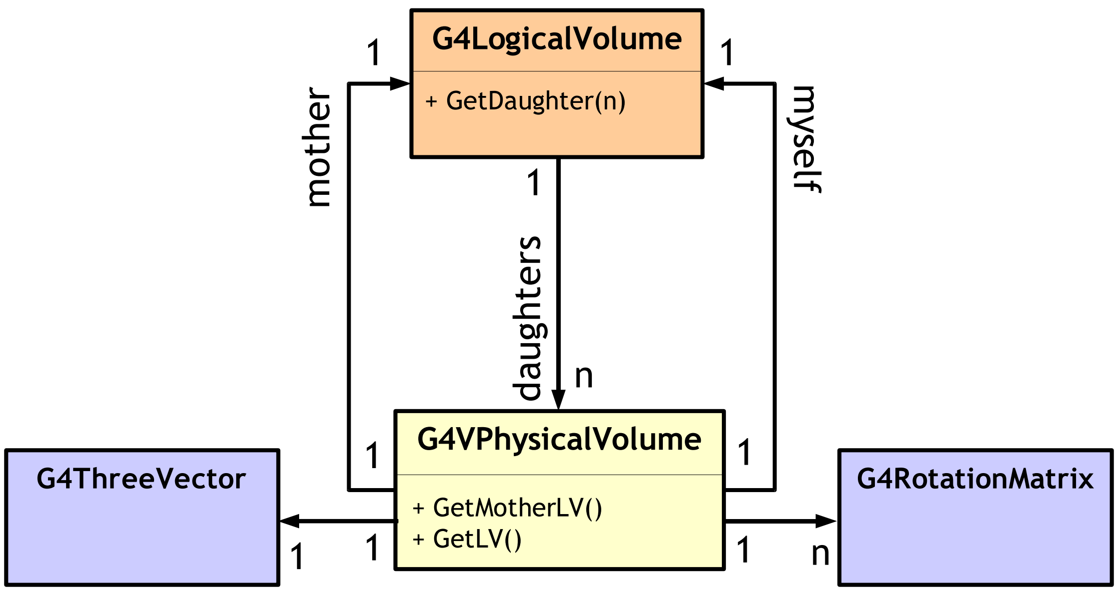


reference frames of solids used for relative positioning:

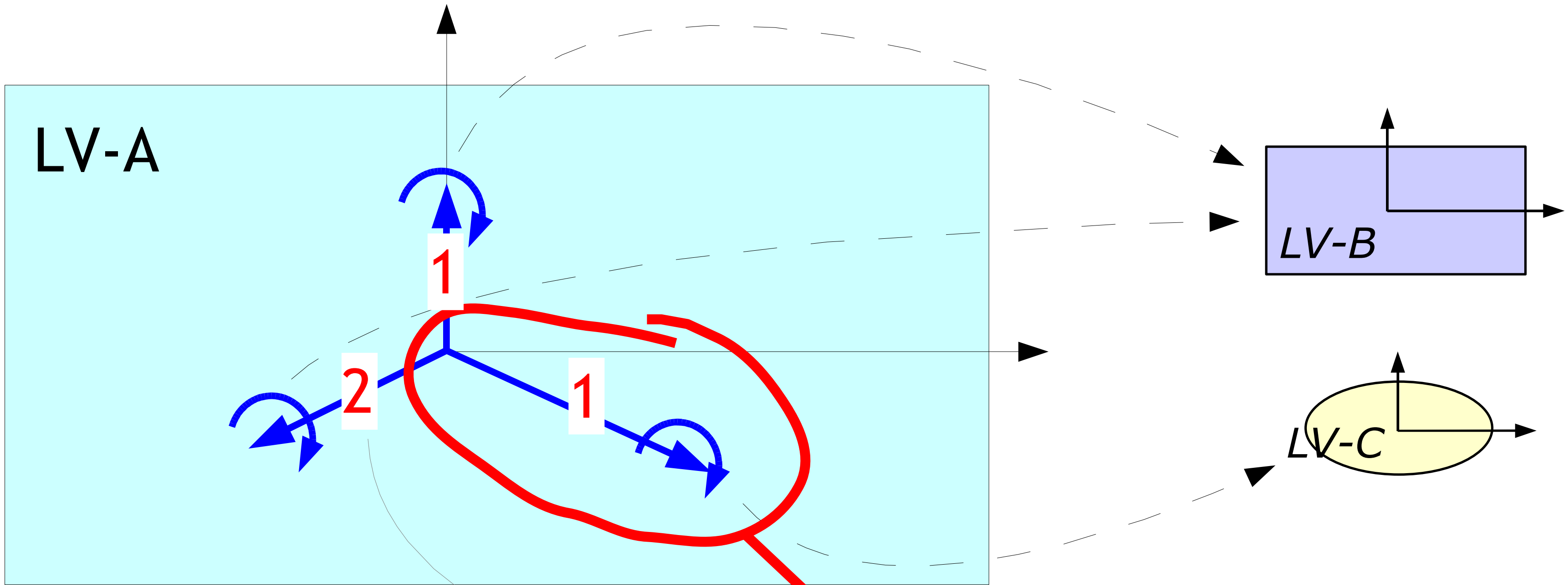




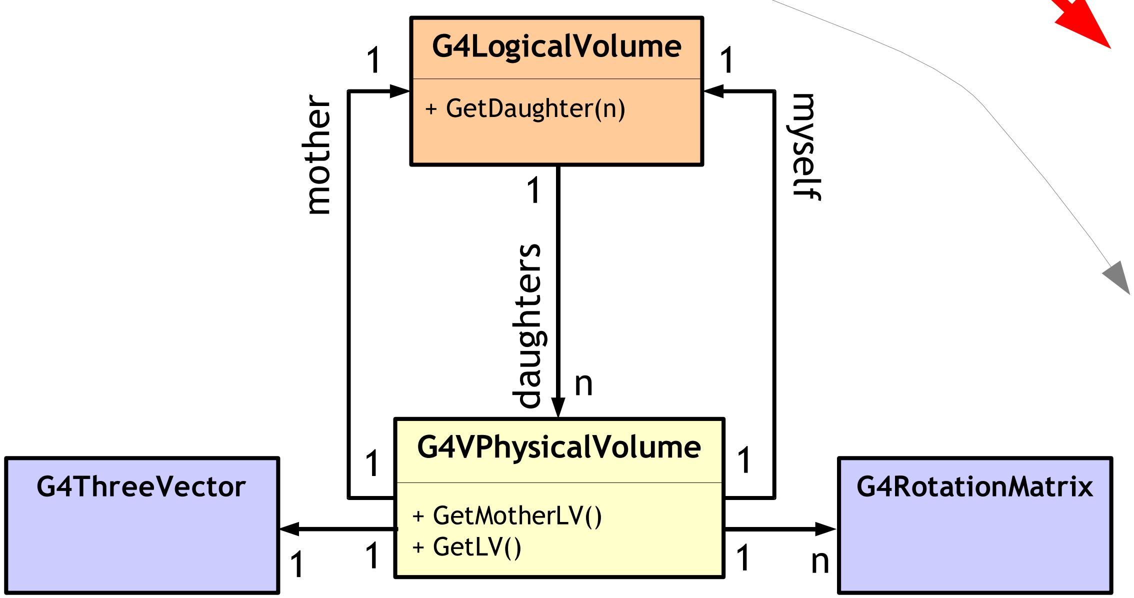




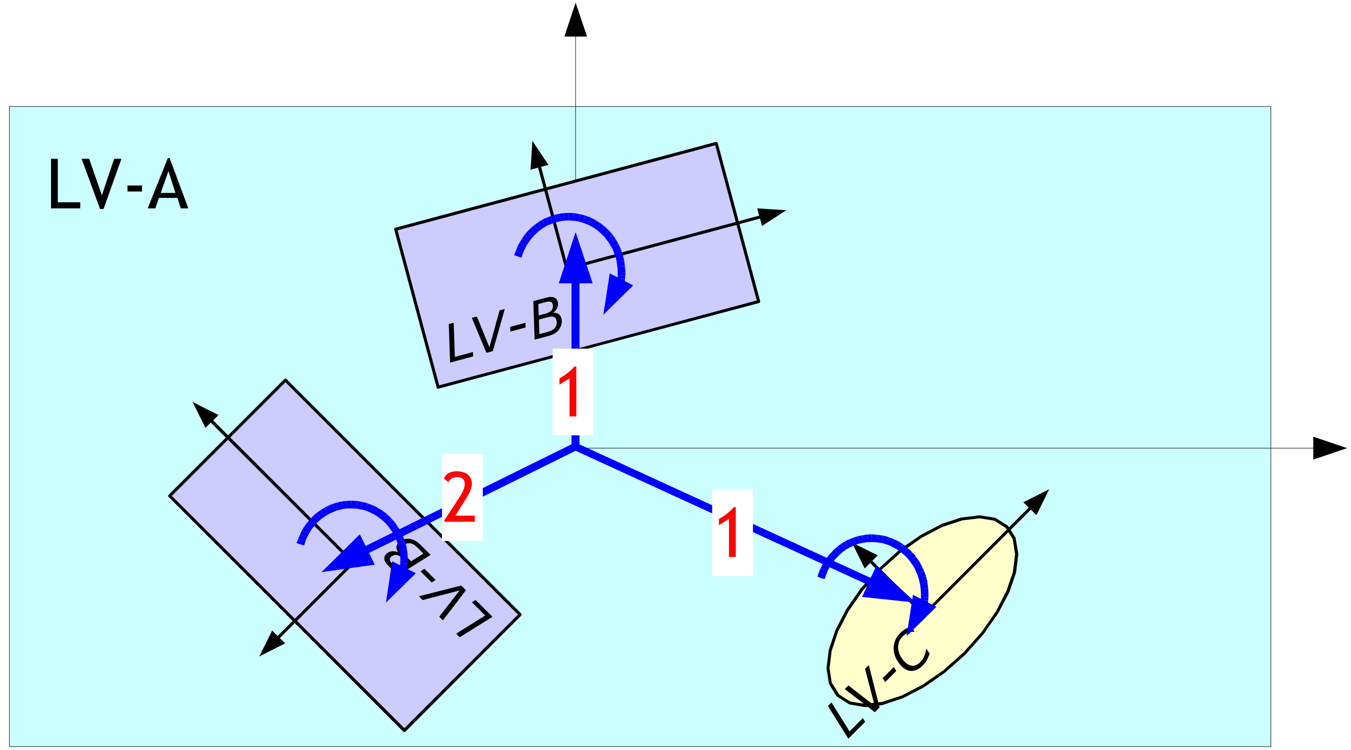
creation of 3 instances of **G4VPhysicalVolume** in LV-A



symbolically:
one instance of a
G4VPhysicalVolume



The numbers are user defined “copy-numbers”; they help us to easier distinguish the copies corresponds to:

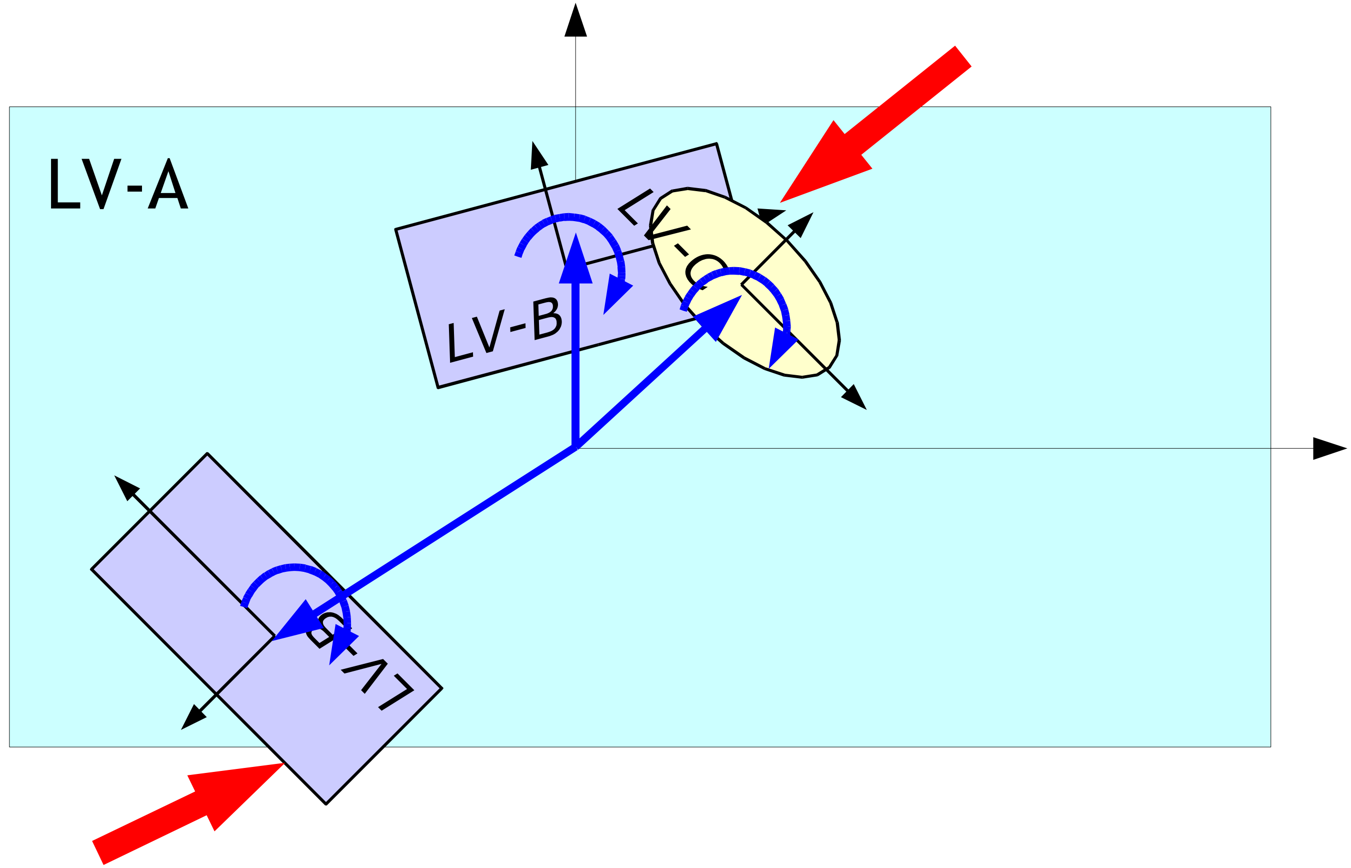


Constraints demanded by GEANT4:

- daughter volumes must be fully contained in the mother
- daughter volumes must not overlap each other

GEANT4 does NOT check this for you, but prefers to behave in an undefined manner during tracking!!

forbidden!!!
forbidden!!!
forbidden!!!

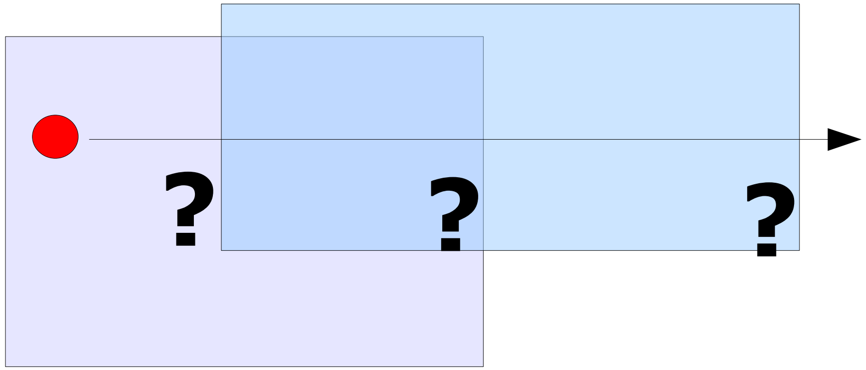


Constraints demanded by GEANT4:

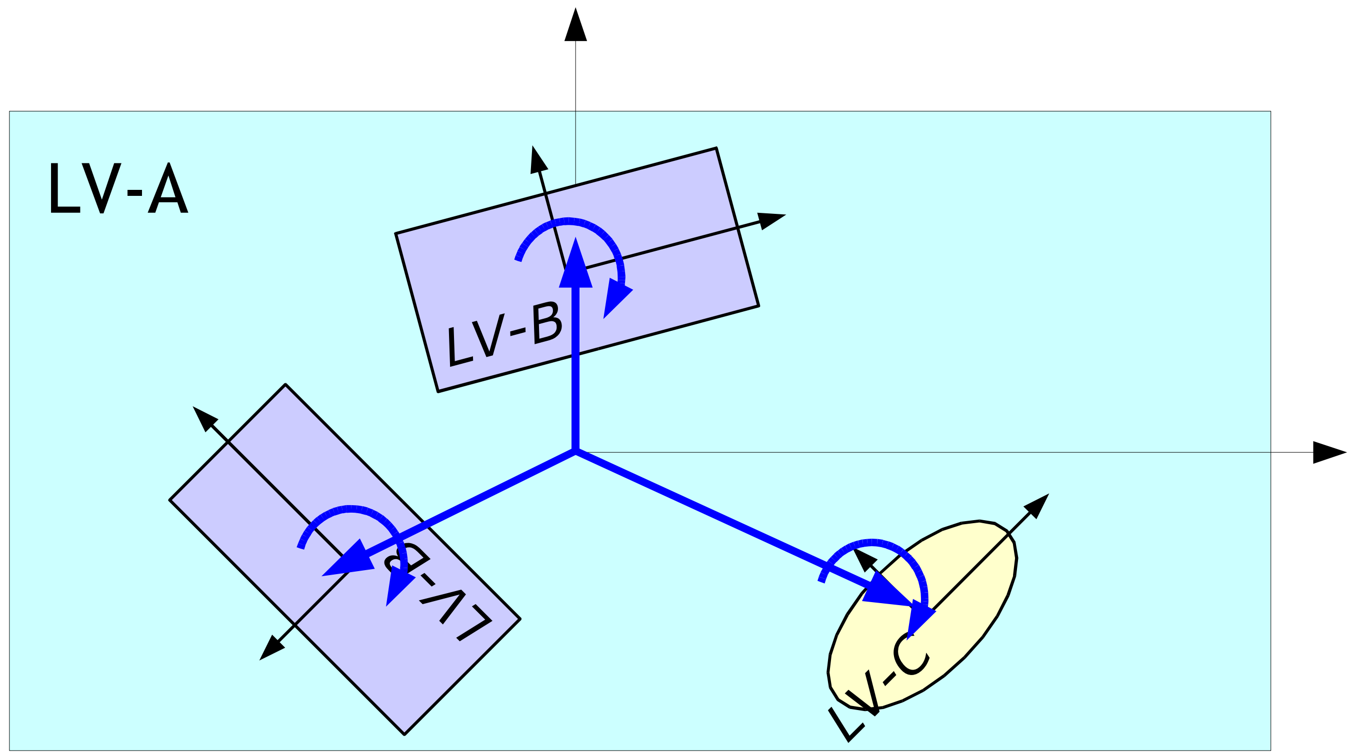
- daughter volumes must be fully contained in the mother
- daughter volumes must not overlap each other

Overlaps lead to ambiguities!

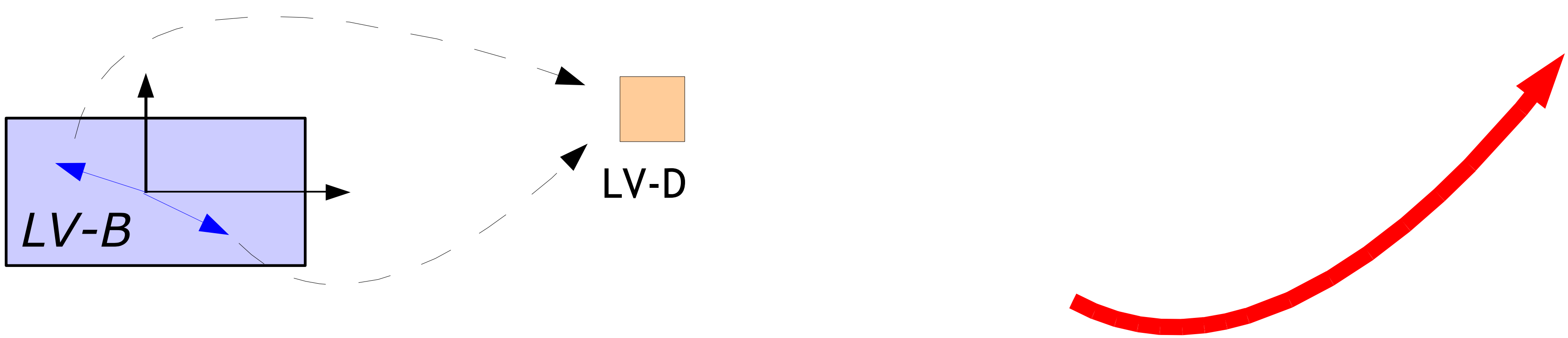
Where should we have the StepPoints on the boundaries?



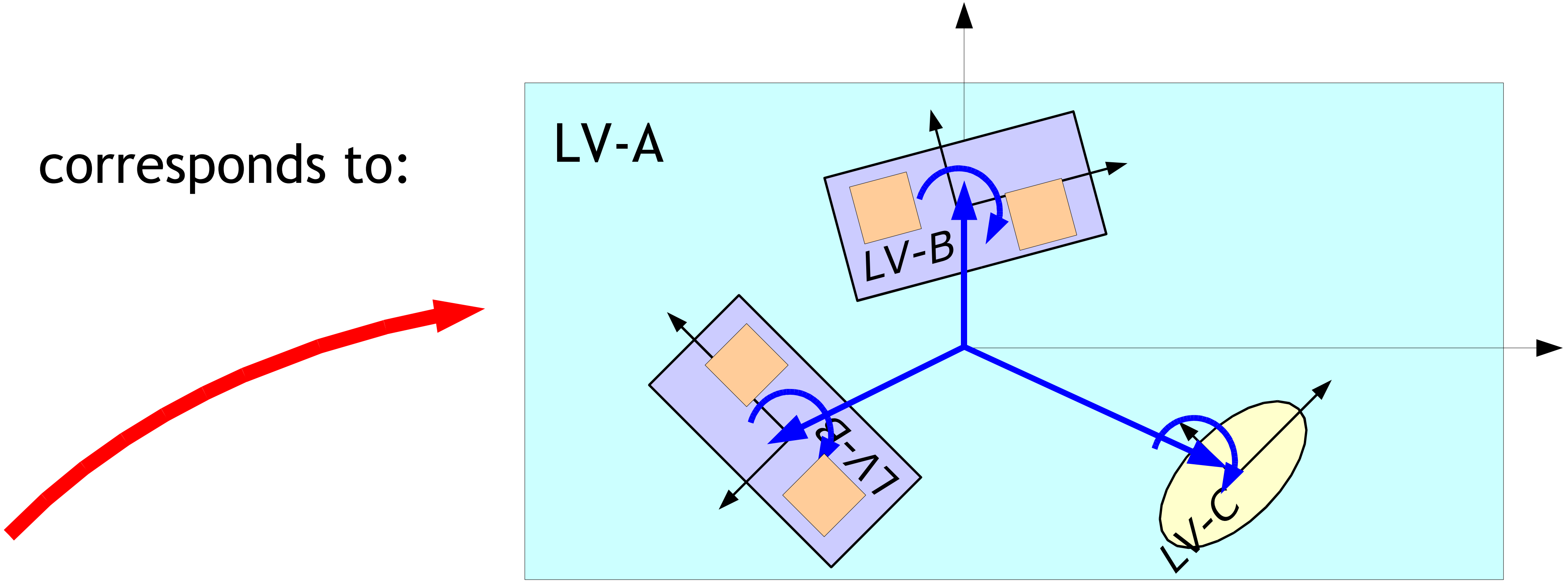
Deeper hierarchies!



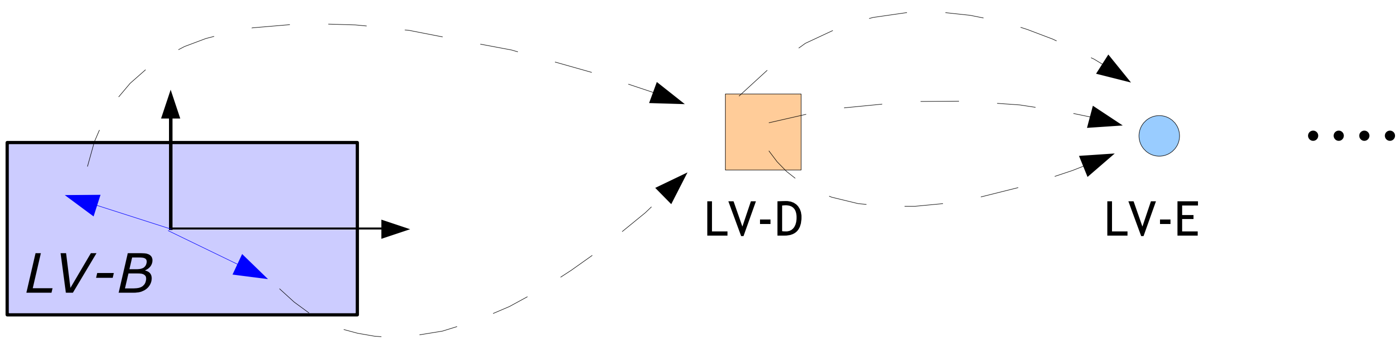
We can apply the positioning procedure recursively!



corresponds to:



and so on and on ...



Properties of the hierarchy

The hierarchy of volumes is a

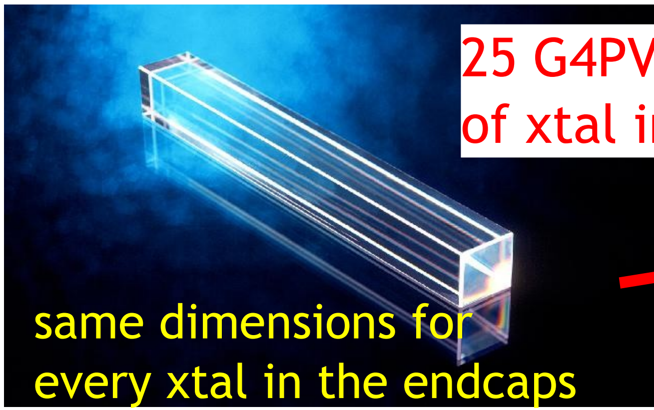
- single rooted (one placement without parent)
- acyclic (preserve strict ancestor ordering -> constraints ...)
- directed (G4LogicalVolume::getDaugther(G4int))

multi-graph.

- The **nodes** of this graph are **logical volumes**
  - material & solid information
- The **edges** of this graph are **physical volumes**
  - position information
- The **root** of the hierarchy **is called the world volume**.

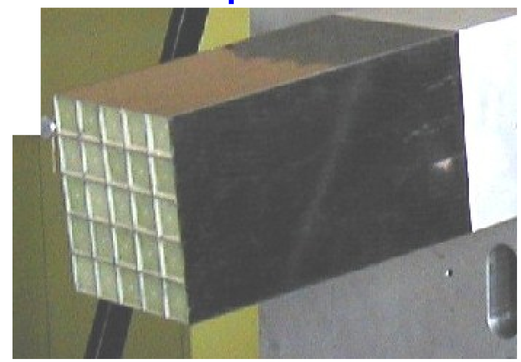
The example ..

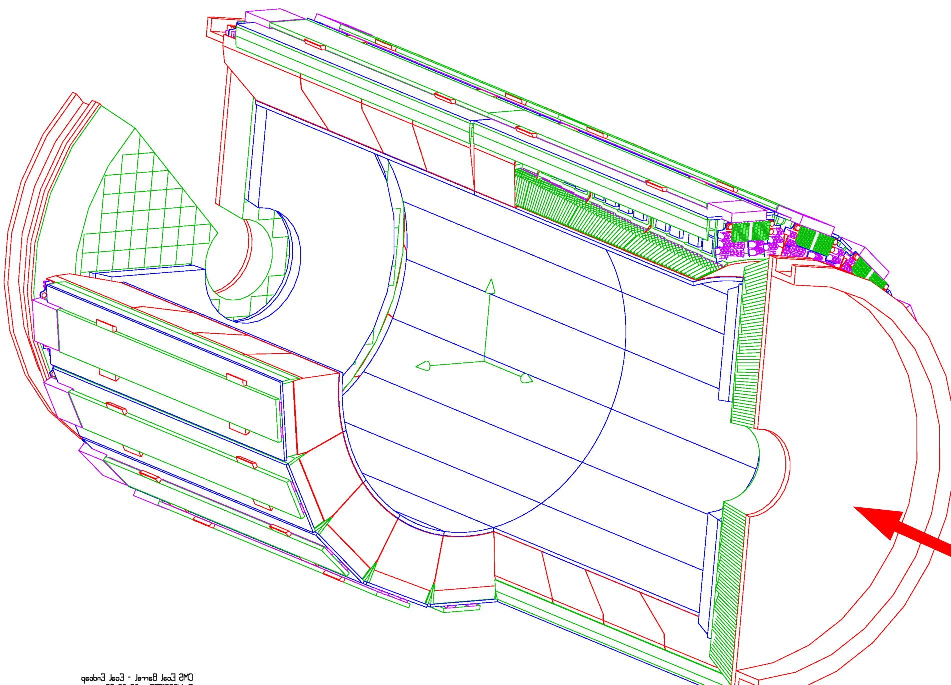
1 G4LogicalVolume
for the xtal

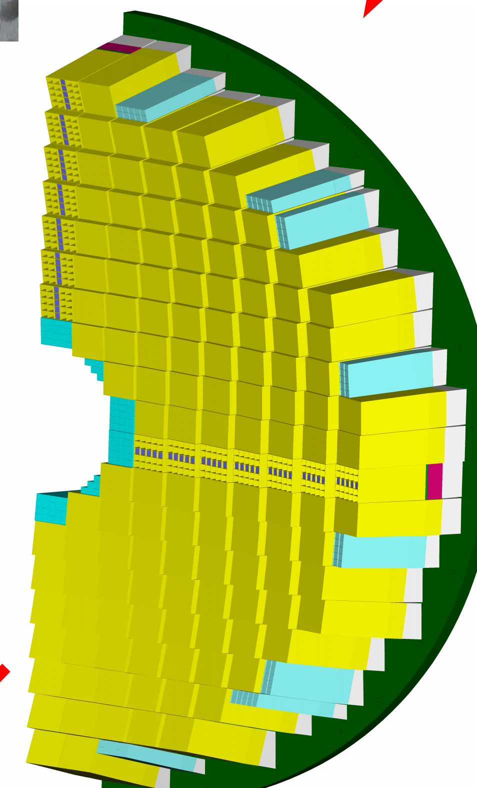


25 G4PVPlacements
of xtal in superxtal

1 G4LogicalVolume
for the super-xtal

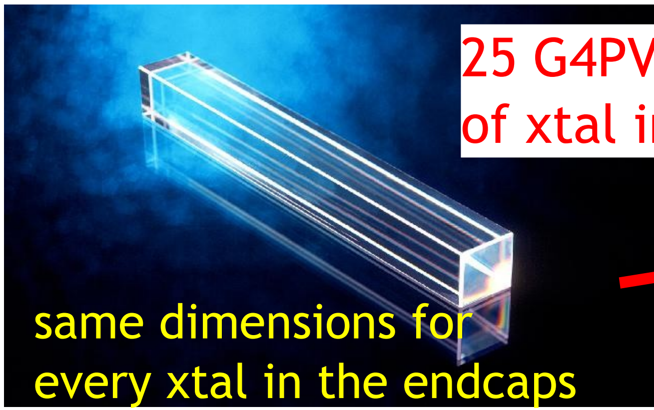






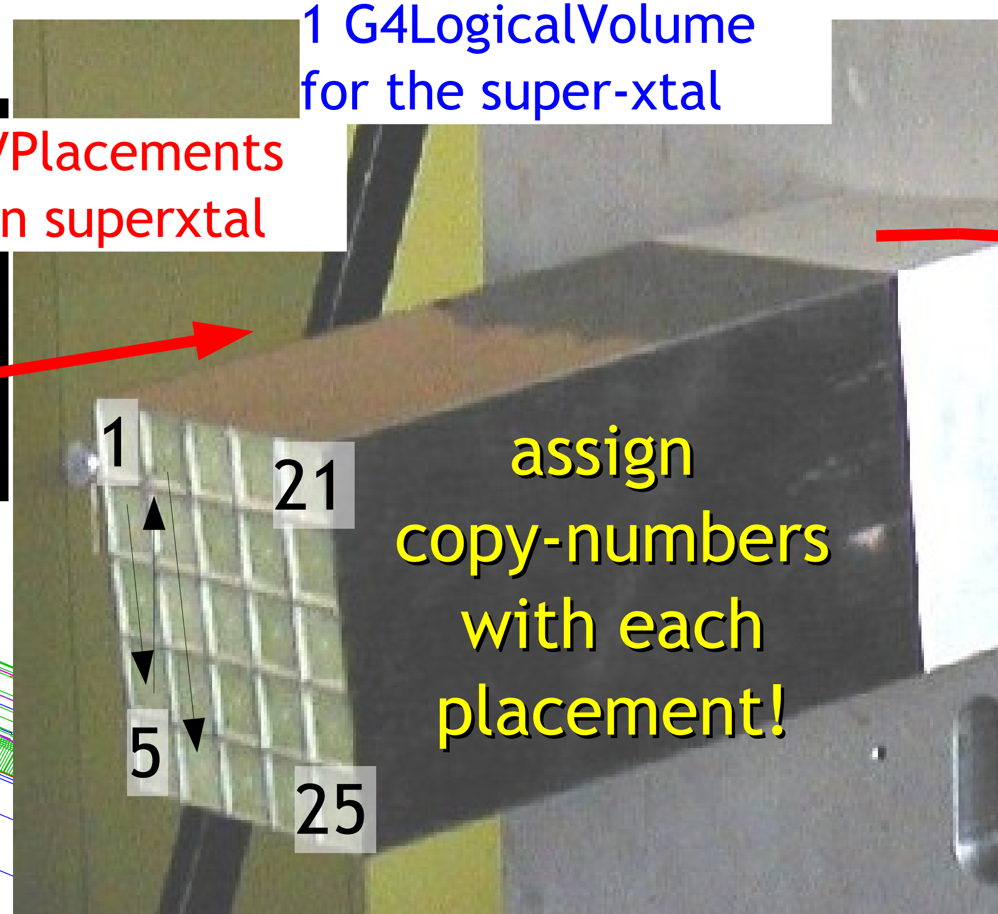
The example ..

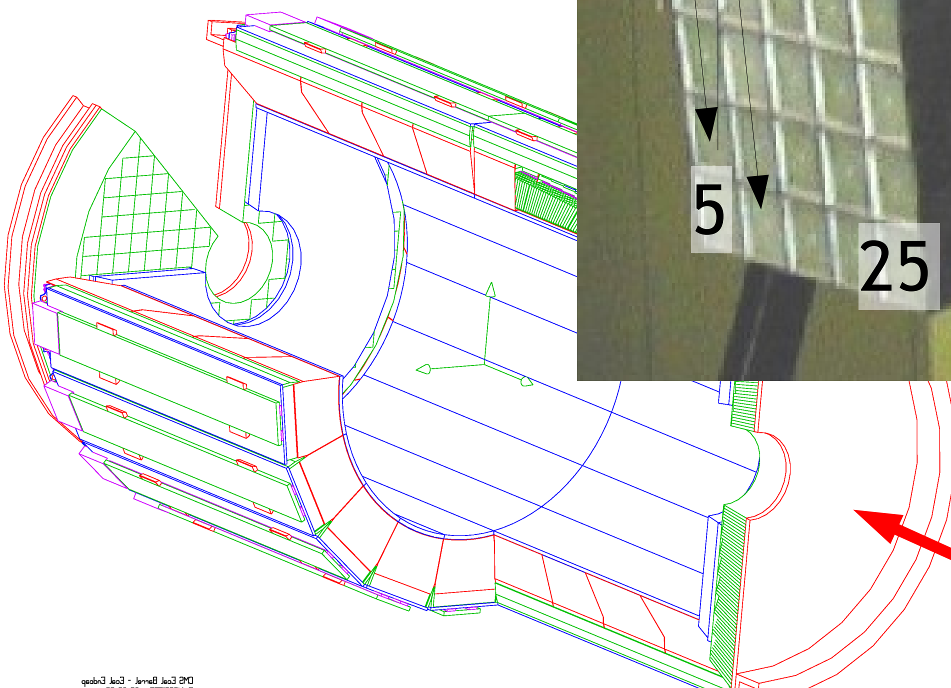
1 G4LogicalVolume
for the xtal

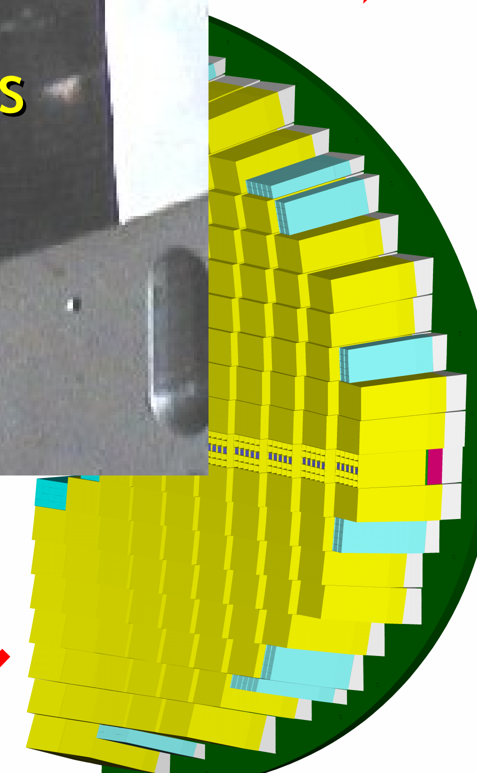


1 G4LogicalVolume
for the super-xtal

25 G4PVPlacements
of xtal in superxtal

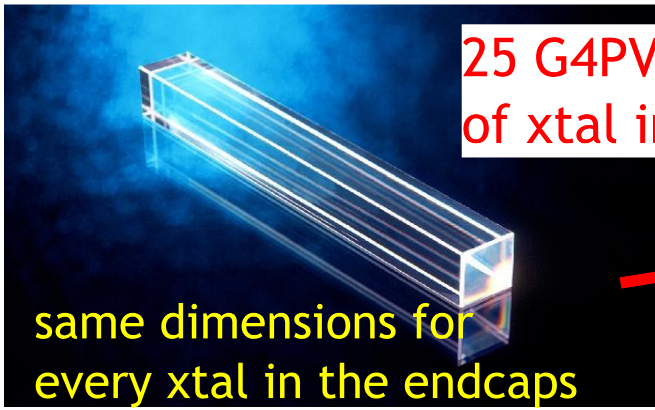






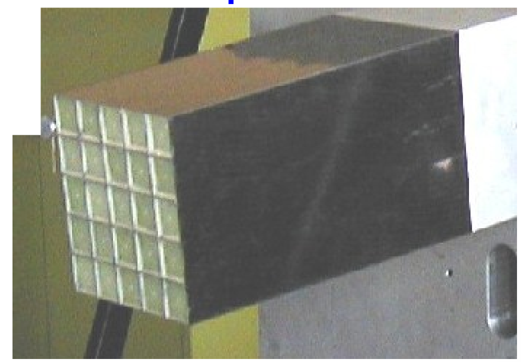
The example ..

1 G4LogicalVolume
for the xtal

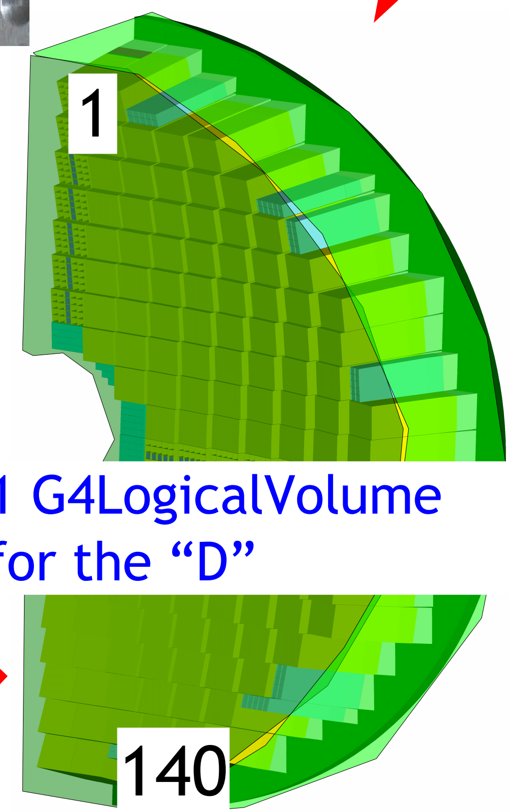


25 G4PVPlacements
of xtal in superxtal

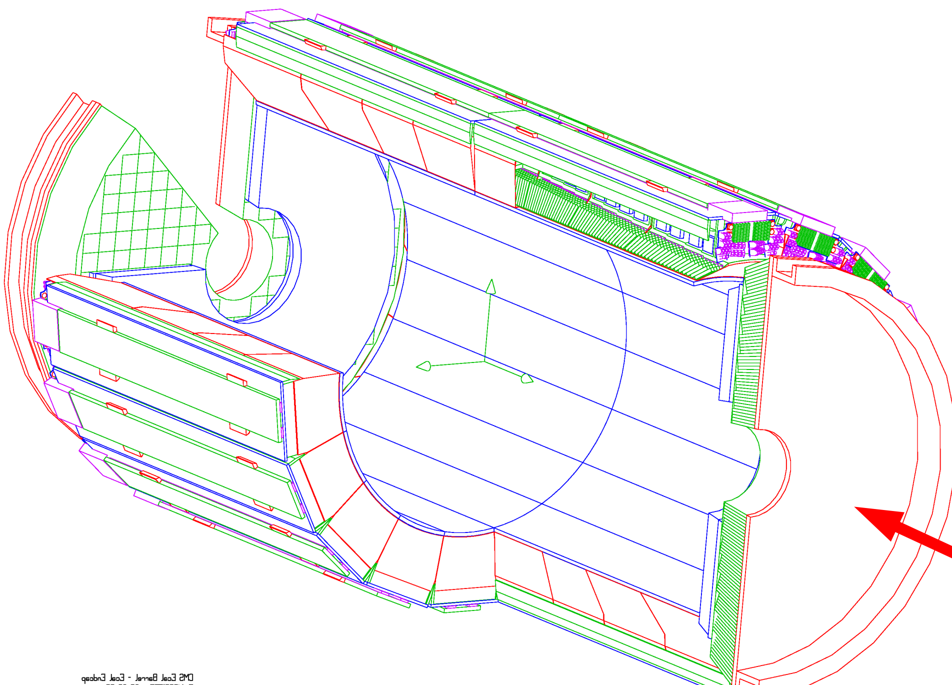
1 G4LogicalVolume
for the super-xtal



140 placements
of super-xtals in D

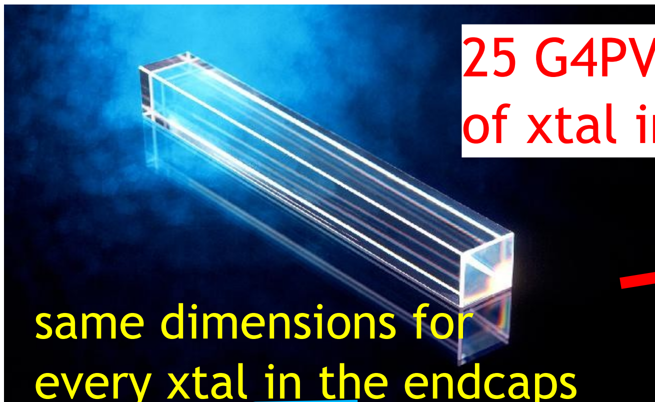


1 G4LogicalVolume
for the “D”



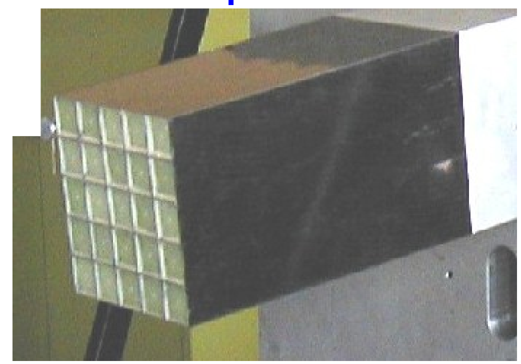
The example ..

1 G4LogicalVolume
for the xtal

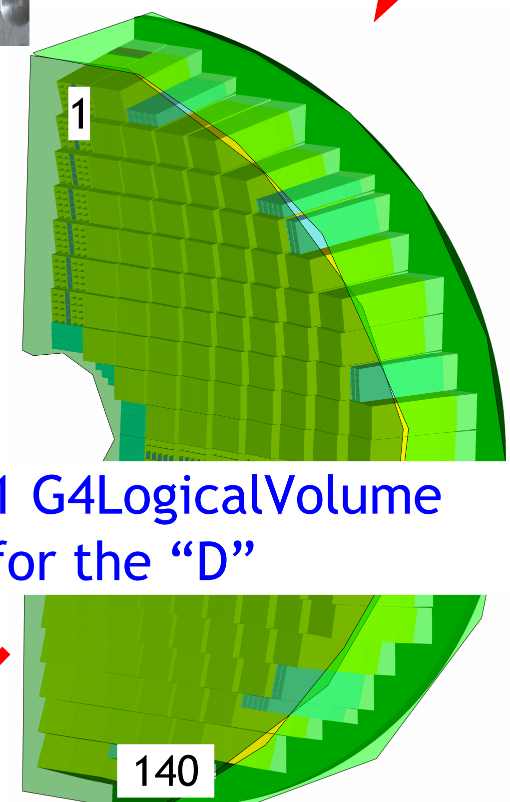


25 G4PVPlacements
of xtal in superxtal

1 G4LogicalVolume
for the super-xtal

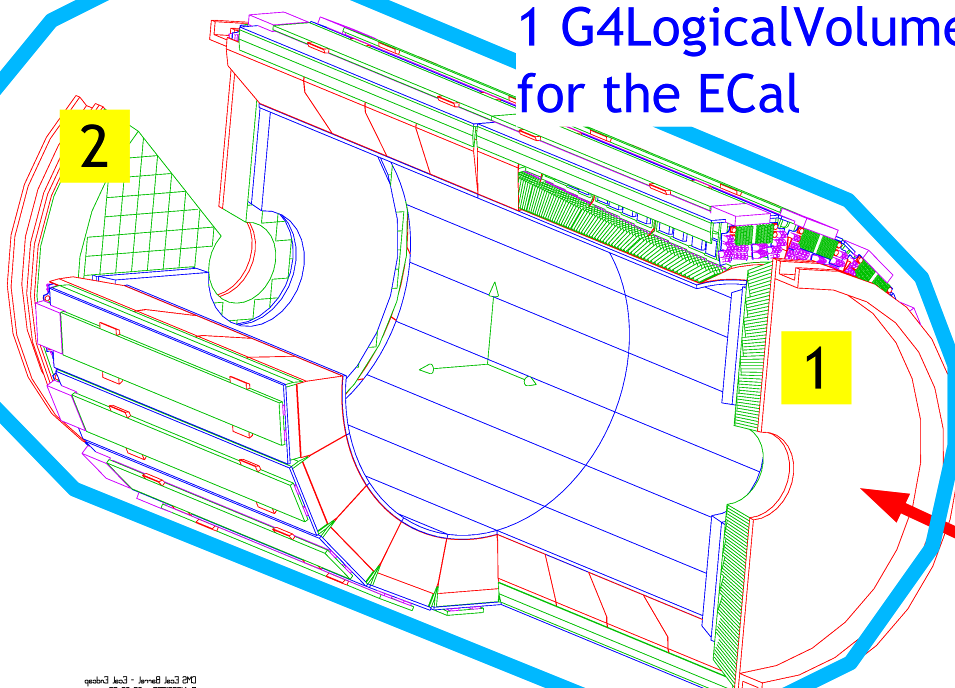


170 placements
of super-xtals in D



1 G4LogicalVolume
for the "D"

1 G4LogicalVolume
for the ECal



2 placements
of "D" in ECal

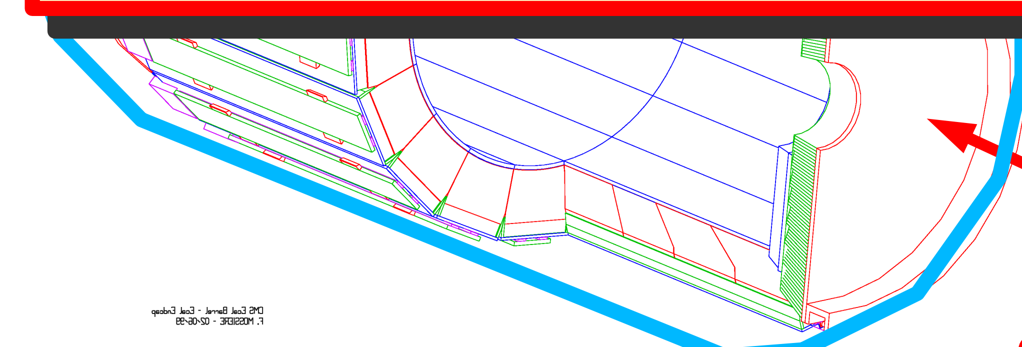
The example ..

1 G4LogicalVolume

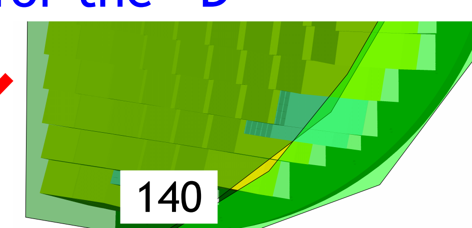
1 G4LogicalVolume

G4LogicalVolumes	**G4PVPlacements**
1 xtal	
1 super-xtal	25 xtal -> super-xtal
1 “D”	140 superxtal -> “D”
1 ECal	2 “D” -> ECal
-----------	-----------------------
4	**167**
===========	=======================

Need only 4 + 167 instances (+ max. 167 rot, trans) to represent ~14.000 crystals

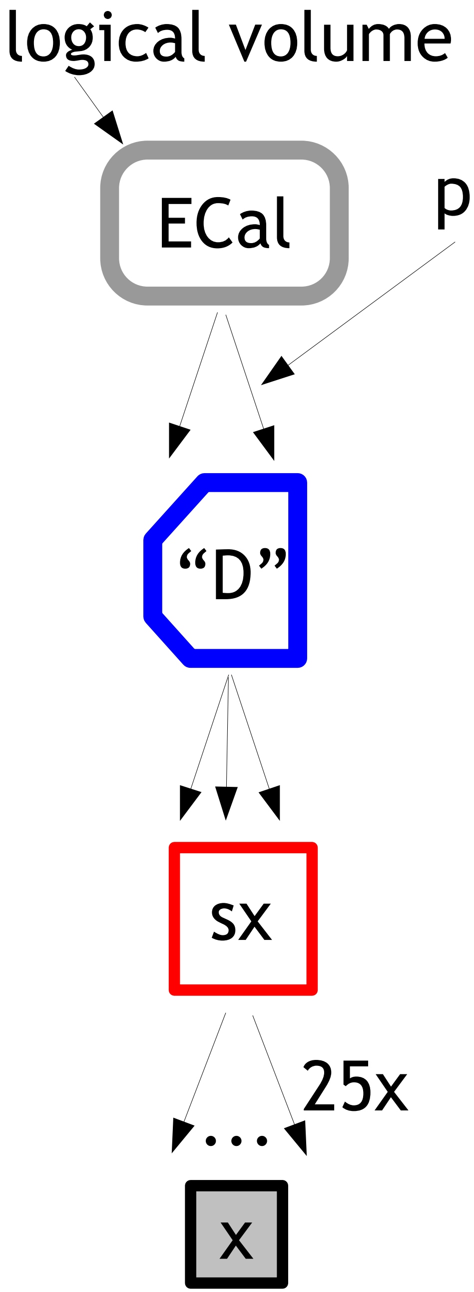


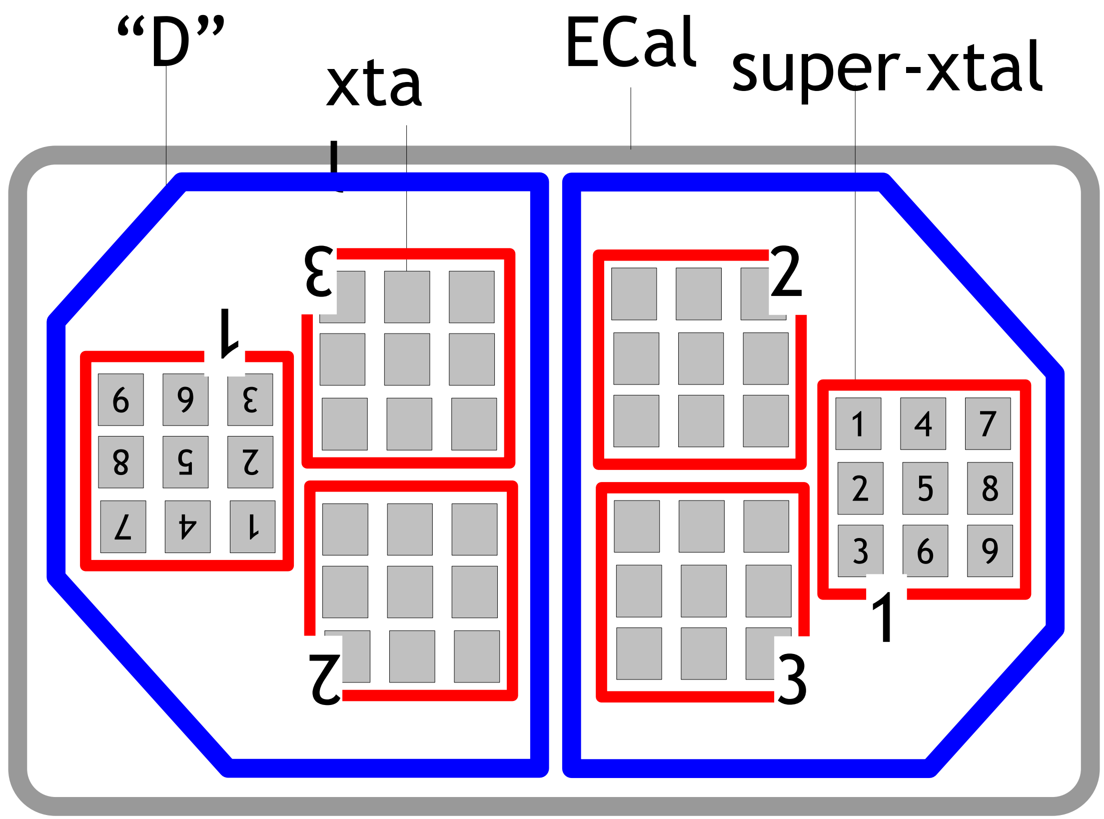
for the “D”



2 placements of “D” in ECal

Graph Structure





simplified version of an ECal ...

Summary ~3~

- A - Physics Model in GEANT4
  - Stepping: moving in free path lengths
  - Physics Processes
- B - Detector Description in GEANT4
  - Solids/Shape Model
  - Volumes
  - Hierarchy of Volumes
- Combining A + B
  - Stepping through a detector description