

Experiment Simulation

CERN School of Computing 2005
Saint Malo

Lecture 3

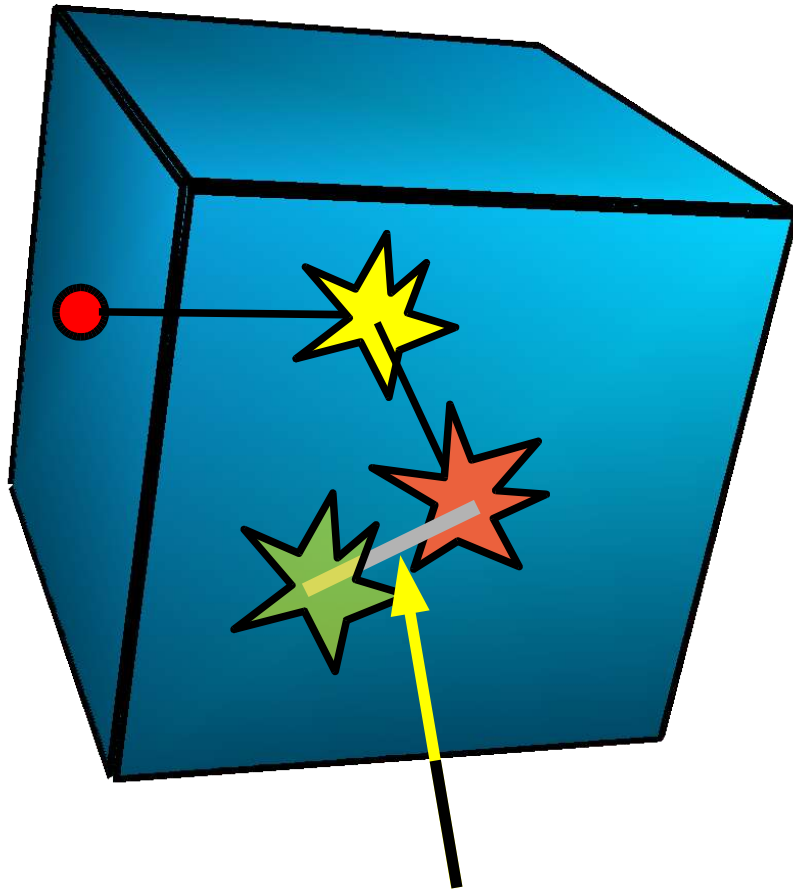


Martin Liendl
SWX Swiss Exchange

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 ...



Woah! A **G4Step**

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

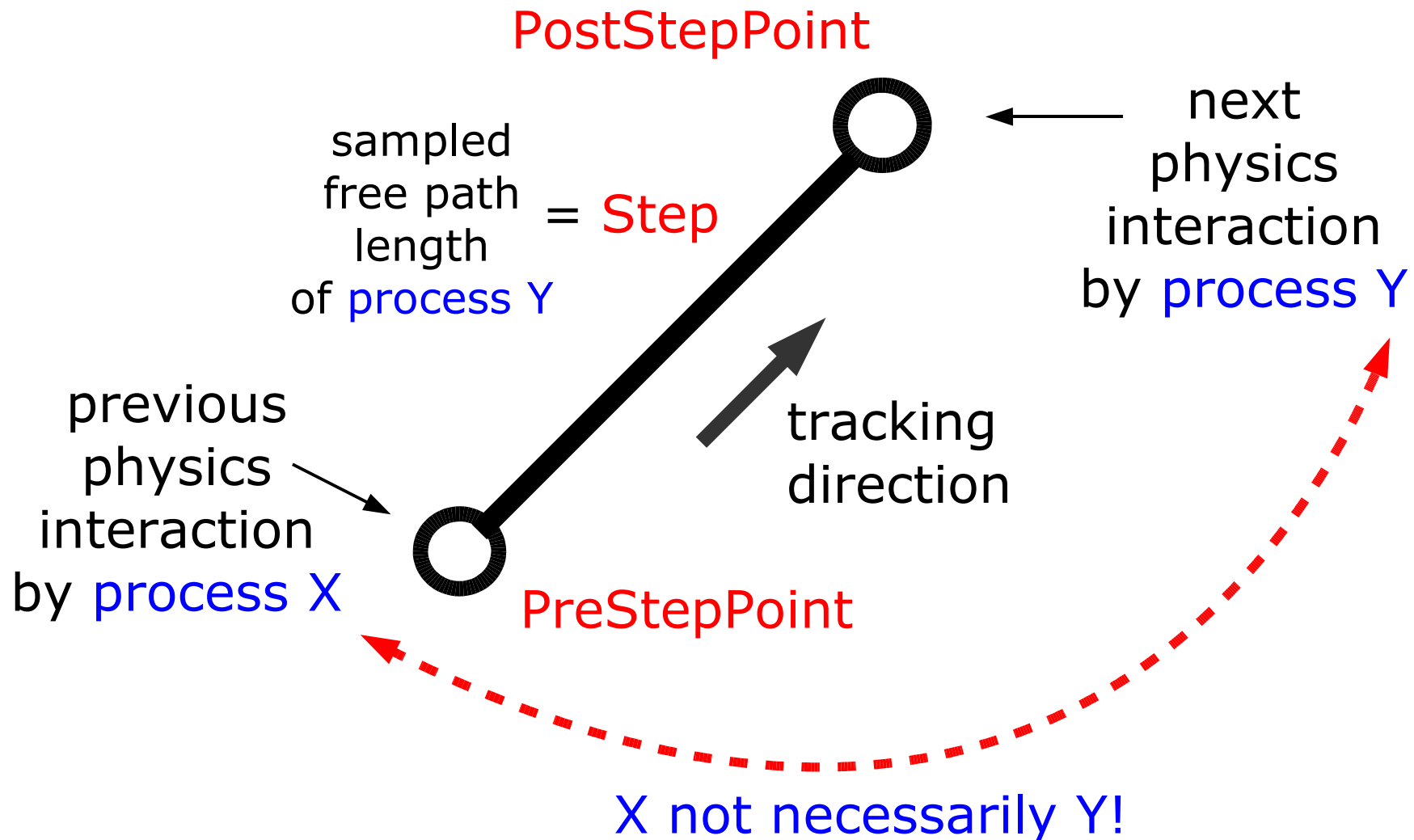
Probability of having the first interaction due to **process i** after a free path length **L**

Monte Carlo Algorithm:

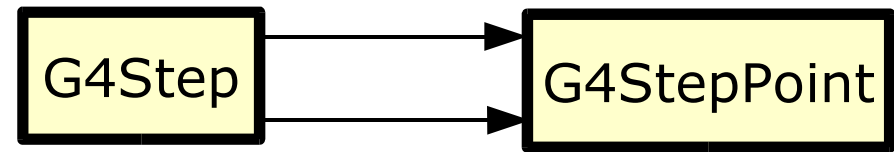
- Sample the free path length from the distributions of all participating processes
- Select the smallest path length
- Move the particle by this **step**
- Simulate the interaction

“That’s one small Step for G4 ..”

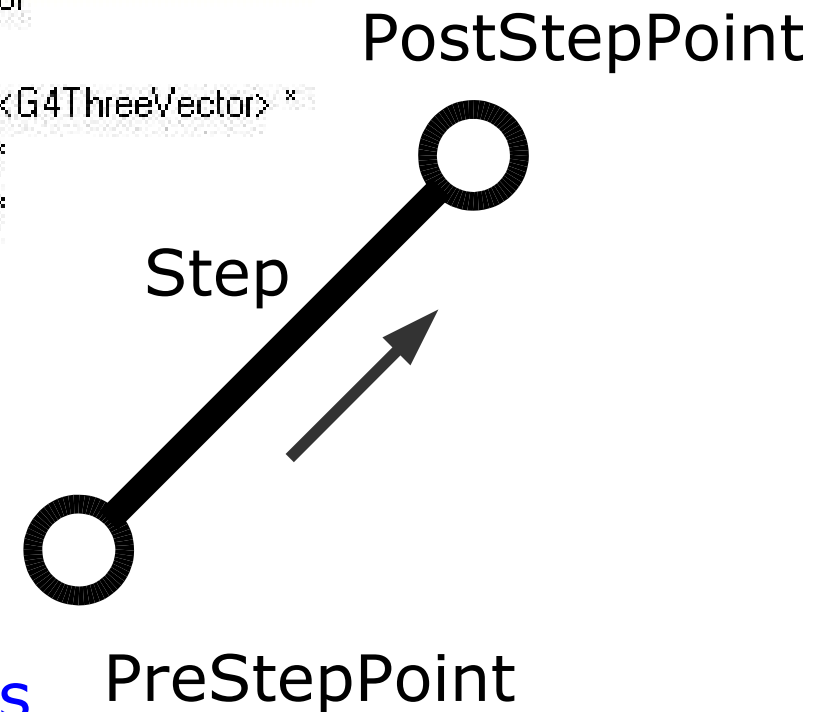
class **G4Step** and class **G4StepPoint**



class **G4Step**

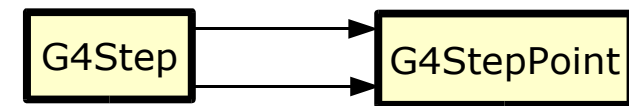


Name	Class	Type
◇ GetDeltaPosition(md)		G4ThreeVector
◇ GetDeltaTime(md)		G4double
◇ GetPointerToVectorOfAuxiliaryPoints(md)		int std::vector<G4ThreeVector> *
◇ GetPostStepPoint(md)		G4StepPoint *
◇ GetPreStepPoint(md)		G4StepPoint *
◇ GetStepLength(md)		G4double
◇ GetTotalEnergyDeposit(md)		G4double
◇ GetTrack(md)		G4Track *

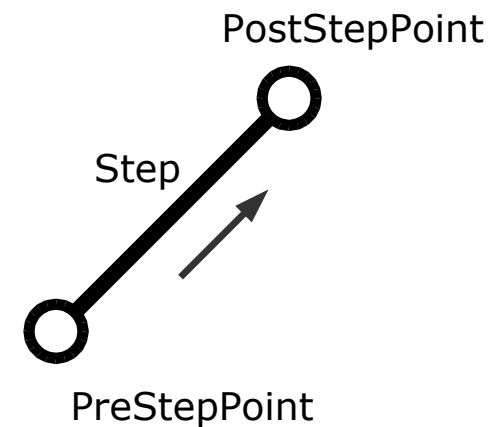


Access to "Delta"-Information,
i.e. kinematic differences of
the particle at the two StepPoints

class **G4StepPoint**



Name	Class	Type
◇ GetCharge(md)		G4double
◇ GetGamma(md)		G4double
◇ GetGlobalTime(md)		G4double
◇ GetKineticEnergy(md)		G4double
◇ GetLocalTime(md)		G4double
◇ GetMass(md)		G4double
◇ GetMaterial(md)		G4Material *
◇ GetMaterialCutsCouple(md)		const G4MaterialCutsC
◇ GetMomentum(md)		G4ThreeVector
◇ GetMomentumDirection(md)		const G4ThreeVector &
◇ GetPhysicalVolume(md)		G4VPhysicalVolume *
◇ GetPolarization(md)		const G4ThreeVector &
◇ GetPosition(md)		const G4ThreeVector &
◇ GetProcessDefinedStep(md)		const G4VProcess *
◇ GetProperTime(md)		G4double
◇ GetSafety(md)		G4double
◇ GetSensitiveDetector(md)		G4VSensitiveDetector

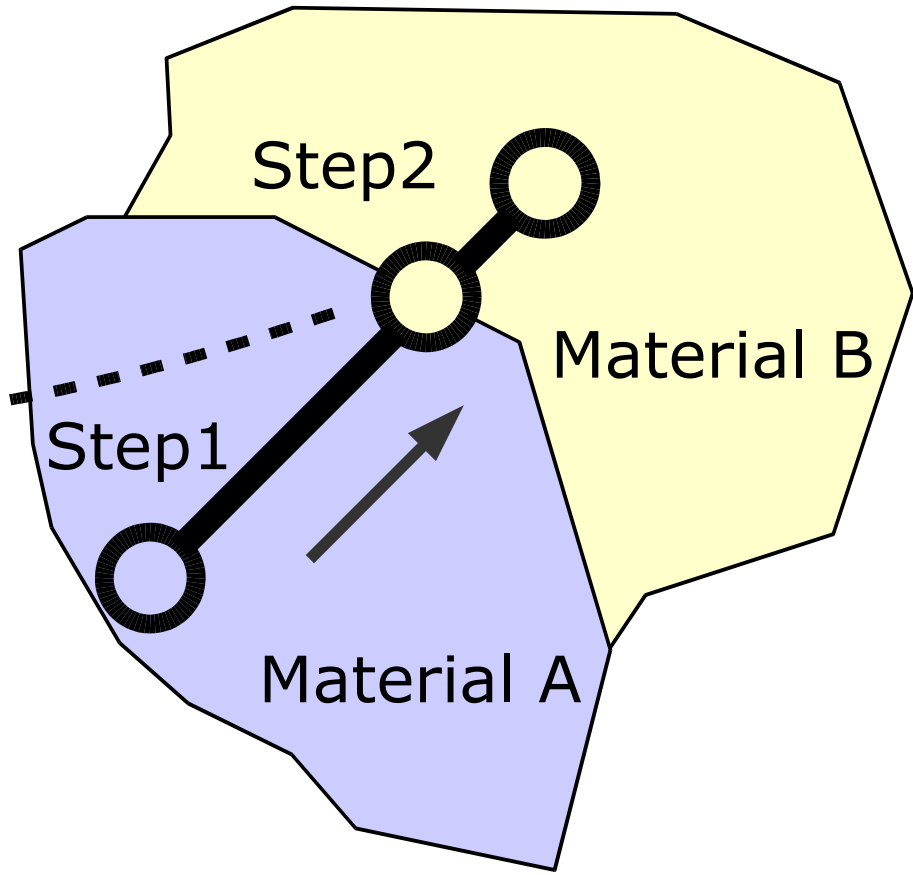


- Kinematic information of the particle at this point
- Convenience methods for accessing **G4ParticleDefinition**
- Access to Physics Process
- Access to **Detector Description Data: G4Material & volume hierarchy** (see later)

Specialty

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

PostStepPoint1
=
PreStepPoint2

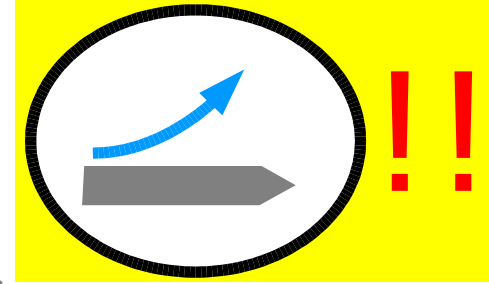


It's good for us to know,
when particles are going
from one region in the
detector to another,
e.g. from support structure
into a **sensitive element!**

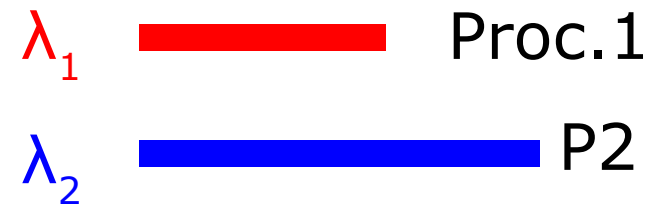
Recap: Monte Carlo Algorithm

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

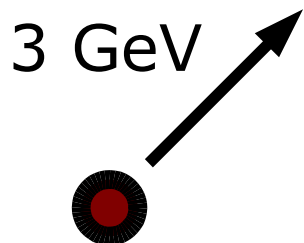
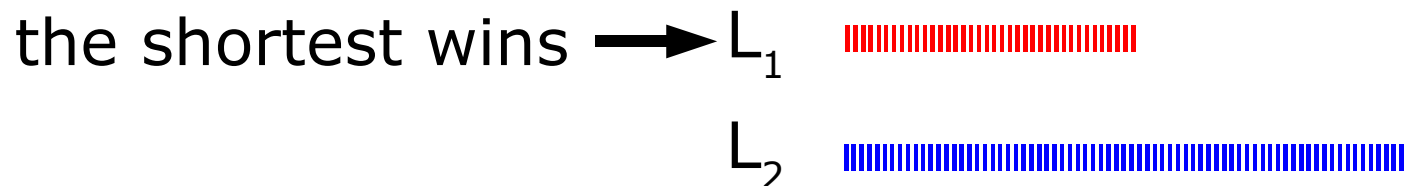
- (1) set properties for incident particle (energy, momentum, ..)
- (2) get values for λ_i for all relevant processes $i=1,2,\dots,m$
- (3) for each process i ($i=1,2,\dots,m$):
 - sample L_i from $p_i(x)$
- (4) $L_c = \min(L_i)$ from all sampled L_i , c in $(1,2,\dots,m)$
- (5) transport incident particle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



Step 1:



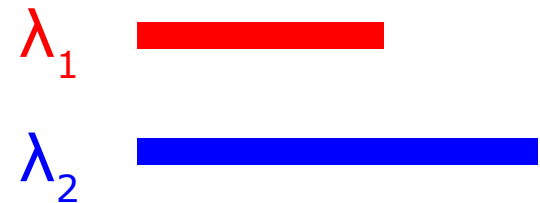
Drawing the random free path length from both processes:



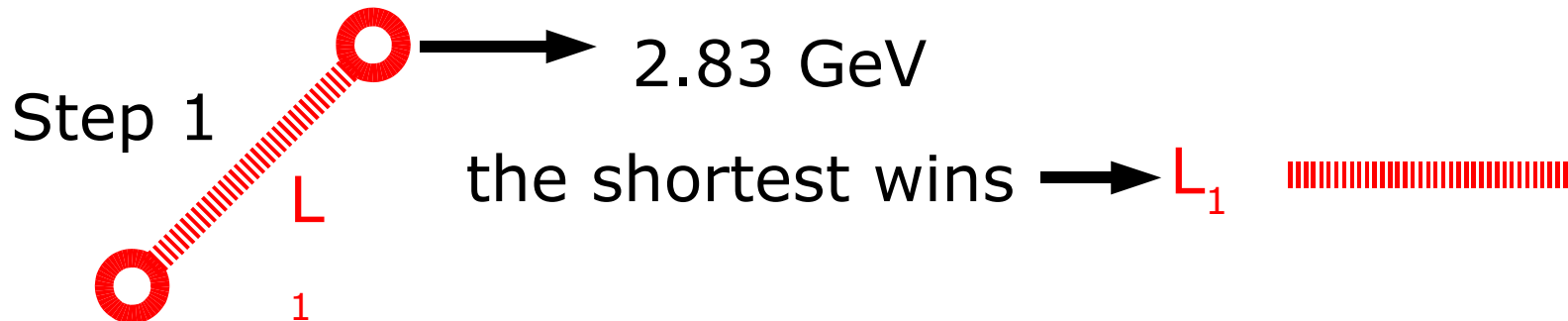
Recap: Monte Carlo Algorithm

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

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



PostStepPoint: Interaction **P1**

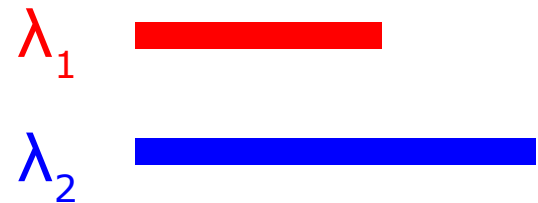


Recap: Monte Carlo Algorithm

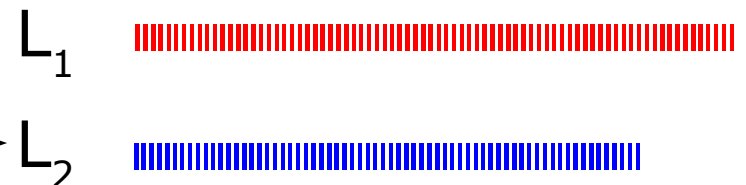
$$p_i(x) = 1 - \exp(-x/\lambda_i)$$

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

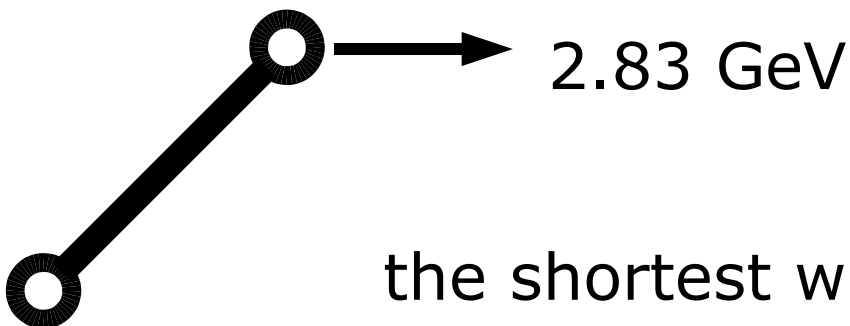
Step 2:



Drawing the random free path length from both processes:



the shortest wins \rightarrow



Recap: Monte Carlo Algorithm

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

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

(2) get values for λ_i for all relevant processes $i=1,2,\dots,m$

(3) for each process i ($i=1,2,\dots,m$):

sample L_i from $p_i(x)$

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

(5) transport incident particle by L_c

(6) simulate interaction

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

Step 2:

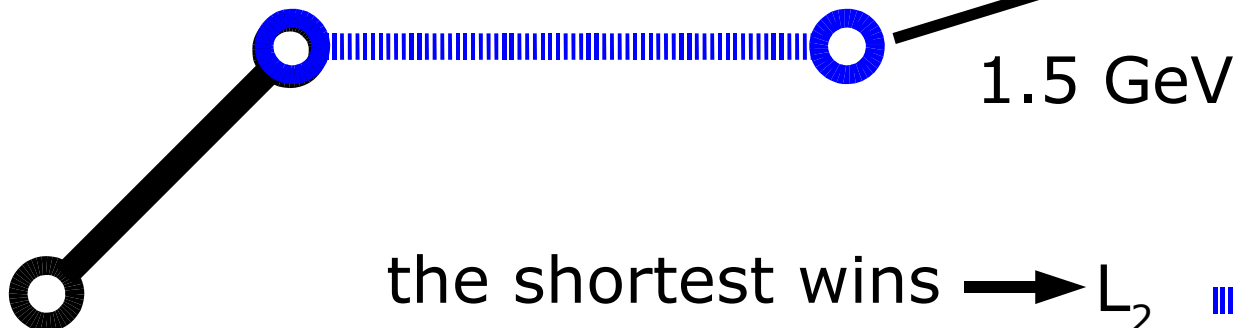
λ_1




λ_2



Interaction P2



the shortest wins $\rightarrow L_2$ 

same with 2 Materials

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

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

(2) get values for λ_i for all relevant processes $i=1,2,\dots,m$

(3) for each process i ($i=1,2,\dots,m$):

sample L_i from $p_i(x)$

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

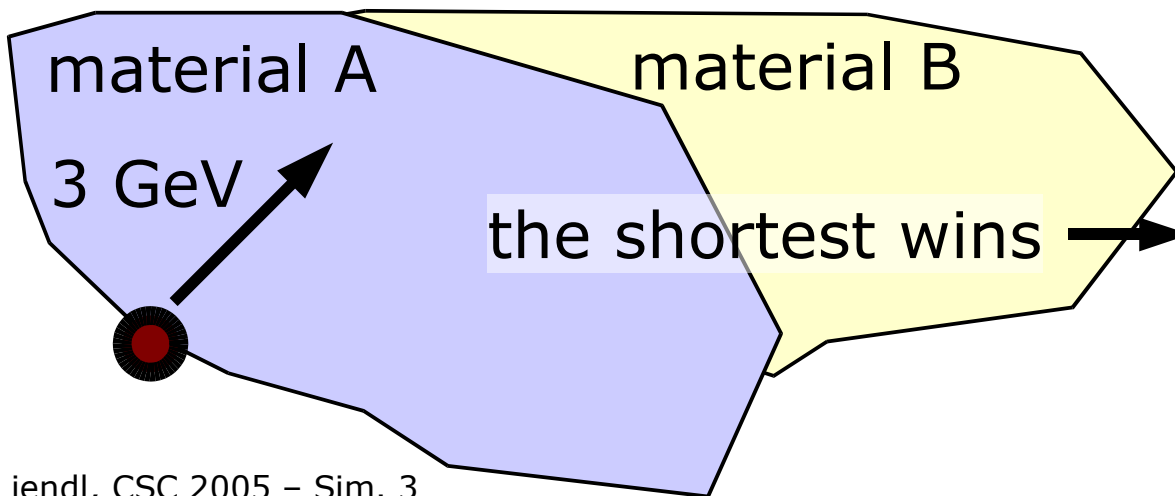
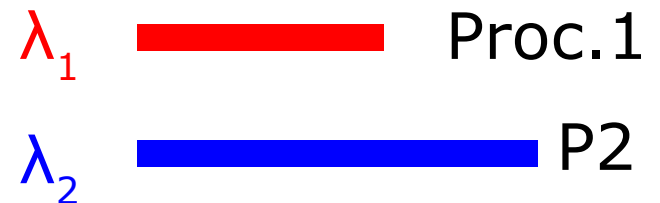
(5) transport incident particle by L_c

(6) simulate interaction

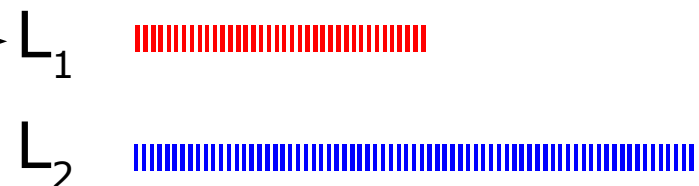
(7) if particle still exists: goto (1)

values for
material A

Step 1:



Drawing the random
free path length from
both processes:



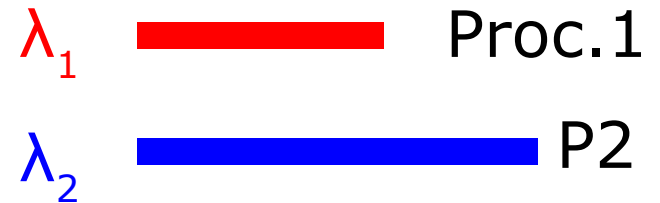
same with 2 Materials

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

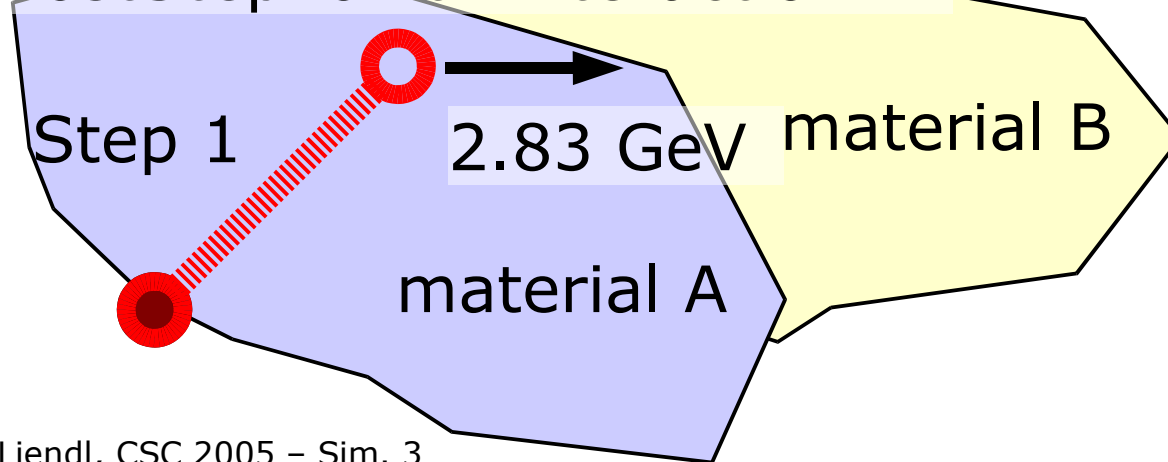
- (1) set properties for incident particle (energy, momentum, ..)
- (2) get values for λ_i for all relevant processes $i=1,2,\dots,m$
- (3) for each process i ($i=1,2,\dots,m$):
sample L_i from $p_i(x)$
- (4) $L_c = \min(L_i)$ from all sampled L_i , c in $(1,2,\dots,m)$
- (5) transport incident particle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

values for
material A

Step 1:



PostStepPoint: Interaction **P1**



L_1

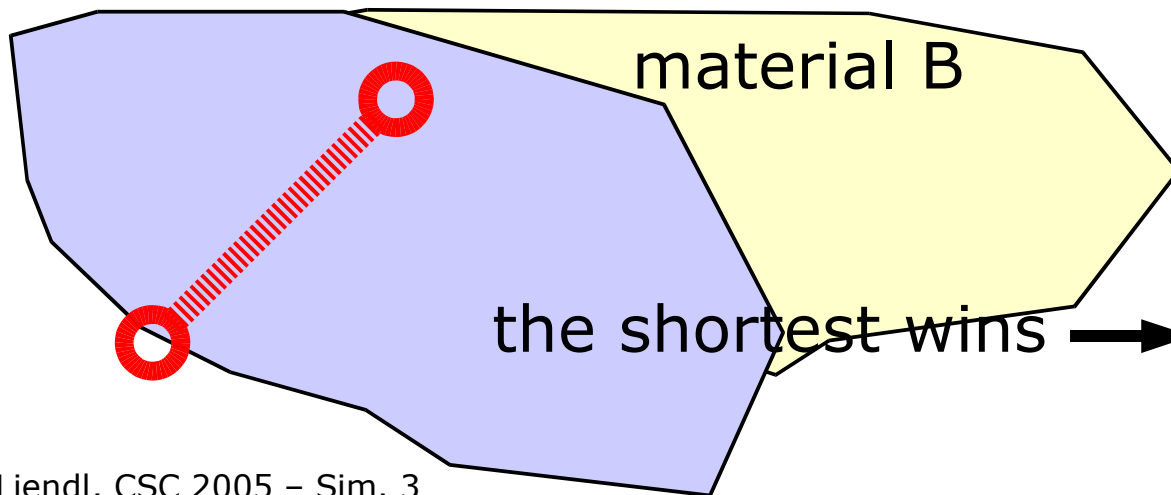
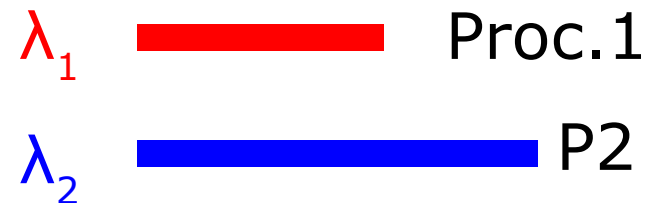
same with 2 Materials

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

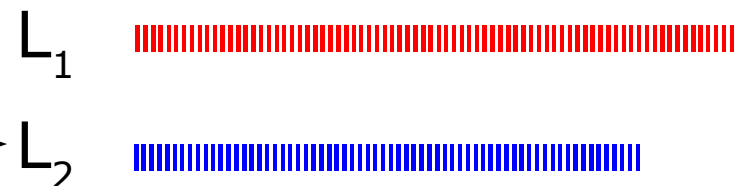
- (1) set properties for incident particle (energy, momentum, ..)
- (2) get values for λ_i for all relevant processes $i=1,2,\dots,m$
- (3) for each process i ($i=1,2,\dots,m$):
sample L_i from $p_i(x)$
- (4) $L_c = \min(L_i)$ from all sampled L_i , c in $(1,2,\dots,m)$
- (5) transport incident particle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

values for
material A

Step 2:



Drawing the random
free path length from
both processes:



same with 2 Materials

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

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

(2) get values for λ_i for all relevant processes $i=1,2,\dots,m$

(3) for each process i ($i=1,2,\dots,m$):

sample L_i from $p_i(x)$

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

(5) transport incident particle by L_c

(6) simulate interaction

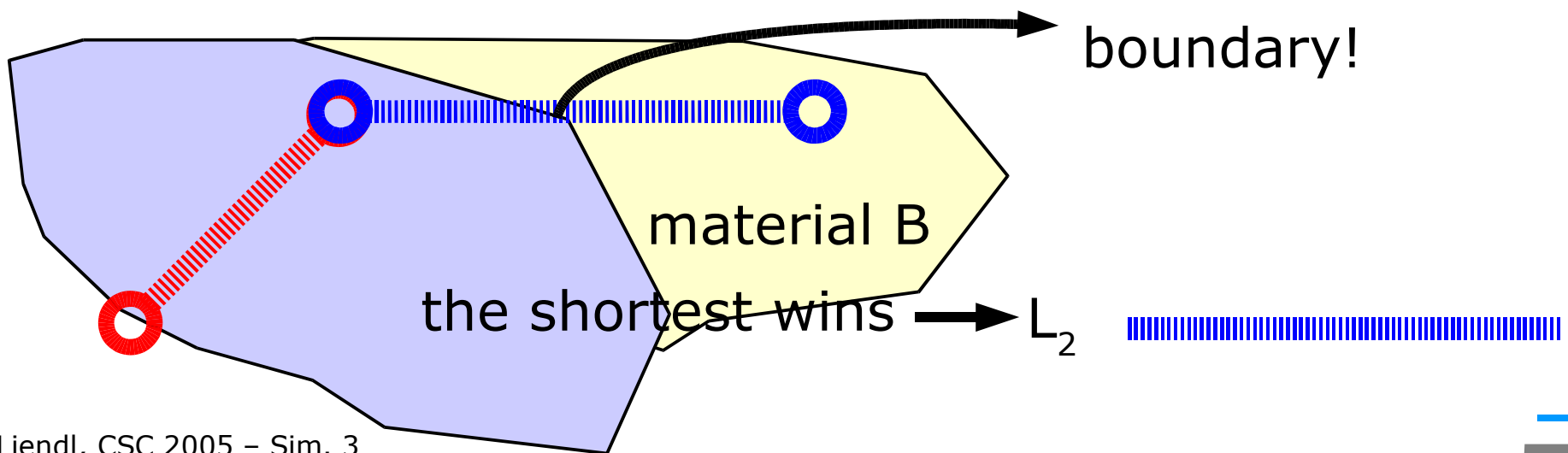
(7) if particle still exists: goto (1)

values for
material A

Step 2:

λ_1  Proc.1

λ_2  P2



same with 2 Materials

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

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

(2) get values for λ_i for all relevant processes $i=1,2,\dots,m$

(3) for each process i ($i=1,2,\dots,m$):

sample L_i from $p_i(x)$

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

(5) transport incident particle by L_c

(6) simulate interaction

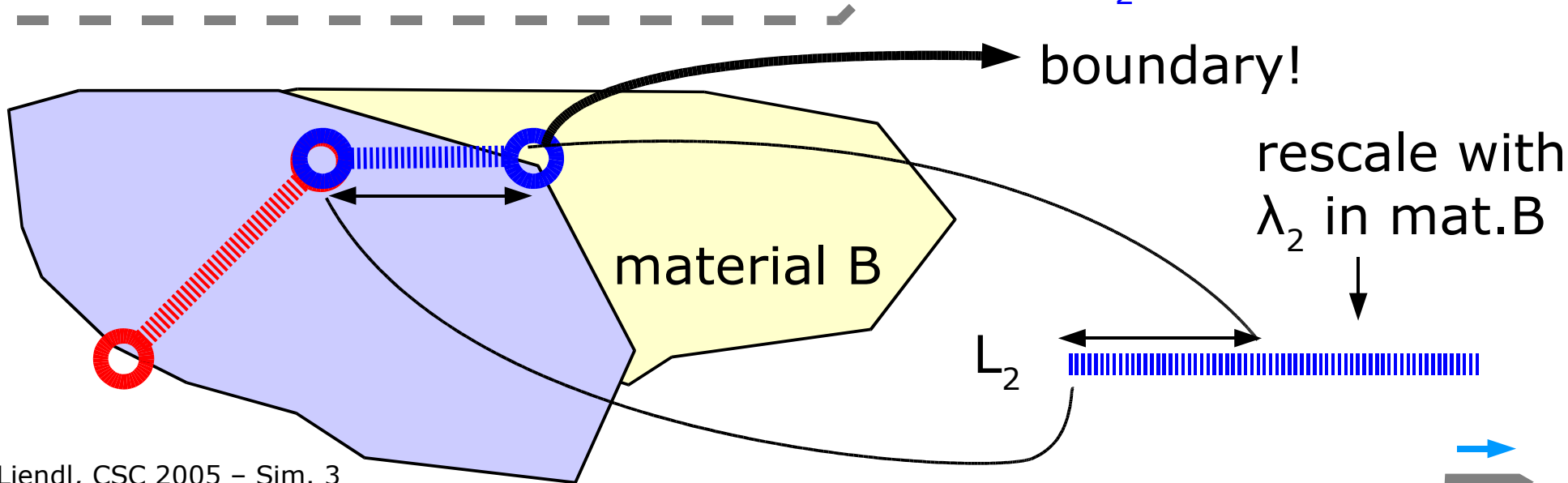
(7) if particle still exists: goto (1)

values for
material A

Step 2:

λ_1  Proc.1

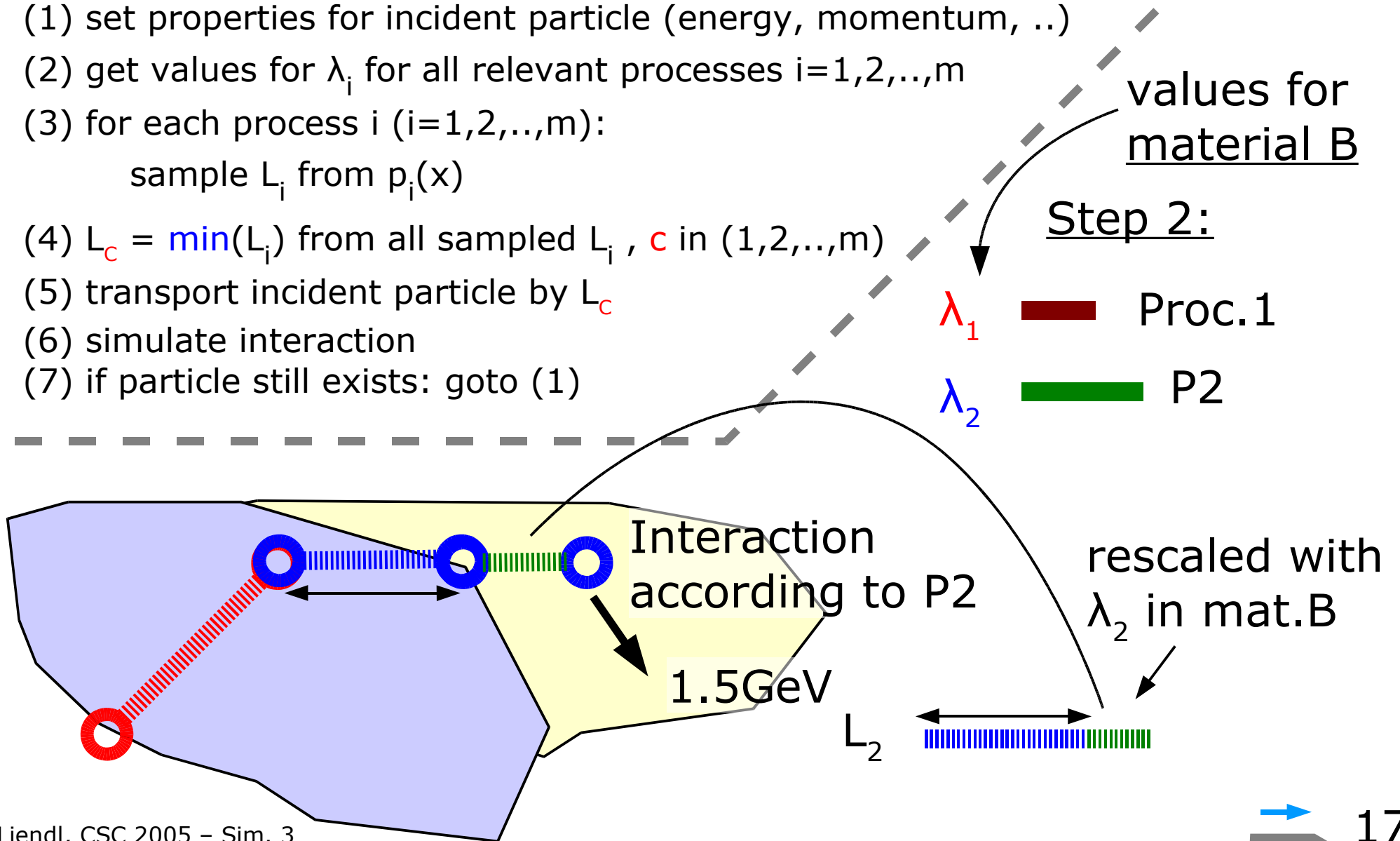
λ_2  P2



same with 2 materials

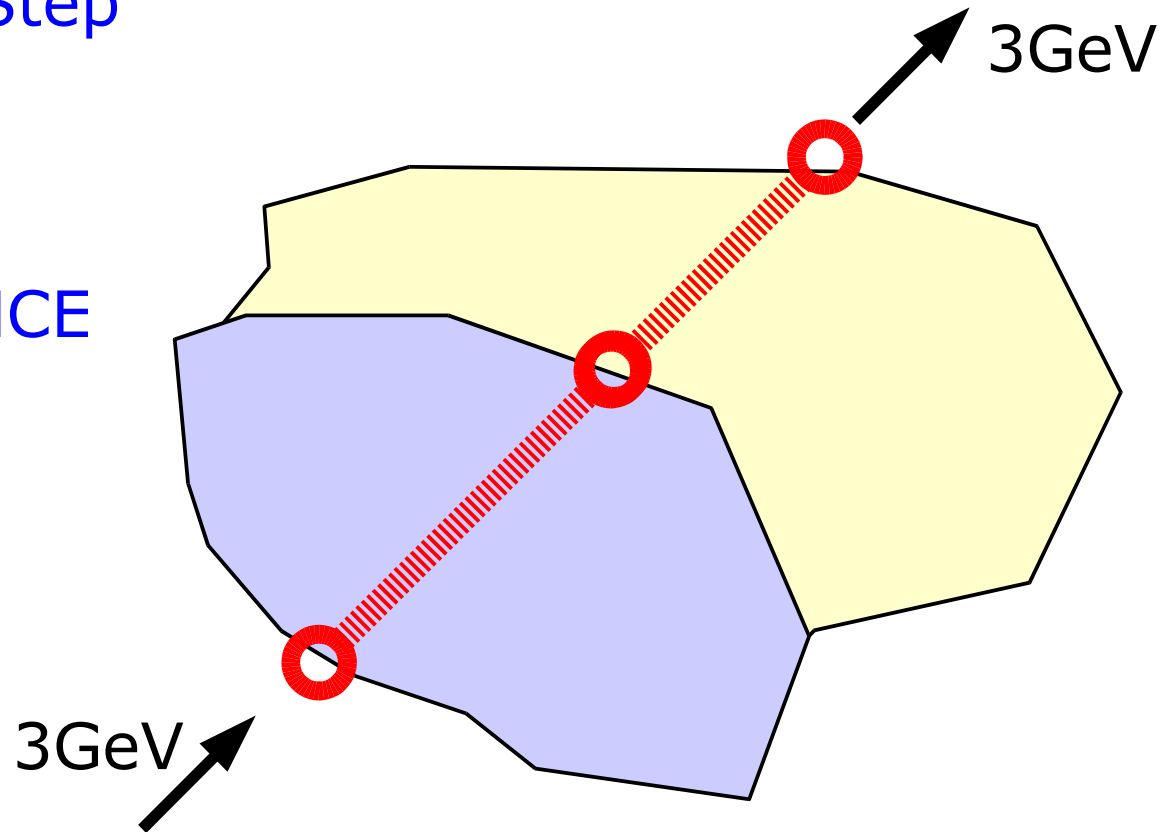
$$p_i(x) = 1 - \exp(-x/\lambda_i)$$

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

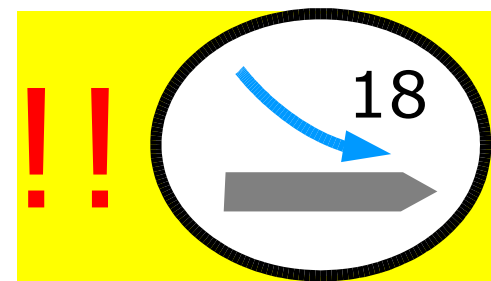


Absence of Physics

An instance of a G4Step corresponds to the trajectory of a free, undisturbed particle ONLY IN THE ABSENCE OF "CONTINUOUS" PHYSICAL PROCESSES AND EXTERNAL FIELDS!



Nevertheless, boundary crossings are marked by introducing StepPoints

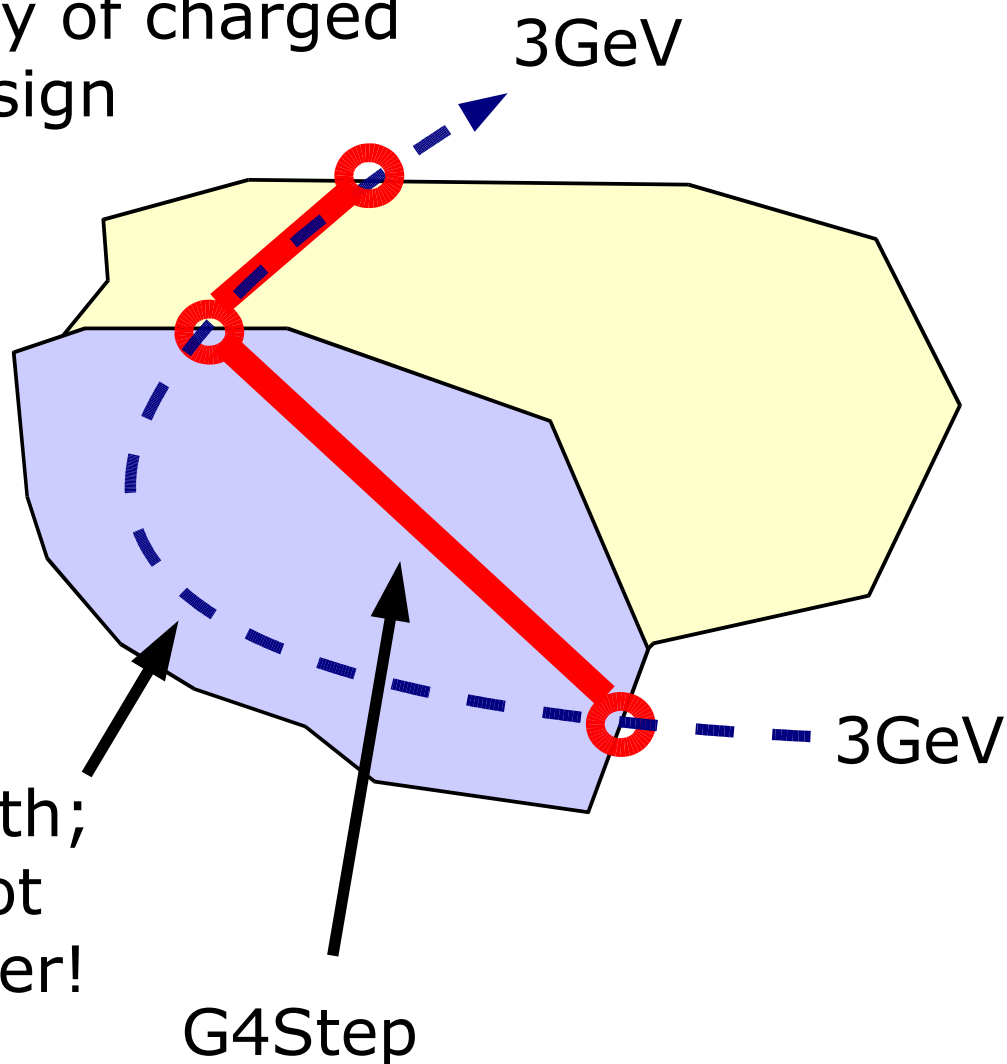


Magnetostatic Fields

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

Static B-fields don't change the energy of particles – all physics processes assumed to be switched off!

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

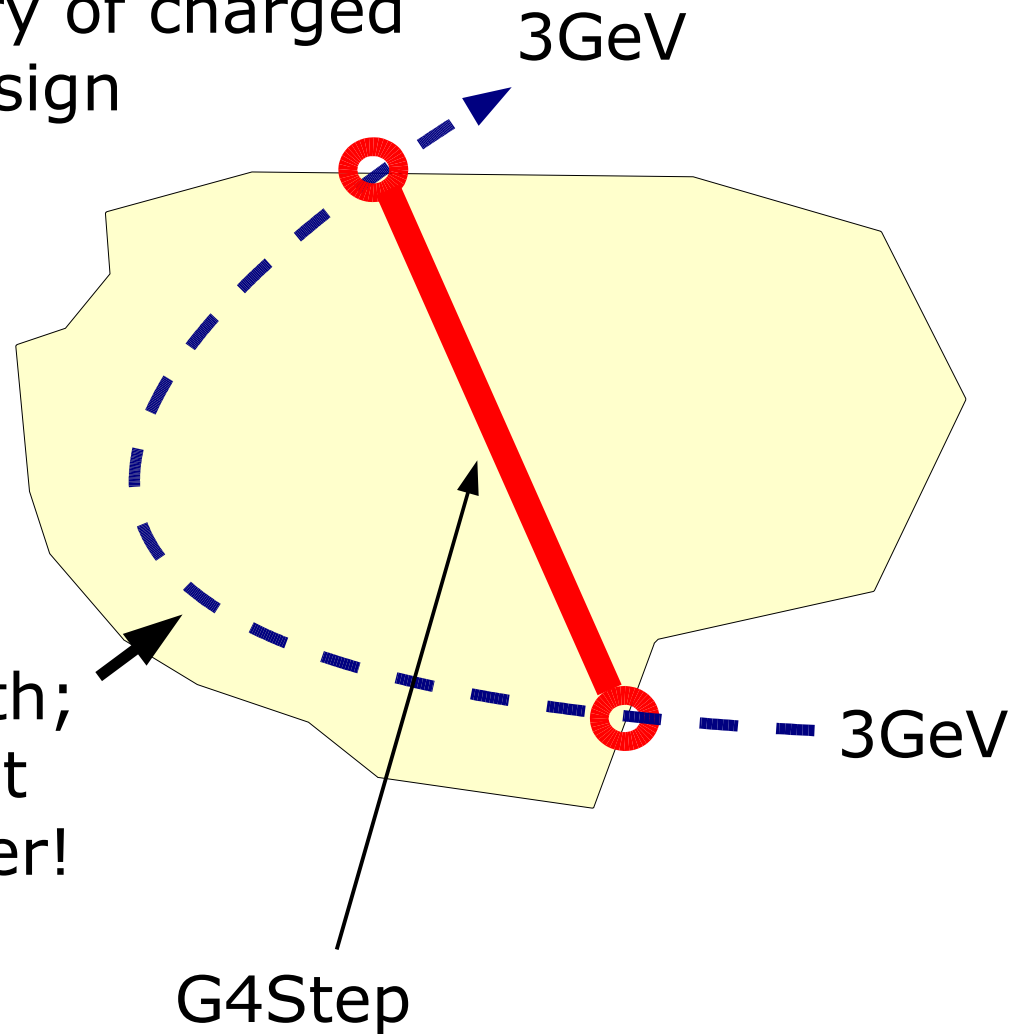


Magnetostatic Fields

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

Static B-fields don't change the energy of particles!

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



Implementing a B-field

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] } global space point  
point[2] } (x,y,z)  
  
point[3] global, laboratory time
```

Output, provided by

```
*(field+0)  
*(field+1) } magnetic  
*(field+2) } component  
  
*(field+3) } electric  
*(field+4) } component  
*(field+5)
```

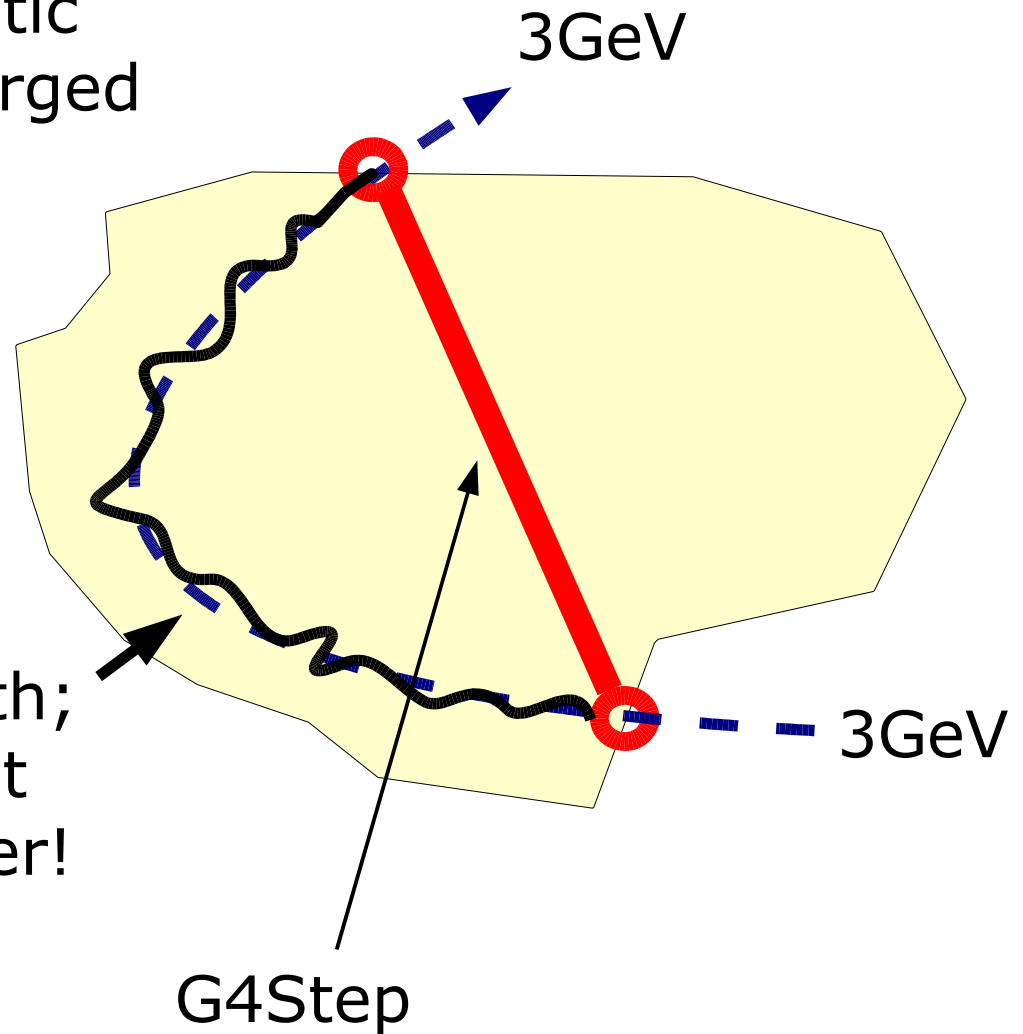


Multiple Coulomb Scattering

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

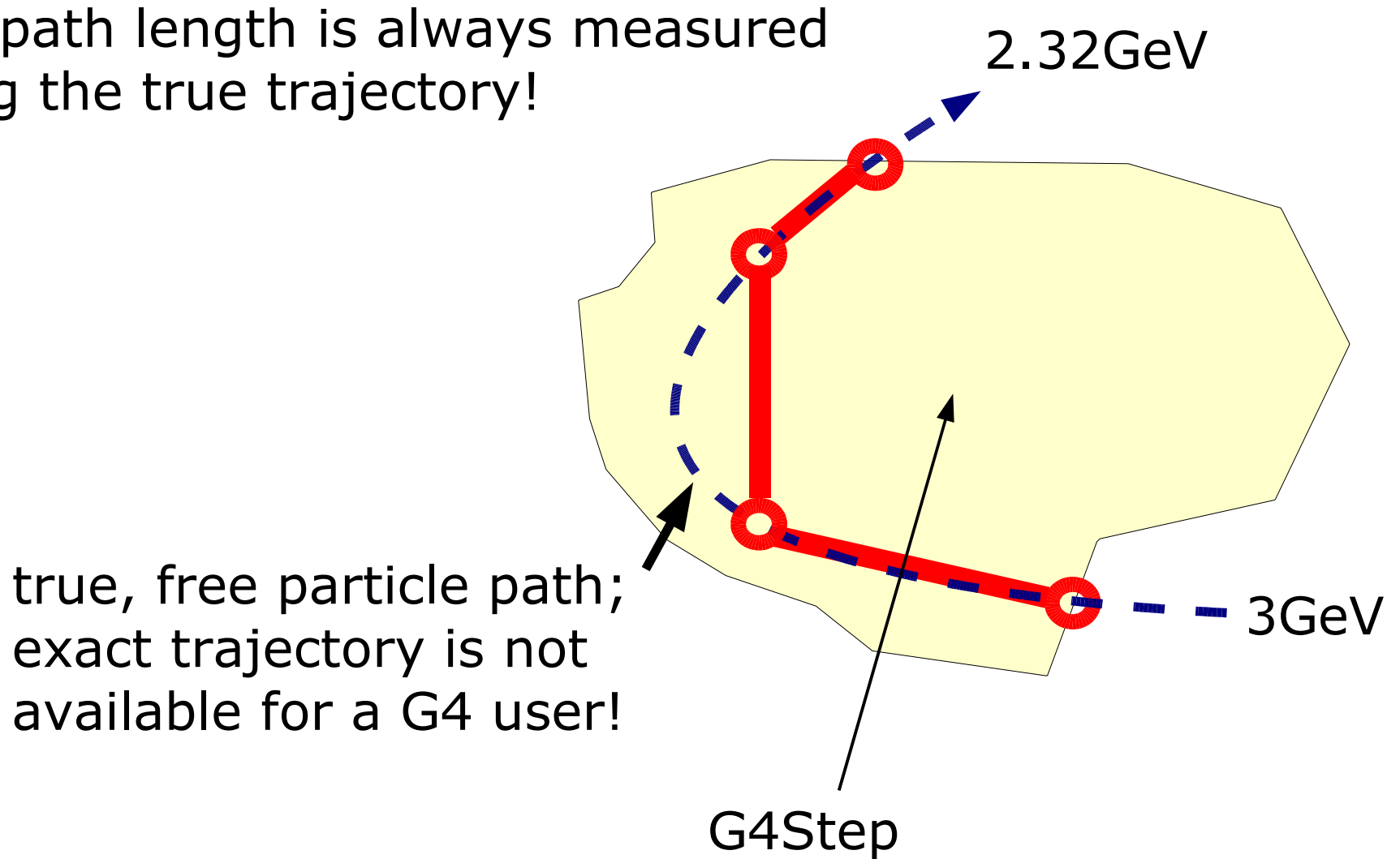
Is respected by G4 -
but not visible to
the user!

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

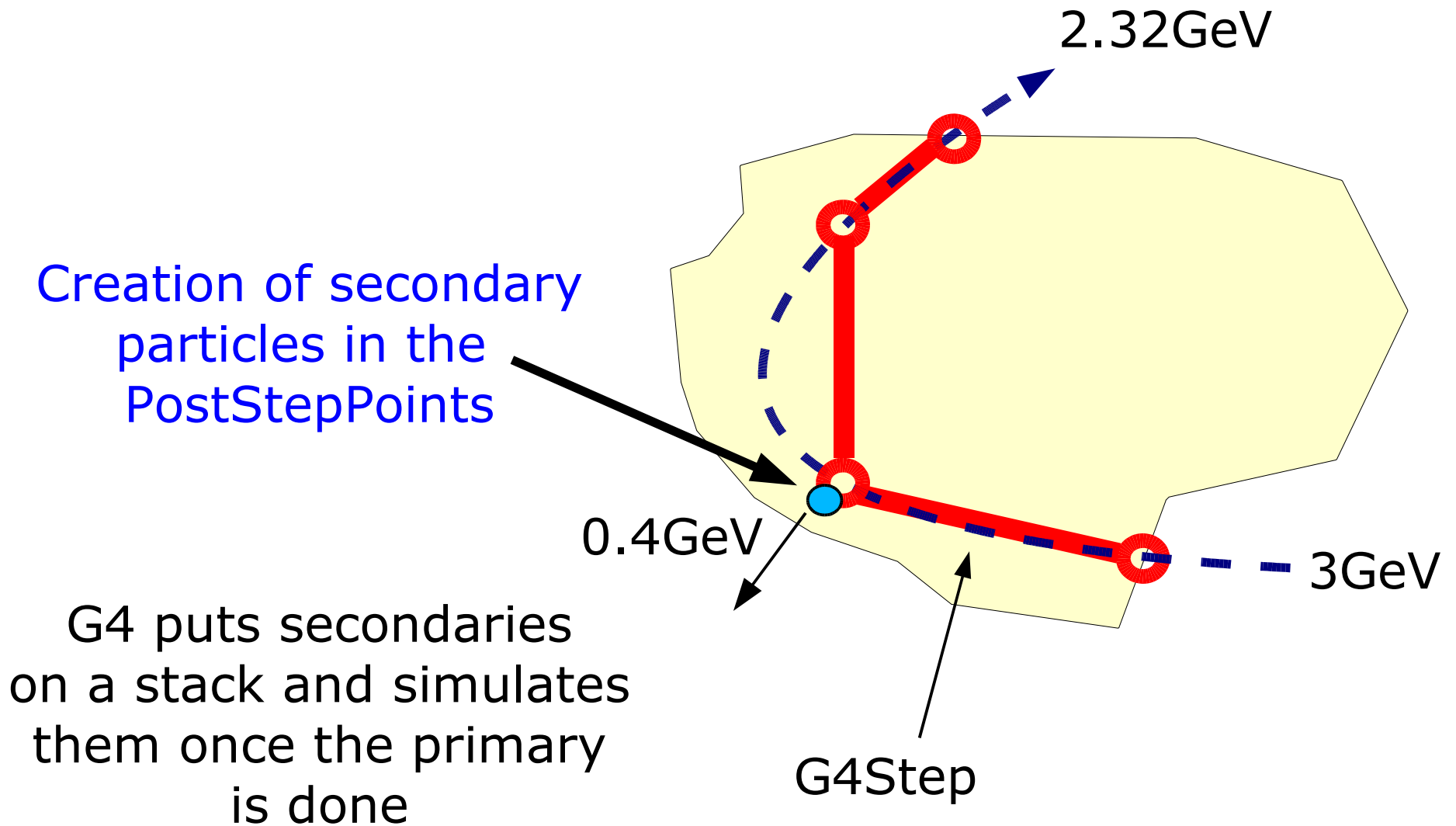


Processes “turned on”

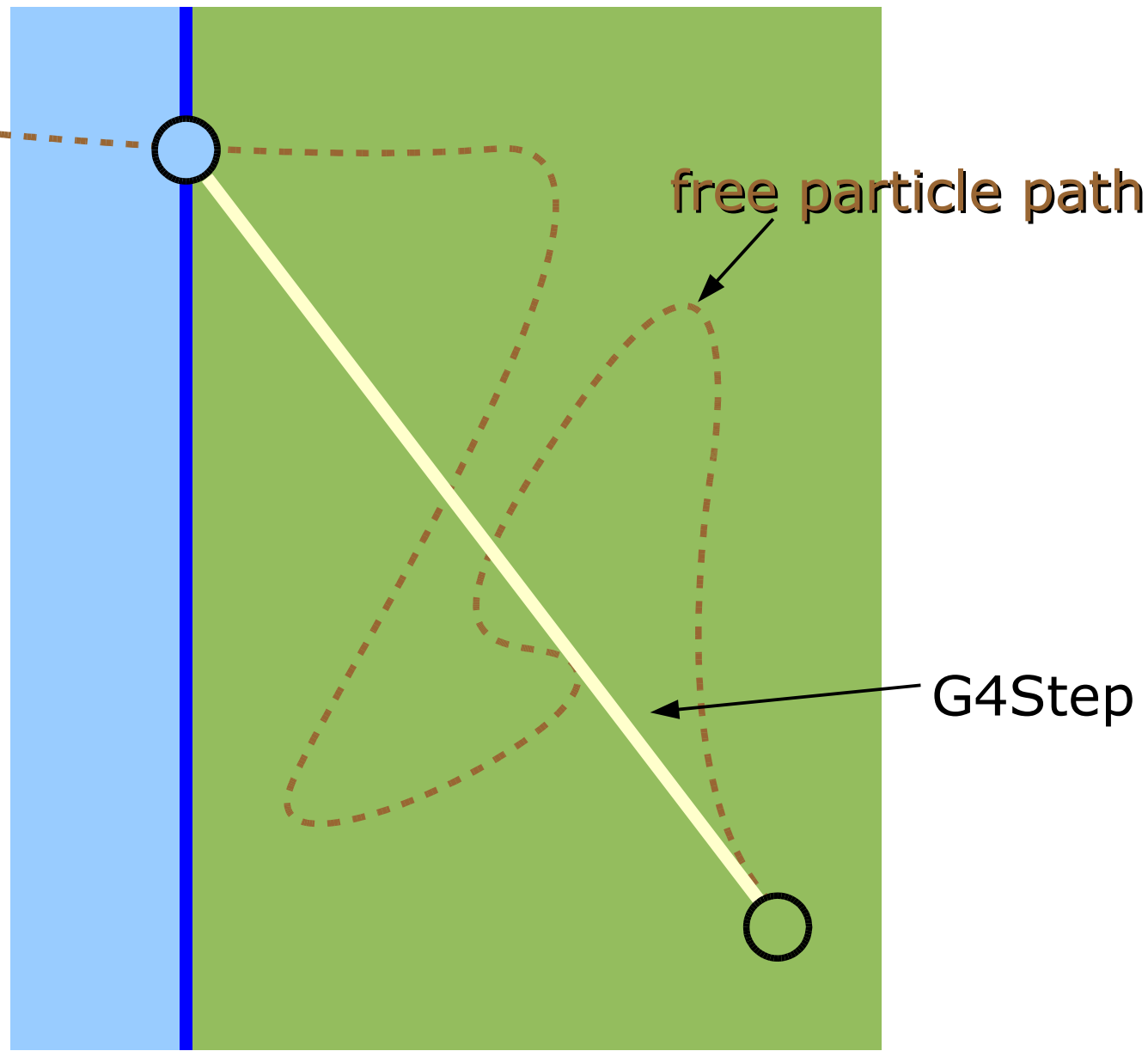
free path length is always measured along the true trajectory!



Processes “turned on”



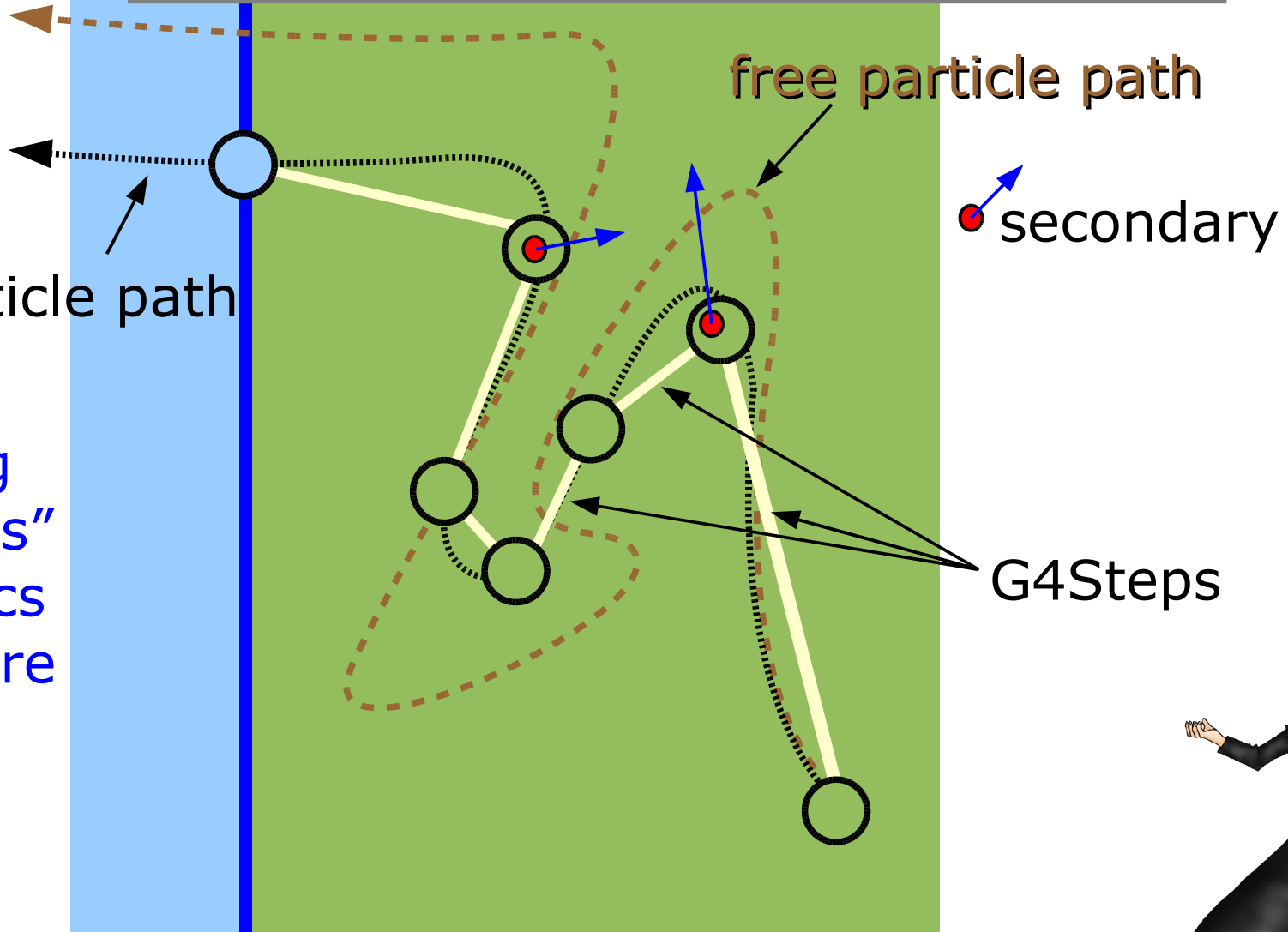
Elegant "Fred Astaire - Steps" with B-field and no G4-physics processes





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

physical particle path



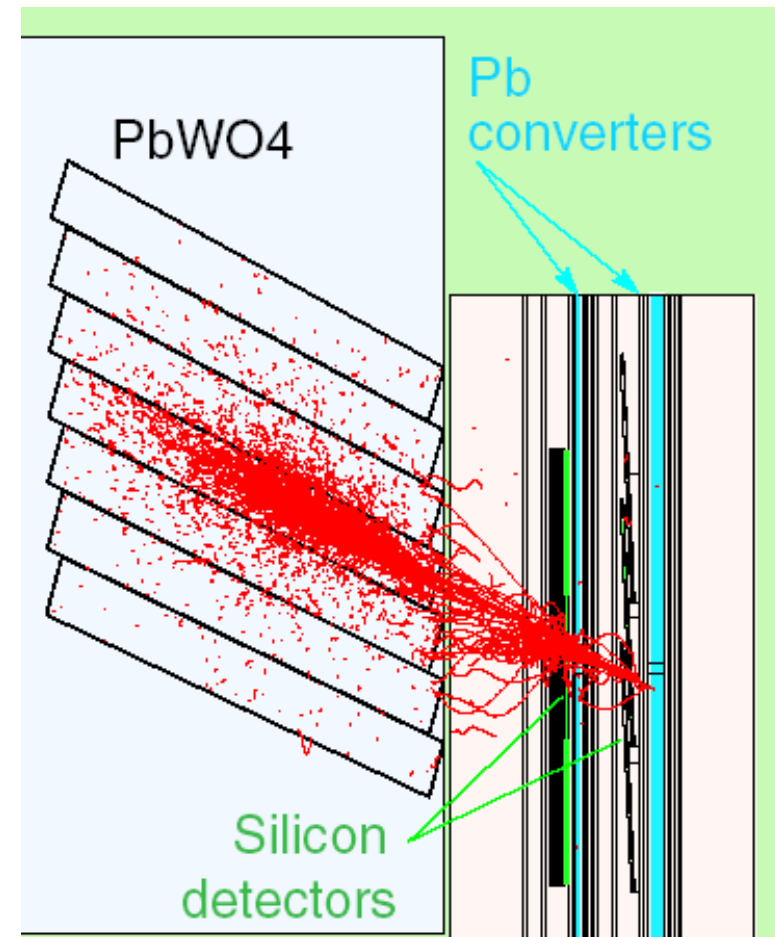
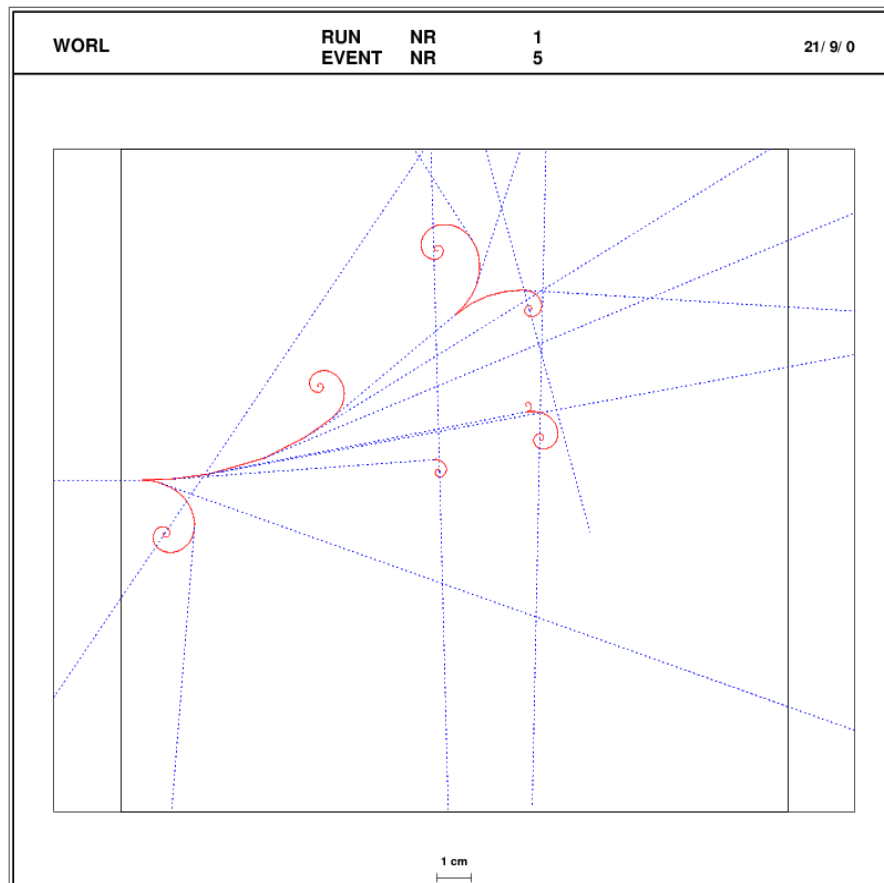
vs. "fighting Matrix-Steps" when physics processes are enabled



Example: electromagnetic shower

Especially when high energetic e^- , e^+ , or photons hit some dense material, electromagnetic showers develop. Many, many secondaries. Demand lots of computing time!

γ 200 MeV in $2 X_0$ Aluminium. Pair + brem

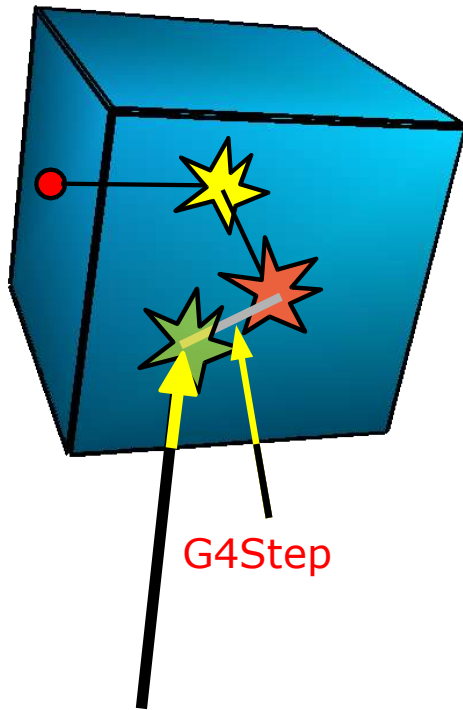


The GEANT4 Physics Model

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 the first interaction due to **process i** after a free 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 particles get!

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 particles get!

Introduction of "Cuts":

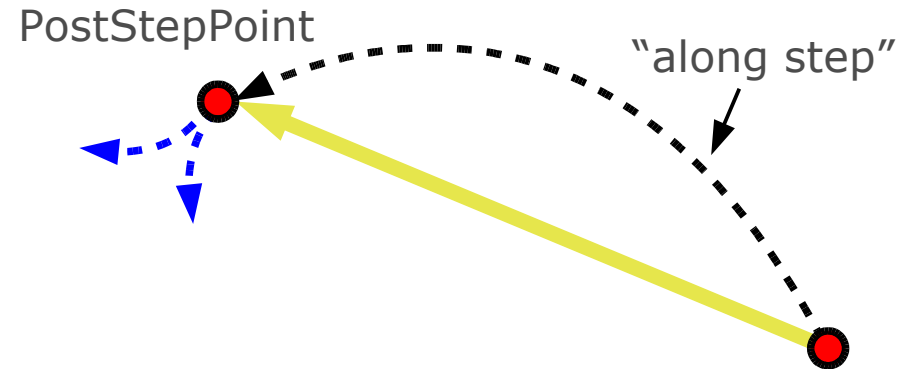
- > below a certain energetic threshold, we only take the average effect of a process into account!
- > in GEANT4: "along" the step
- > above the threshold: exact treatment according to quantum mechanics

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!

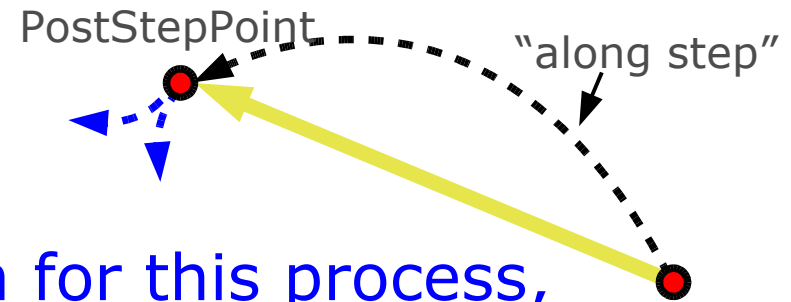


double PostStep
double AlongStep
double AtRest } GetPhysicalInteractionLength()

G4VParticleChange * PostStep
G4VParticleChange * AlongStep
G4VParticleChange * AtRest } DoIt() won't look at them ..
(lack of time)

G4VProcess: PostStep

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



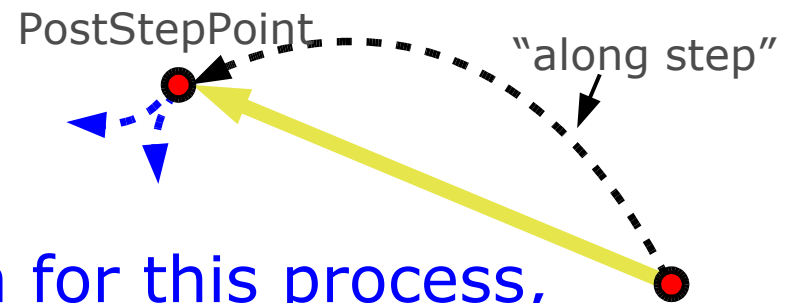
returns the mean free path for this process,
used to sample the free path from the exp.distribution

double PostStep }
double AlongStep } GetPhysicalInteractionLength()

G4VParticleChange * PostStep }
G4VParticleChange * AlongStep } DoIt()

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 PostStep }
double AlongStep } GetPhysicalInteractionLength()

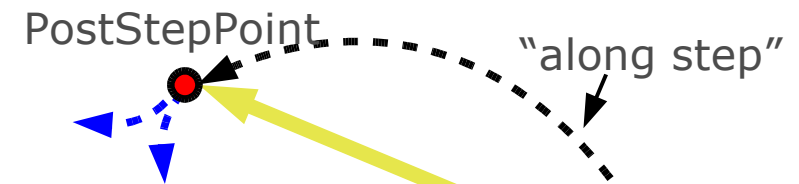
G4VParticleChange * PostStep }
G4VParticleChange * AlongStep } DoIt()

PostStepDoit is called if this process returned the shortest λ .
Sampling of the new state of the tracked particle,
generation of new particles, ... using the Monte Carlo method
based on distributions of the quantum theory of the interaction

G4VProcess: AlongStep

Along step methods for average process description

```
double PostStep }  
double AlongStep } GetPhysicalInteractionLength()
```



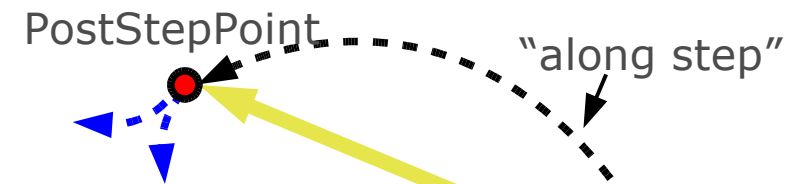
Returns the **distance** beyond which the average approximation is too crude. This distance, **if shorter** than any other along-step distance or any sampled free path of other processes, limits the step width!

```
G4VParticleChange * PostStep }  
G4VParticleChange * AlongStep } DoIt()
```

G4VProcess: AlongStep

Along step methods for average process description

```
double PostStep }  
double AlongStep } GetPhysicalInteractionLength()
```



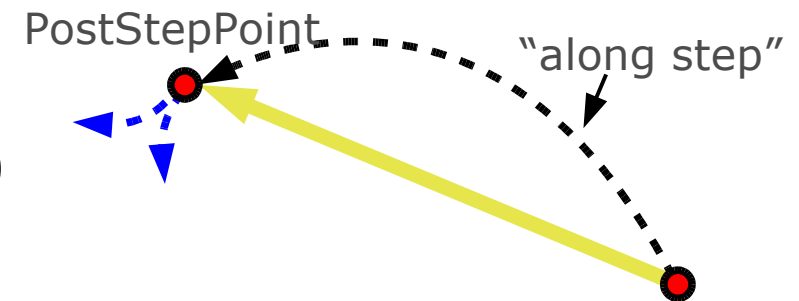
Returns the **distance** beyond which the average approximation is too crude. This distance, **if shorter** than any other along-step distance or any sampled free path of other processes, limits the step width!

```
G4VParticleChange * PostStep }  
G4VParticleChange * AlongStep } DoIt()
```

The AlongStepDoIts() of all processes are **always called before the PostStepDoIt()** to account for the average description of the processes, e.g. average energy loss along the step

G4VProcess & Particles

```
G4VParticleChange * PostStep  
G4VParticleChange * AlongStep } DoIt()
```



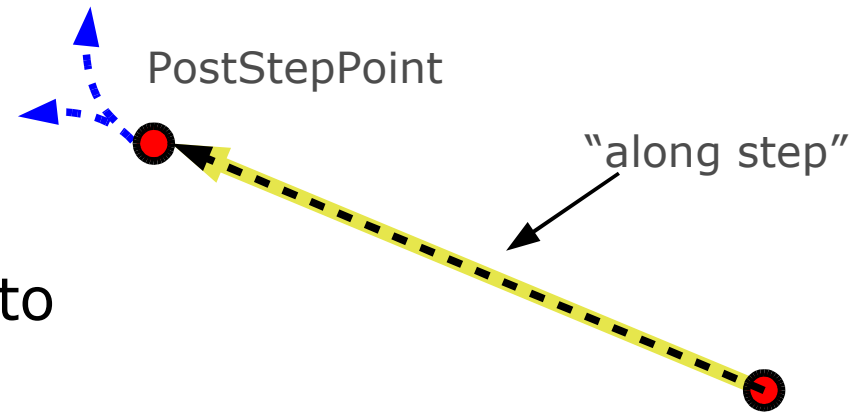
DoIt methods return **G4VParticleChange**. A G4VParticleChange contains all the “delta” information of the dynamic particle between the PreStep and PostStep, including a list of newly created dynamic particles!

Processes are only invoked for particles which have been “declared” by the user by instantiating one static instance of a **G4ParticleDefinition**. Therefore, the method:
virtual G4boolean G4VProcess::isApplicable(G4ParticleDefinition *)
is implemented for each process.

Example:

```
class G4GammaConversion:  
  public ... G4VProcess :
```

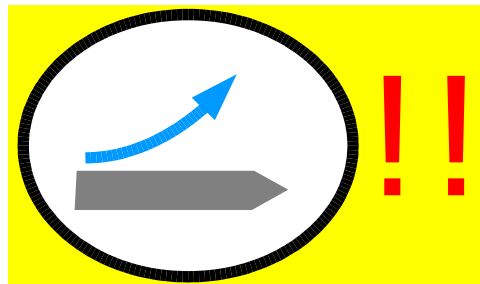
- the tracked photon is converted into an electron – positron pair



GetPhysicalInteractionLength()

returns the free path length (x-section of gamma-conversion, material properties, sampling from $P(L) = \exp(-L/\lambda)$)

DoIt() creates the (e-, e+) pair (sampling of energy, momentum, killing of photon)

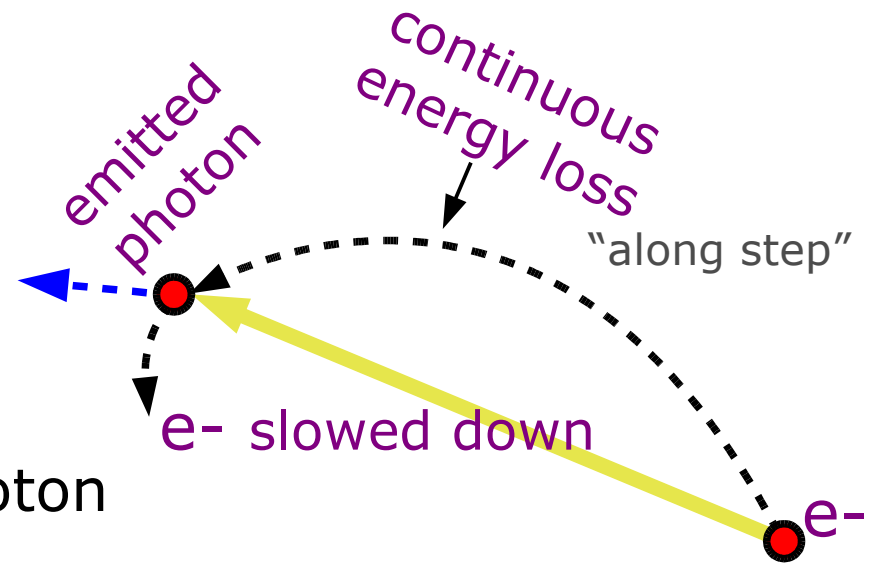


Example:

`class G4eBremsstrahlung :`

`public G4VProcess`

- along step: average energy loss of the e⁻
- post step: emission of a brems-photon



`PostStepPhysicalInteractionLength()`

returns a free path length until the photon emission

`PostStepDoit()` emits the photon

`AlongStepPhysicalInteractionLength()`

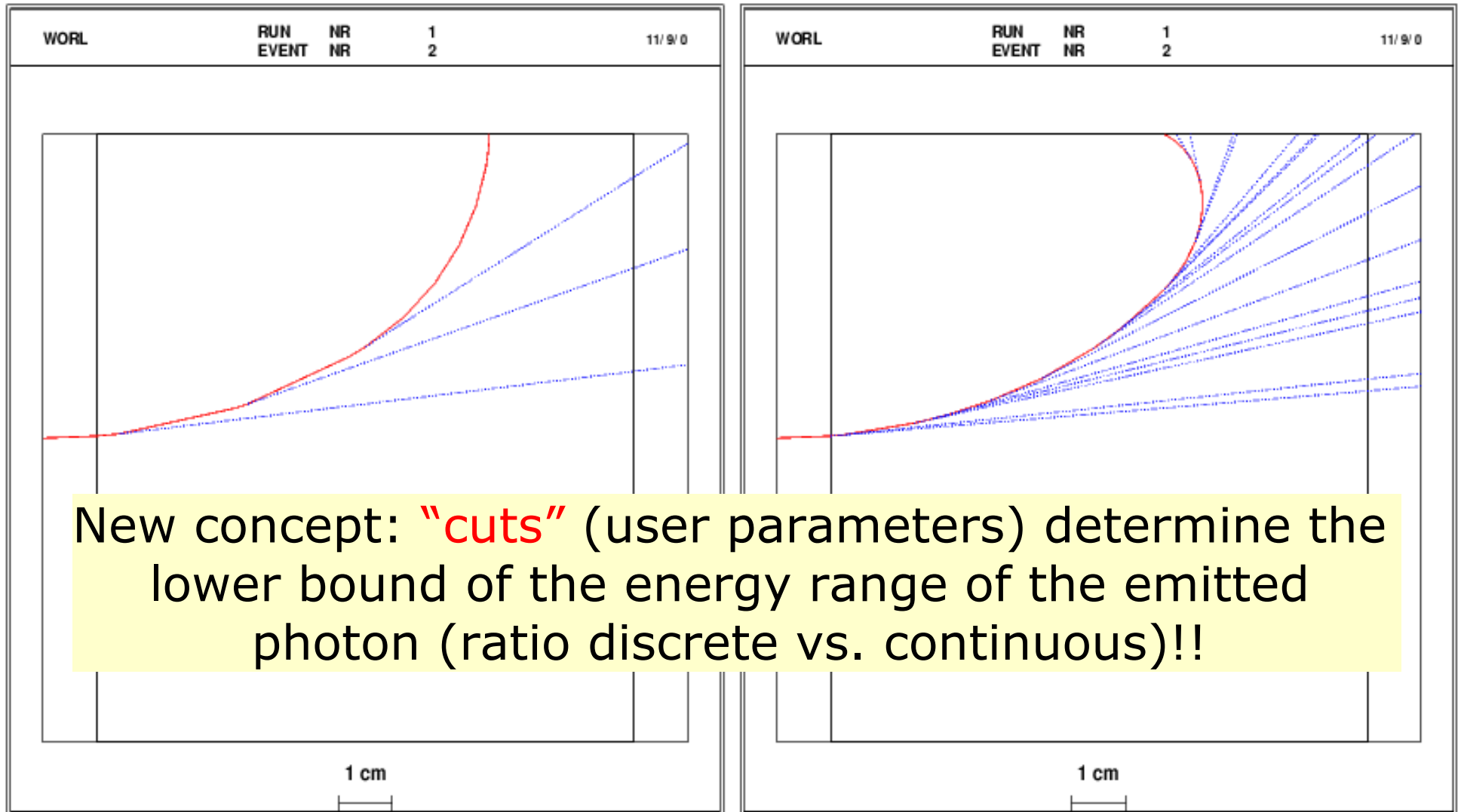
returns a step limit compatible with the employed dE/dx model

`AlongStepDoit()` calculate dE/dx for the actual step taken

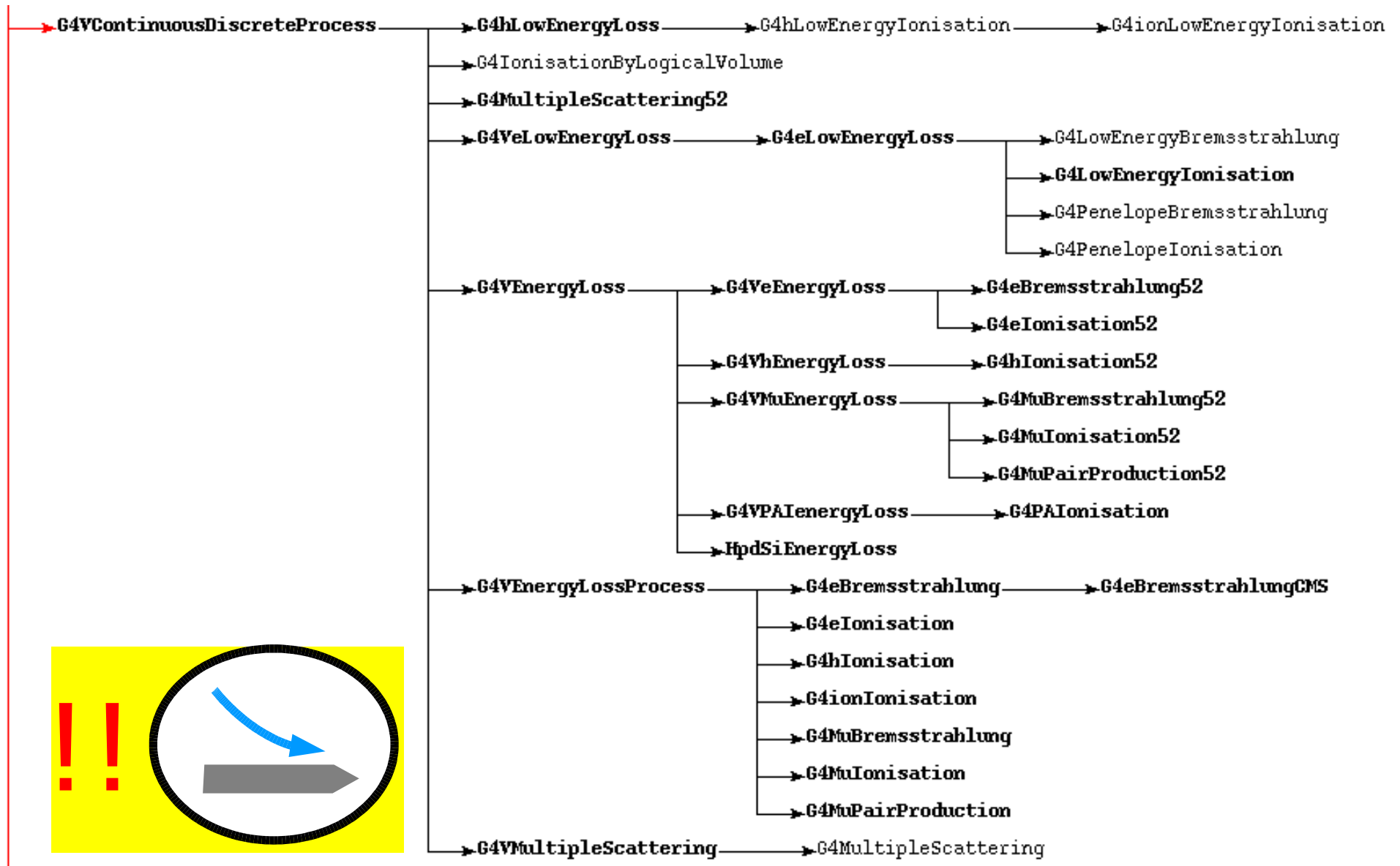
(note: the step taken is not necessarily due to Bremsstrahlung ...)

Example:

e^- 200 MeV in 10 cm Aluminium (cut: 1 MeV, 10 keV). Field 5 tesla



Lots of Processes!!



(only continuous-discrete electromagnetic processes shown)

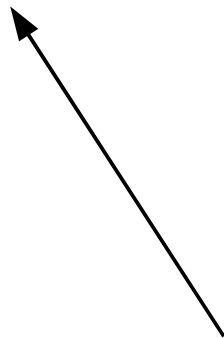
Categories

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

Categories

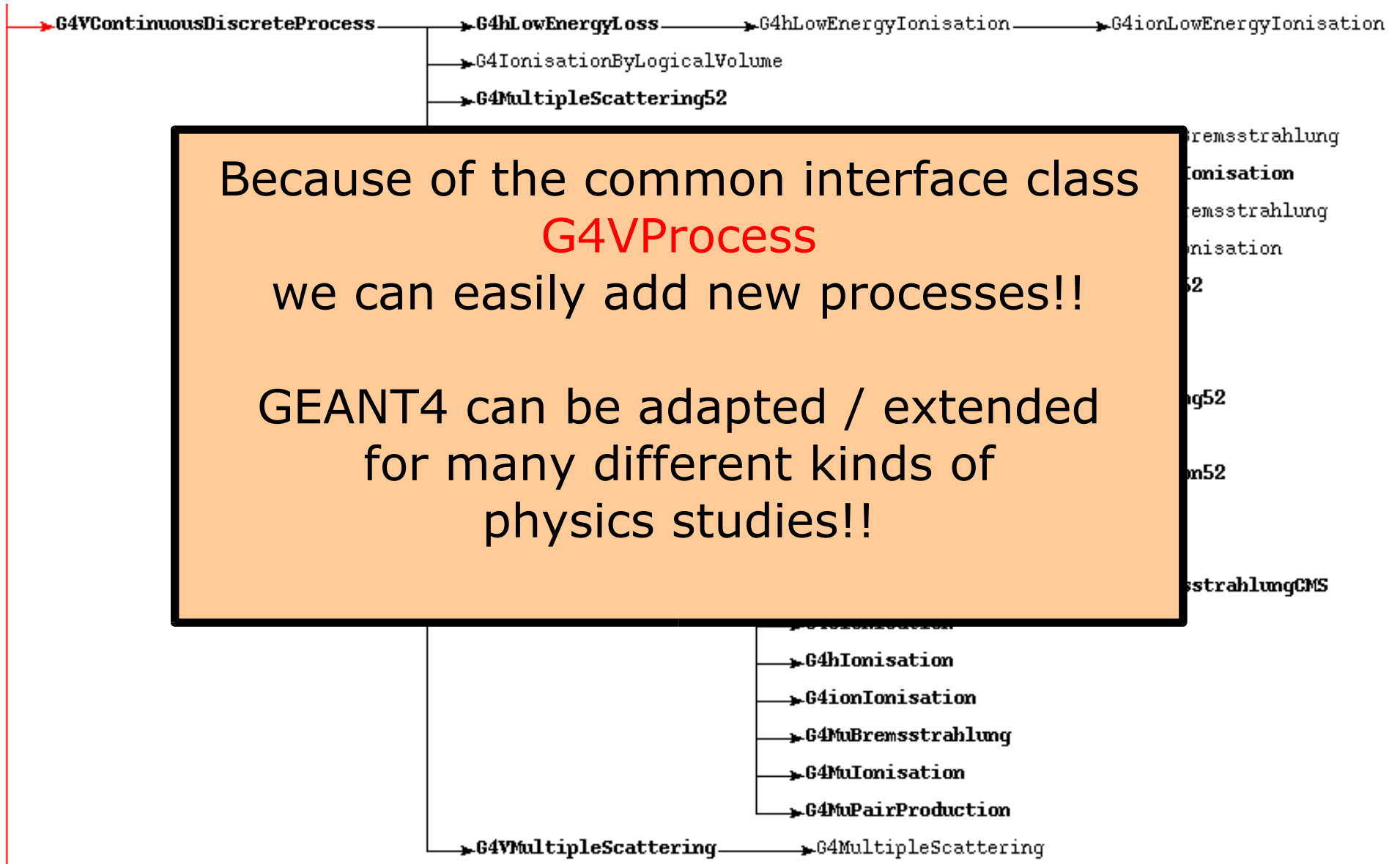
Stepping to volume boundaries is implemented as **G4VProcess!**

Free path length is simply the distance to the next boundary, and the `DoIt()` moves the particle!!



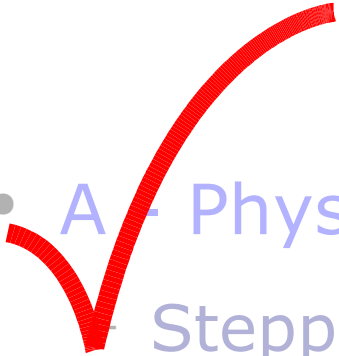
- **Electromagnetic**
 - standard
 - low energy
- **Hadronic**
 - pure hadronic
 - radioactive decay
 - photo- and electro-nuclear
- Decay
- Optical photon
- Parameterization
- **Transportation**

Processes and more!!



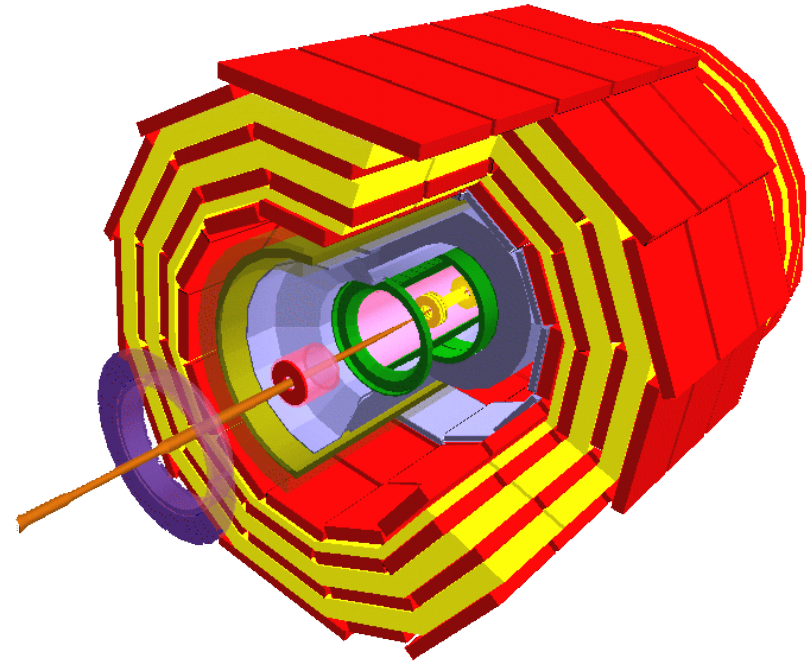
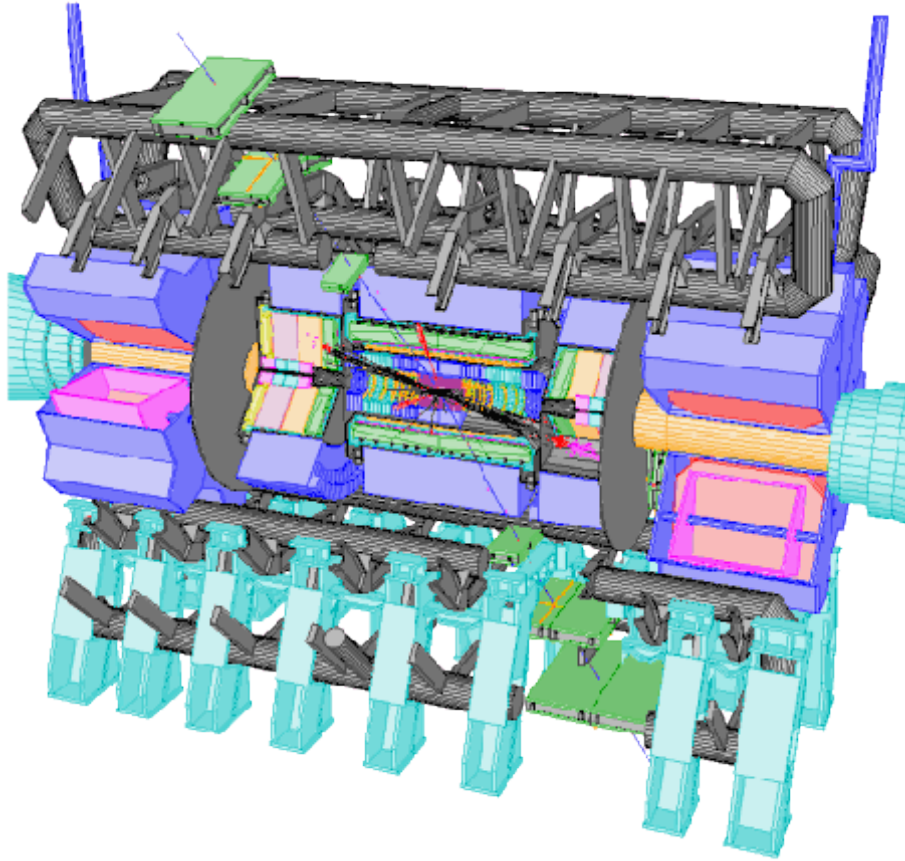
In the last lecture, we will see how we **associate a process with a tracked particle ...**

- > GEANT4 physics list
- > turning on / off of processes

- 
- **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!

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 fulfill these constraints, e.g. introduction of artificial envelope volumes

General Requirements

- **Must be compatible with detector models employed by other experiment software:**
 - Engineering data / CADs: describe the same detector as it is actually planned to be built!
 - Event reconstruction: 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 ...)
 - Field calculations: 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

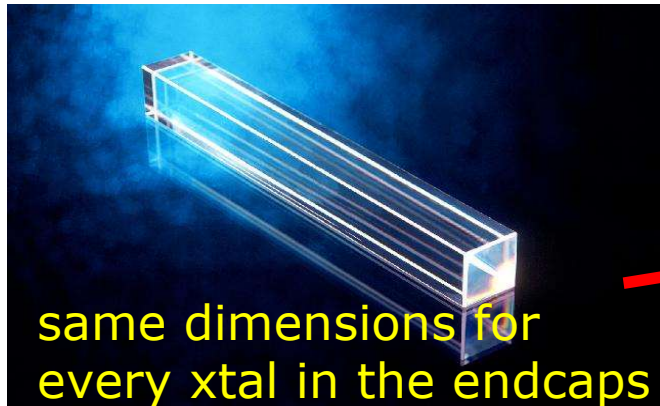
Compatibility between different descriptions of the same detector

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
- Extensible model
 - define custom shapes / solids

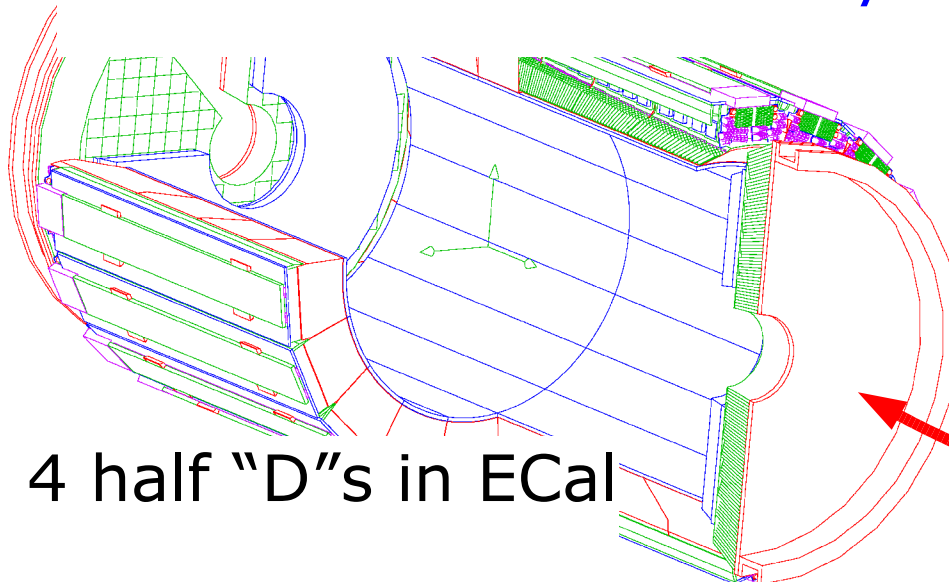
Example: CMS Endcap ECal

Lead-tungstate crystal
 $\sim 2 \times 2 \times 20$ cm

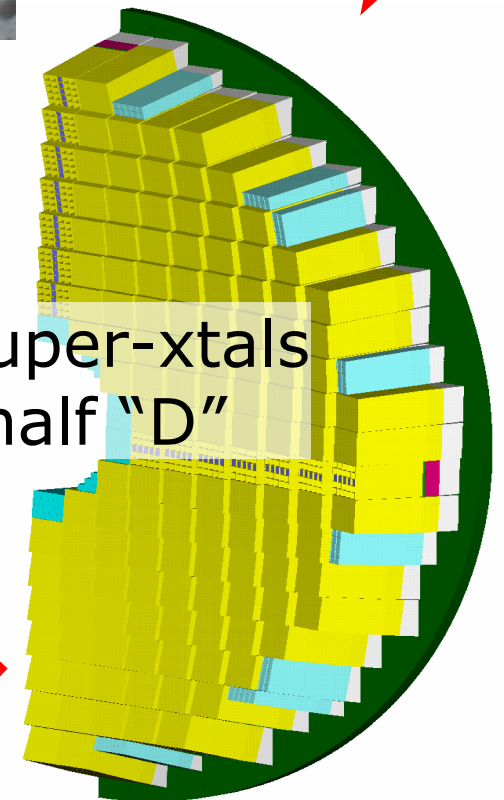


5 x 5 crystals
in a super-xtal

CMS ECal endcaps consist of
 $\sim 5 \times 5 \times 140 \times 4 = 14000$ crystals!



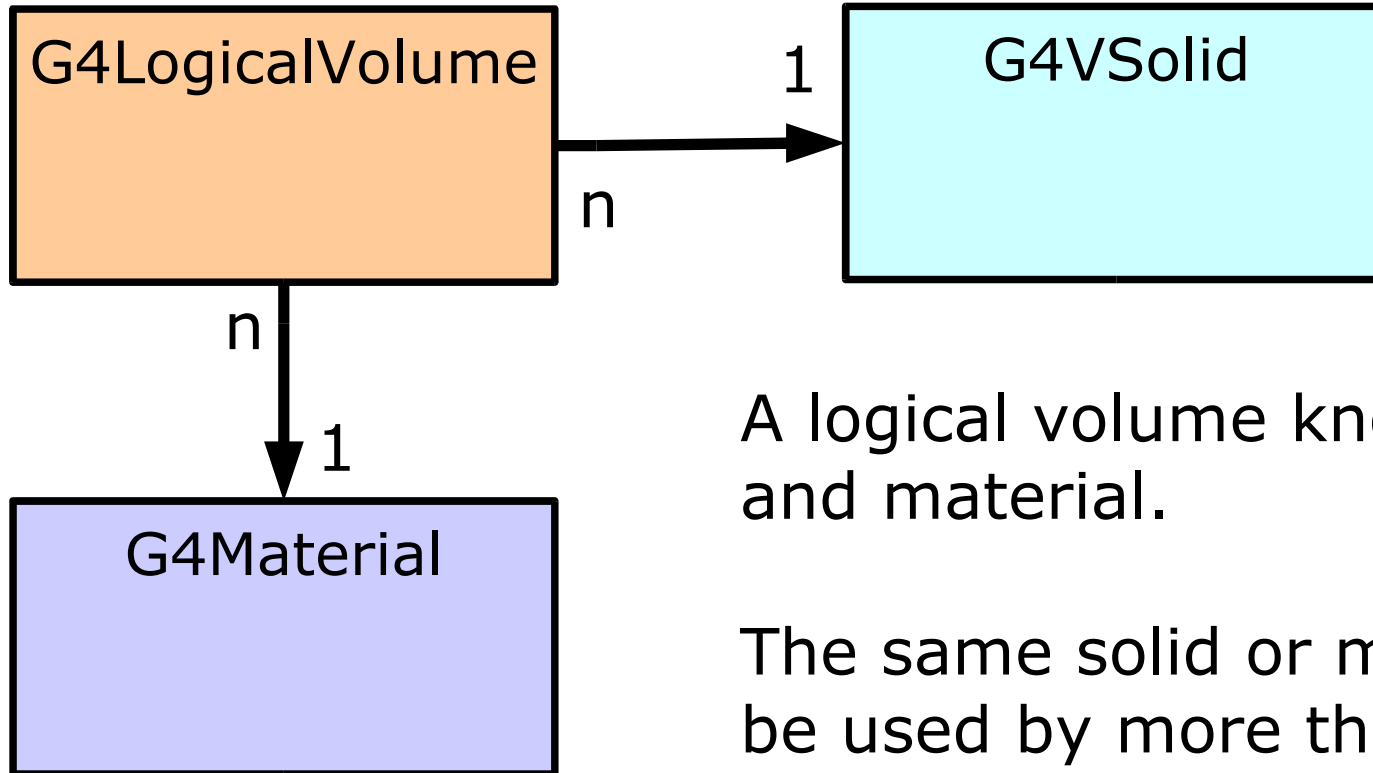
~ 140 super-xtals
in a half "D"



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



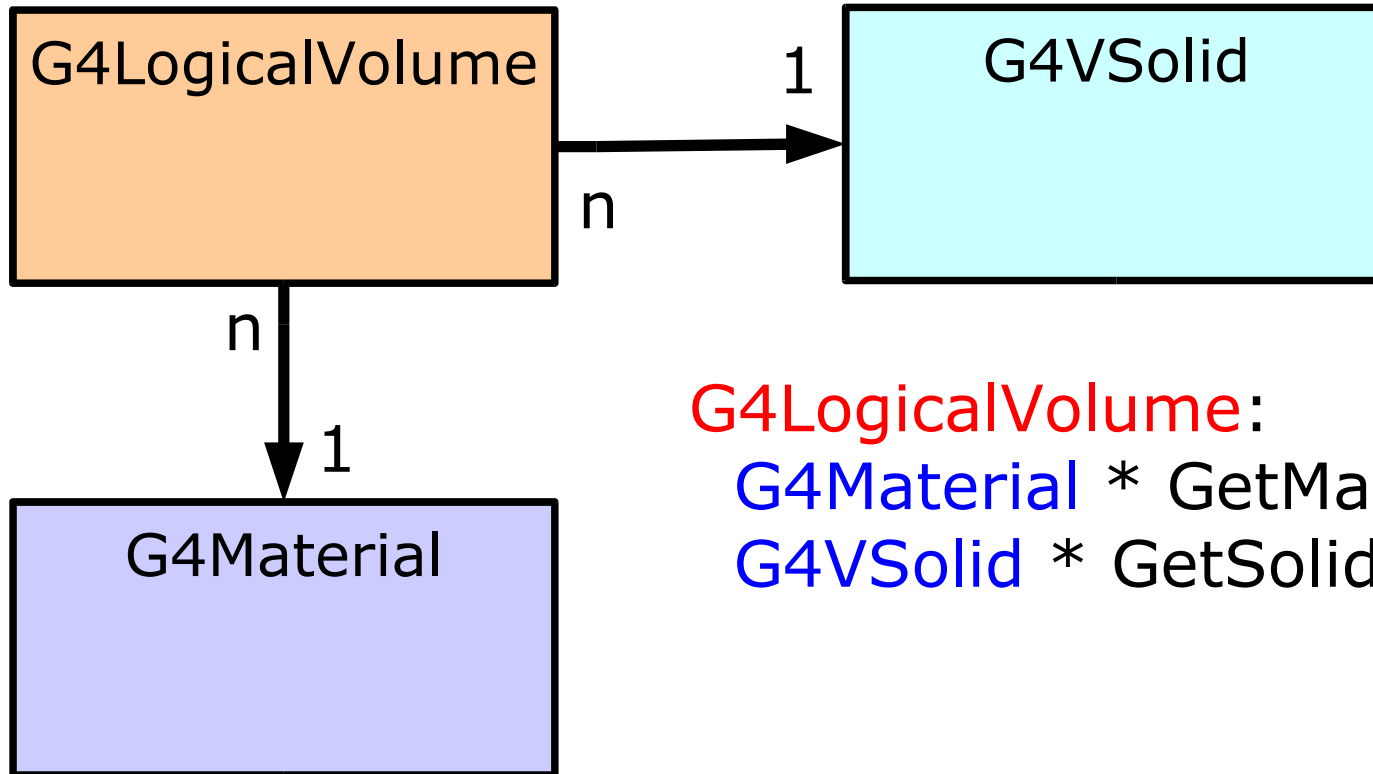
A logical volume knows its solid and material.

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!

Logical Volume



G4LogicalVolume:

`G4Material * GetMaterial() const;`

`G4VSolid * GetSolid() const;`

but **not**:

G4Material:

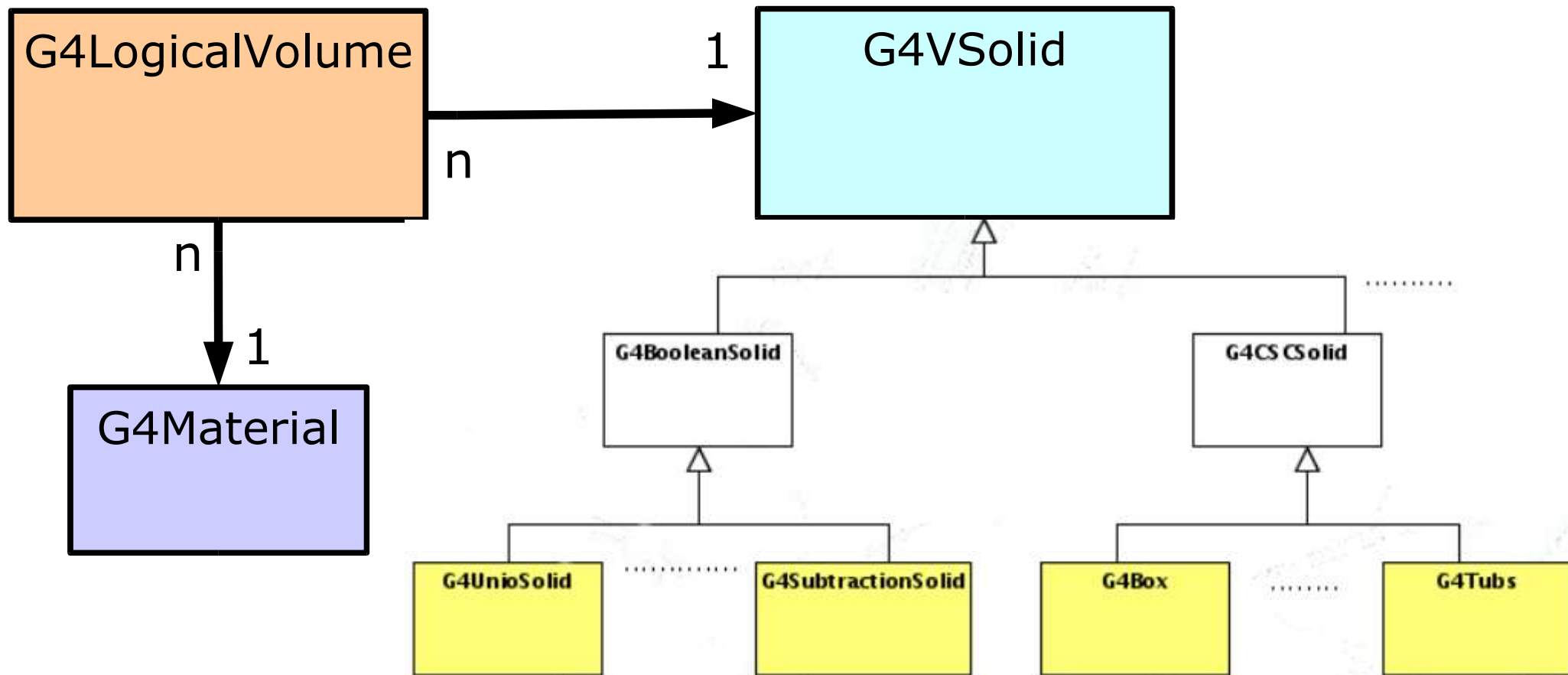
`vector<G4LogicalVolume*> GetVolumes()`

G4VSolid:

`vector<G4LogicalVolume*> GetVolumes()`

Remember:
G4Material – we know
already about it!

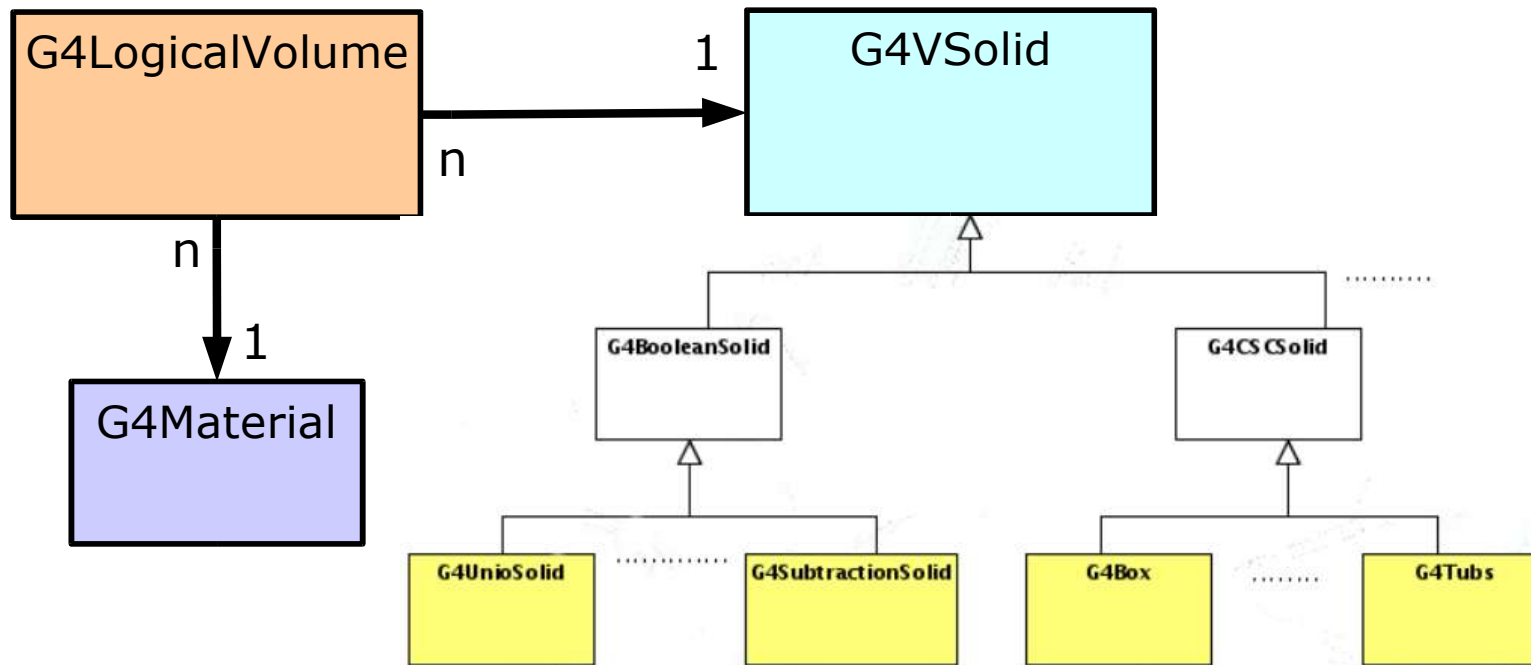
Solids



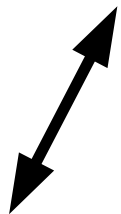
GEANT4 comes with a wide variety of solids

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

Solids



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

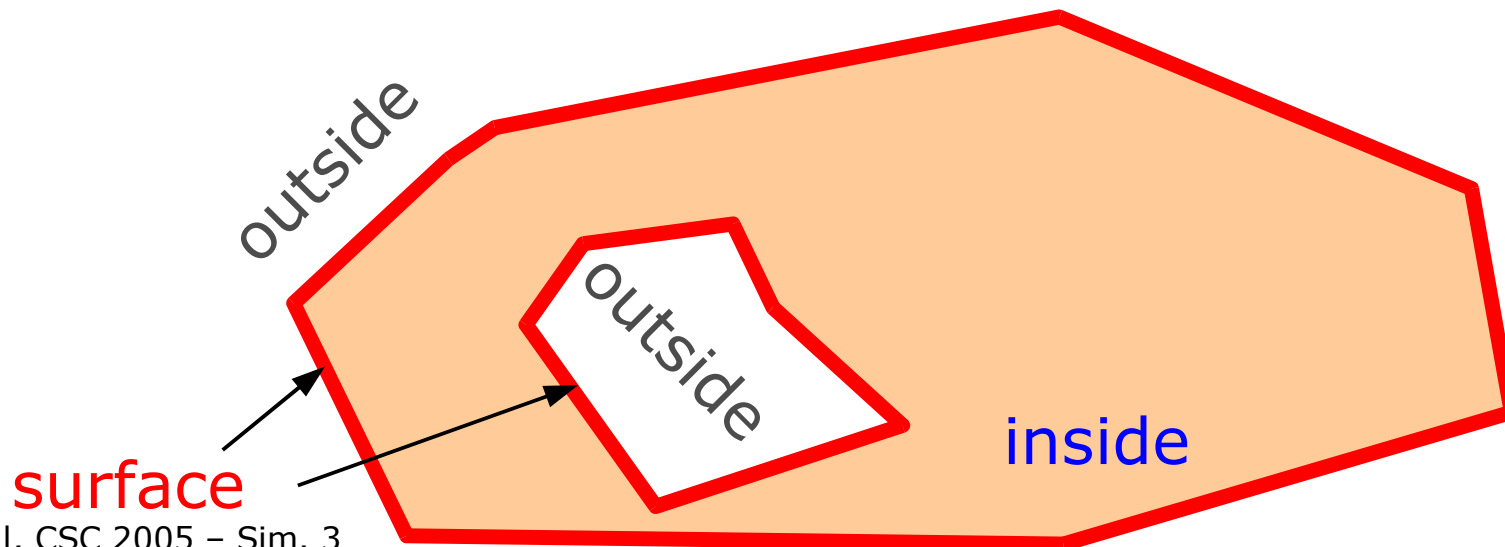


GEANT4 comes with a wide variety of solids

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

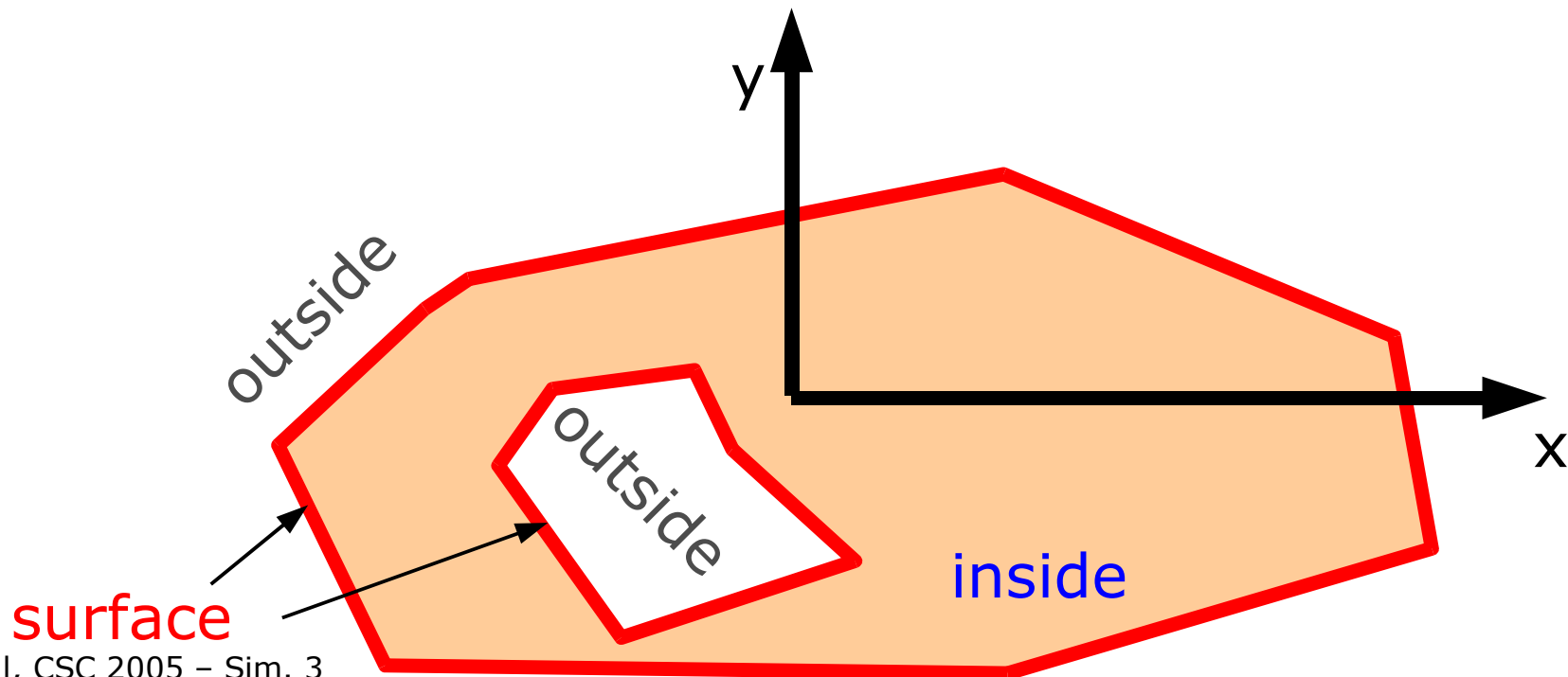
Properties of a G4VSolid

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

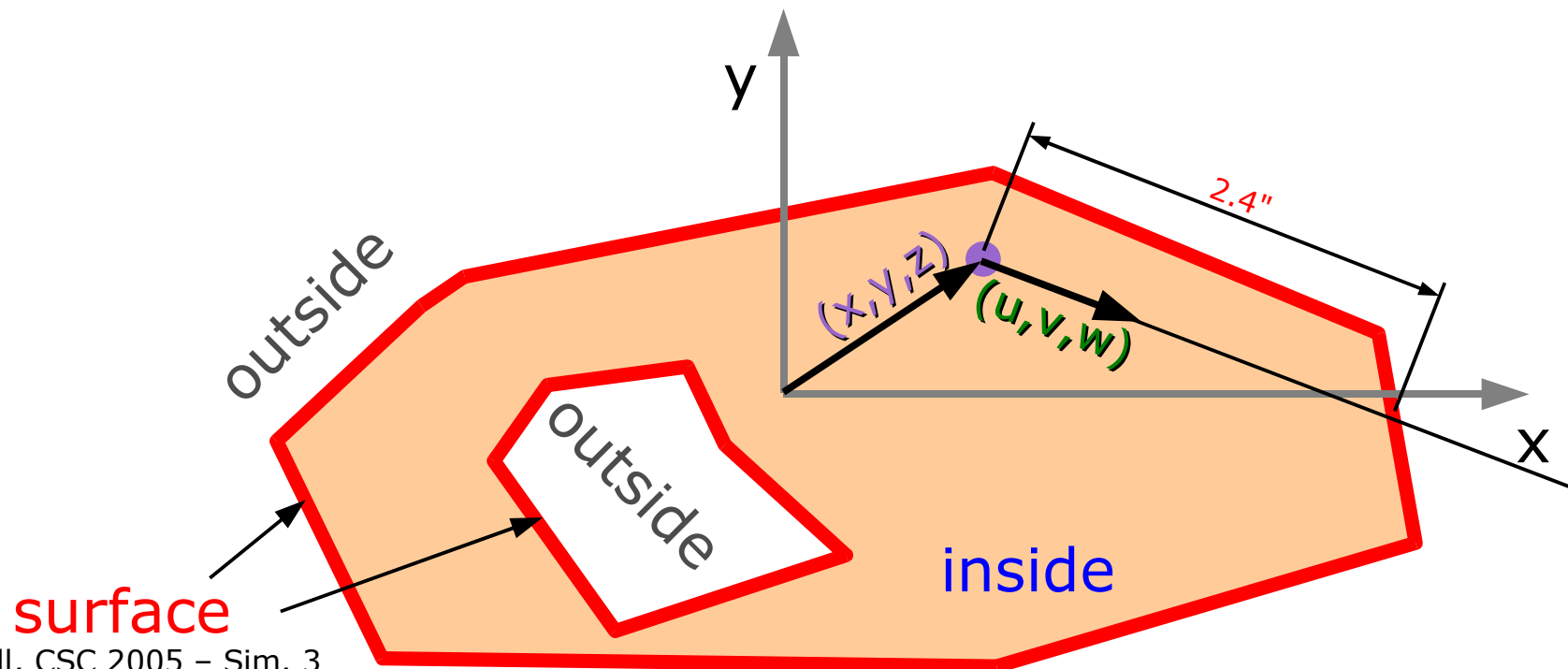


Properties of a G4VSolid

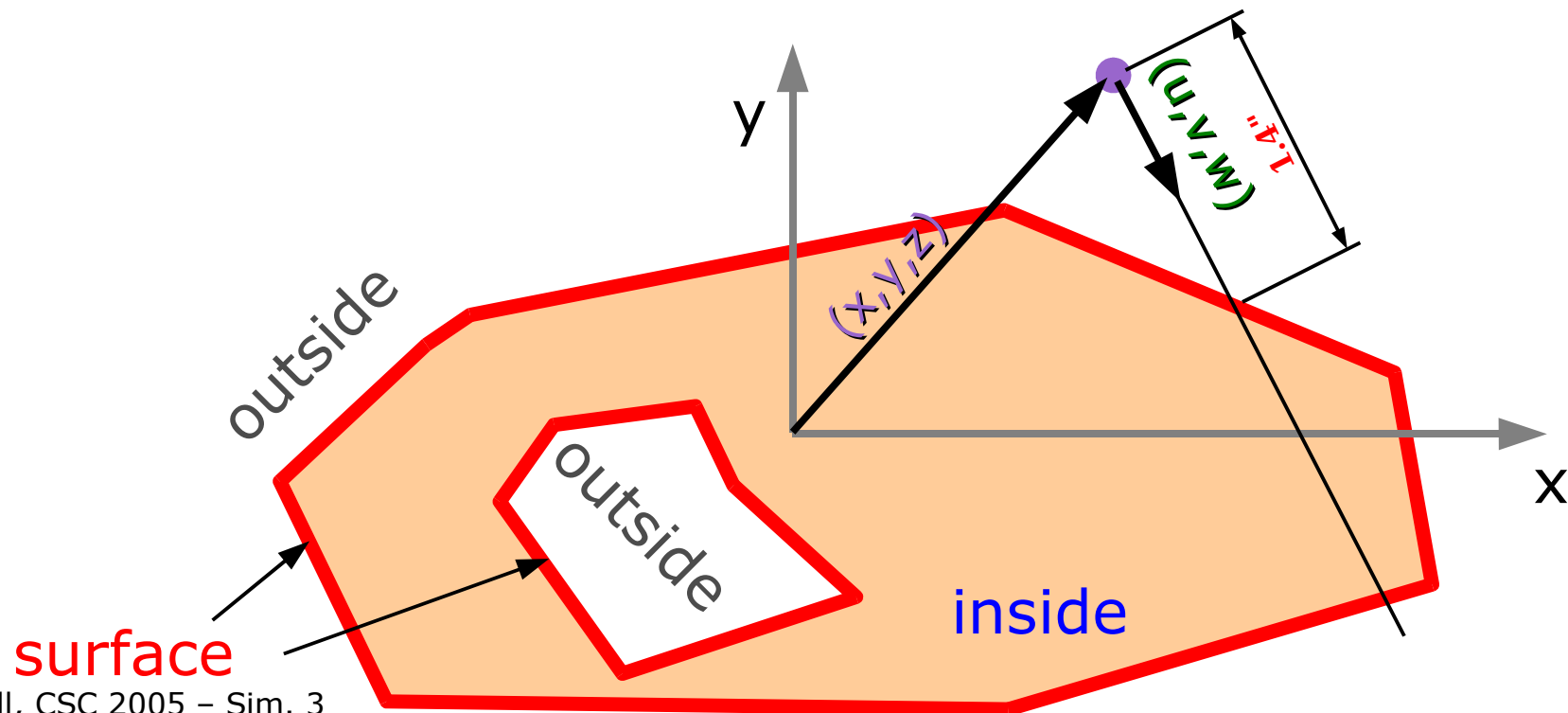
- A G4VSolid has
 - a well defined inside, outside, and boundary/surface within certain numerical tolerances
 - has a cartesian system of reference



- A G4VSolid has
 - a well defined inside, outside, and boundary/surface
 - has a cartesian system of reference
 - supports a set of geometrical calculations w.r.t. its reference system, e.g.
 - Is point (x,y,z) inside, outside, or on surface?
 - Being at point (x,y,z) and looking into direction (u,v,w) , what is the **distance** to the surface?



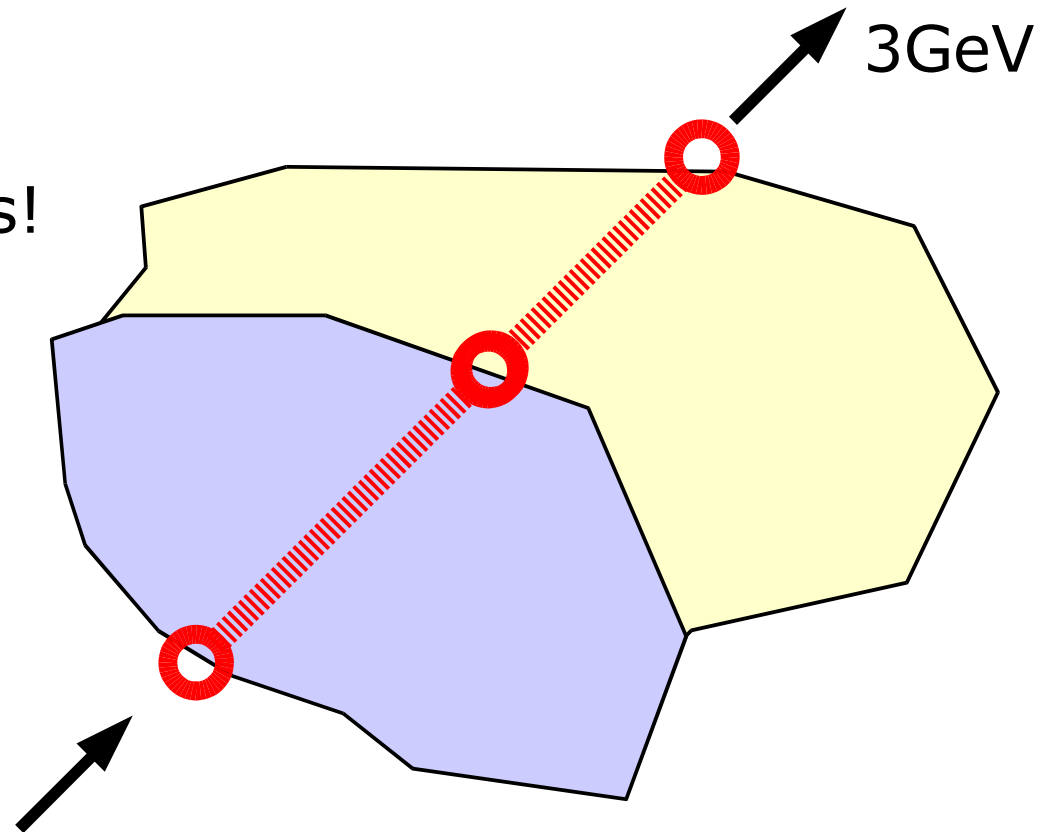
- A G4VSolid has
 - a well defined inside, outside, and boundary/surface
 - has a cartesian system of reference
 - supports a set of geometrical calculations w.r.t. its reference system, e.g.
 - Is point (x,y,z) inside, outside, or on surface?
 - Being at point (x,y,z) and looking into direction (u,v,w) , what is the **distance** to the surface?



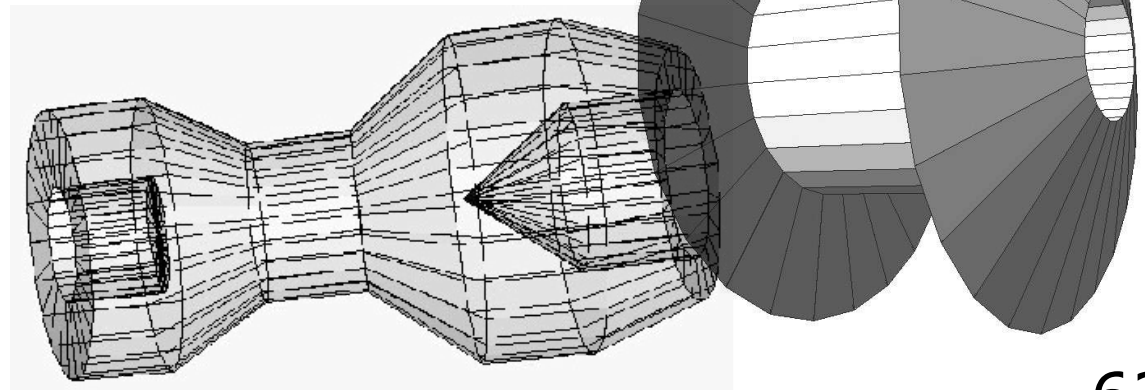
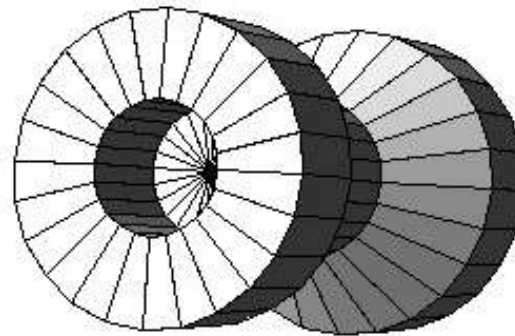
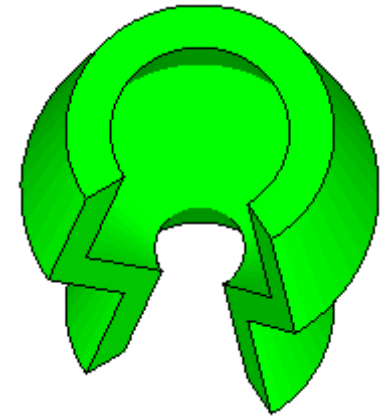
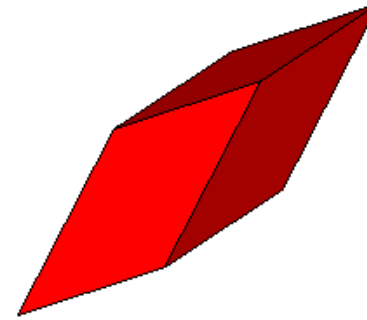
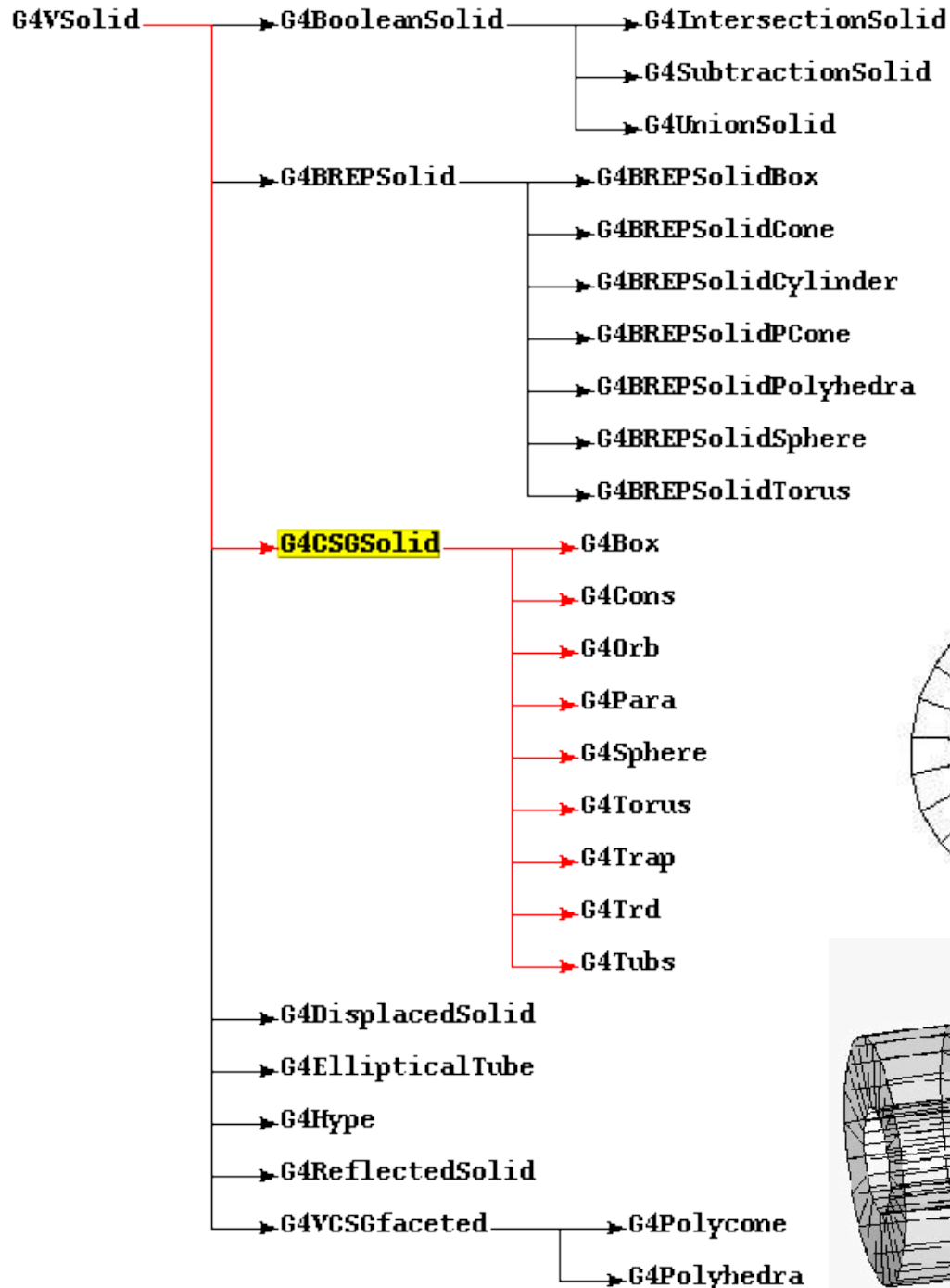
Remember?

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

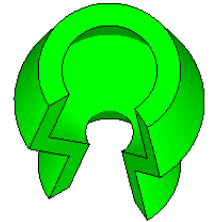
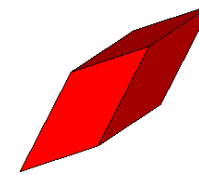
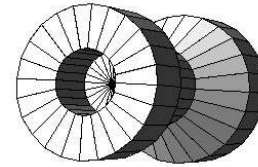
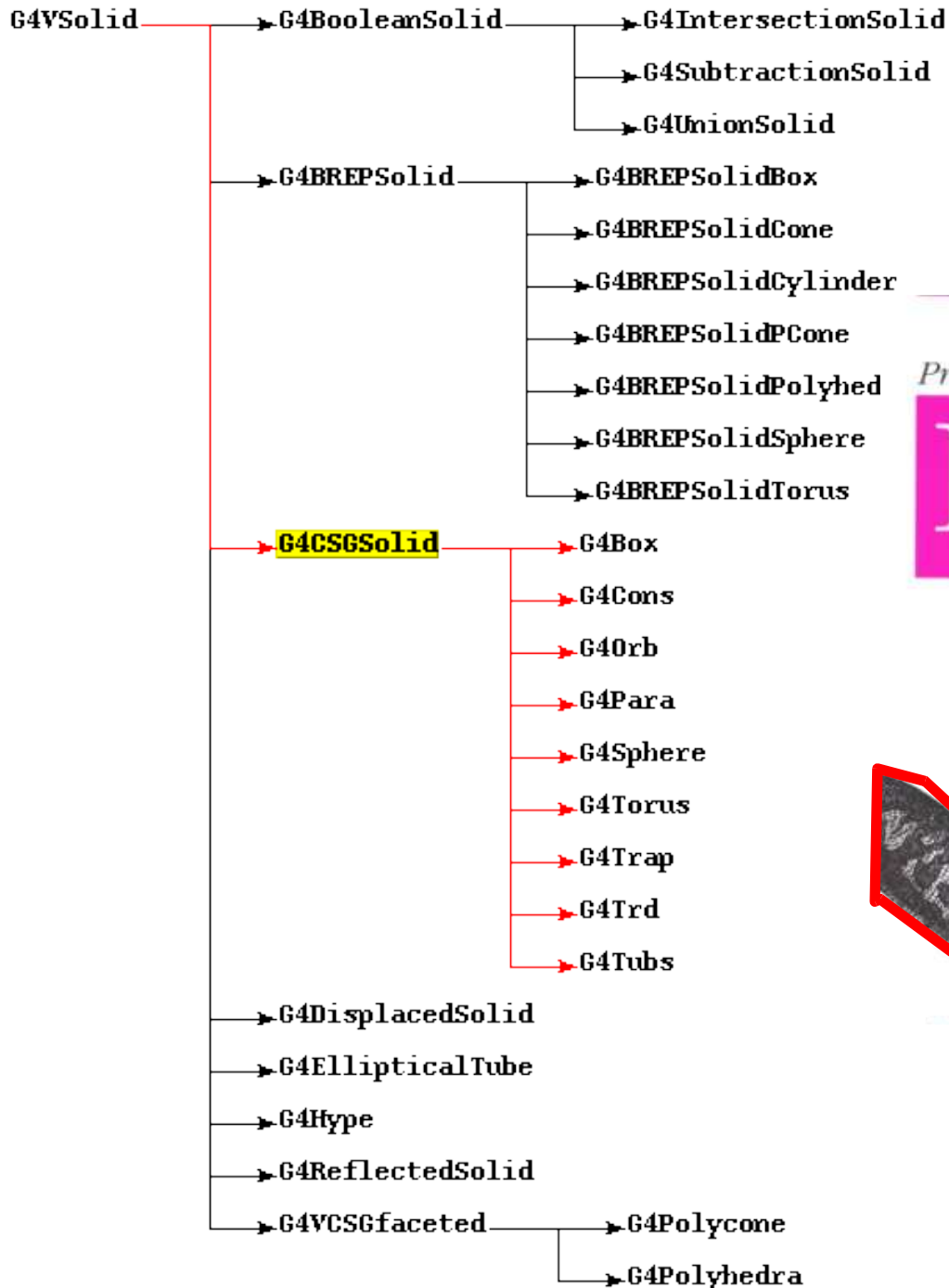
The solid of the volume is asked for distance information towards its boundary!



“Batteries included”



“Batteries included”



Implement your own
Python solid!

```
class G4Python
: public G4VSolid
```

Recap:

What we have up to now:

- Material
 - simple, composites
- Solid
 - frame of reference, inside, outside, surface
- Logical Volume
 - points to a mandatory solid and a mandatory material

Example:

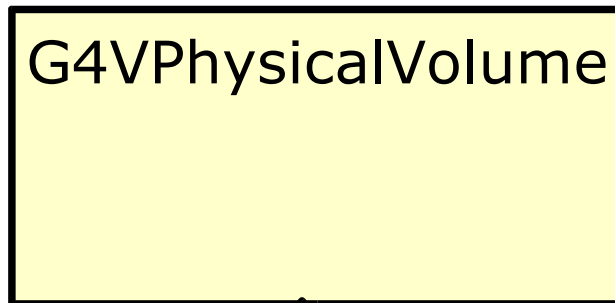


? How can we describe a ?
complex detector
using these ingredients?

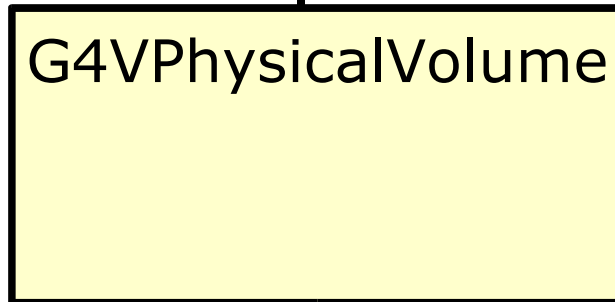
Volume Hierarchies

- 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
 - parameterized multiple placement

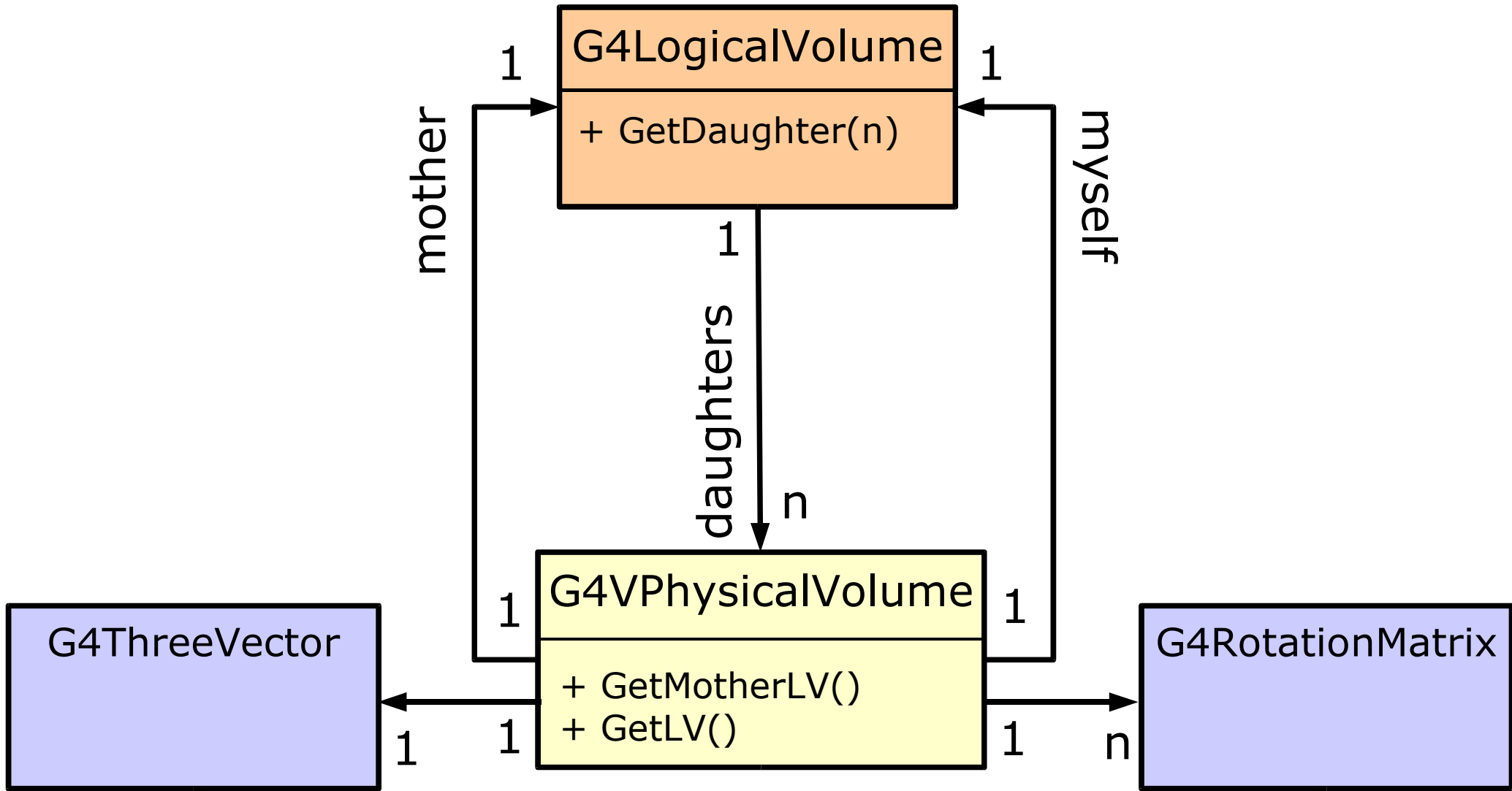
G4VPhysicalVolume

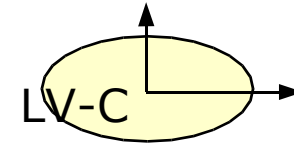
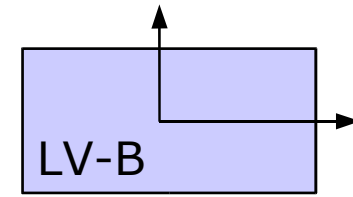
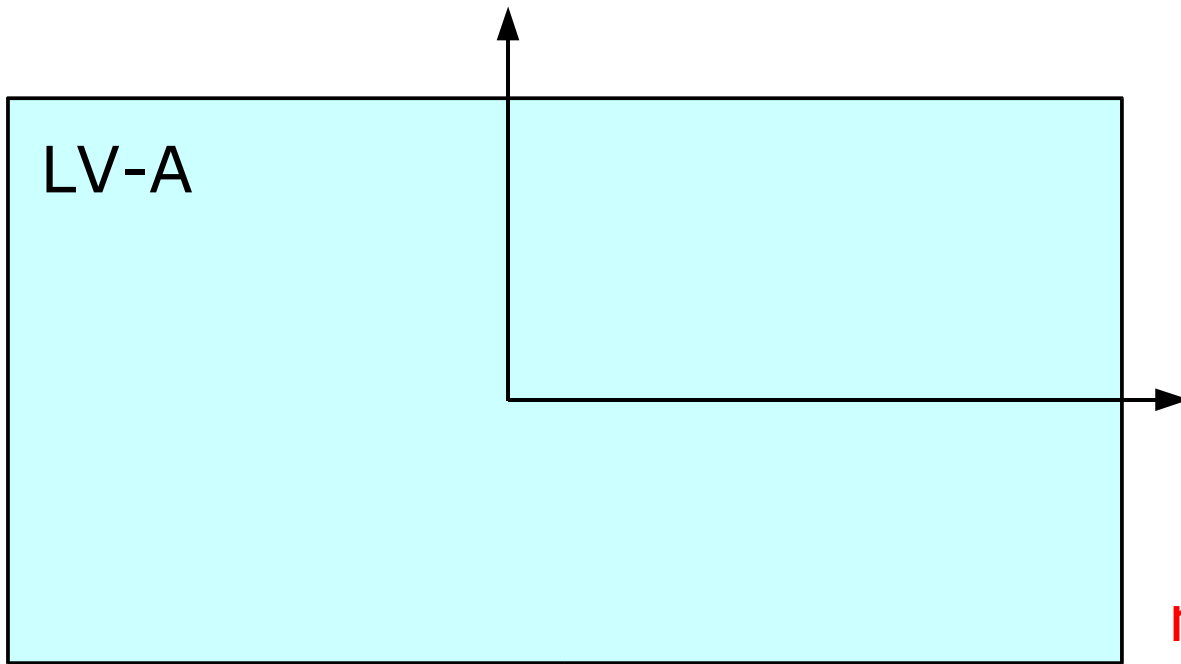


abstract base

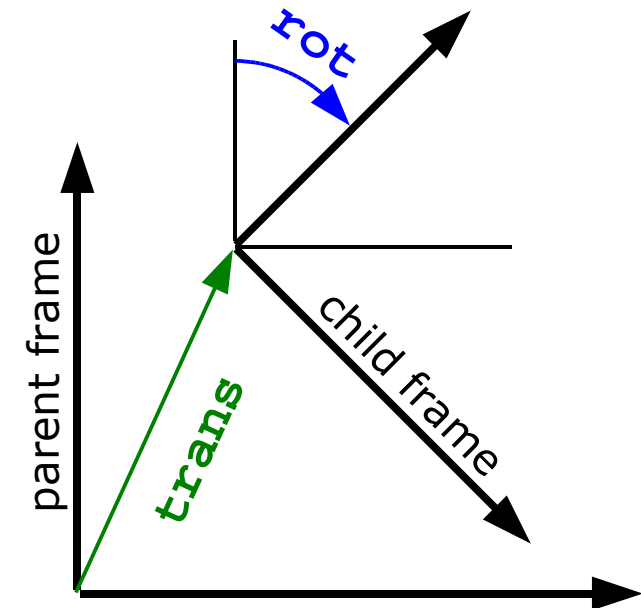
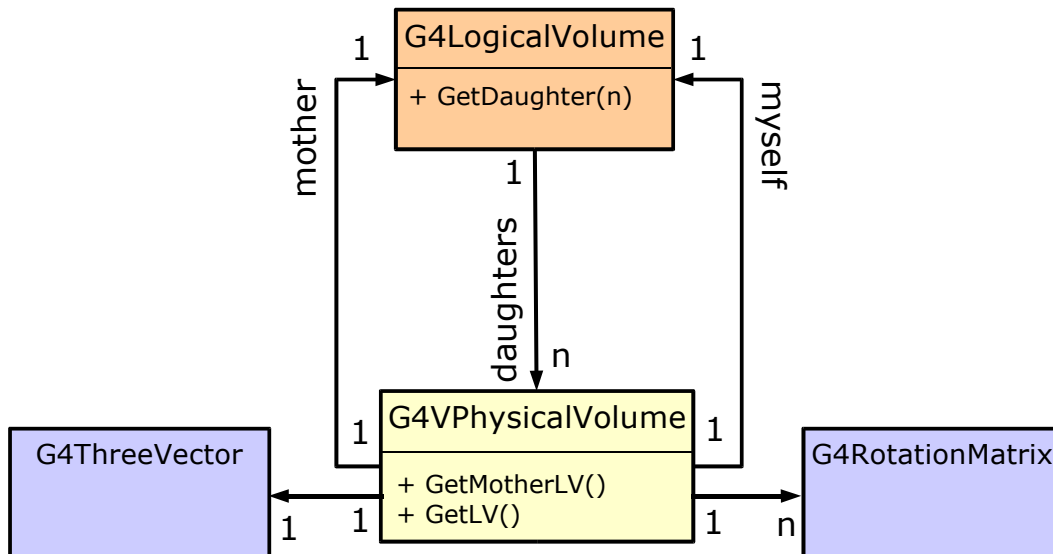


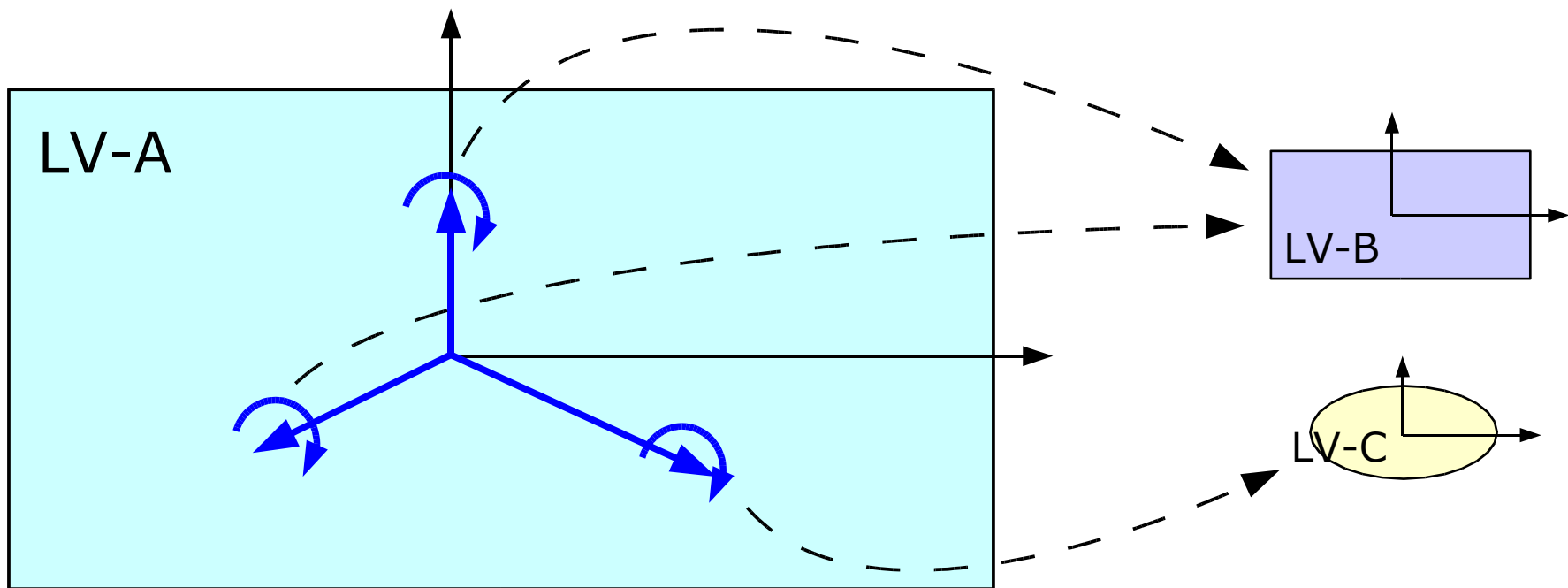
concrete implementation -
implements a single
parent-child relation of
two volumes.



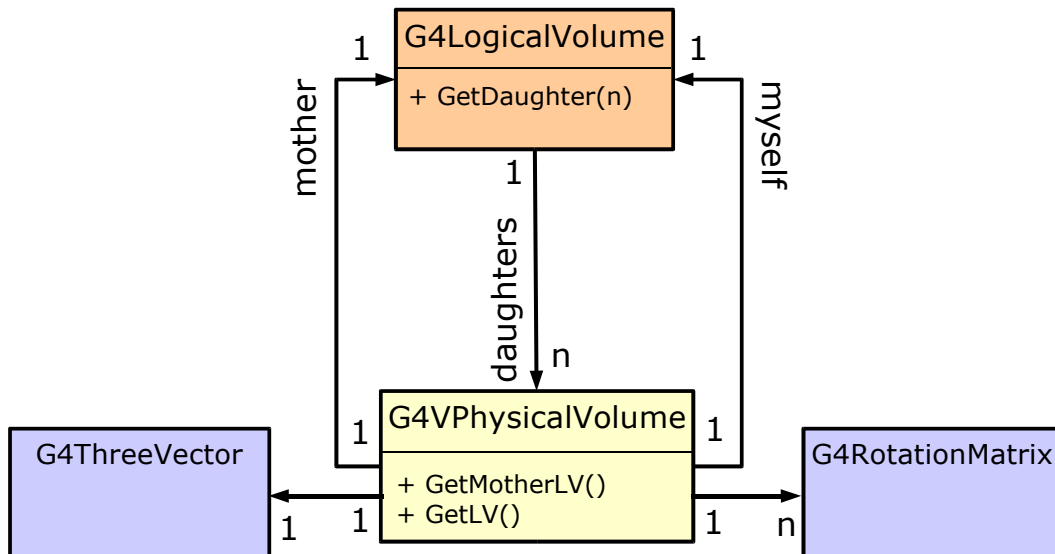


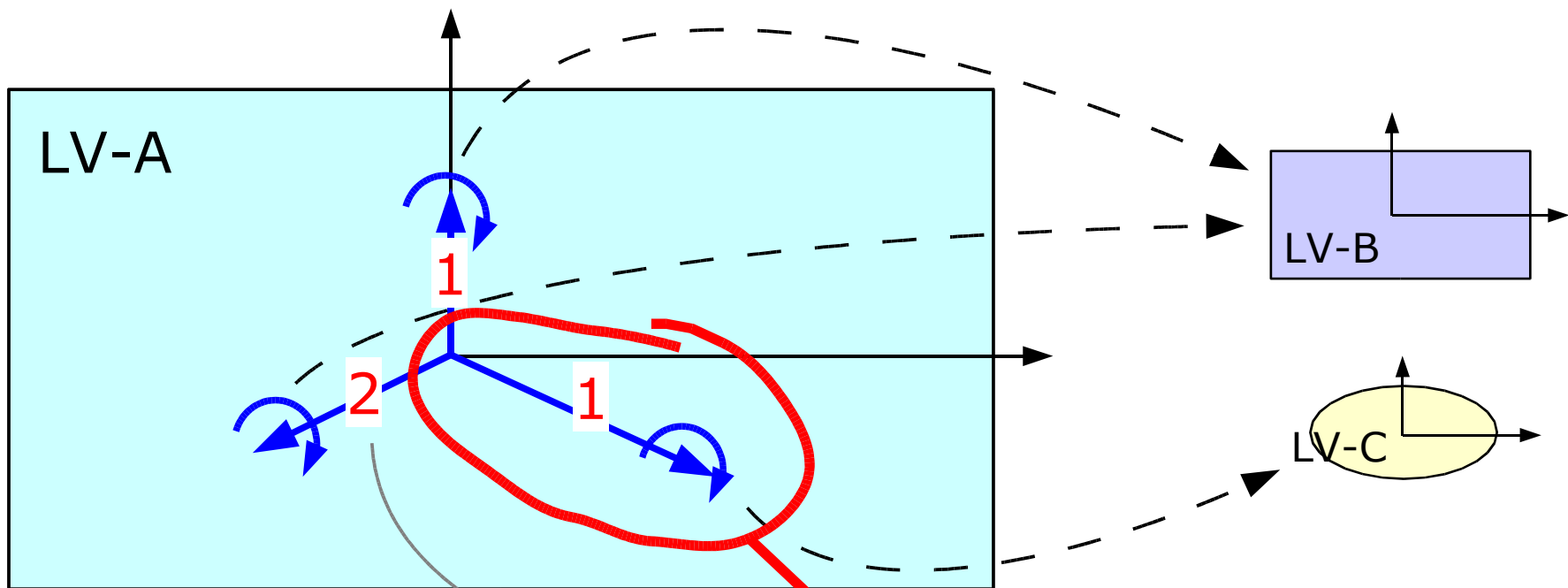
reference frames of solids used for relative positioning:





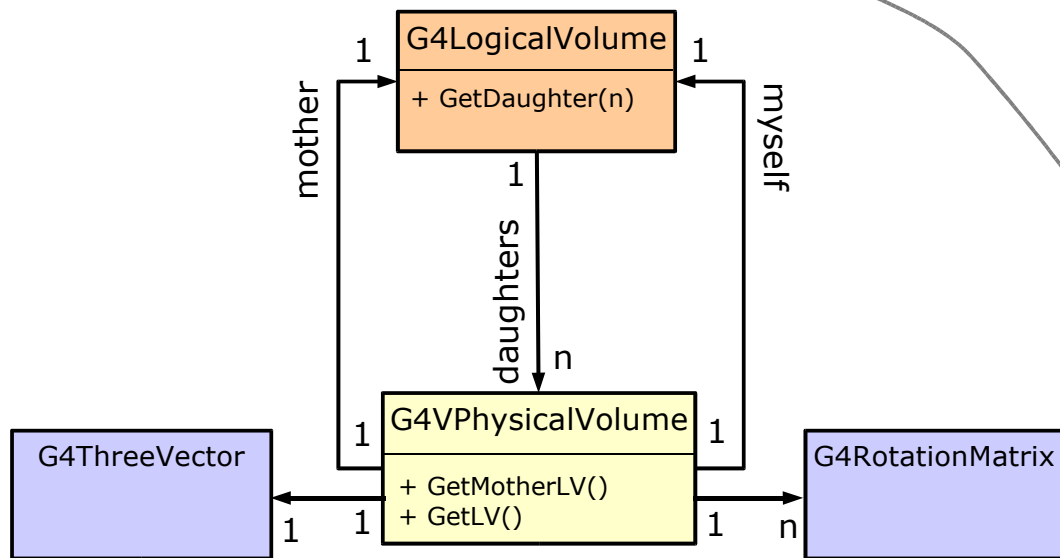
creation 3 instances
of **G4VPhysicalVolume**
in LV-A



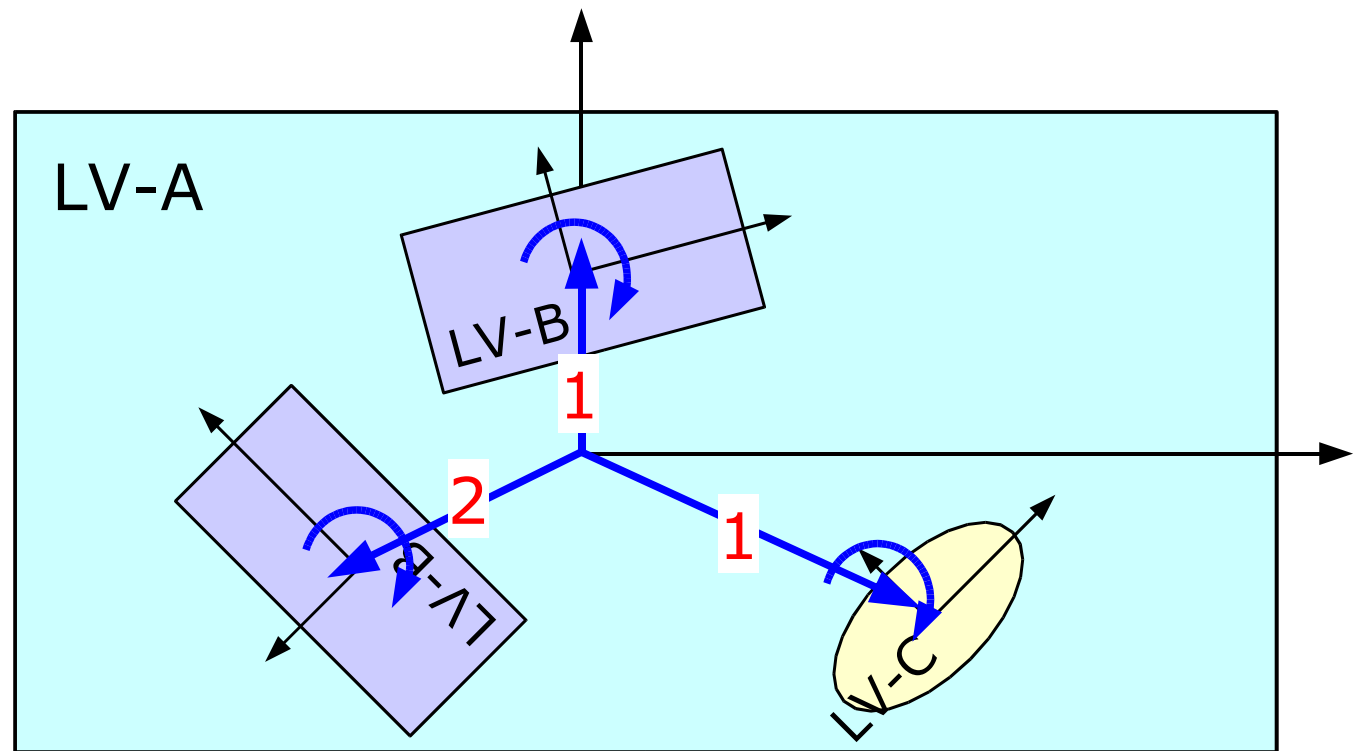


symbolically:
 one instance of a
G4VPhysicalVolume

The numbers are user defined "copy-numbers"; they help us to easier distinguish the copies



corresponds to:

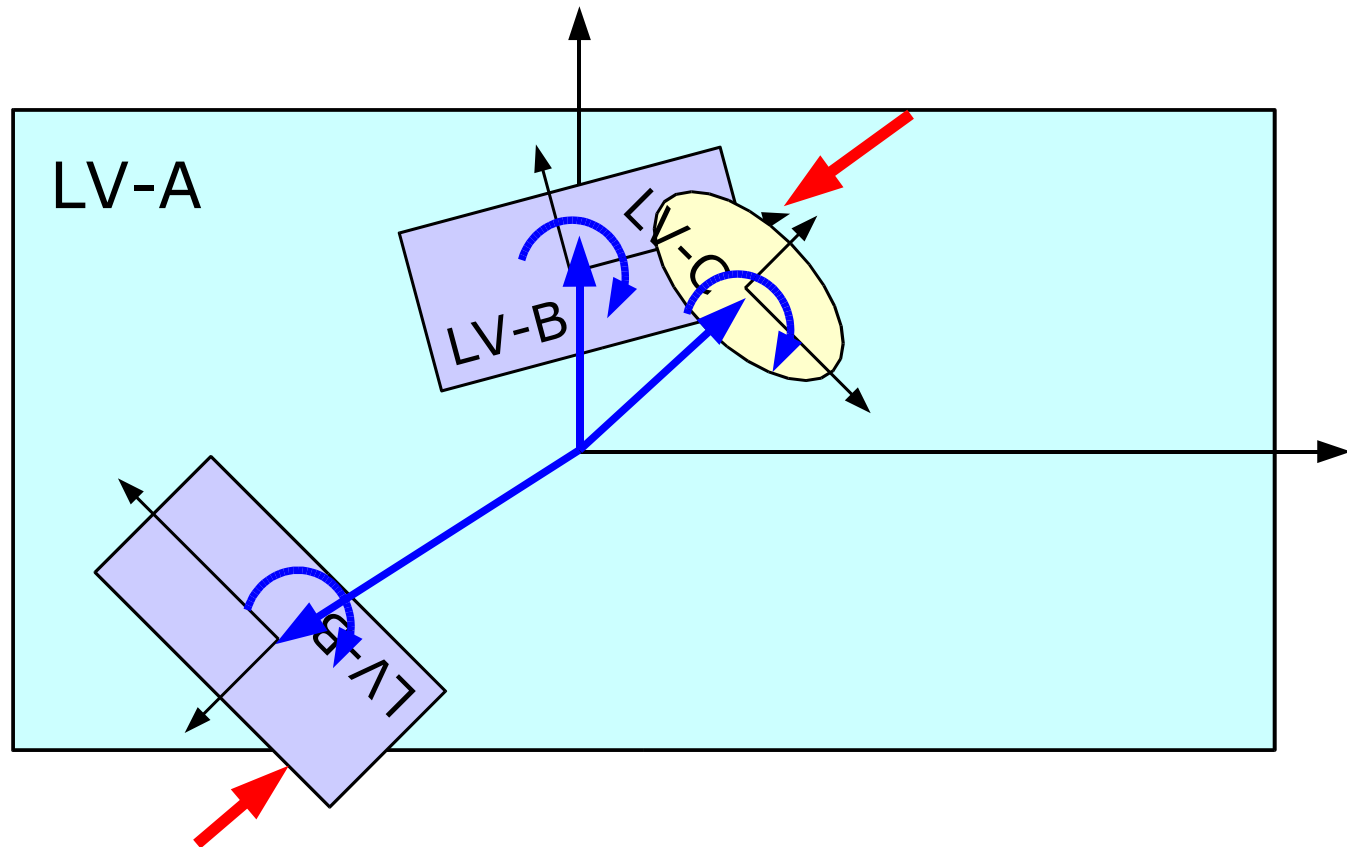


Constraints demanded by GEANT4:

- 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!!

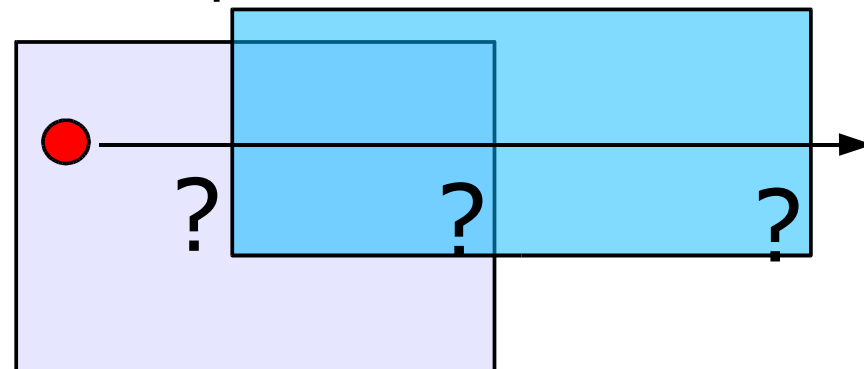
forbidden!!!
forbidden!!!
forbidden!!!



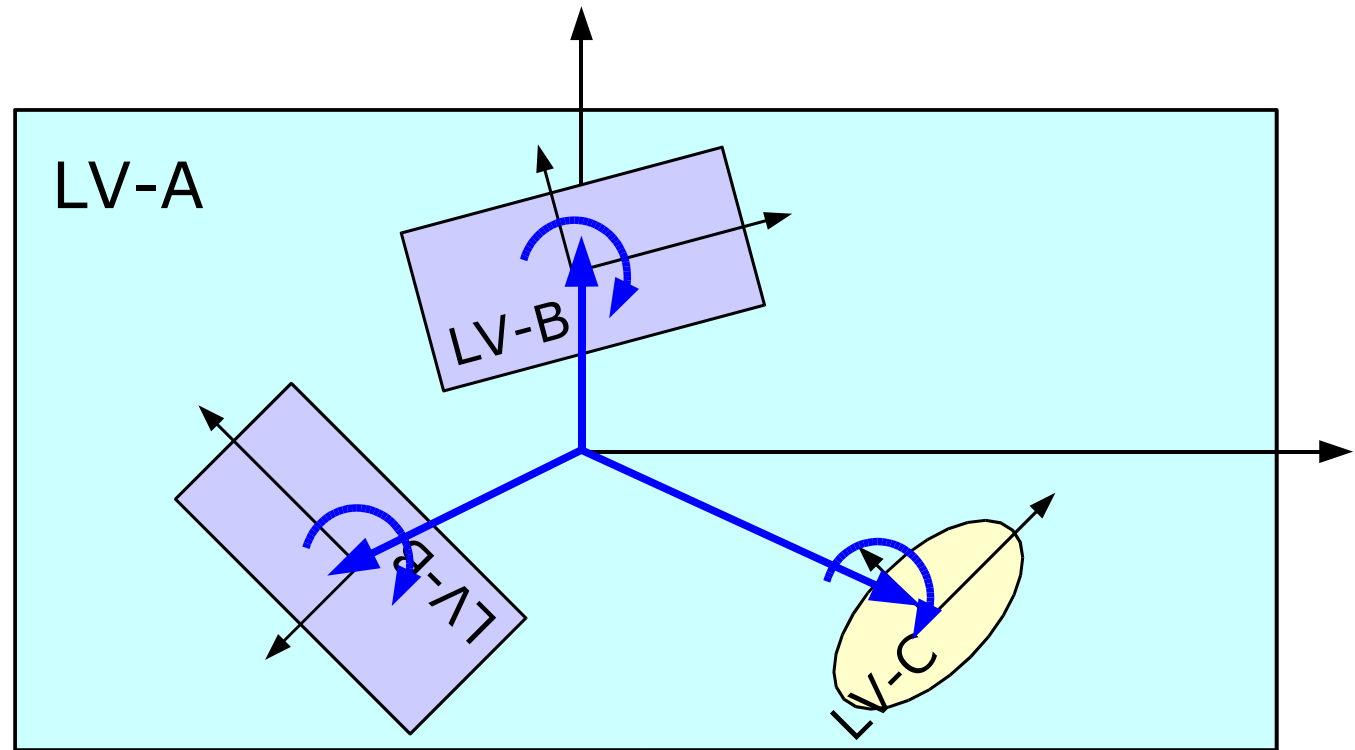
Constraints demanded by GEANT4:

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

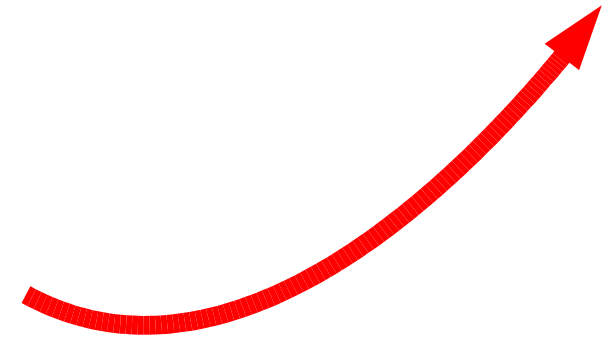
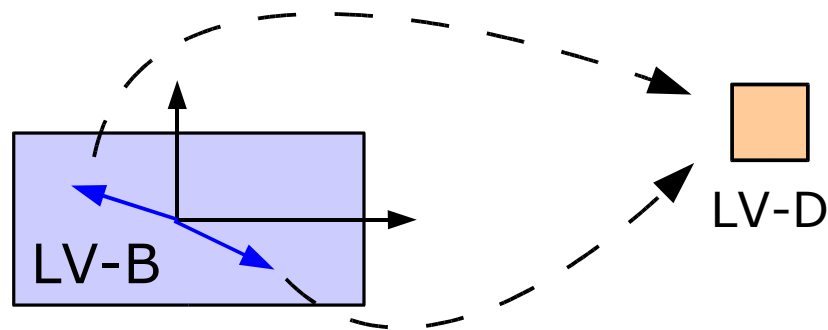
Overlaps lead to ambiguities!
Where should we have the
StepPoints on the boundaries?



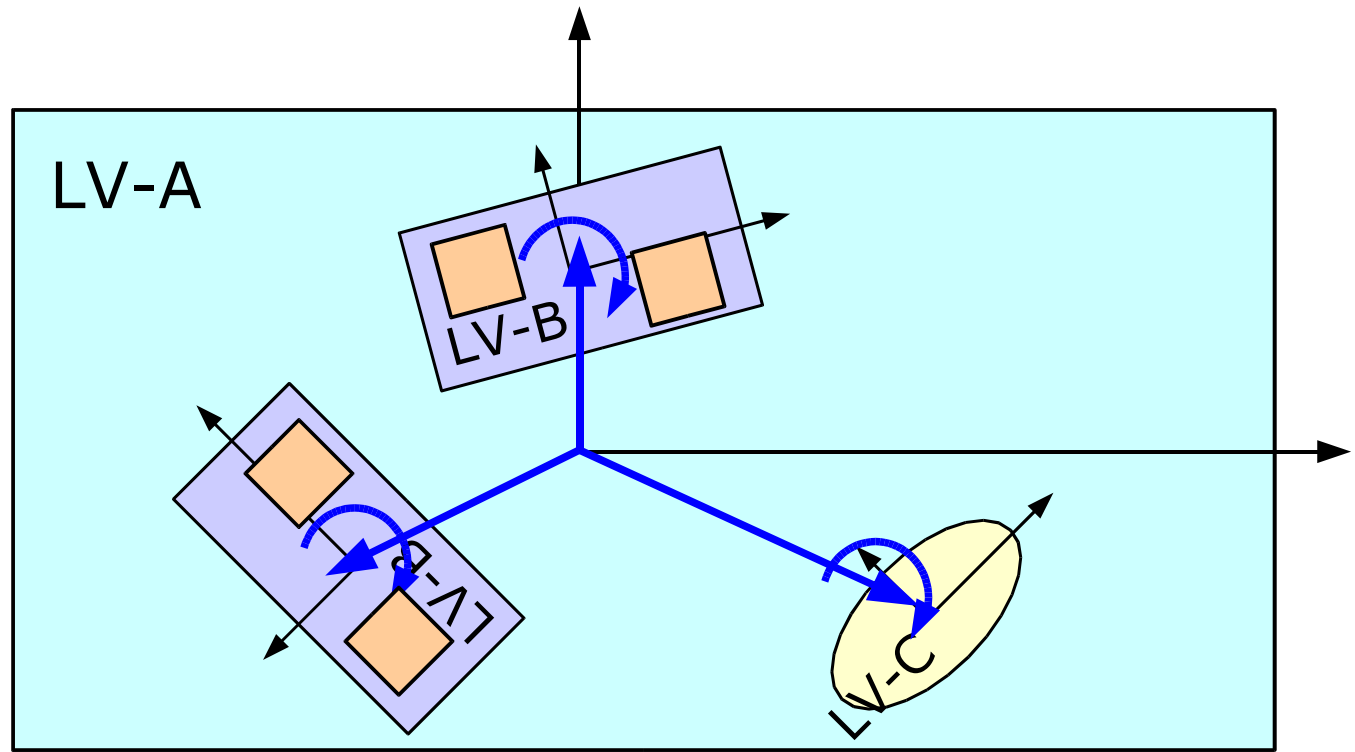
Deeper hierarchies!



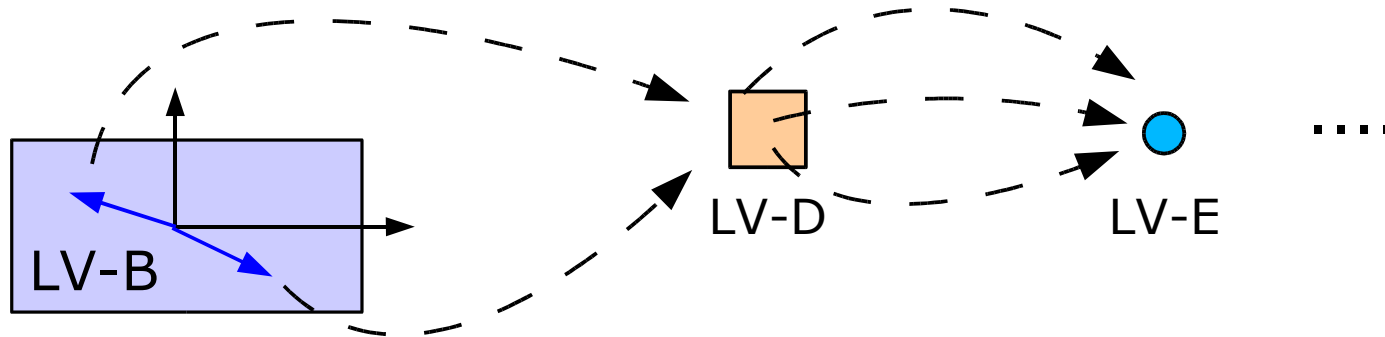
We can apply the positioning procedure recursively!



corresponds to:



and so on and on ...



Implications

The hierarchy of volumes is a

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

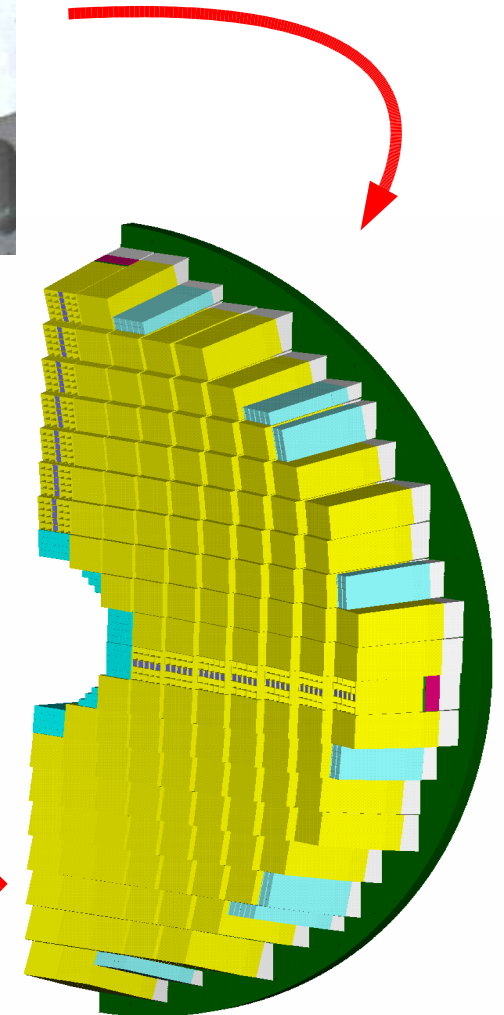
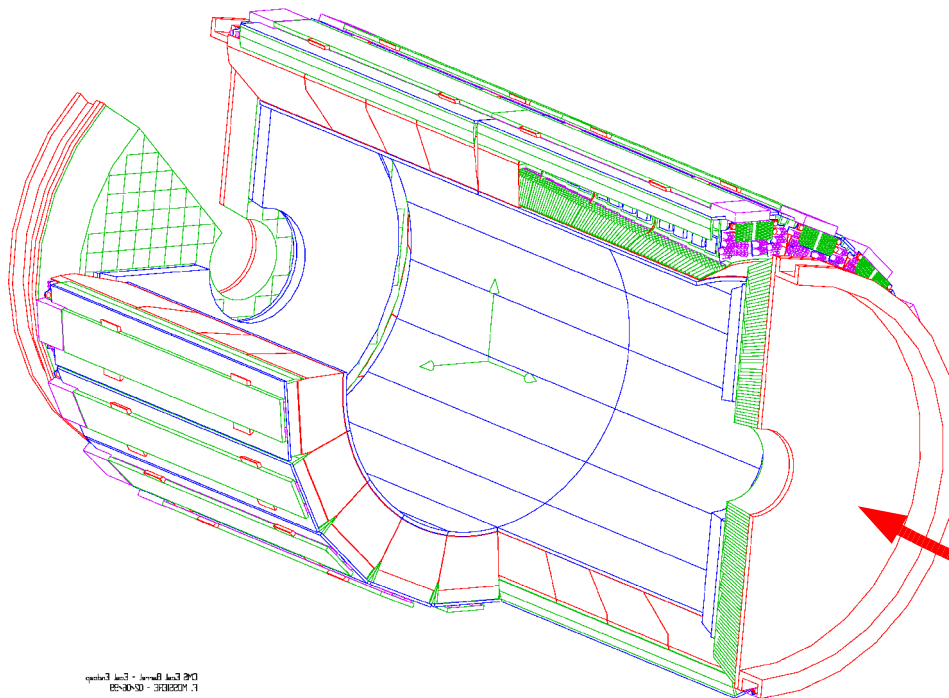
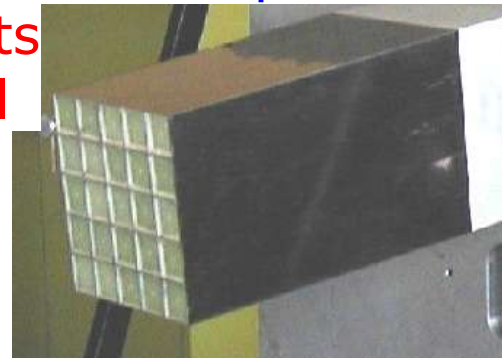
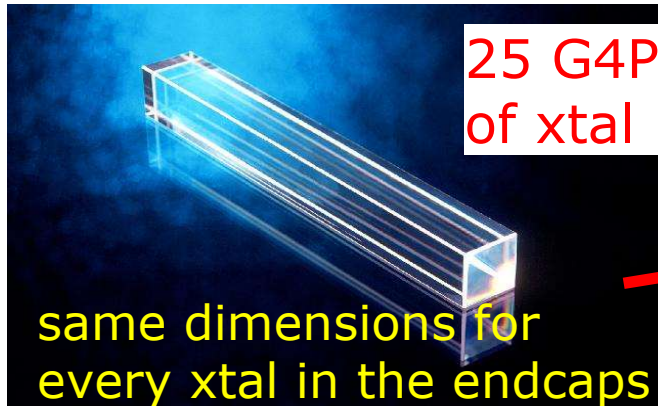
multi-graph.

- 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

1 G4LogicalVolume
for the super-xtal



© 2005 - 2006
L. Liendl - CSC

The example ..

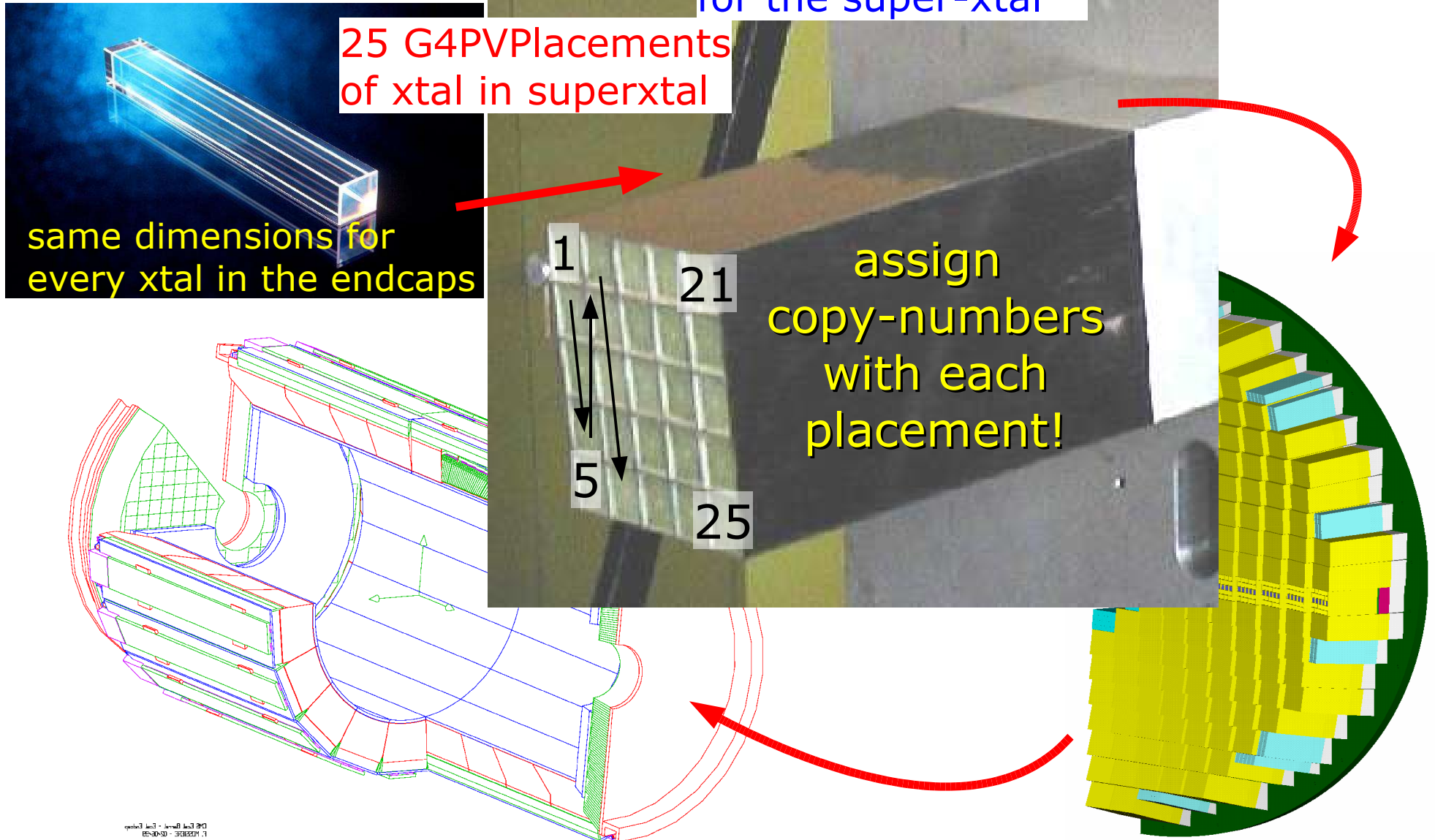
1 G4LogicalVolume
for the xtal

1 G4LogicalVolume
for the super-xtal

25 G4PVPlacements
of xtal in superxtal

same dimensions for
every xtal in the endcaps

assign
copy-numbers
with each
placement!

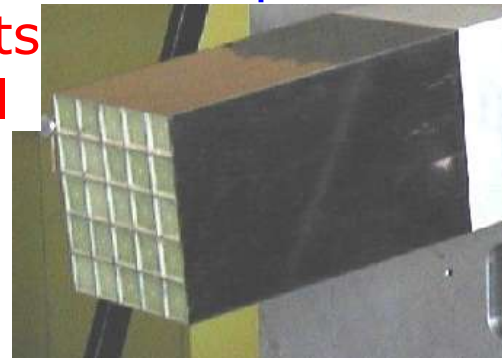
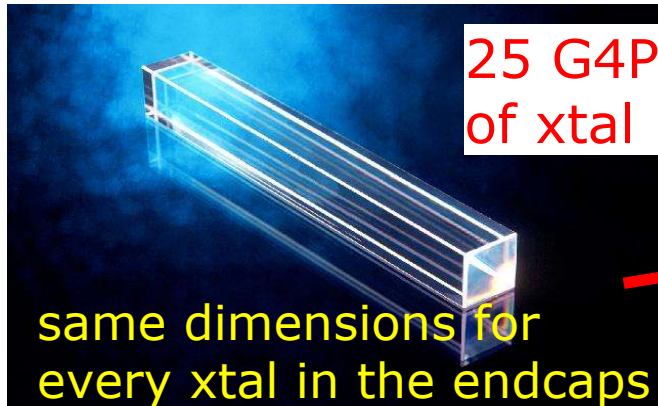


© 2005-2006, GE Healthcare
All rights reserved.

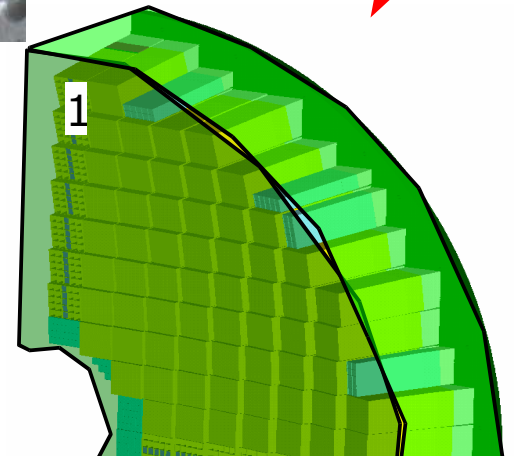
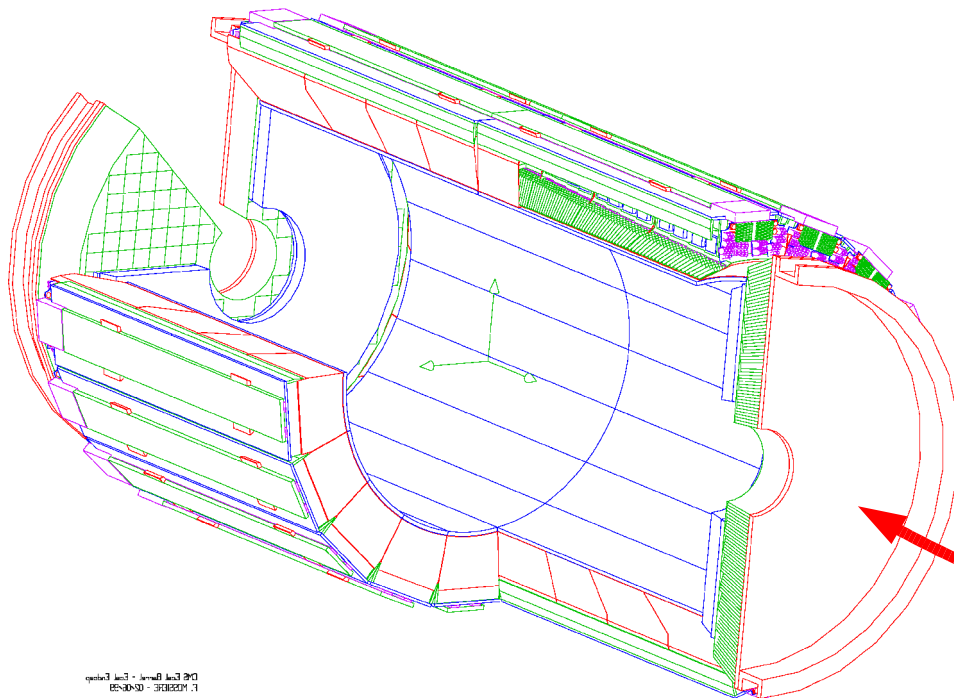
The example ..

1 G4LogicalVolume
for the xtal

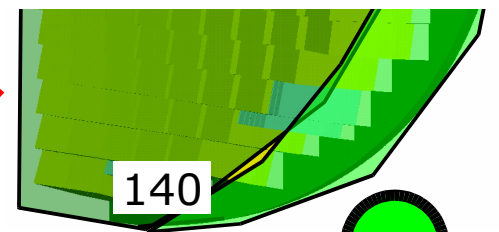
1 G4LogicalVolume
for the super-xtal



140 placements
of super-xtals in D



1 G4LogicalVolume
for the "D"

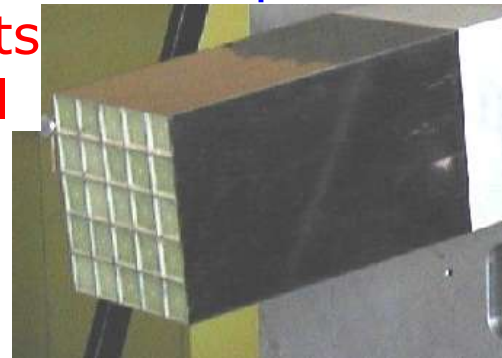
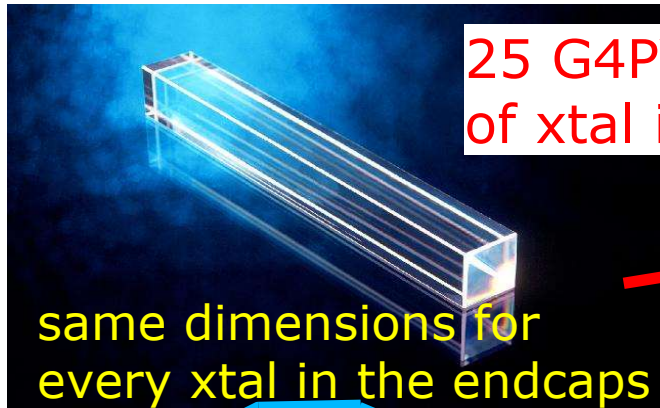


© 2005 GEANT4 Collaboration

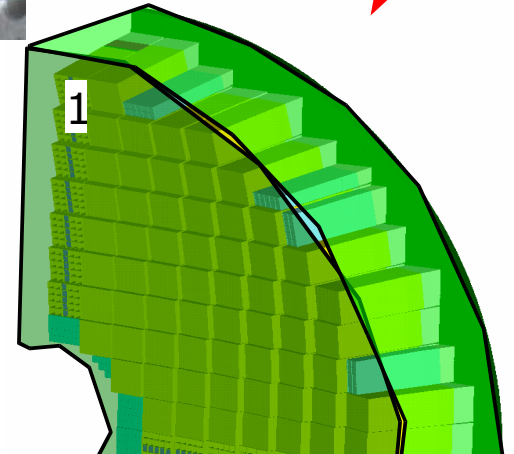
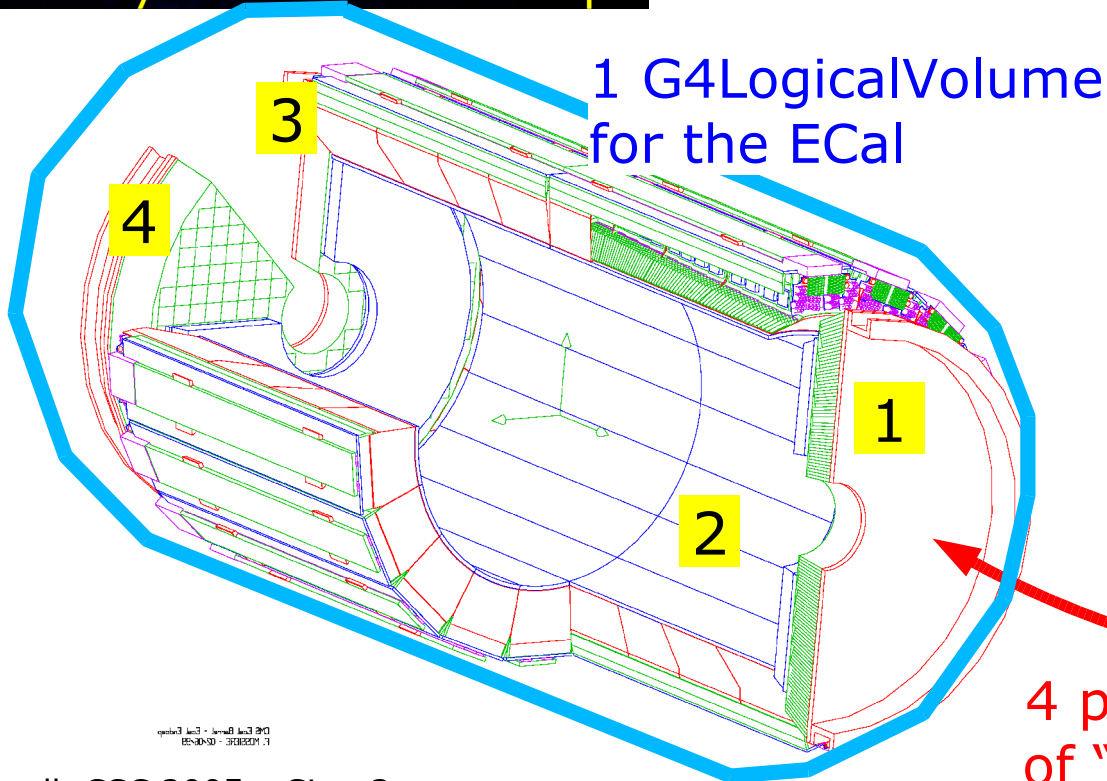
The example ..

1 G4LogicalVolume
for the xtal

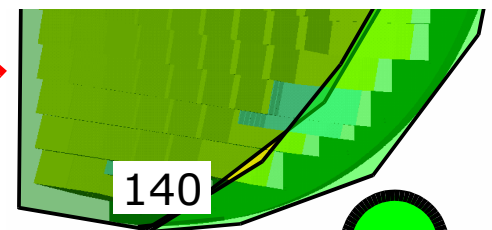
1 G4LogicalVolume
for the super-xtal



140 placements
of super-xtals in D



1 G4LogicalVolume
for the "D"



4 placements
of "D" in ECal

The example ..

1 G4LogicalVolume
for the xtal

1 G4LogicalVolume
for the super-xtal

Of super-xtal
140 D
placements

G4LogicalVolumes

G4PVPlacements

1 xtal

1 super-xtal

1 "D"

1 ECal

25 xtal -> super-xtal

140 superxtal -> "D"

4 "D" -> ECal

4
=====

171
=====

**Need only 4 + 171 instances (+ max. 171 rot, trans)
to represent ~14.000 crystals**

2

for the "D"

4 placements
of "D" in ECal

140

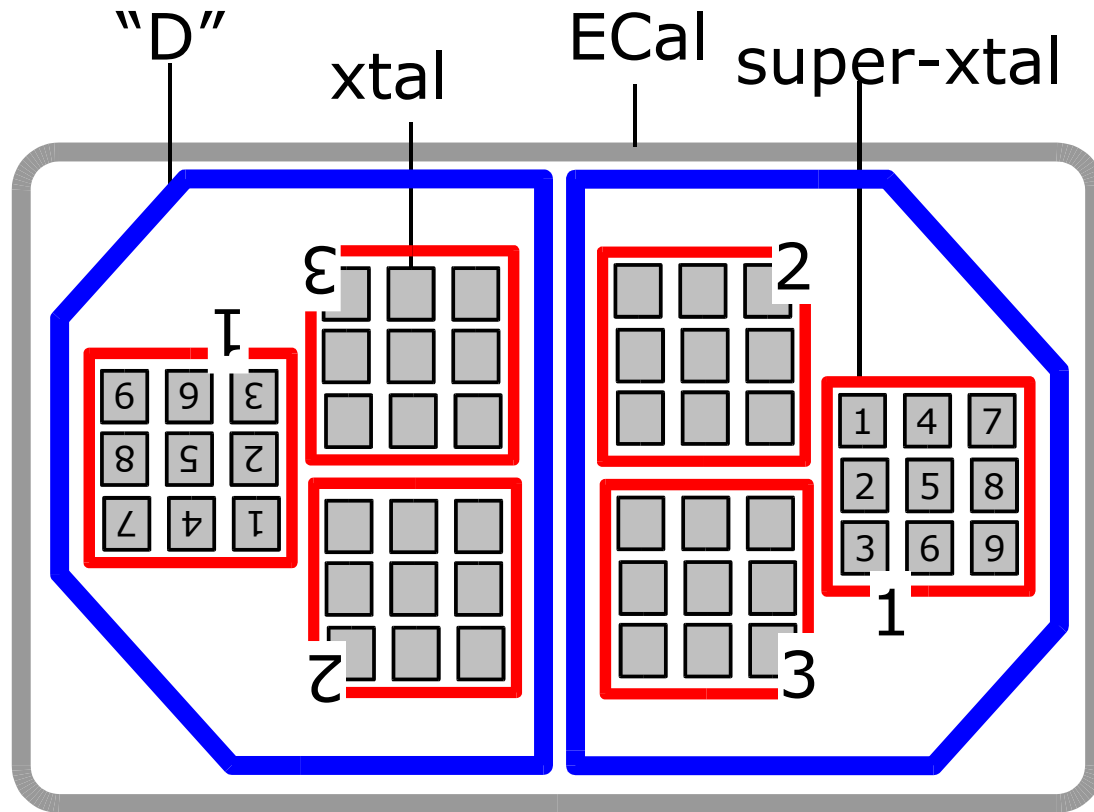
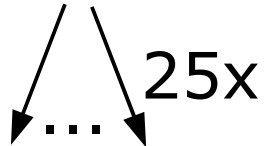
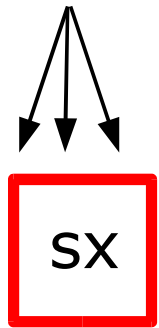
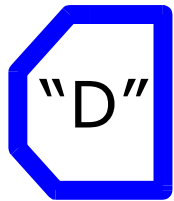
Copyright 2005 - GEANT4 Collaboration

Graph Structure

logical volume



physical volume



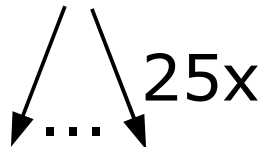
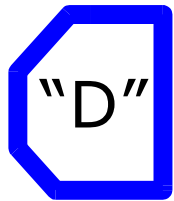
simplified version of an ECal ...

Graph Structure

logical volume

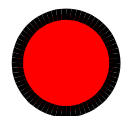


physical volume



Remark:

Instead of using instances of **G4PVPlacements** to represent a physical volume, Geant4 allows to use **parameterizations** (~user supplied formulas) that calculate the physical volumes (relative rotations & translations) at runtime whenever a particles are tracked through these volumes!!



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