

PART III

Detector Description: Geometrical Hierarchies

Tracking of interactionless Particles

Parents, Children, Siblings:

We are Family / Hierarchy!

- A logical volume can have children logical volumes
- The position of a child within its parent is defined by
 - a translation vector and a rotation matrix
 - specifying the 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
- Geant4 offers 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

Parents, Children, Siblings: We are Family / Hierarchy!

- A logical volume can have children logical volumes
- The concepts of Geant4 geometrical hierarchies will be explained by using single placements. The other methods are refinements thereof and don't change the behavior!
- There is a parent's solid relationship
- Geant4 parent-child relationship:
 - **single placement of a child in a parent**
 - dividing the parent into several children
 - parameterized multiple placement

Parents, Children, Siblings: We are Family / Hierarchy!

Remarks on Geant4 nomenclature:

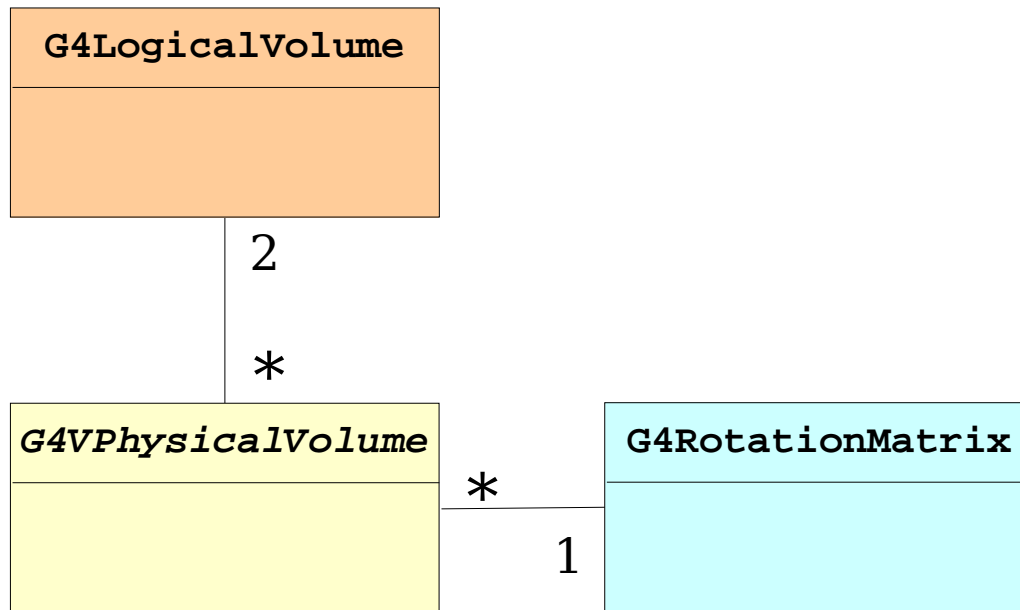
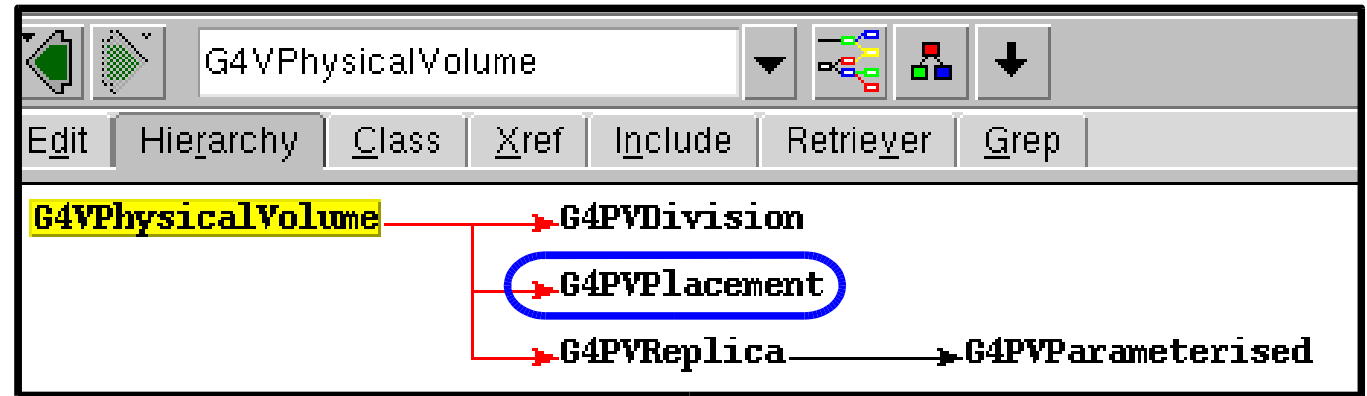
- Parents are always **mothers**
- Children are always **daughters**
- One mother – daughter relationship (including the placement information) is called a **Physical Volume**

Remark on the last remark:

I find the term “Physical Volume” misleading. As we will see later, a “Physical Volume” does not necessarily have a 1:1 correspondence with a physical volume in the real detector ...

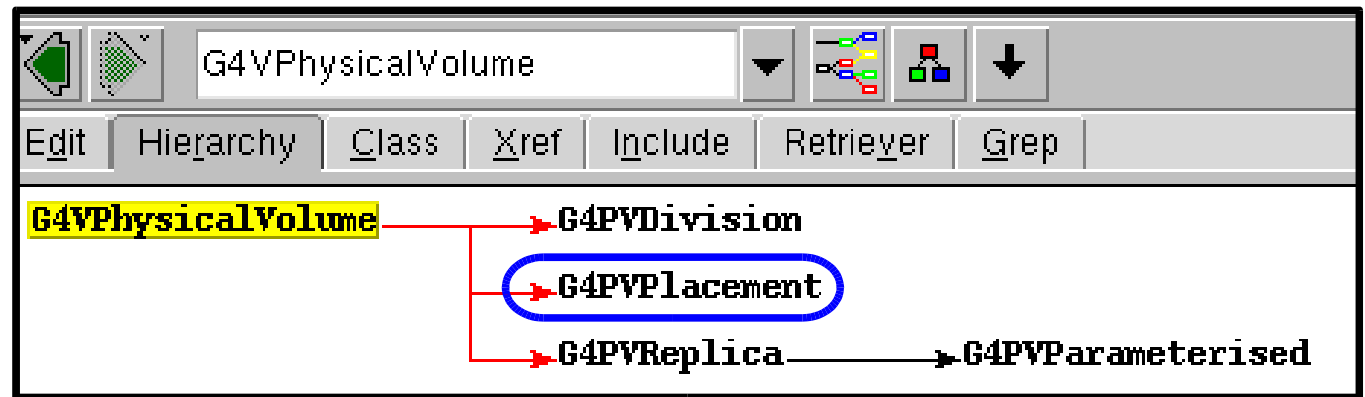
G4VPhysicalVolume

Inheritance from
G4VPhysicalVolume

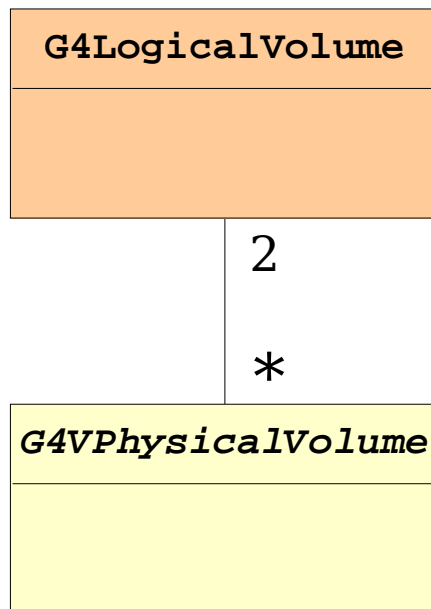


G4VPhysicalVolume

Inheritance from
G4VPhysicalVolume



Read:



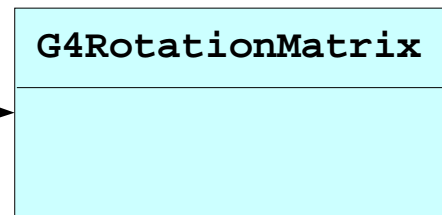
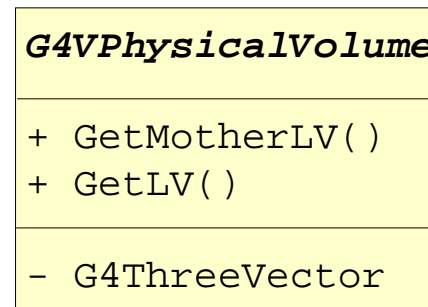
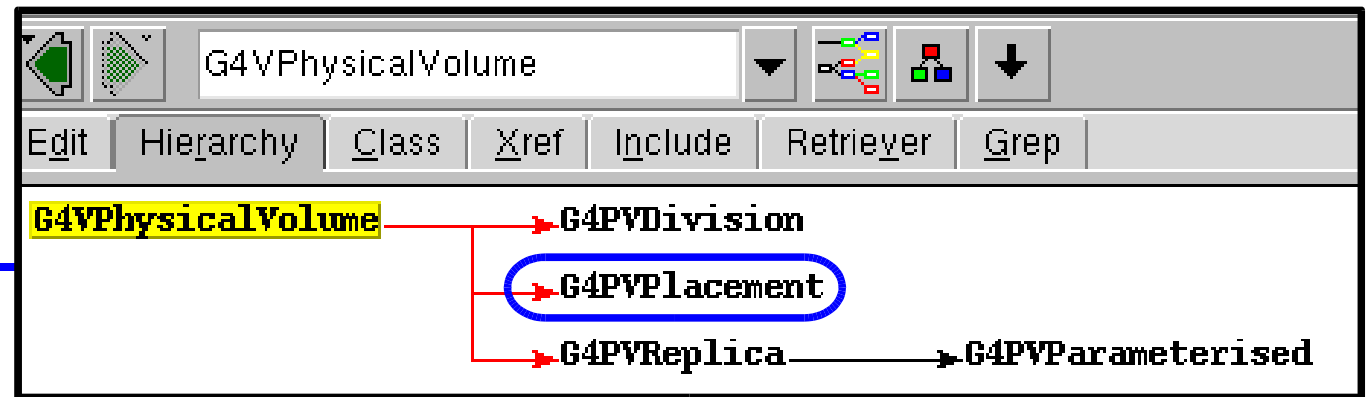
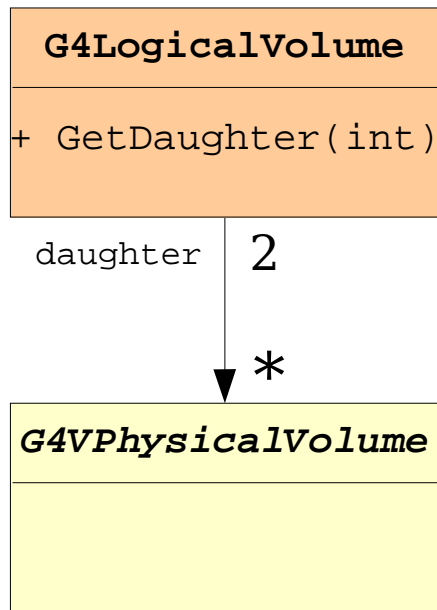
- **Top->Down: One** instance of **G4LogicalVolume** is associated with **0-n** instances of **G4VPhysicalVolume**
- **Down->Top: One** instance of **G4VPhysicalVolume** is associated with **2** instances of **G4LogicalVolume**
- **BUT:** The accessibility of one instance from the other one is NOT symmetric!

(Note: This cannot be read from the diagram)

G4VPhysicalVolume

Inheritance from
G4VPhysicalVolume

as seen from
G4LogicalVolume



as seen from
G4VPhysicalVolume

Single Placement

```
class G4PVPlacement // (short for Physical Volume Placement)
: public G4VPhysicalVolume {
public:
  G4PVPlacement(
    const G4Transform3D& trans,
    G4LogicalVolume* child,
    const G4String & name, // provide a name
    G4LogicalVolume* parent,
    G4bool many, // NOT USED!
    G4int copyNumber) ; // numbering, useful when placing
                          // the same child more times
  ... };
```

```
typedef HepTransform3D G4Transform3D;
```

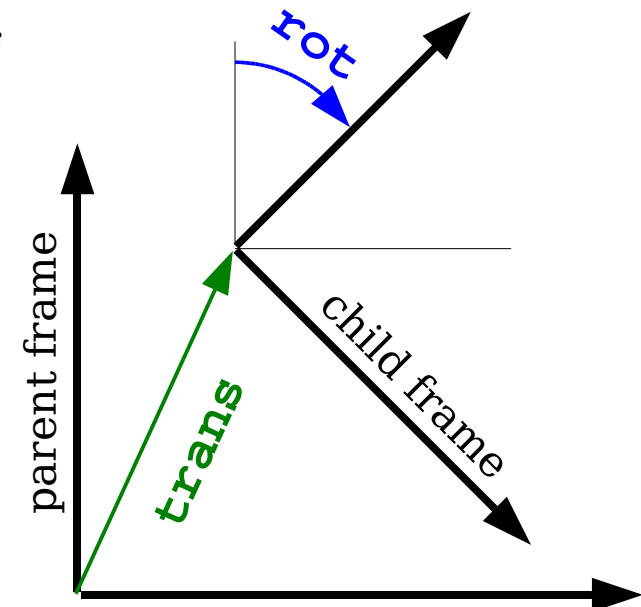
```
HepTransform3D( // constructor
  const HepRotation & rot,
  const Hep3Vector & trans);
```

HepRotation:

an orthonormal rotation

Hep3Vector:

a 3-dim. translation vector



Constraints

Two very important constraints on placements apply because of the way Geant4 is tracking particles:

The solids of the children **must** be fully inside the solid of the parent.

The solids of siblings **must not** overlap.

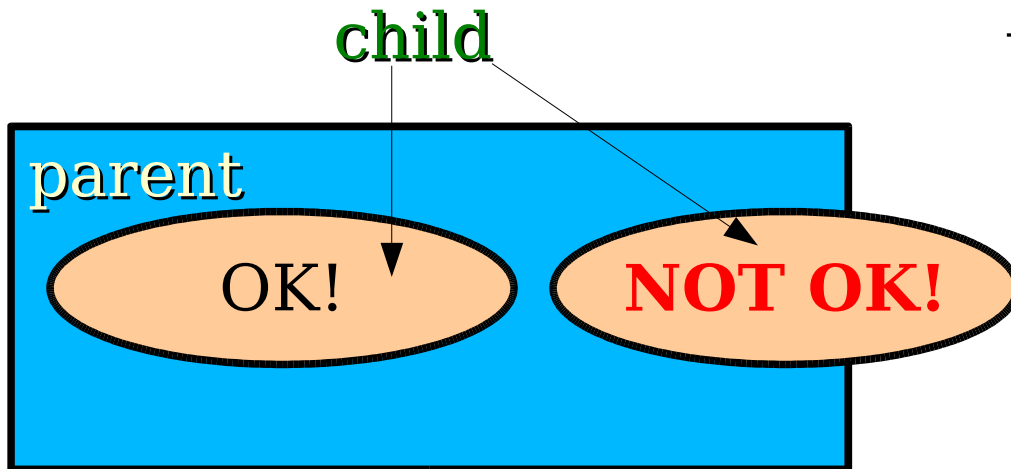
Geant4 will not warn you in case your setup violates these constraints. But expect undefined behavior - if not crashes ...

Constraints

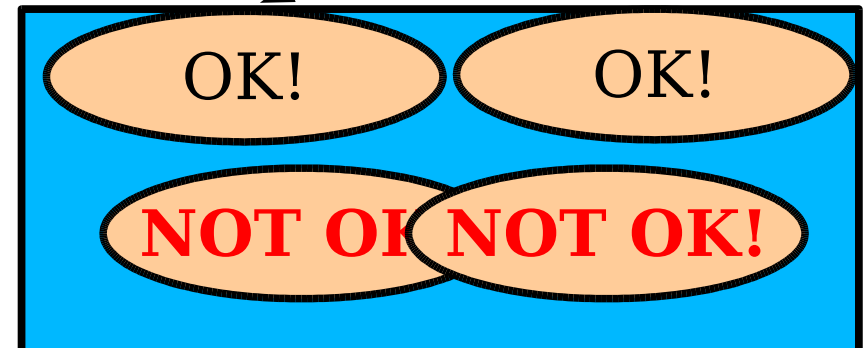
Two very important constraints on placements apply because of the way Geant4 is tracking particles:

The solids of the children **must** be fully inside the solid of the parent.

The solids of siblings **must not** overlap.



touching surfaces are allowed!



Constraints

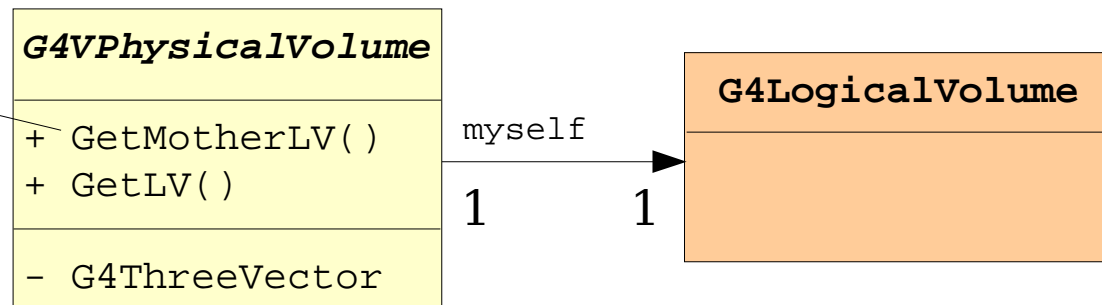
Two very important constraints on placements apply because of the way Geant4 is tracking particles:

The solids of the children **must** be fully inside the solid of the parent.


The solids of siblings **must not** overlap.

There must be **one physical volume without a parent and without rotation & translation.**

will return
the 0 pointer!

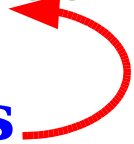


Implications

- **Logical volumes can have children**
 - **Children are described by logical volumes**
- + constraints
- 

=>

Implications

- **Logical volumes can have children**
 - **Children are described by logical volumes**
- + constraints 

=>

The hierarchy of volumes is a

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

multigraph.

- 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.

Behavior

- **Logical volumes can have children**
- **Children are described by logical volumes**

:=

The solid of a child takes the space of the parent, e.g. you put an air bubble into a homogeneous cheese brick, you get Emmentaler.



Behavior

- **Logical volumes can have children**
- **Children are described by logical volumes**

:=

The solid of a child takes the space of the parent, e.g. you put an air bubble into a homogeneous cheese brick, you get Emmentaler. You put a mouse into the air bubble, you are mean, because the mouse must be inside its parent bubble ...



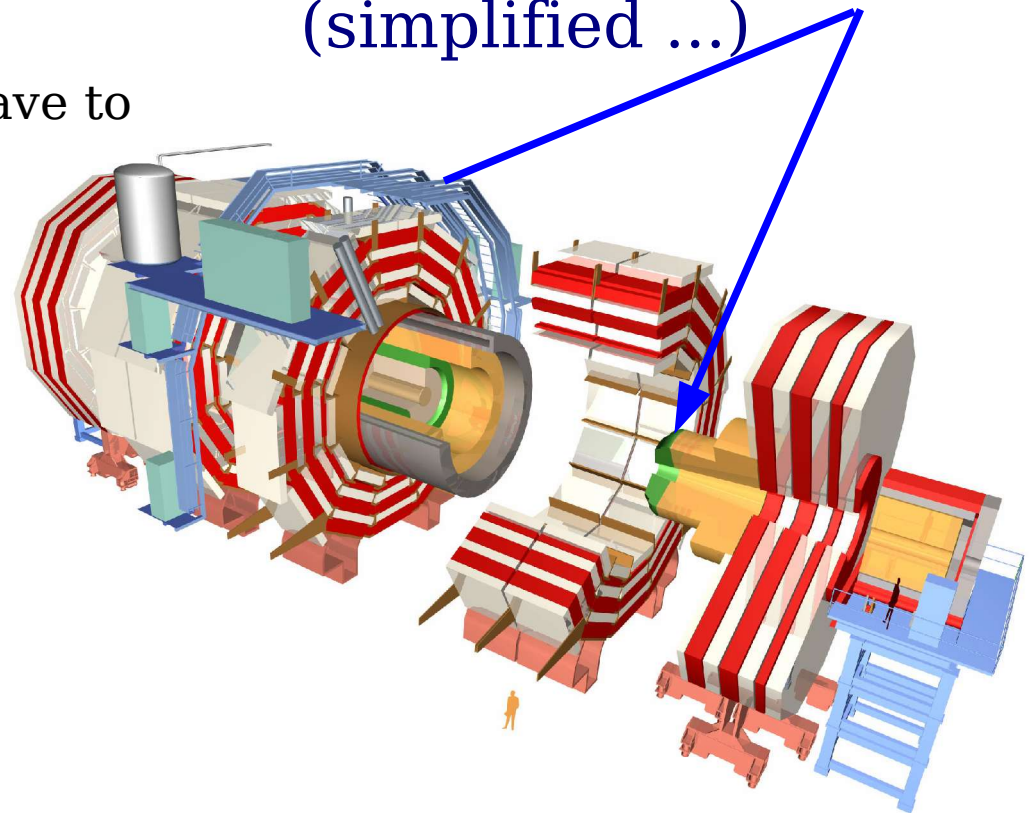
Implications

- **Logical volumes can have children**
- **Children are described by logical volumes**

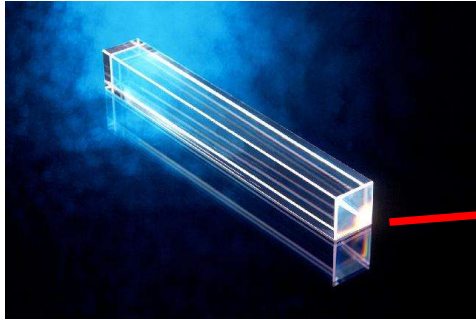
=>

Example:
CMS ECal Endcap
(simplified ...)

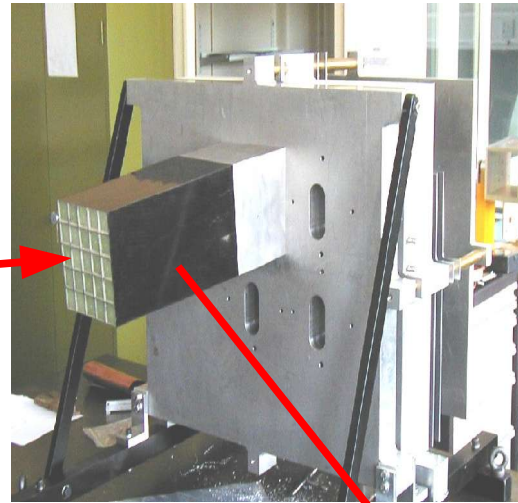
- All positions are relative
 - global values (w.r.t. the world) have to be calculated
- Exploiting the graph character:
 - symmetries of the geometry are easily accounted for
 - Number of logical volumes \ll Number of "real" volumes of the detector
 - rigid geometry ...



CMS ECal Endcap

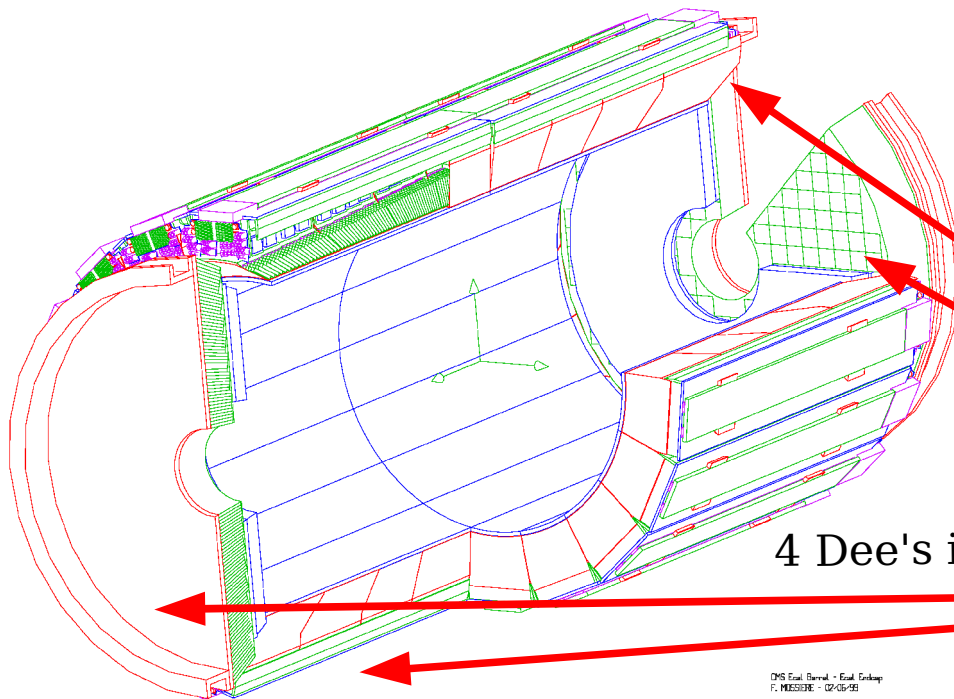


25 crystals of same shape per supercrystal



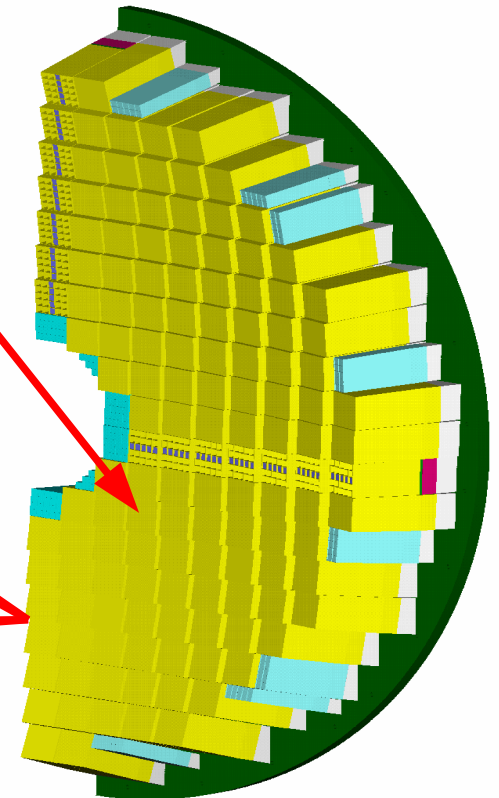
3662 crystals per Dee
14648 crystals in total

(Crystal mass: 5.4 tonnes per Dee)



138 fully filled +
14 partially filled
supercrystals per "Dee"

4 Dee's in total

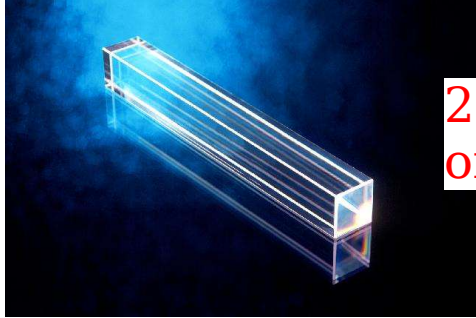


CMS Endcap - Endcap
F. MESENE - 02-05-98

G4 Geometry

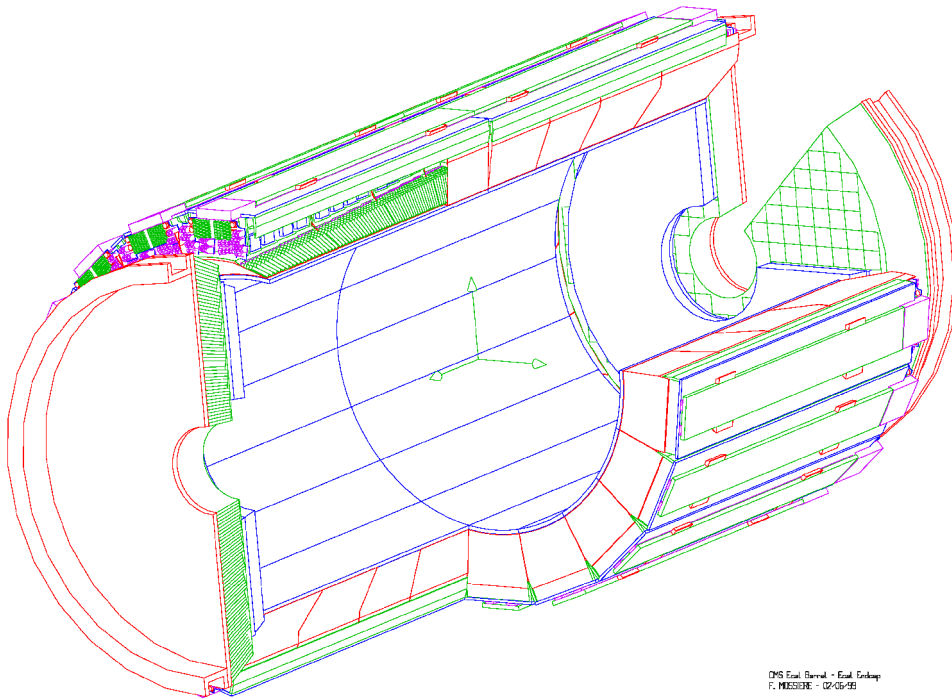
modified & simplified from
the official CMS description

1 G4LogicalVolume
for the xtal

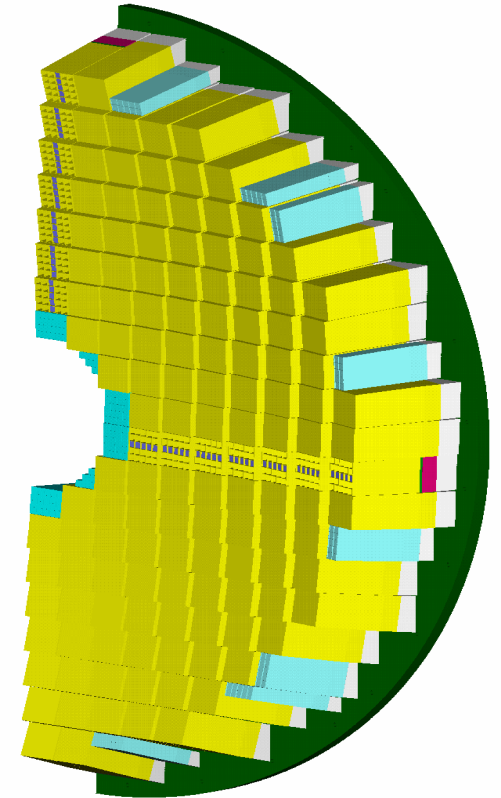


25 G4PVPlacements
of xtal in superxtal

1 G4LogicalVolume
for the superxtal

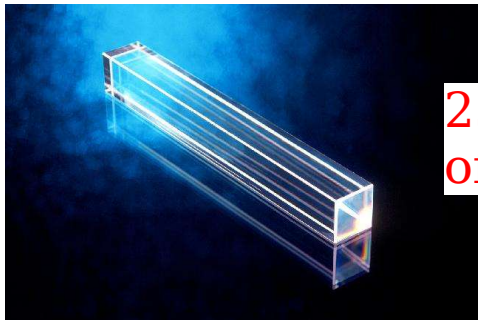


CMS End Cap and End Cap
F. MEISSE - 02-05-98

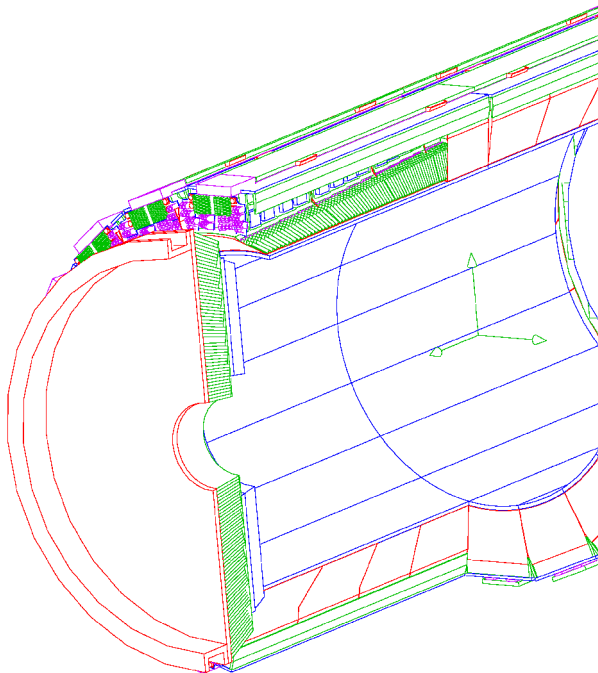


G4 Geo

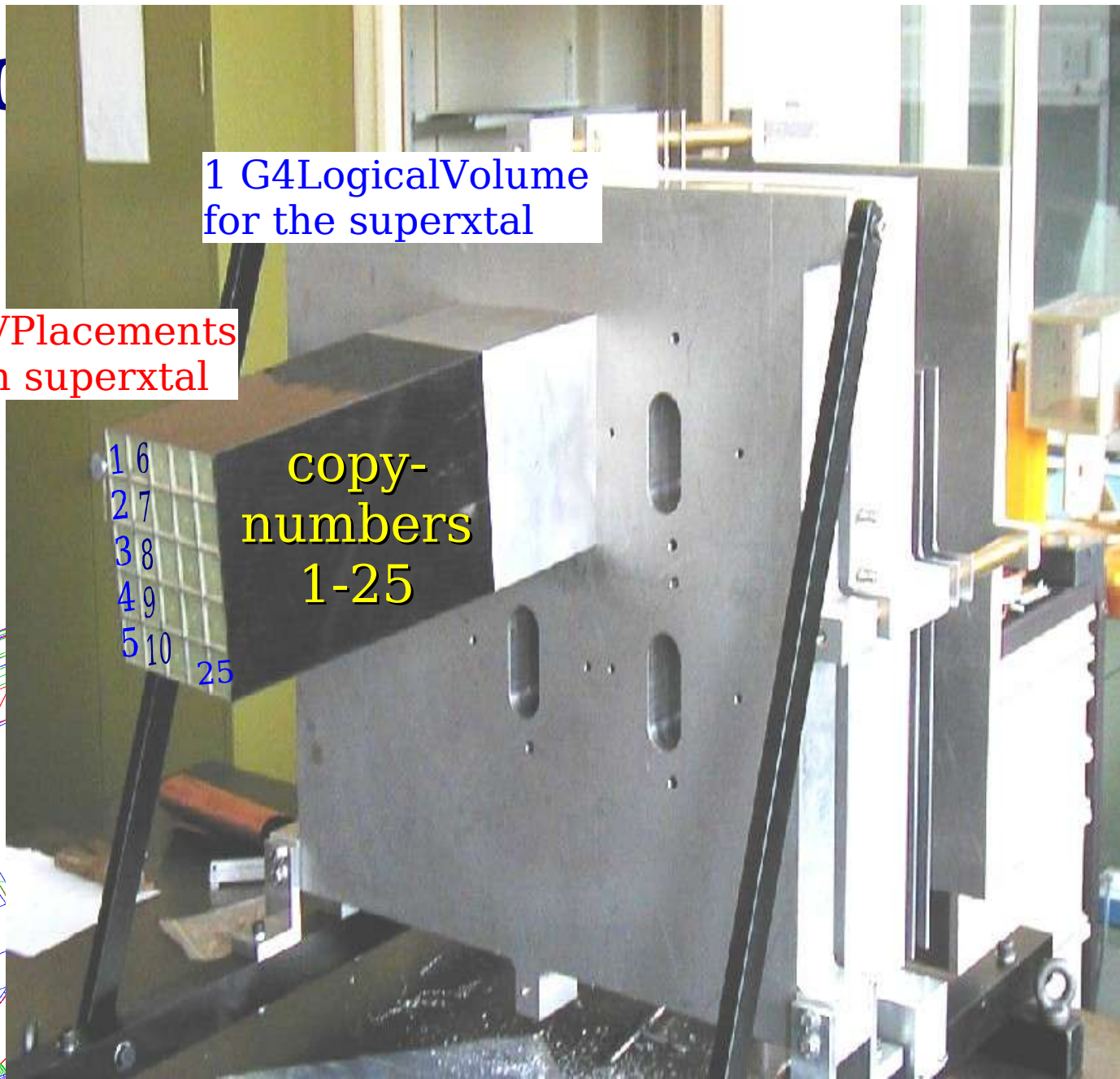
1 G4LogicalVolume
for the xtal



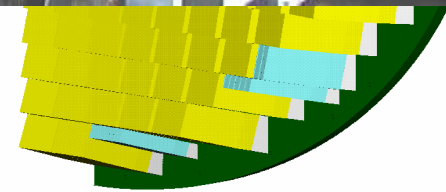
25 G4PVPlacements
of xtal in superxtal



1 G4LogicalVolume
for the superxtal



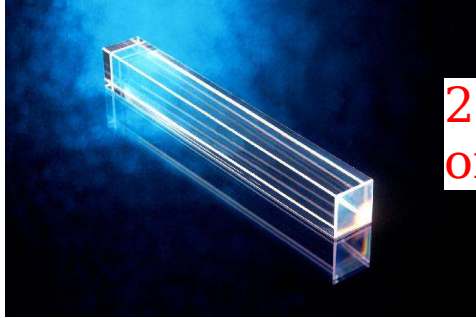
GMS Eval. Dienst - Eval. Endung
F. MEISER - 02-05-98



G4 Geometry

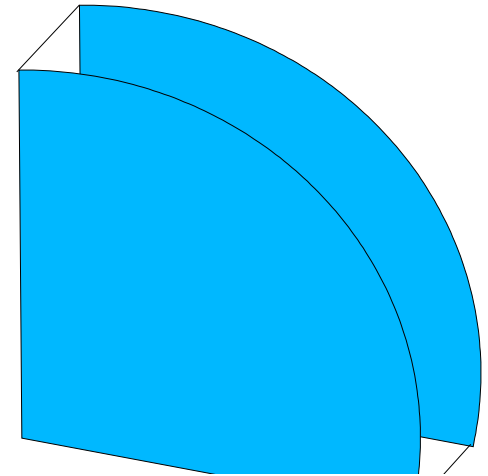
modified & simplified from the official CMS description

1 G4LogicalVolume for the xtal

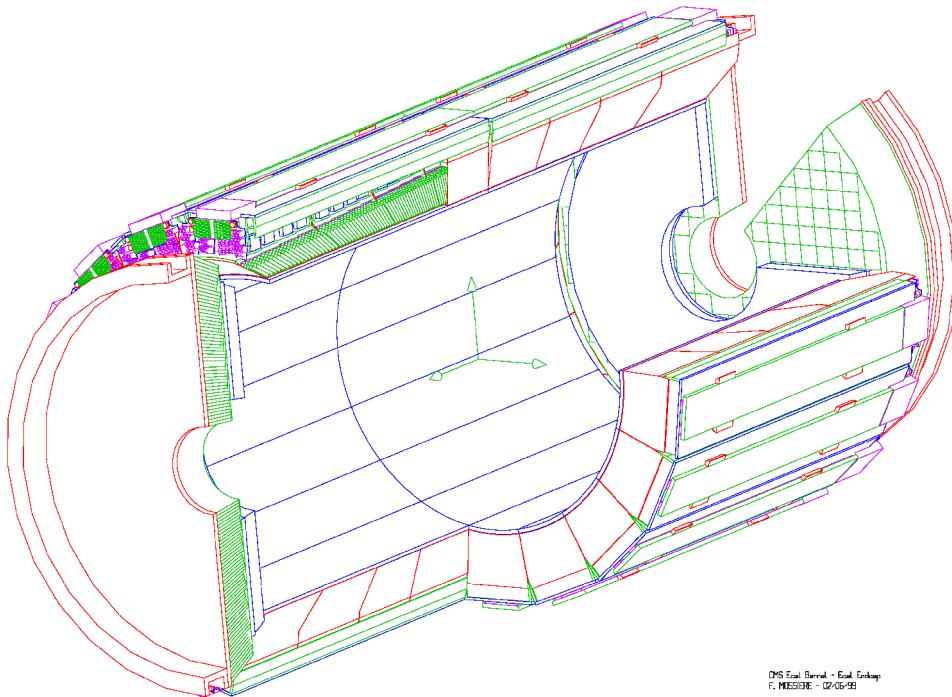


25 G4PVPlacements of xtal in superxtal

1 G4LogicalVolume for the superxtal



1 G4LogicalVolume for a HalfDee, filled with air



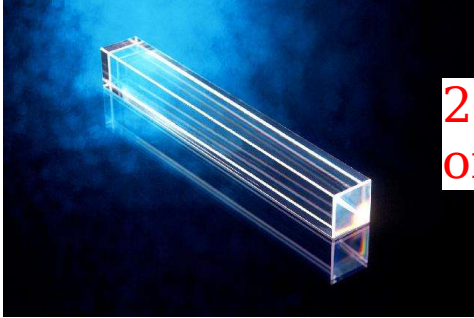
Introducing envelope volumes is a common practice to exploit geometrical symmetries

CMS Det. Conv. - Eval. Endcap
F. MESEBE - 02-05-98

G4 Geometry

modified & simplified from
the official CMS description

1 G4LogicalVolume
for the xtal

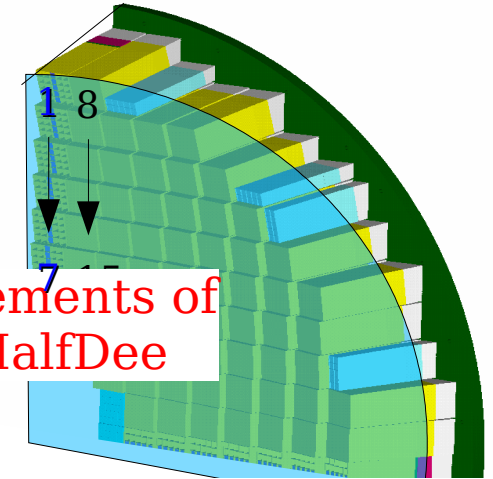


25 G4PVPlacements
of xtal in superxtal

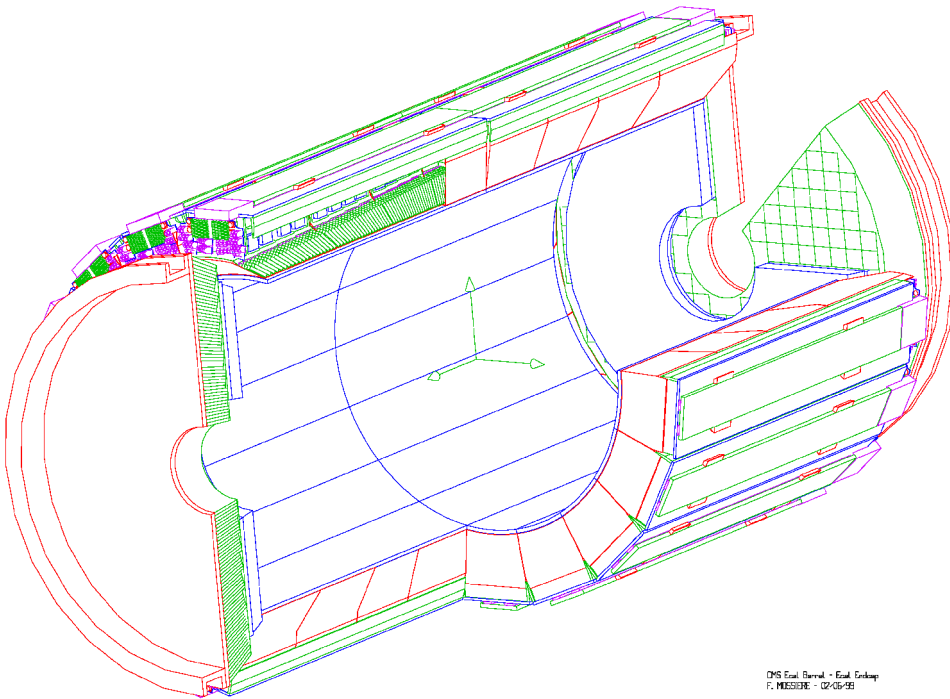
1 G4LogicalVolume
for the superxtal



69 G4PVPlacements
of superxtal in HalfDee



1 G4LogicalVolume for a
HalfDee, filled with air

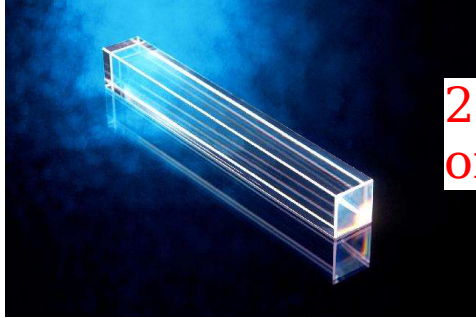


CMS Equal-View - Equal-View
F. MEISER - 02-05-98

G4 Geometry

modified & simplified from
the official CMS description

1 G4LogicalVolume
for the xtal

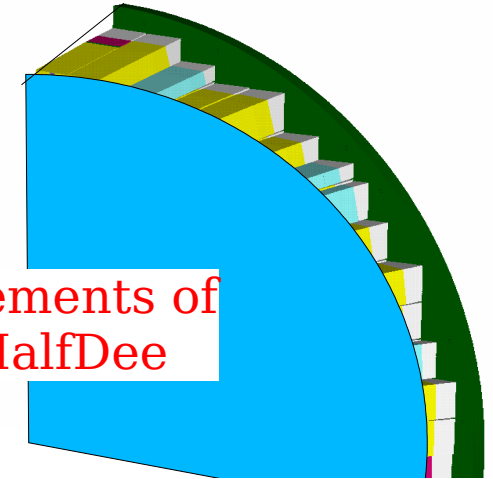


25 G4PVPlacements
of xtal in superxtal

1 G4LogicalVolume
for the superxtal

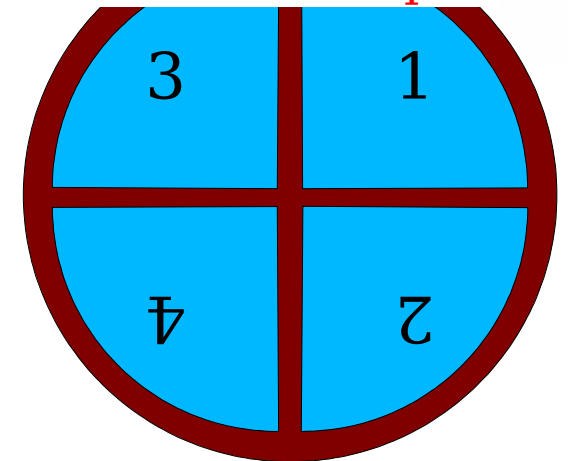


69 G4PVPlacements of
superxtal in HalfDee

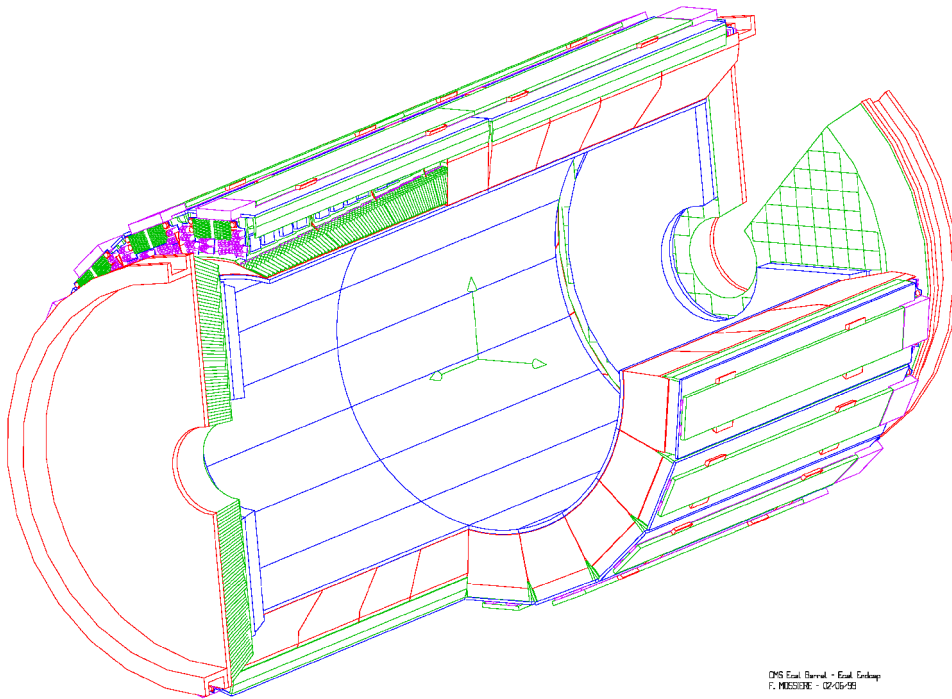


1 G4LogicalVolume for a
HalfDee, filled with air

4 G4PVPlacements of
HalfDee in Endcap



1 G4LogicalVolume for an
Endcap envelope, air

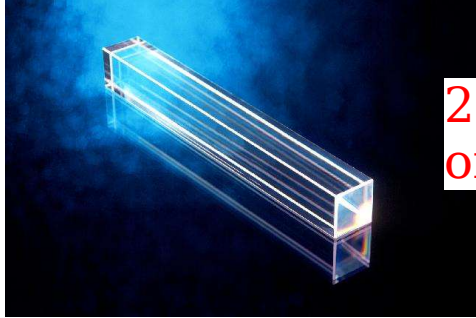


CMS End Cap - End Cap
F. MESEBE - 02-05-98

G4 Geometry

modified & simplified from
the official CMS description

1 G4LogicalVolume
for the xtal

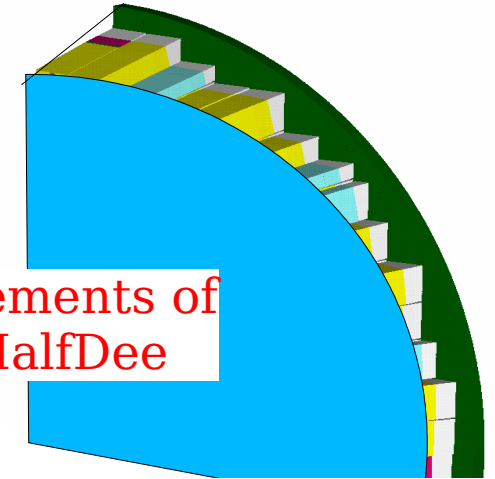


25 G4PVPlacements
of xtal in superxtal

1 G4LogicalVolume
for the superxtal

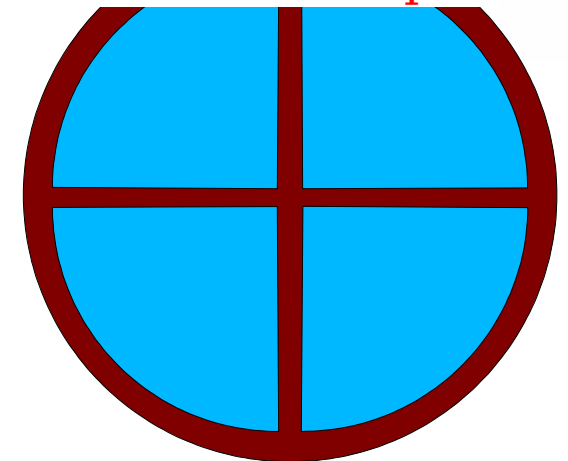


69 G4PVPlacements of
superxtal in HalfDee

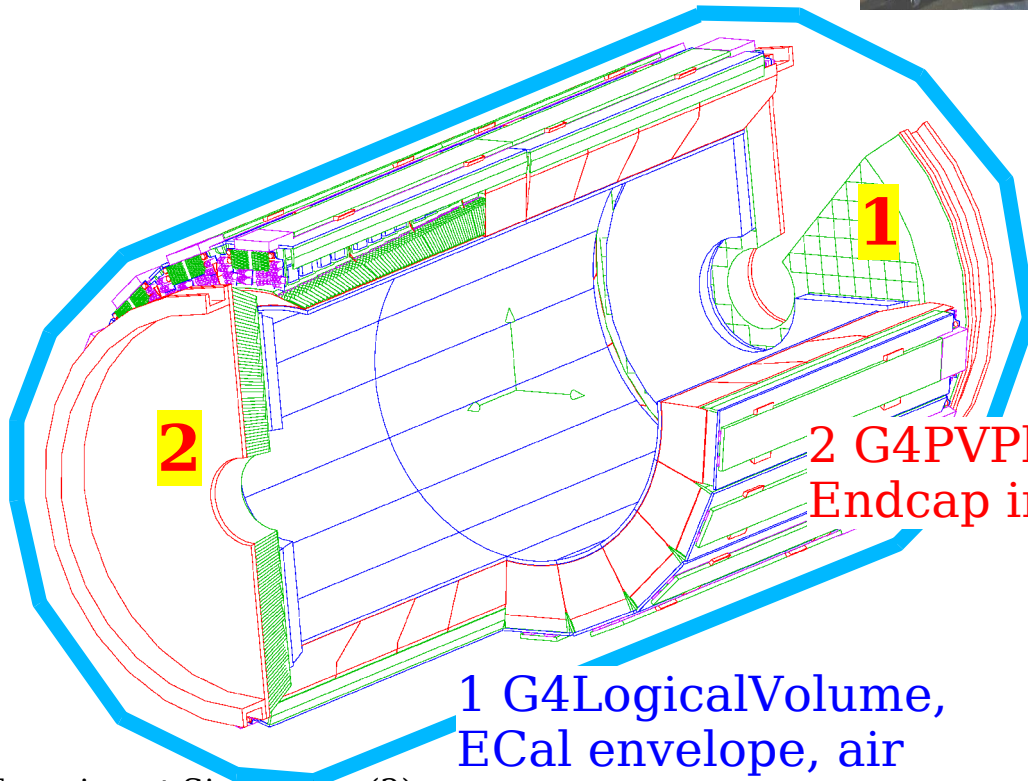


1 G4LogicalVolume for a
HalfDee, filled with air

4 G4PVPlacements of
HalfDee in Endcap



1 G4LogicalVolume for an
Endcap envelope, air



2 G4PVPlacements of
Endcap in ECal

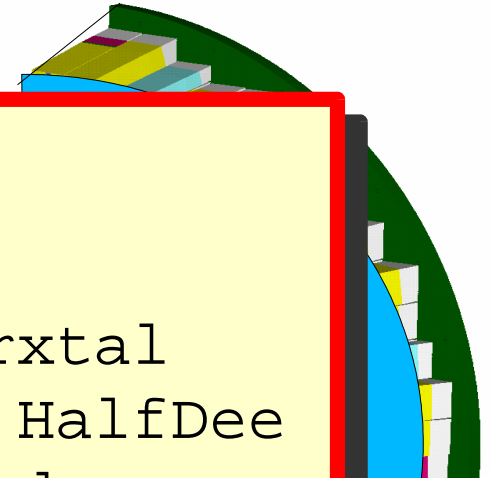
1 G4LogicalVolume,
ECal envelope, air

G4 Geometry

modified & simplified from the official CMS description

1 G4LogicalVolume for the xtal

1 G4LogicalVolume for the superxtal



G4LogicalVolumes

G4PVPlacements

1 xtal

1 superxtal

1 HalfDee

1 EndCap

1 ECal

25 xtal -> superxtal

69 superxtal -> HalfDee

4 HalfDee -> Endcap

2 Endcap -> ECal

5

100

=====

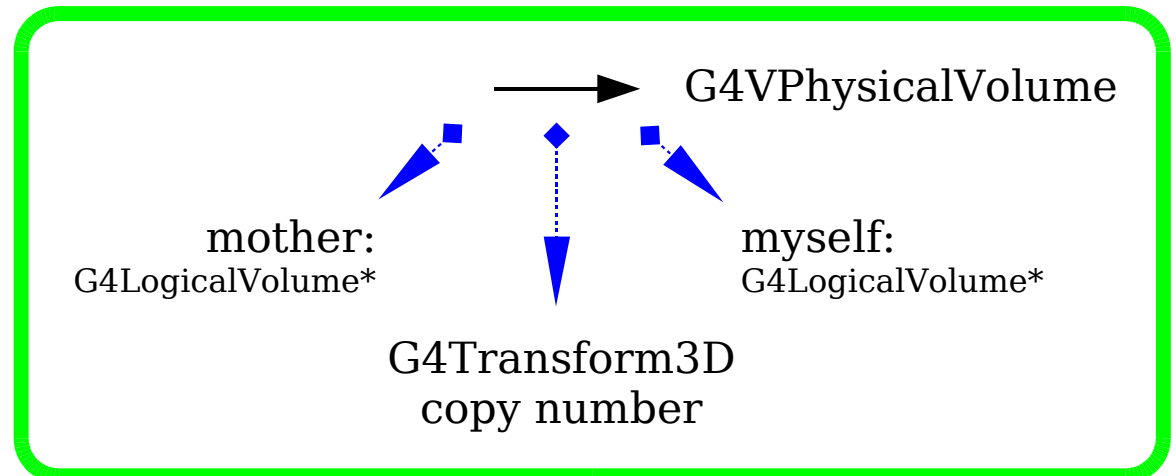
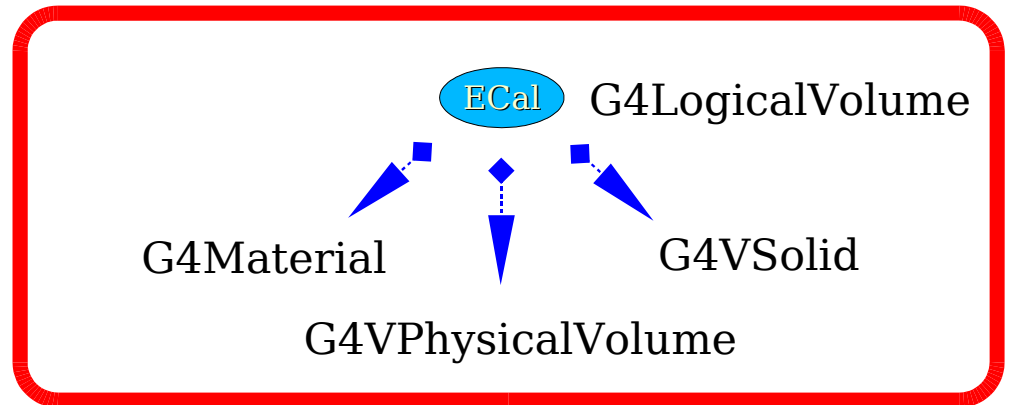
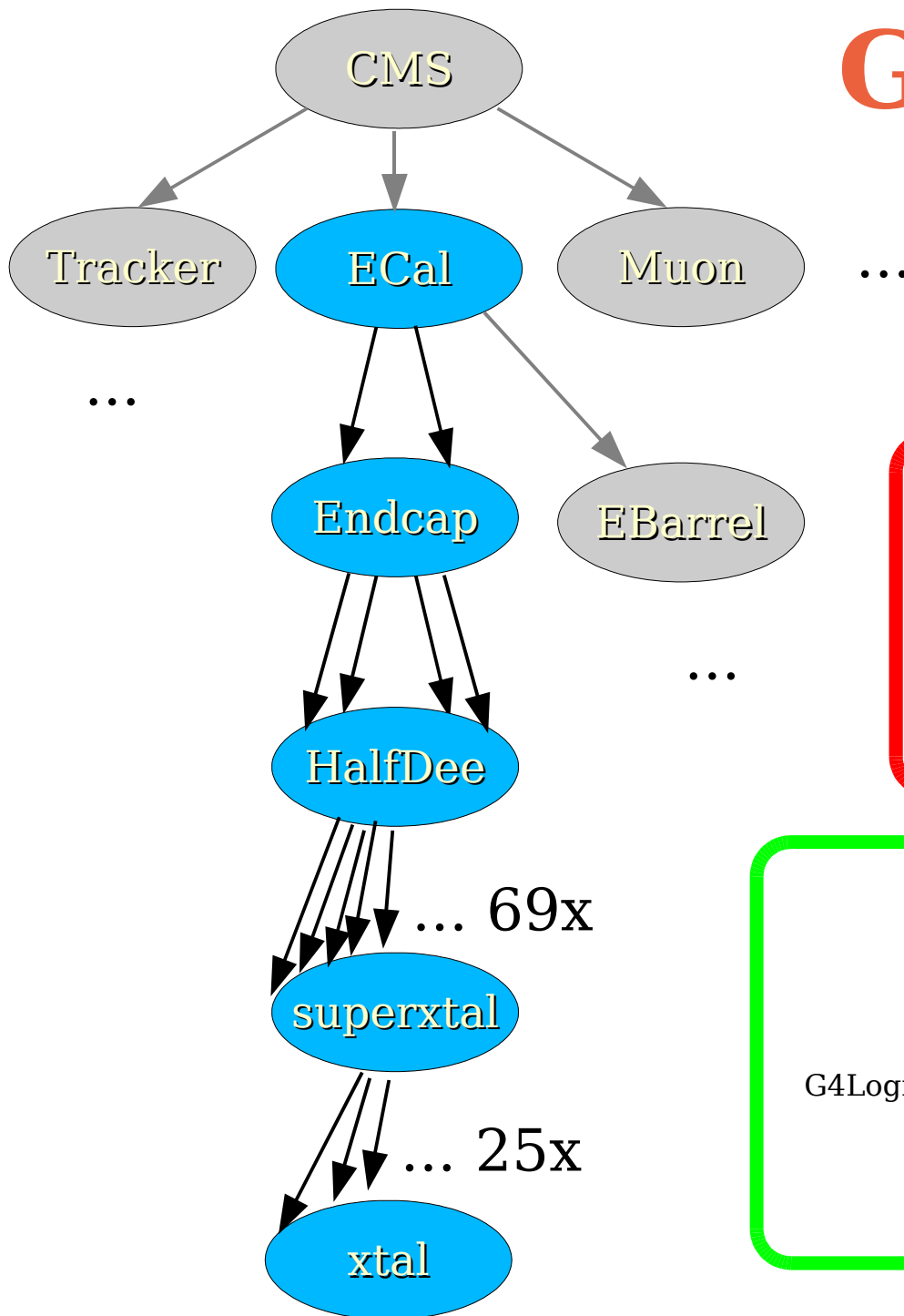
**Need only 105 instances
to represent ~14.000 crystals**

for a
air
of

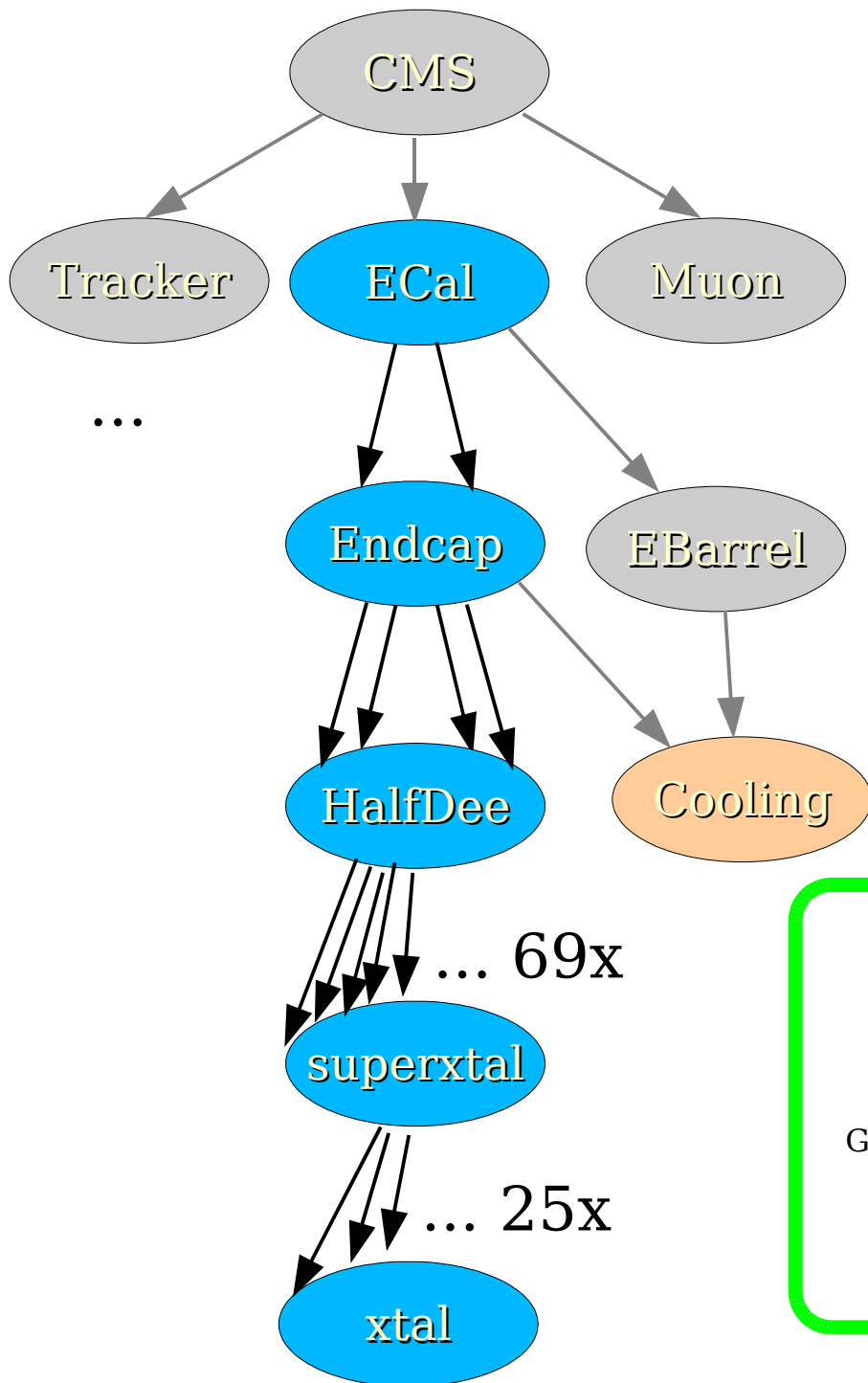
1 G4LogicalVolume,
ECal envelope, air

1 G4LogicalVolume for an
Endcap envelope, air

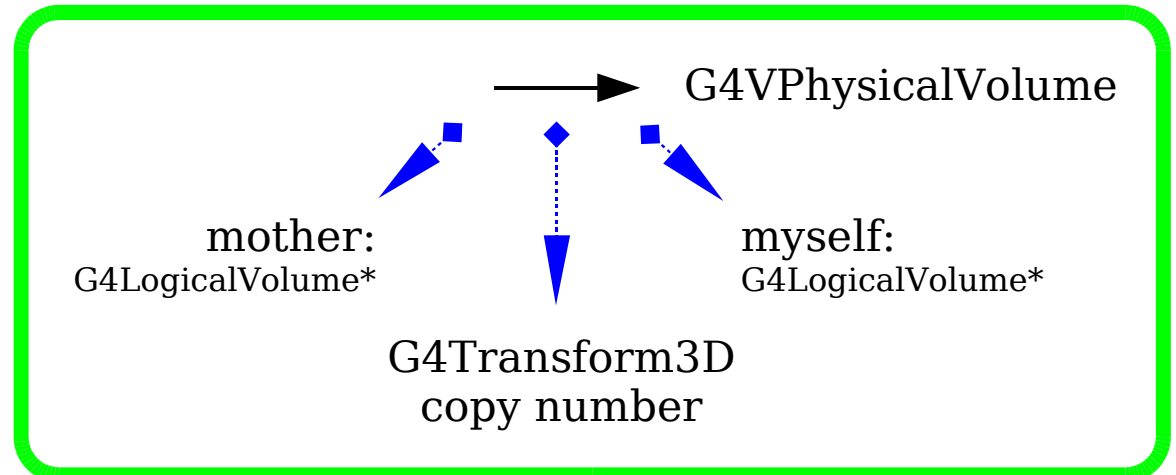
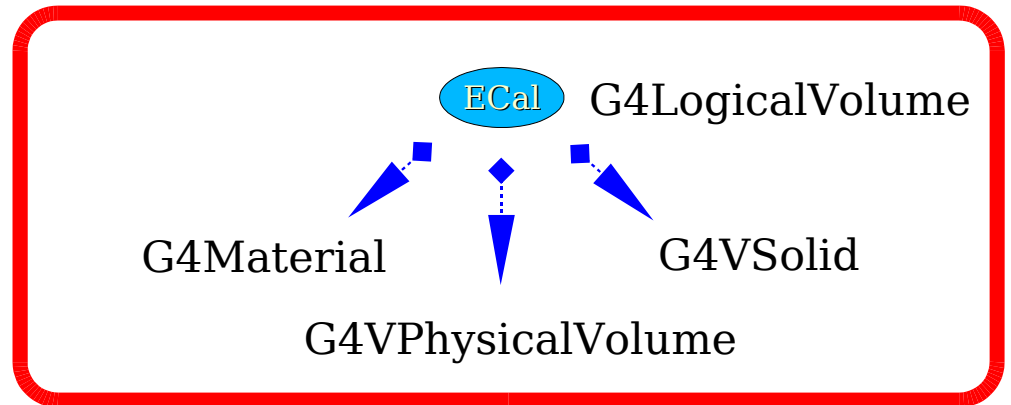
Graph Structure



Graph Structure



...
Can be used to re-use components in different geometrical branches!



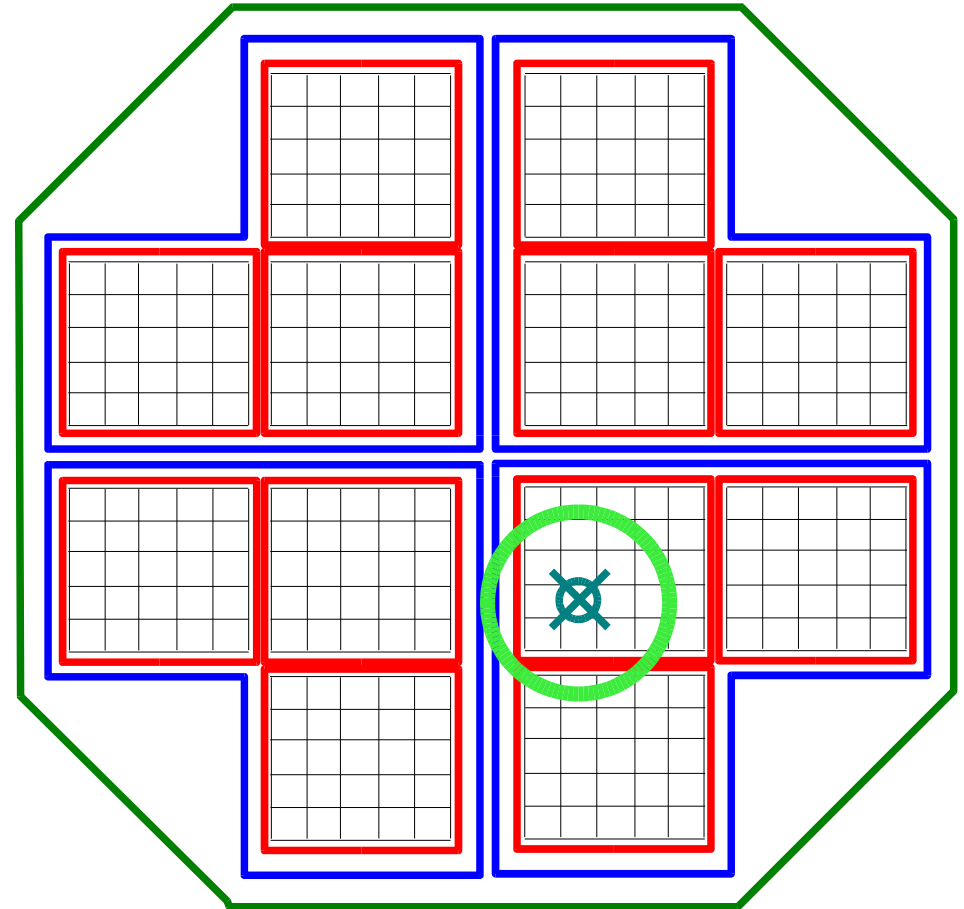
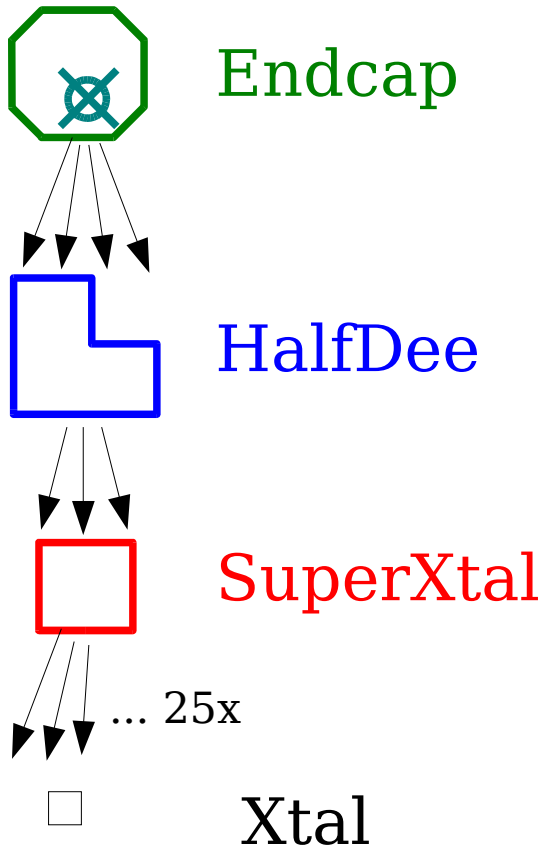
Summary: Geometry Ingredients

- G4LogicalVolume
 - basic building block
 - points (mandatory) to a G4VSolid and G4Material
- G4VSolid
 - defined within a local reference frame
 - gives answers to geometrical questions (point inside or outside, distance to surface, ...) asked in the local reference frame
- G4Material
- G4VPhysicalVolume
 - points to 2 G4LogicalVolumes: myself and mother
 - defines the relative orientation of the solids of myself w.r.t. mother

Basics of Navigation

⊗ **Where am I** (x,y,z) ?

(x,y,z) in global co-ordinates, i.e. w.r.t. to the world volume



and, again,
I simplify
the geometry
(a lot ...)

Basics of Navigation

Recursive Algorithm:

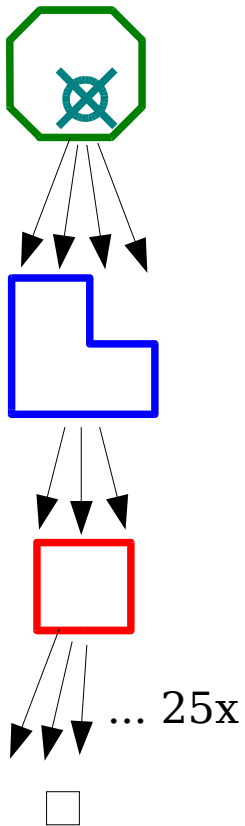
```

STACK S;
pos = (x,y,z);
push(s, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

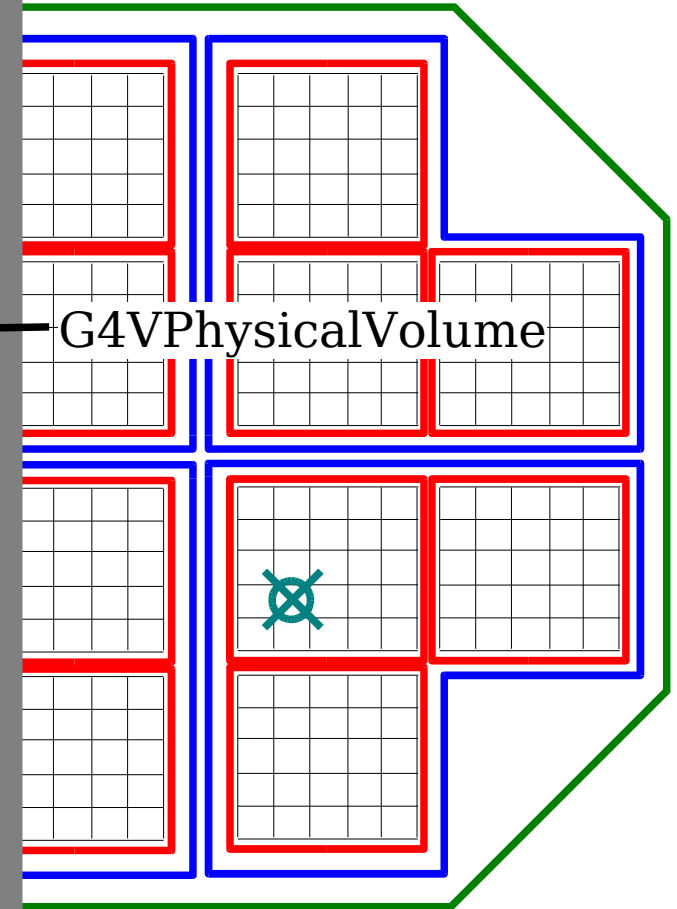
children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

```



G4VPhysicalVolume

G4LogicalVolume: mother,myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

```

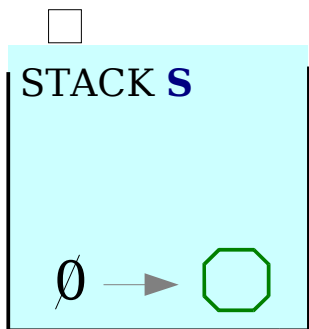
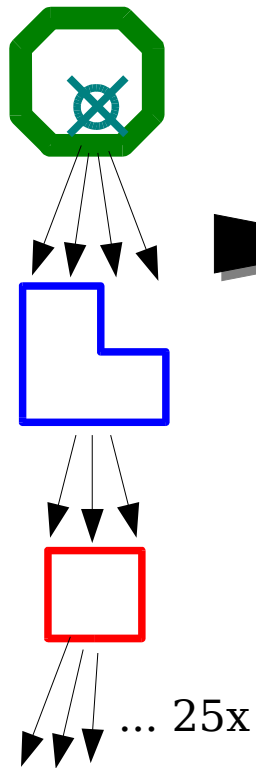
STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);
    
```

```

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}
    
```

```

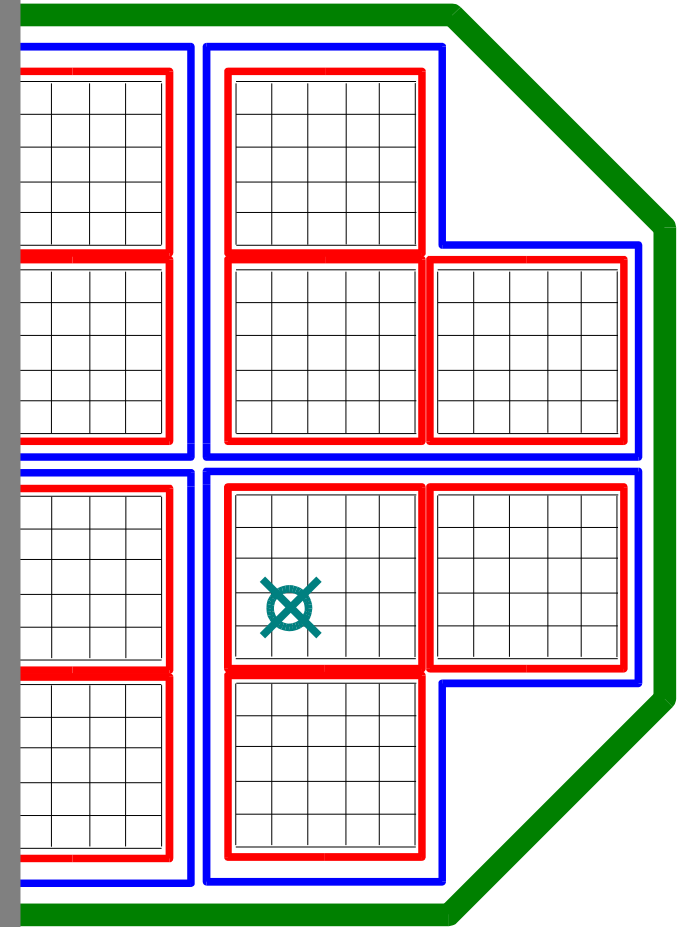
children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}
    
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

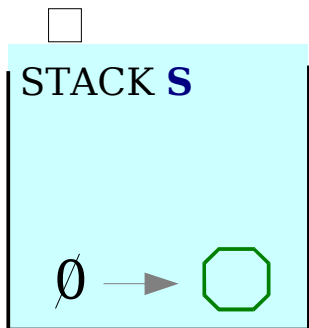
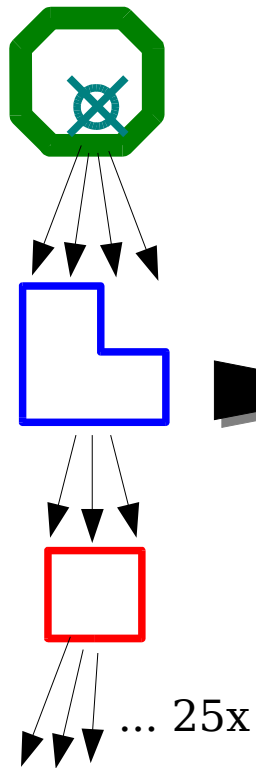
Recursive Algorithm:

```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

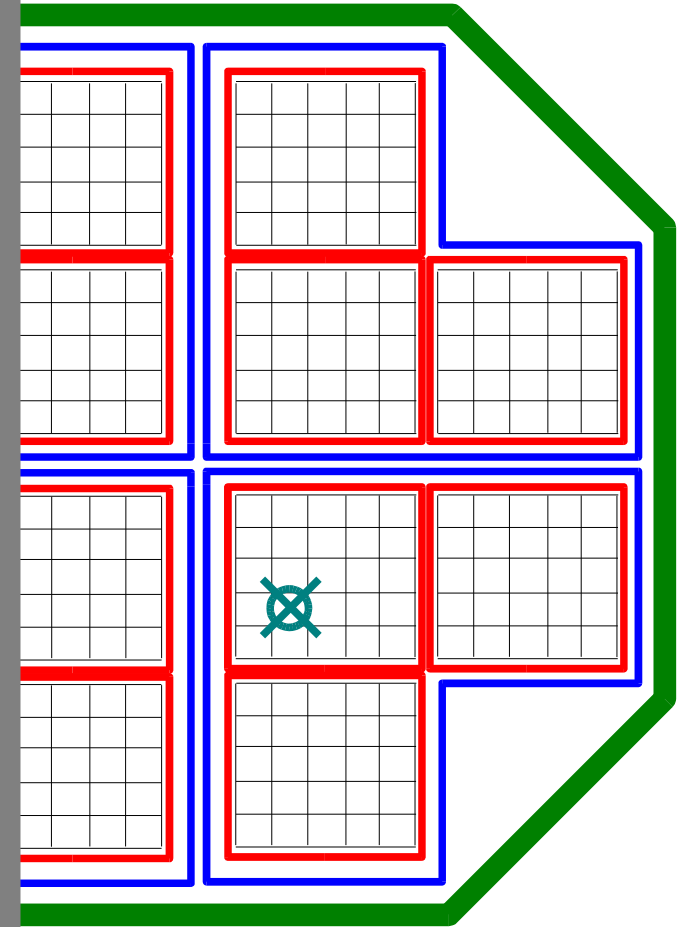
children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}
  
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

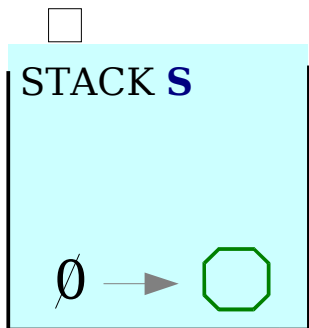
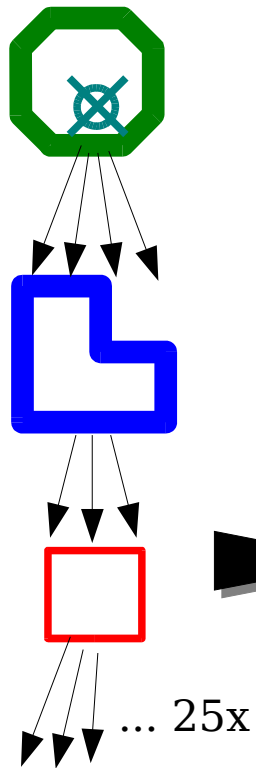
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

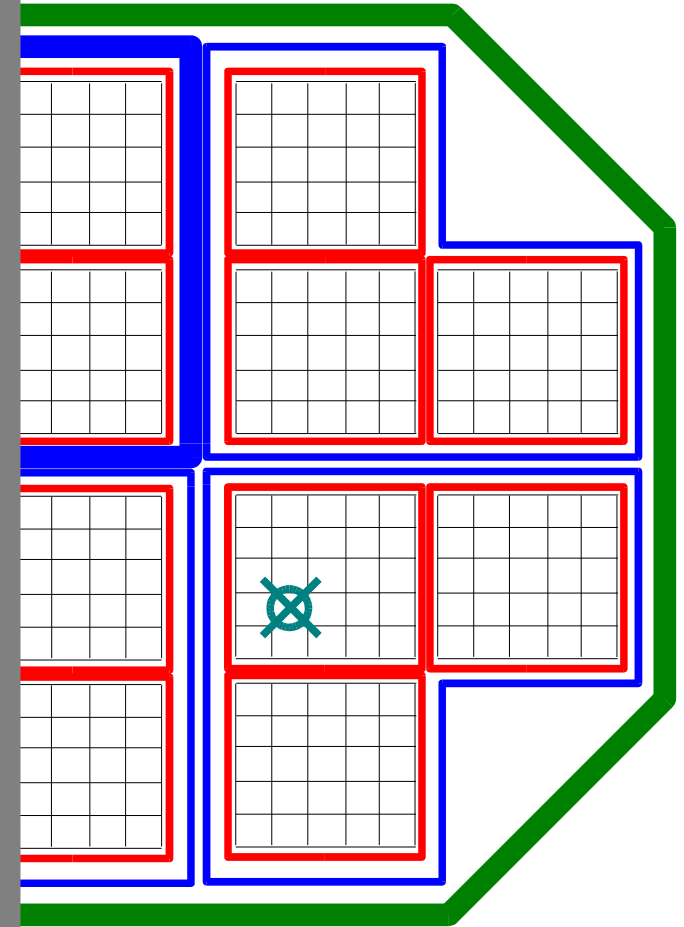
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

```

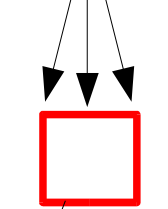
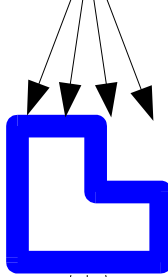
STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

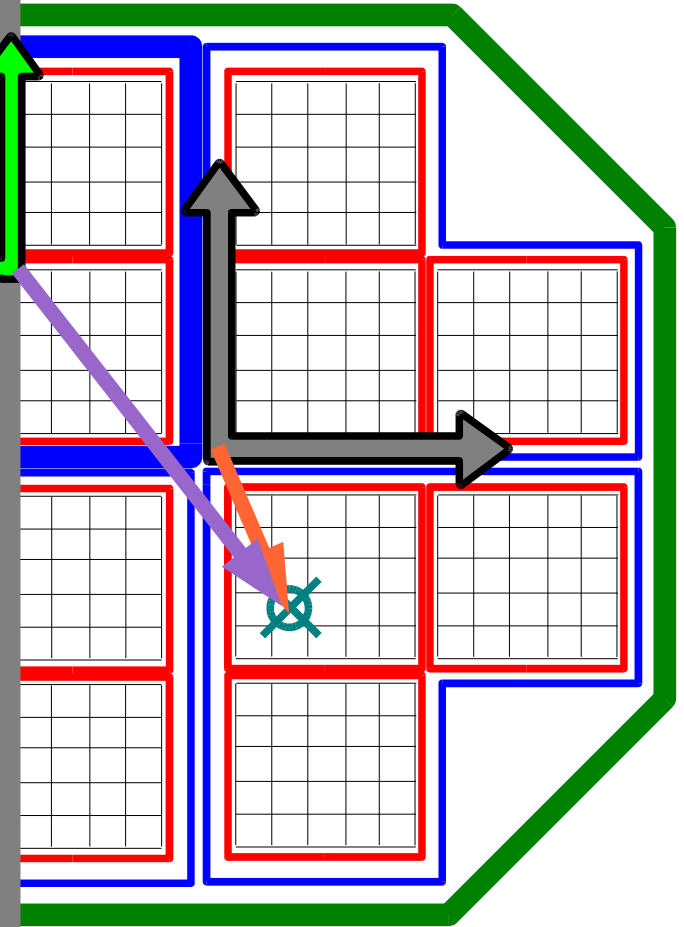
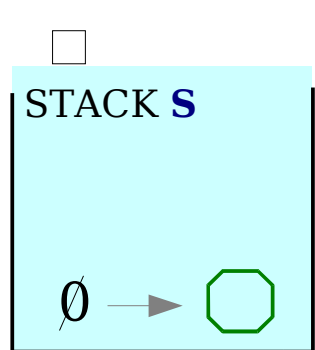
children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}
  
```

G4VPhysicalVolume

G4LogicalVolume: mother,myself
G4Transform3D: relative orientation



... 25x



Basics of Navigation

Recursive Algorithm:

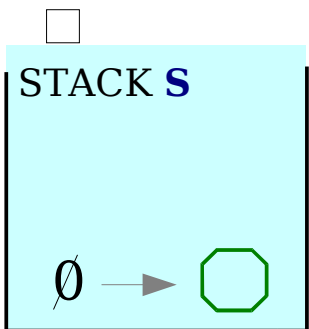
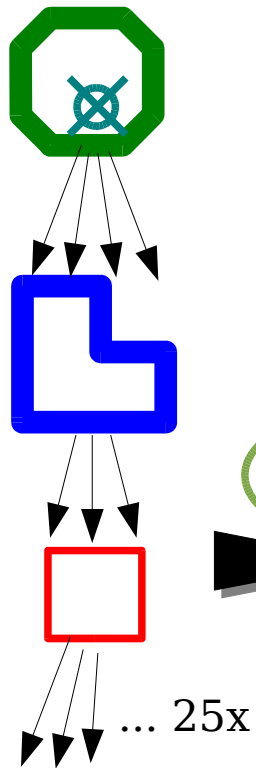
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

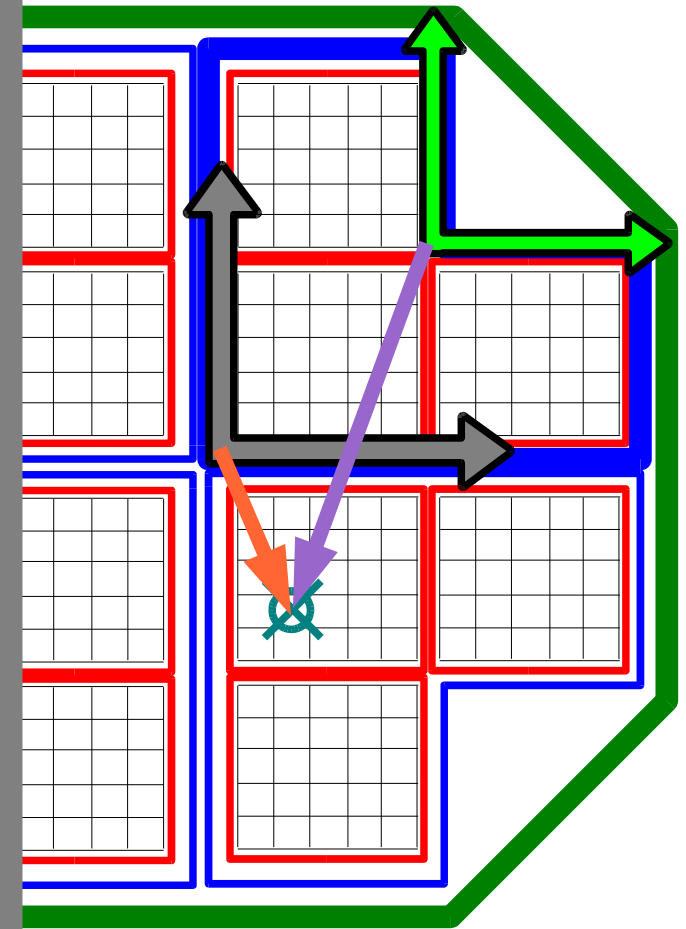
children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

```



G4VPhysicalVolume

G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

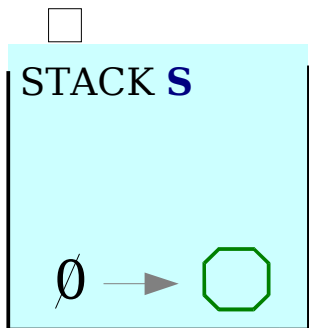
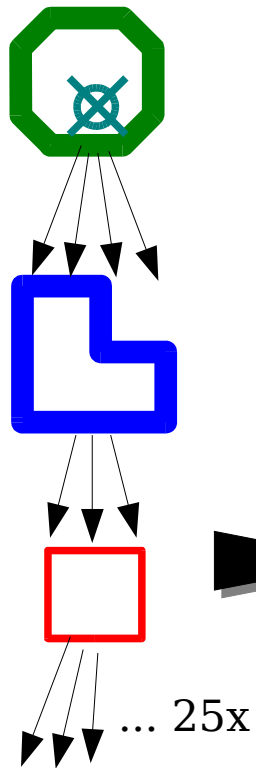
Recursive Algorithm:

```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

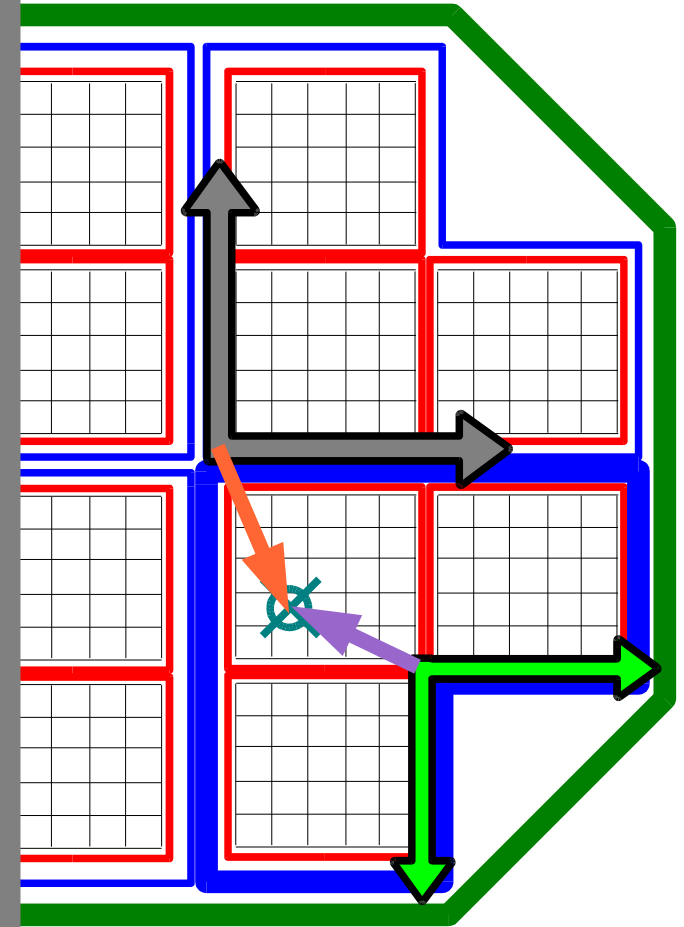
descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}
  
```



G4VPhysicalVolume

G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

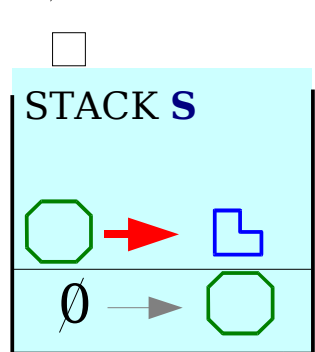
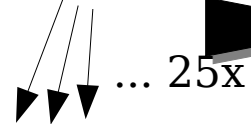
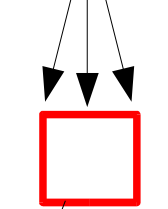
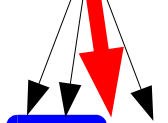
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

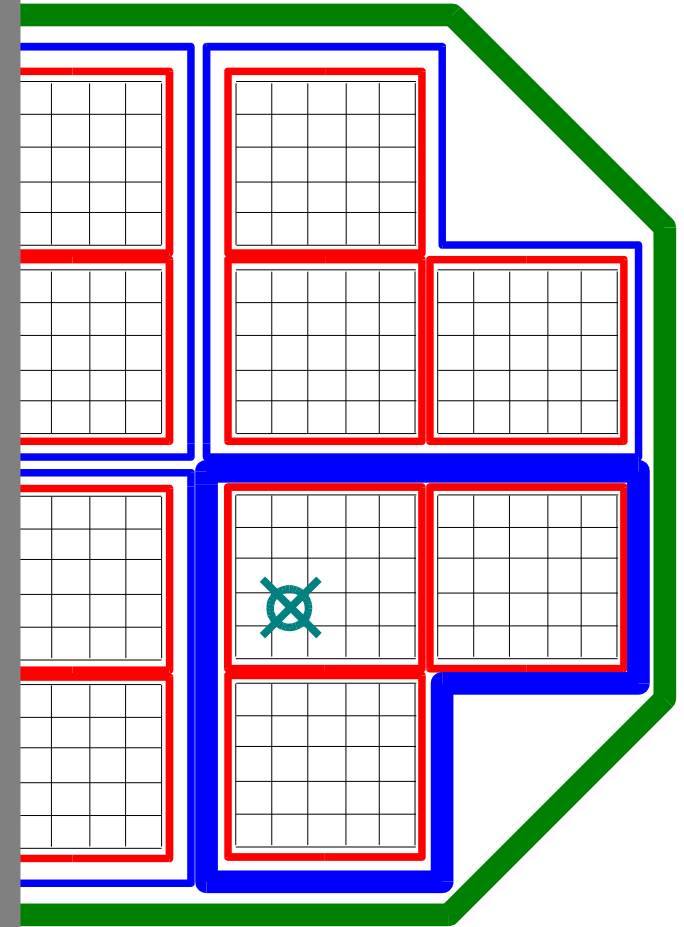
```



G4VPhysicalVolume



G4LogicalVolume: mother,myself
G4Transform3D: relative orientation



Basics of Navigation

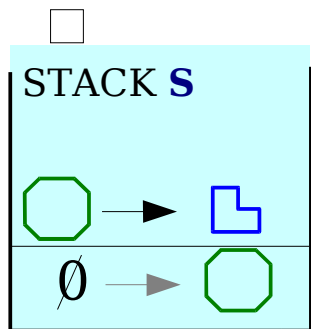
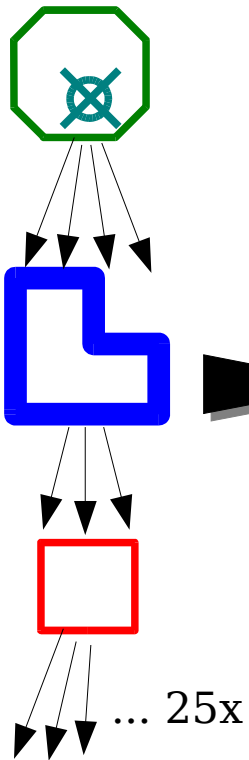
Recursive Algorithm:

```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

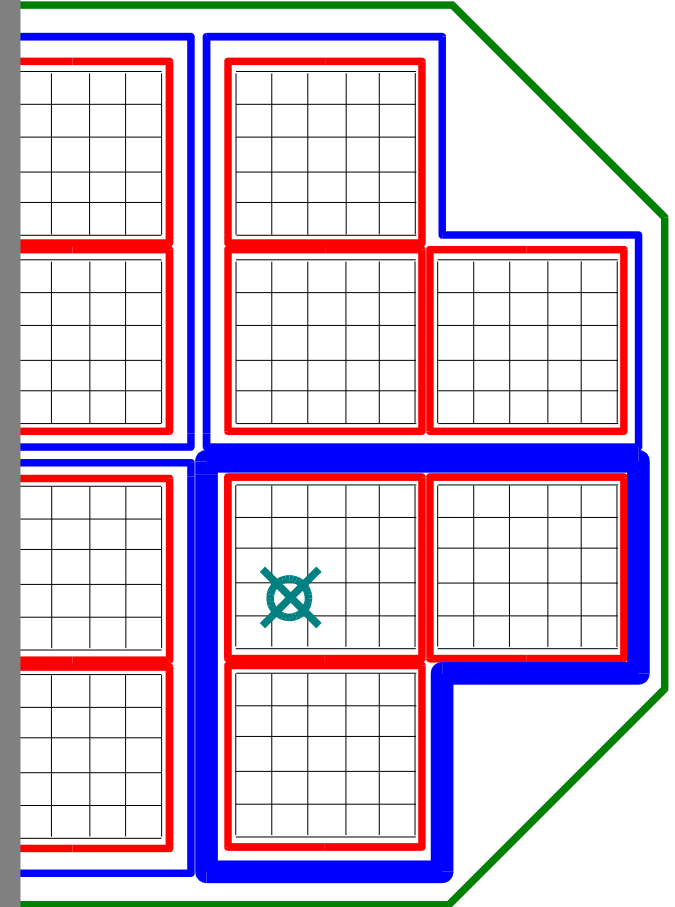
children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}
  
```



G4VPhysicalVolume



G4LogicalVolume: mother,myself
G4Transform3D: relative orientation



Basics of Navigation

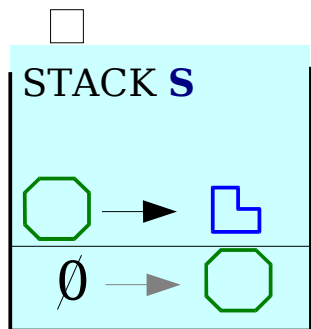
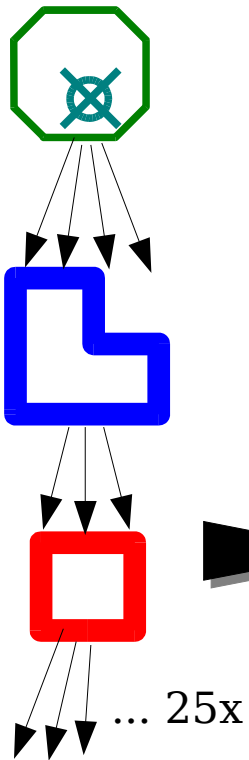
Recursive Algorithm:

```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

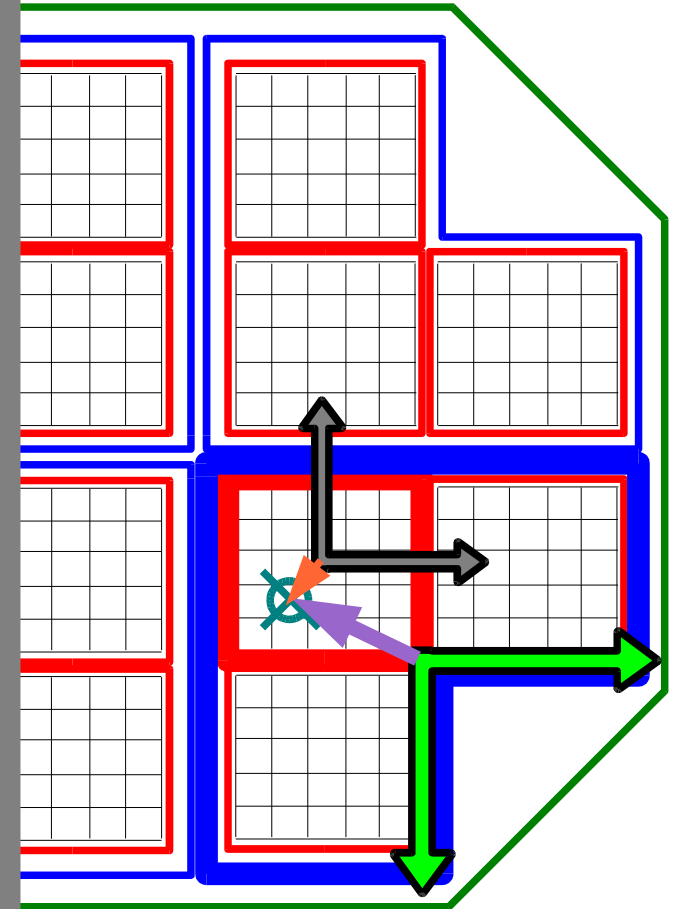
children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}
    
```



G4VPhysicalVolume



G4LogicalVolume: mother,myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

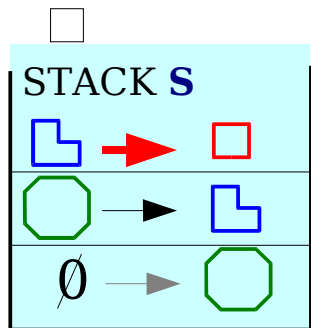
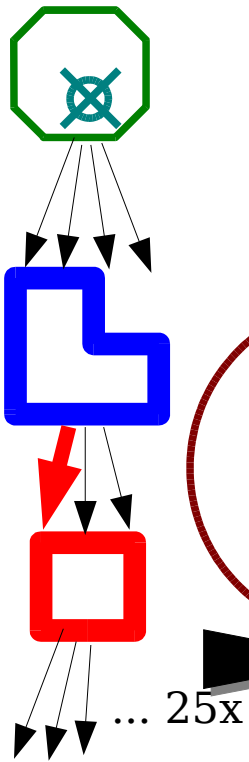
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

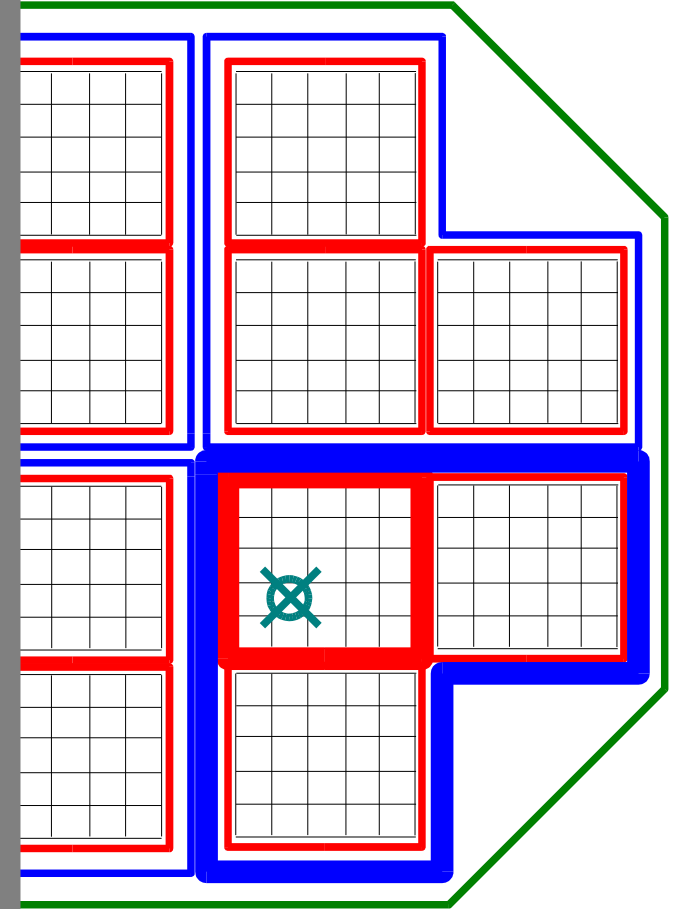
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

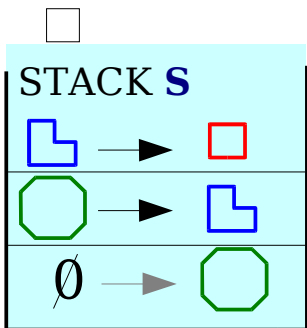
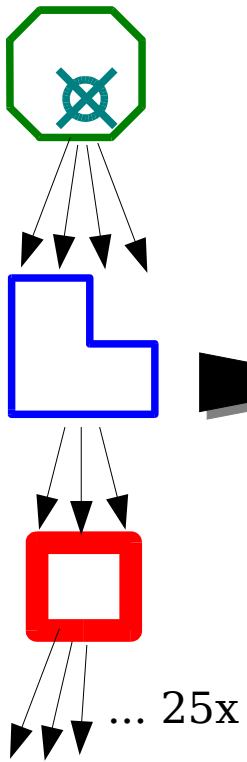
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

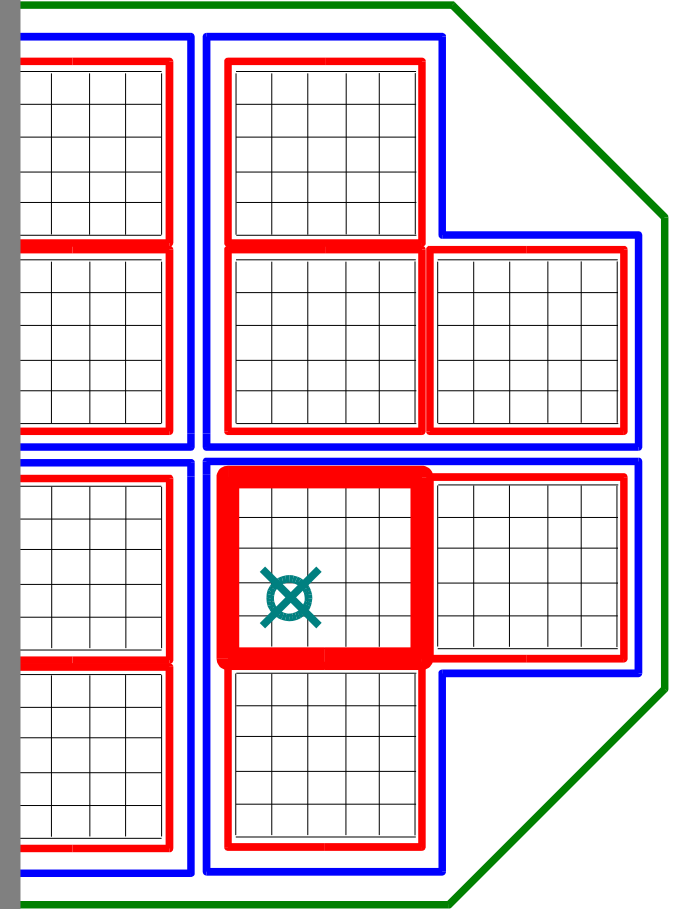
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

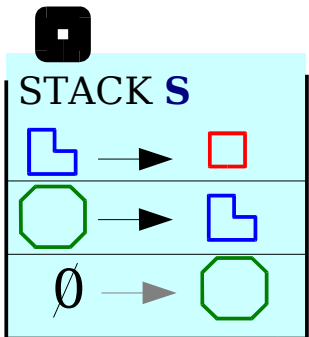
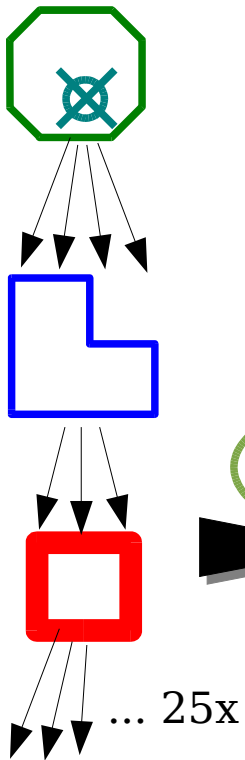
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

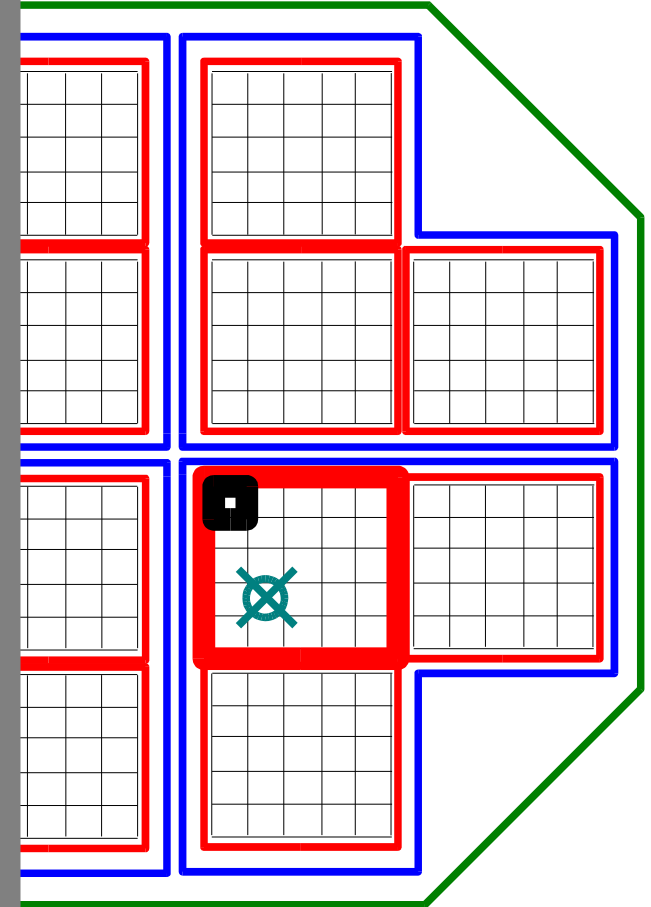
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

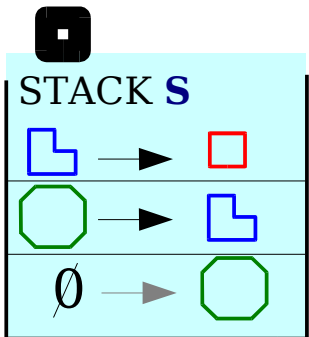
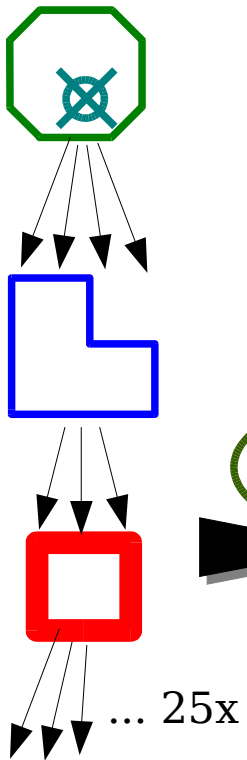
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

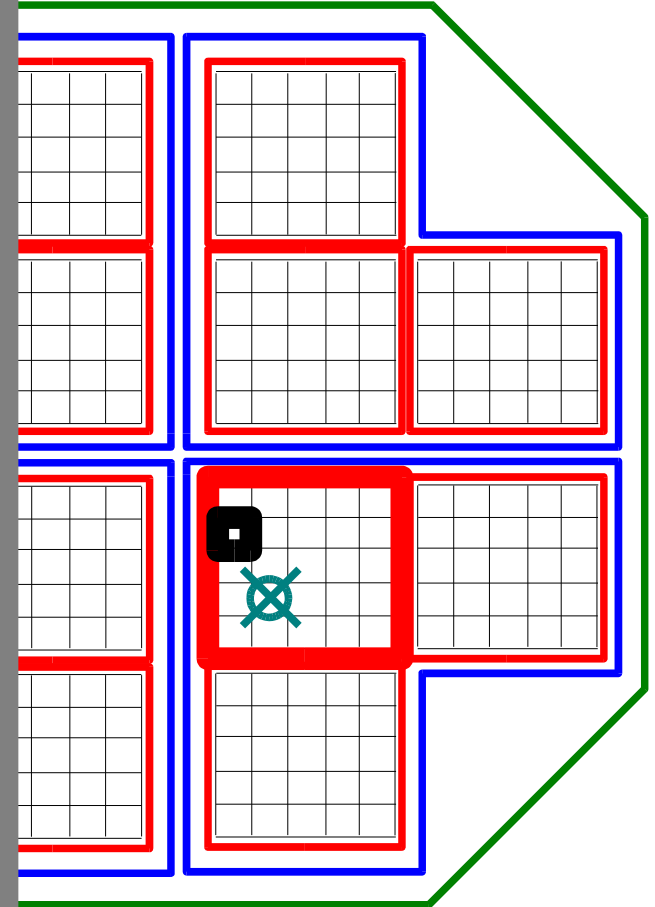
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

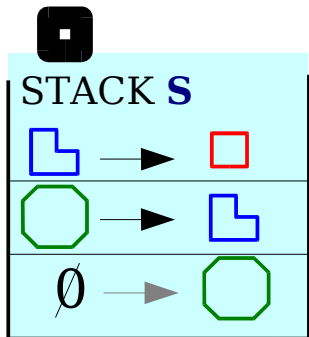
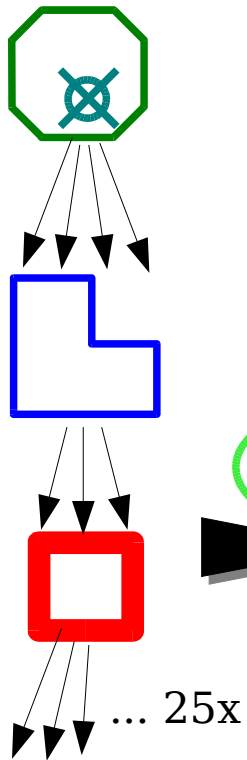
Recursive Algorithm:

```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

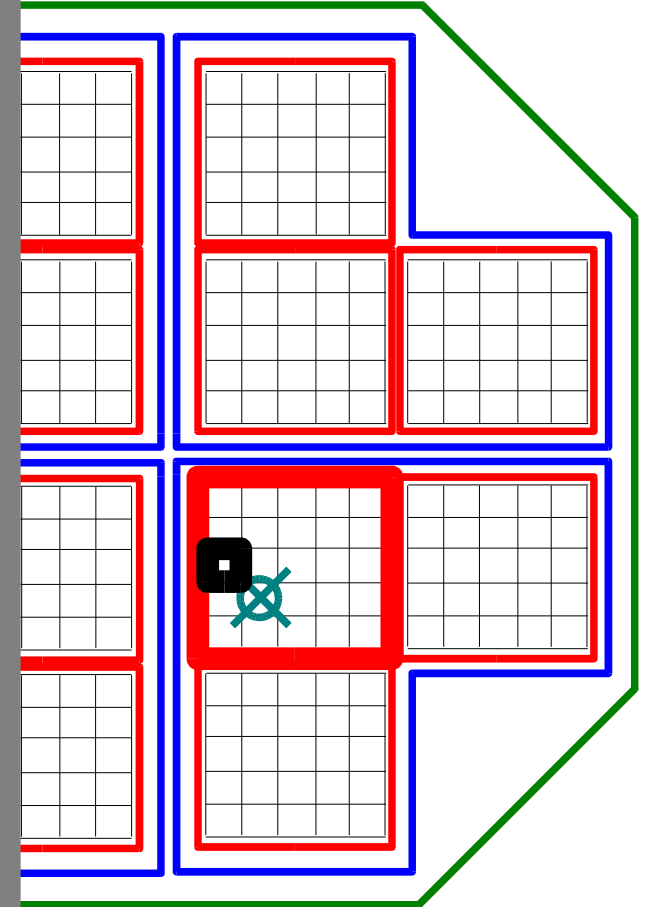
children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}
  
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

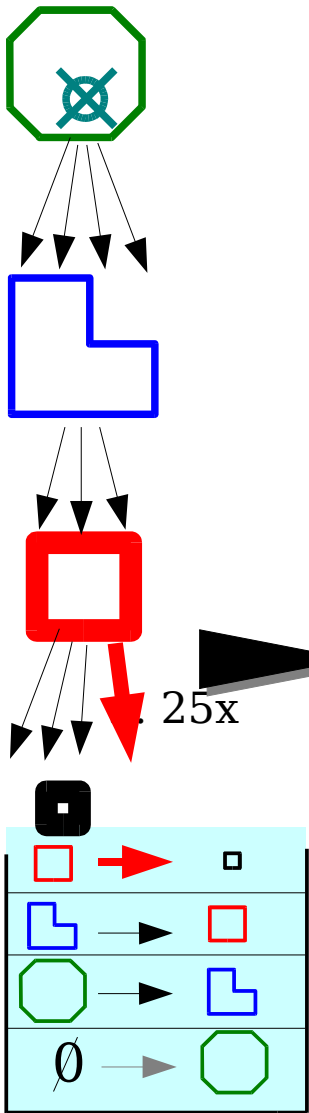
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

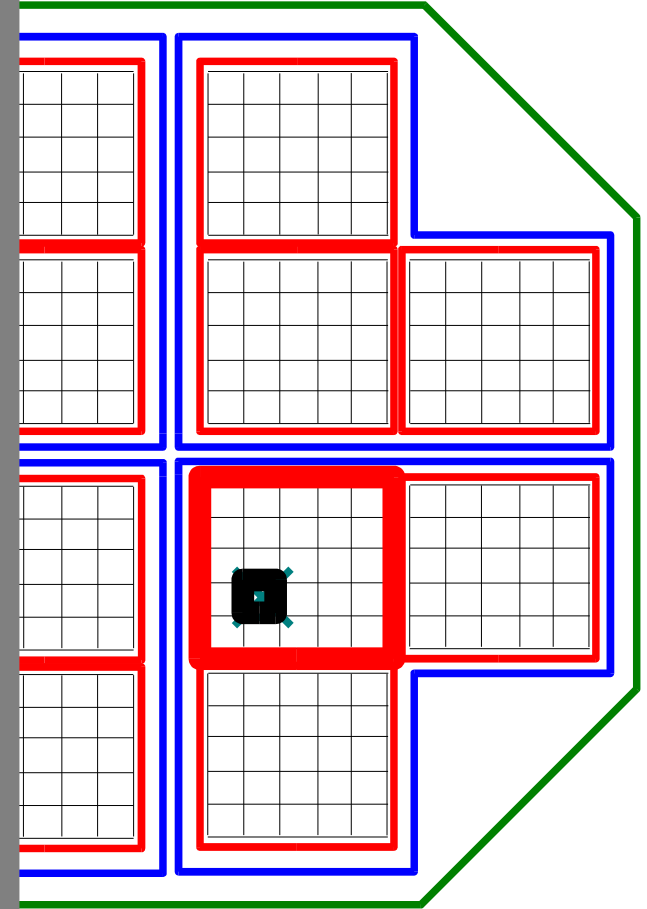
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

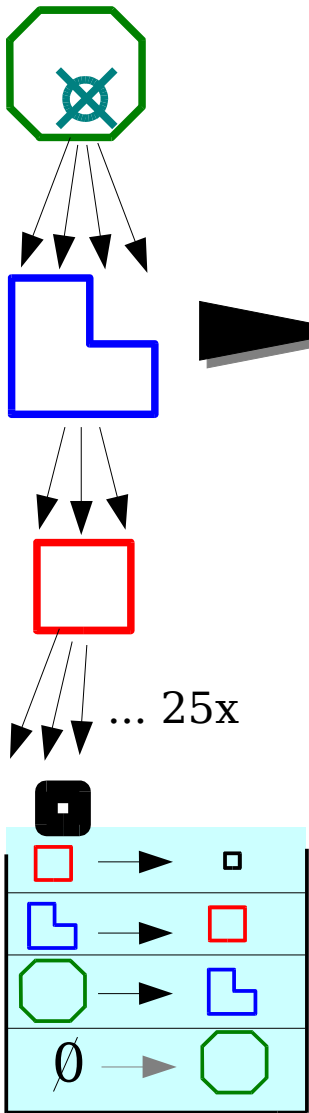
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}

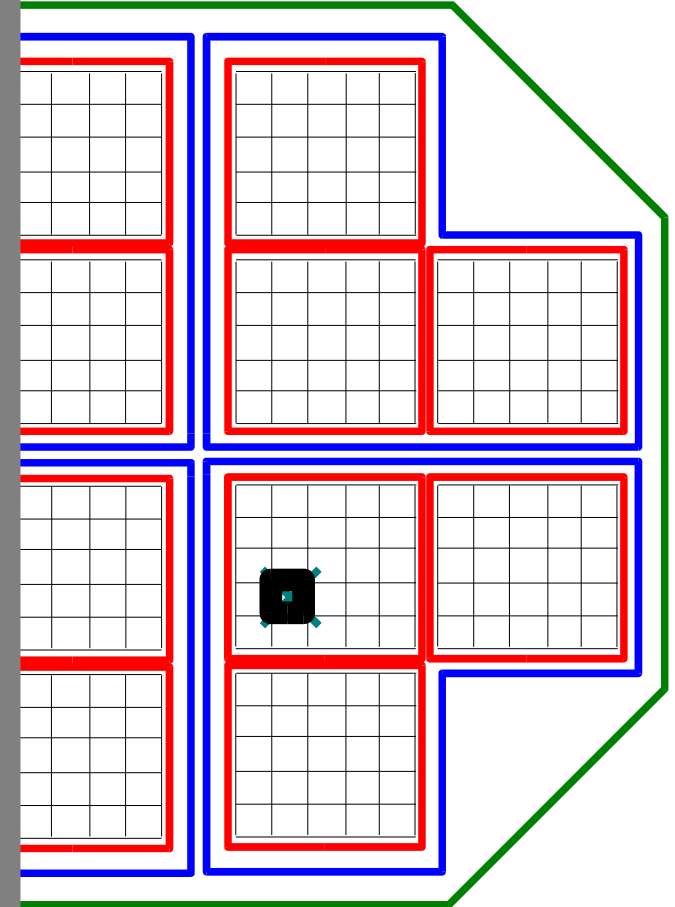
```



G4VPhysicalVolume



G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

Recursive Algorithm:

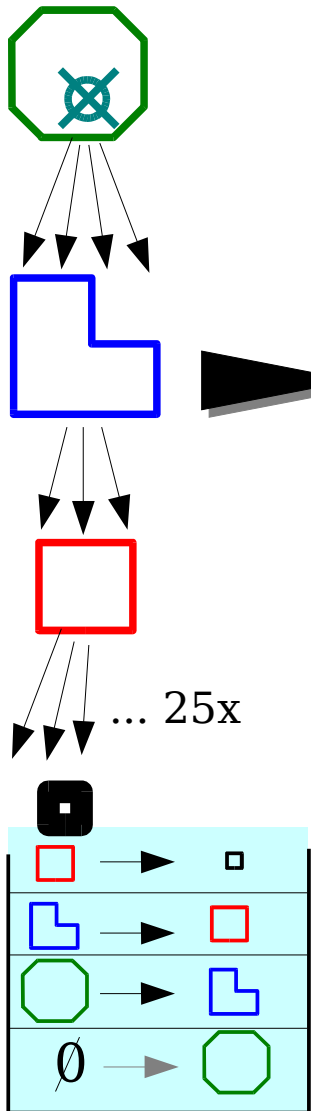
```

STACK S;
pos = (x,y,z);
push(S, world);
descend(world,pos);

descend(vol, pos) {
  if ( children(vol) == 0) return;
  foreach child in children(vol):
    pos' = to_local_frame(child,pos);
    if (child.isInside(pos')) {
      push(S, child);
      descend(child,pos');
    }
}

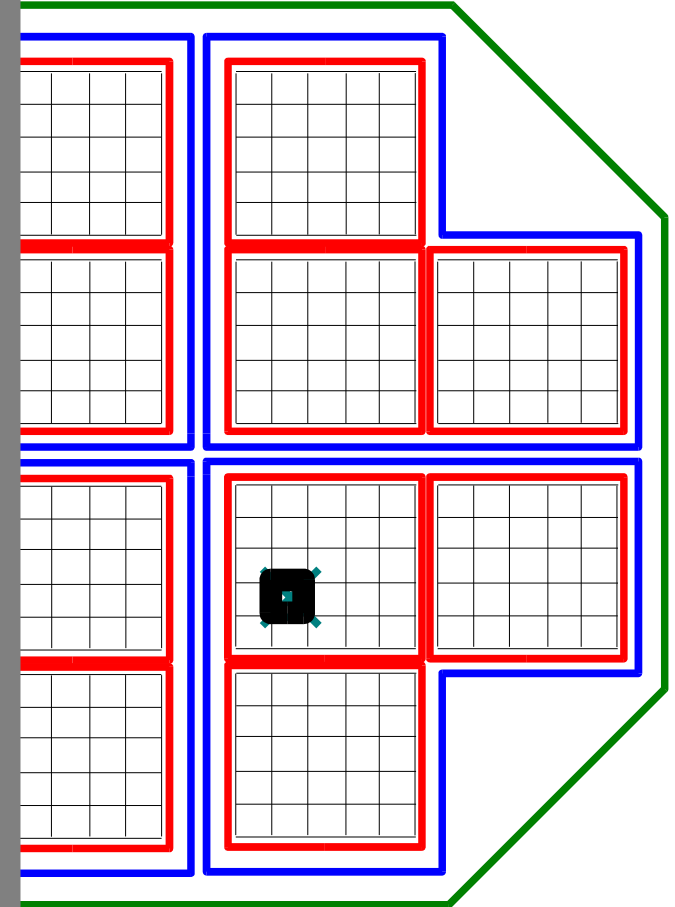
children(physVol) {
  logVol = physVol.myself;
  return logVol.children();
}
    
```

done!



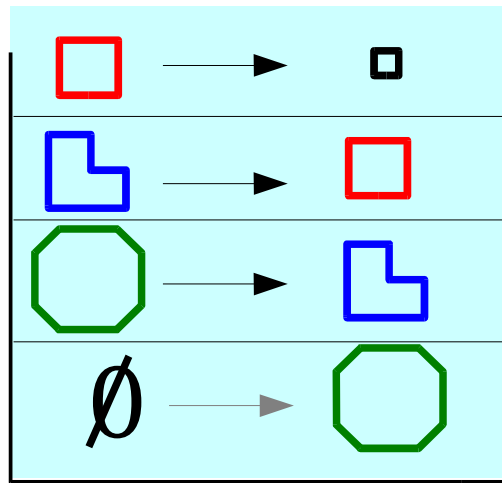
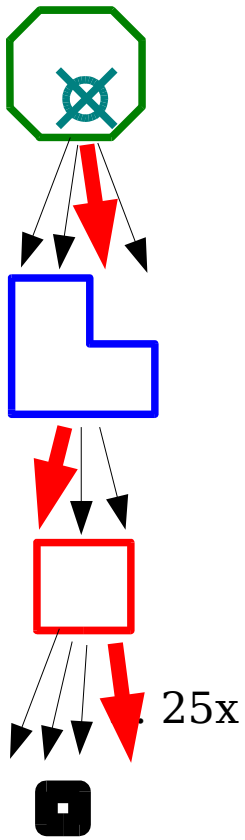
G4VPhysicalVolume

G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

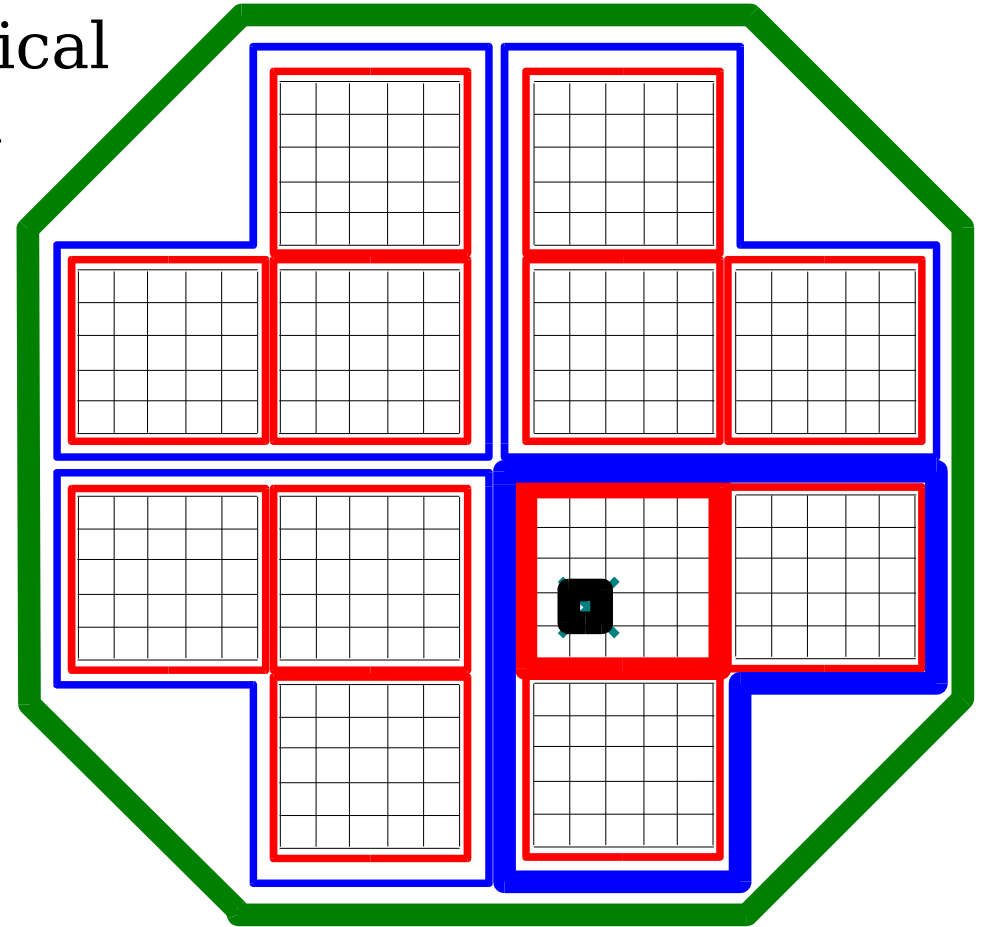
The stack contains the full hierarchy of ancestors including all geometrical transformations to all local frames!



G4VPhysicalVolume

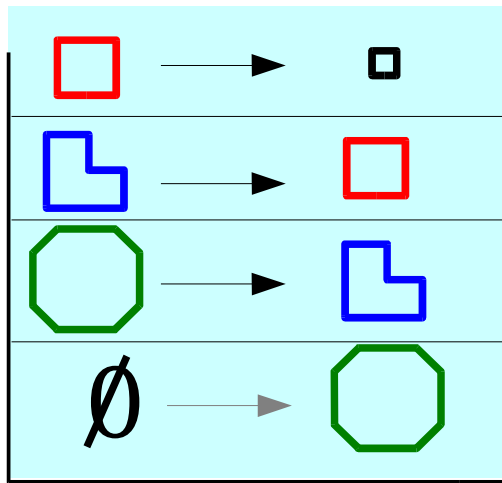
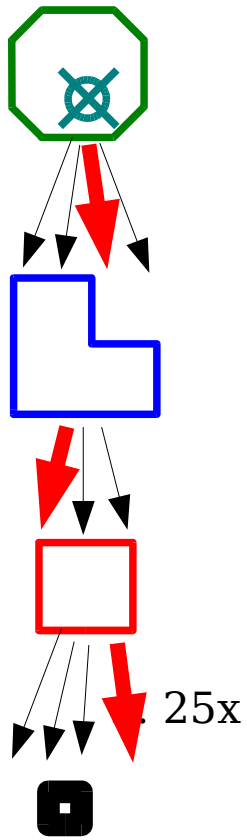


G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



Basics of Navigation

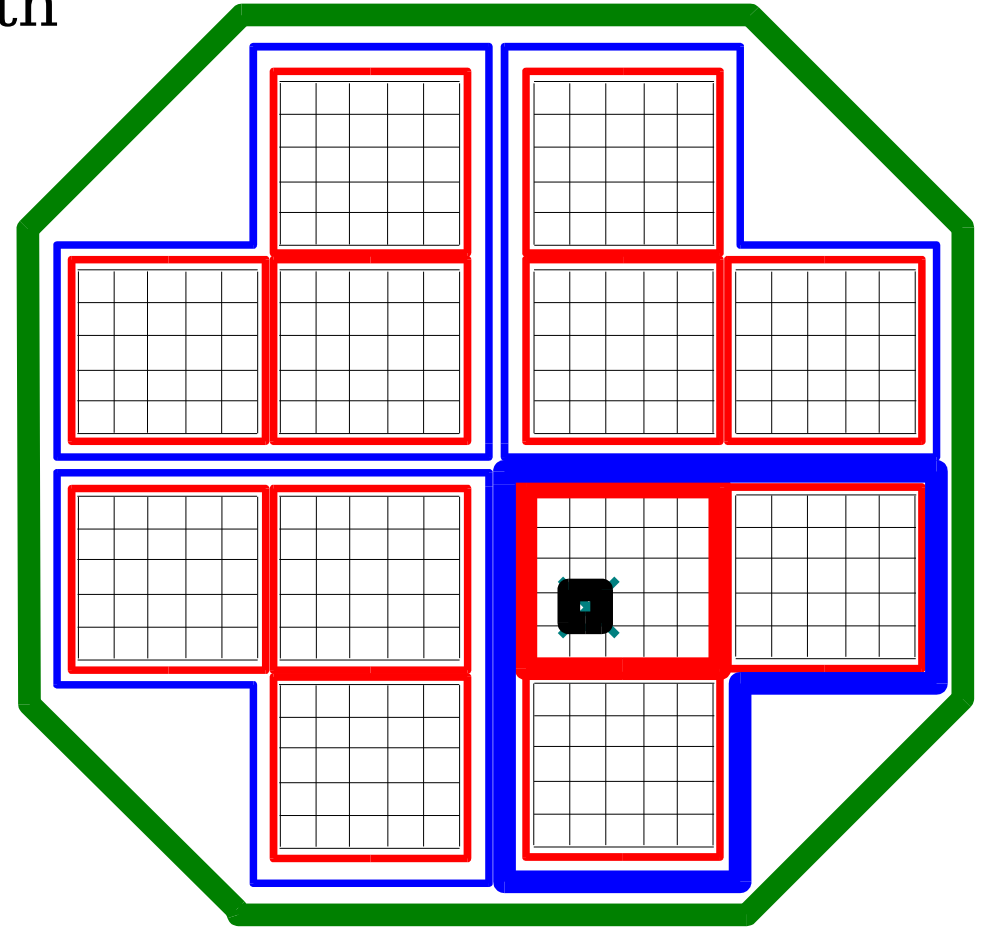
During simulation, G4 will always provide you with this stack. It is your geometrical context!



G4VPhysicalVolume



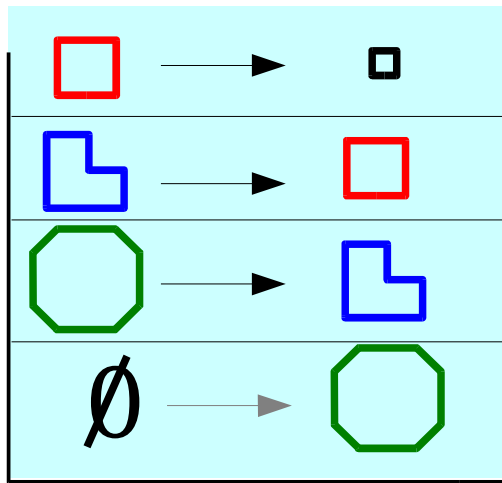
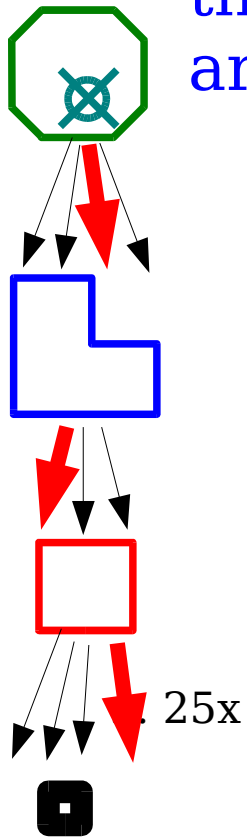
G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



When a particle is tracked through the detector, the stack grows and shrinks as the particle traverses the geometry.

Basics of Navigation

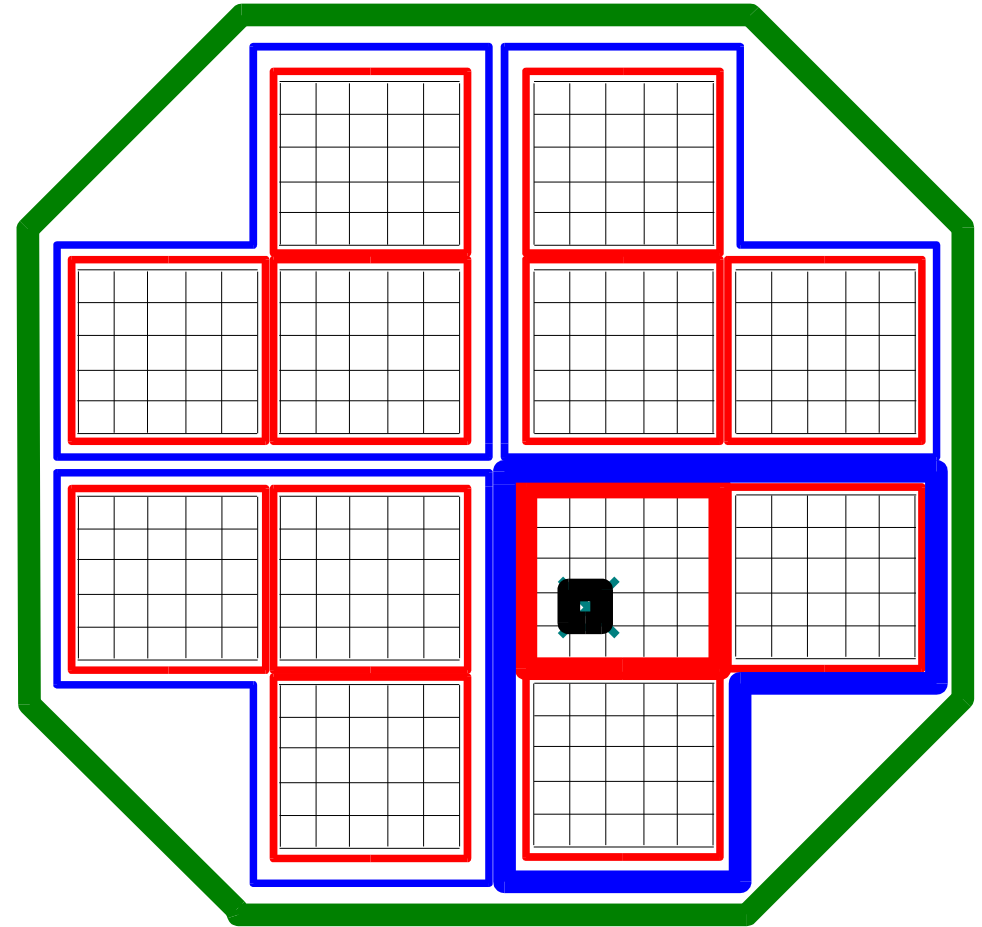
The lookup algorithm assumes that the geometrical constraints are not violated!



G4VPhysicalVolume

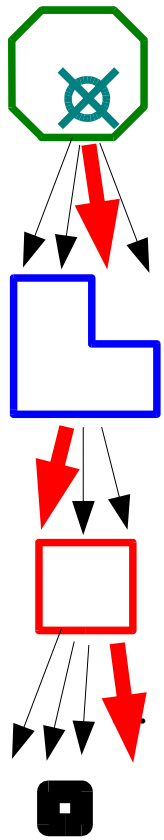


G4LogicalVolume: mother, myself
G4Transform3D: relative orientation



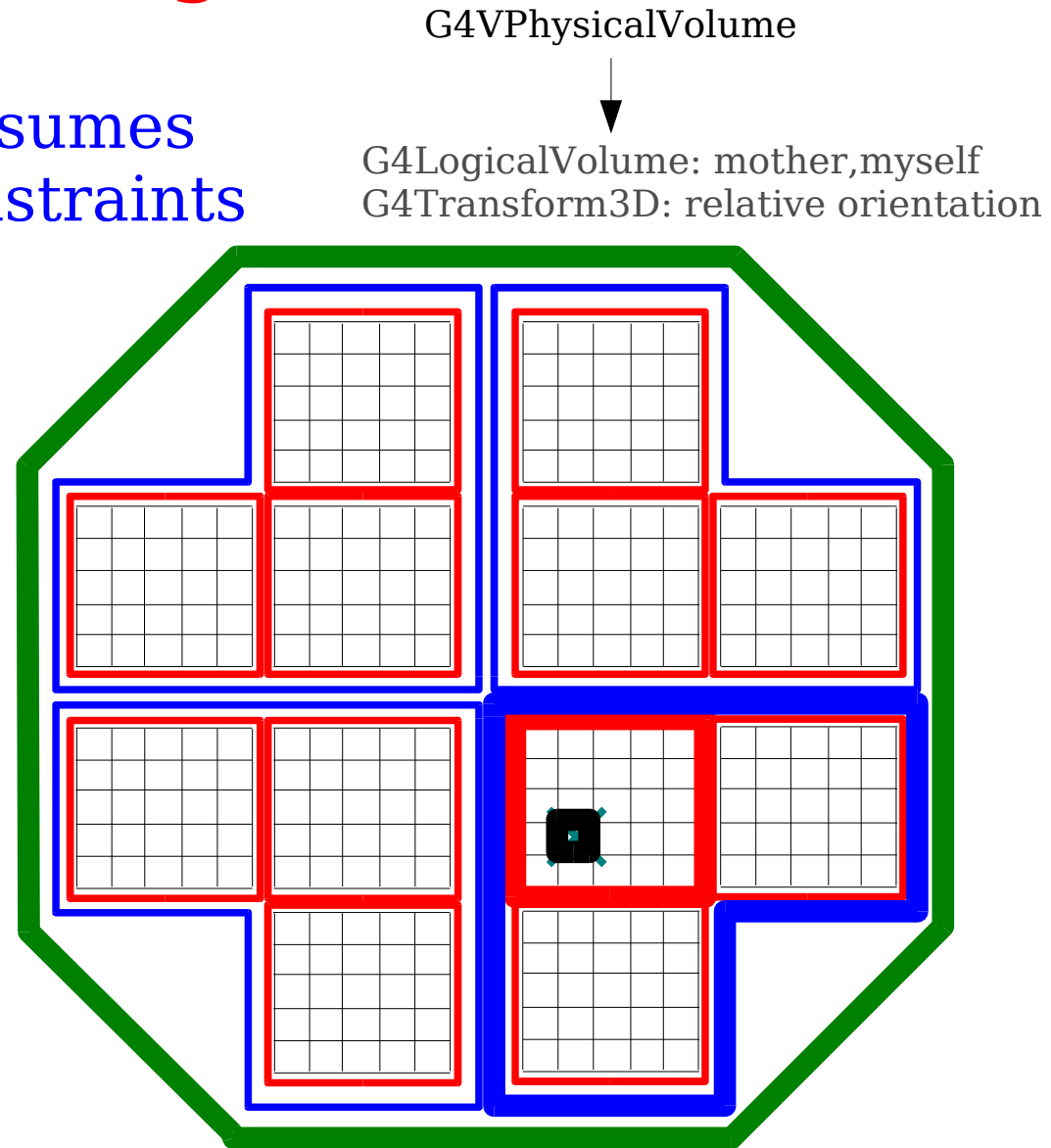
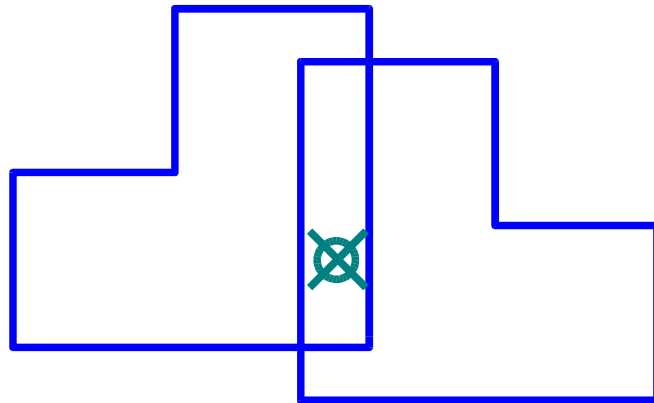
Basics of Navigation

The lookup algorithm assumes that the geometrical constraints are not violated!



Violations -> Ambiguities!

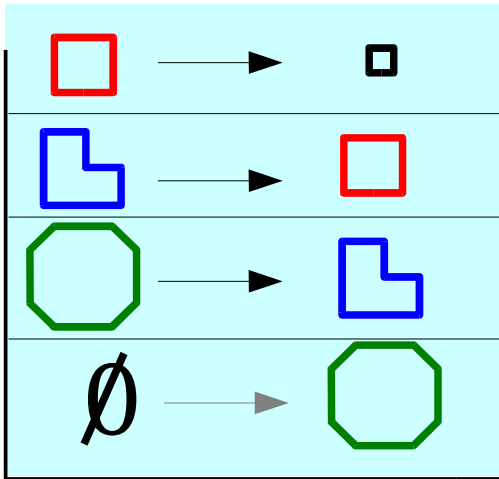
Example:



To which solid does  belong??

Basics of Navigation

0
1
2
3

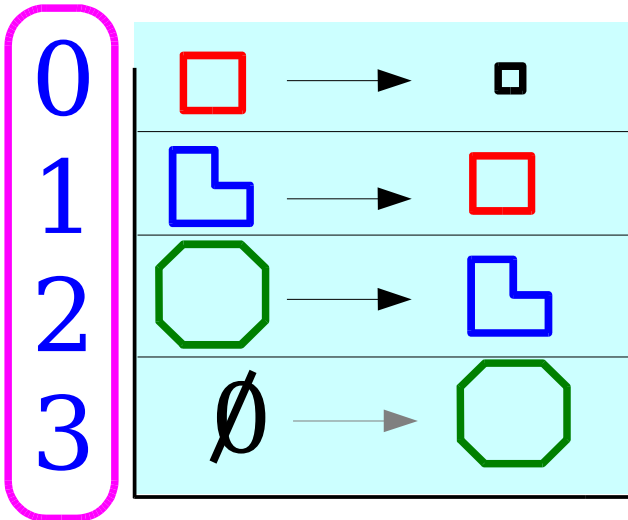


= class **G4TouchableHistory**
: public **G4VTouchable**

will be the geometrical context from which geometrical information can be extracted during simulation.

Name	Class	Type	Parameters
◇ <u>G4VTouchable (md)</u>	<u>G4VTouchable</u>	<u>int</u>	<u>()</u>
◇ ~G4VTouchable (md)	G4VTouchable	virtual int	()
◇ GetHistory (md)	G4VTouchable	virtual const G4NavigationHistory *	()
◇ GetHistoryDepth (md)	G4VTouchable	virtual G4int	()
◇ GetReplicaNumber (md)	G4VTouchable	virtual G4int	(G4int)
◇ GetRotation (md)	G4VTouchable	virtual const G4RotationMatrix *	(G4int)
◇ GetSolid (md)	G4VTouchable	virtual G4VSolid *	(G4int)
◇ GetTranslation (md)	G4VTouchable	virtual const G4ThreeVector &	(G4int)
◇ GetVolume (md)	G4VTouchable	virtual G4VPhysicalVolume *	(G4int)

Basics of Navigation



= class **G4TouchableHistory**
: public **G4VTouchable**

Simulation is performed in this dynamical context;

User interfaces always provide this context (see later)

Name	Class	Type	Parameters
<u>G4VTouchable (md)</u>	<u>G4VTouchable</u>	<u>int</u>	<u>()</u>
~G4VTouchable (md)	G4VTouchable	virtual int	()
GetHistory (md)	G4VTouchable	virtual const G4NavigationHistory *	()
GetHistoryDepth (md)	G4VTouchable	virtual G4int	()
GetReplicaNumber (md)	G4VTouchable	virtual G4int	(G4int)
GetRotation (md)	G4VTouchable	virtual const G4RotationMatrix *	(G4int)
GetSolid (md)	G4VTouchable	virtual G4VSolid *	(G4int)
GetTranslation (md)	G4VTouchable	virtual const G4ThreeVector &	(G4int)
GetVolume (md)	G4VTouchable	virtual G4VPhysicalVolume *	(G4int)

Fine,

now we know how to locate ourselves
in the simulation world ...

Let's see how we can move around!

Tracking

in absence of Physics

“Physics”
=
Geant models of
physical processes

We have seen, how to Geant4 finds out about the geometrical context (G4VTouchable) given a global point in your detector description.

But how are millions of particles tracked through the geometry?

Tracking

in absence of Physics

- Tracking is the process that moves a simulated particle through the detector geometry
- It does all the geometrical calculations and provides the user (=you) with up-to-date information of the state of a particle during tracking
 - volume hierarchy stack, absolute & relative positions
 - kinematic properties of the particle (time, direction, ...)
- Tracking solves the equations of motion of a **pointlike** (=structureless), **non-interacting particle** in the presence of
 - no external fields: (Galileo, Newton, Einstein)
 - Every object in a state of **uniform motion** tends to remain in that state of motion unless an external force is applied to it.
 - external fields: (Maxwell, Einstein)
 - purely magnetic: energy conserved
 - mixed electro-magnetic: energy not conserved
- From the Geant4 point of view, Tracking is a (mandatory) physics

Tracking

in absence of Physics

- Tracking the de
- It does (=you tracking
- vol
- kin
- Tracking (=stru
- no
-
- ext
-
-

Input for Tracking:

Startpoint	\mathbf{x}
Momentum	\mathbf{p}
Rest-mass	m
Electrical charge	q
Time	t
Fields	$\mathbf{E}(\mathbf{r},t), \mathbf{B}(\mathbf{r},t)$
Geometry	world -> ...

$$E^2 = m^2c^4 + \mathbf{p}^2c^2$$

ough

er during

e of

in that

- From the Geant4 point of view, Tracking is a (mandatory) physics process!!

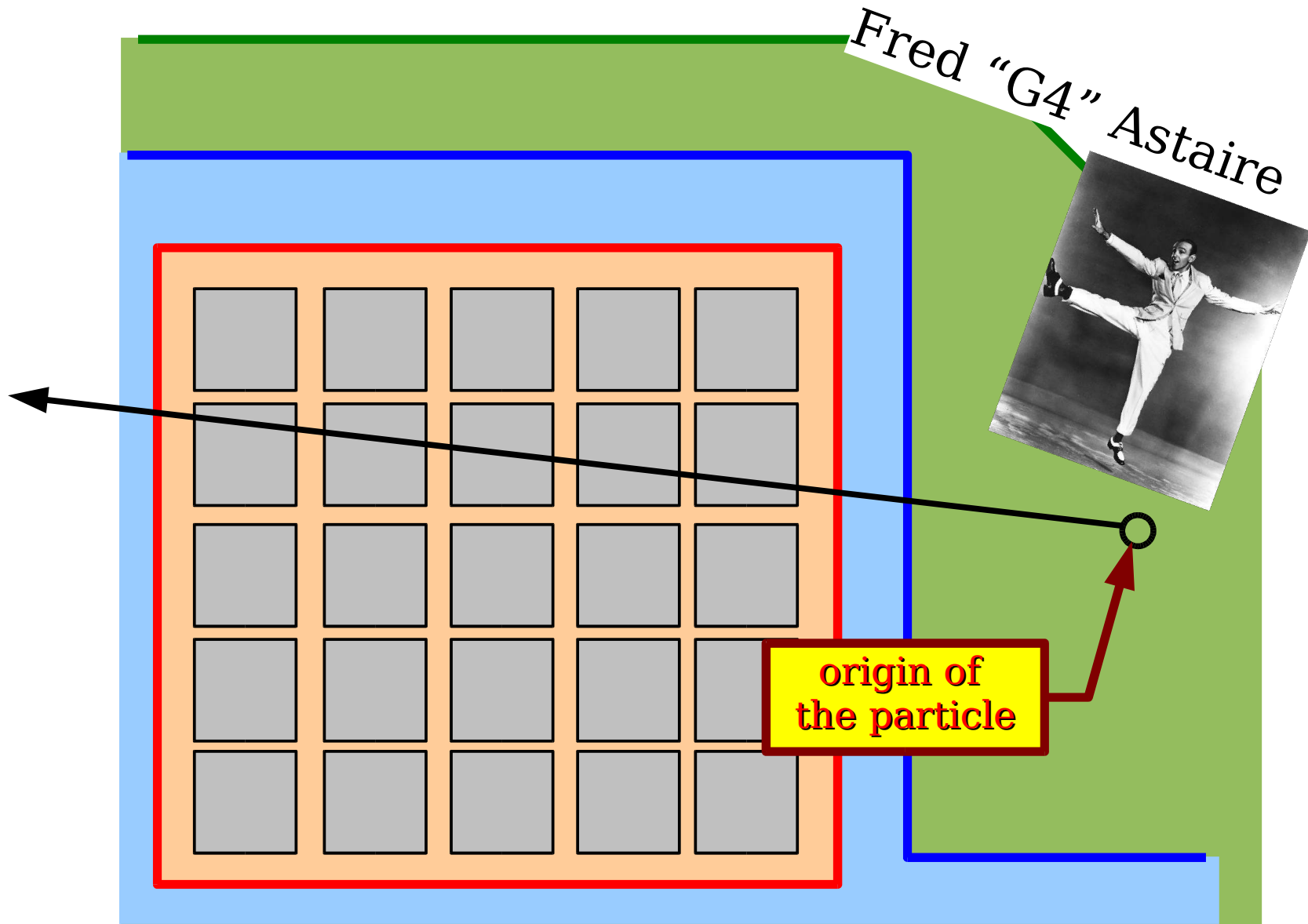
Tracking

in absence of Physics

- One particle is tracked at a time; the others are waiting on a stack
- The G4 tracking system performs a kind of “ray-tracing”
- A ray is called **trajectory**.
 - Geometrically, it is simply a line representing the path of the particle starting at the origin of the particle and ending when crossing the boundary of the world volume
- In Geant4, the trajectory consists of a concatenation of **steps**
- A **step** is delimited by two **step-points**, each on the surface of a solid
- The part of the trajectory belonging to one step does not intersect any surfaces of any solid

no external fields:

The Geant4 Stepdance



no external fields:

G4 classes:

G4Step

G4StepPoint

G4VTouchable

G4StepPoint *

G4Step::GetPreStepPoint();

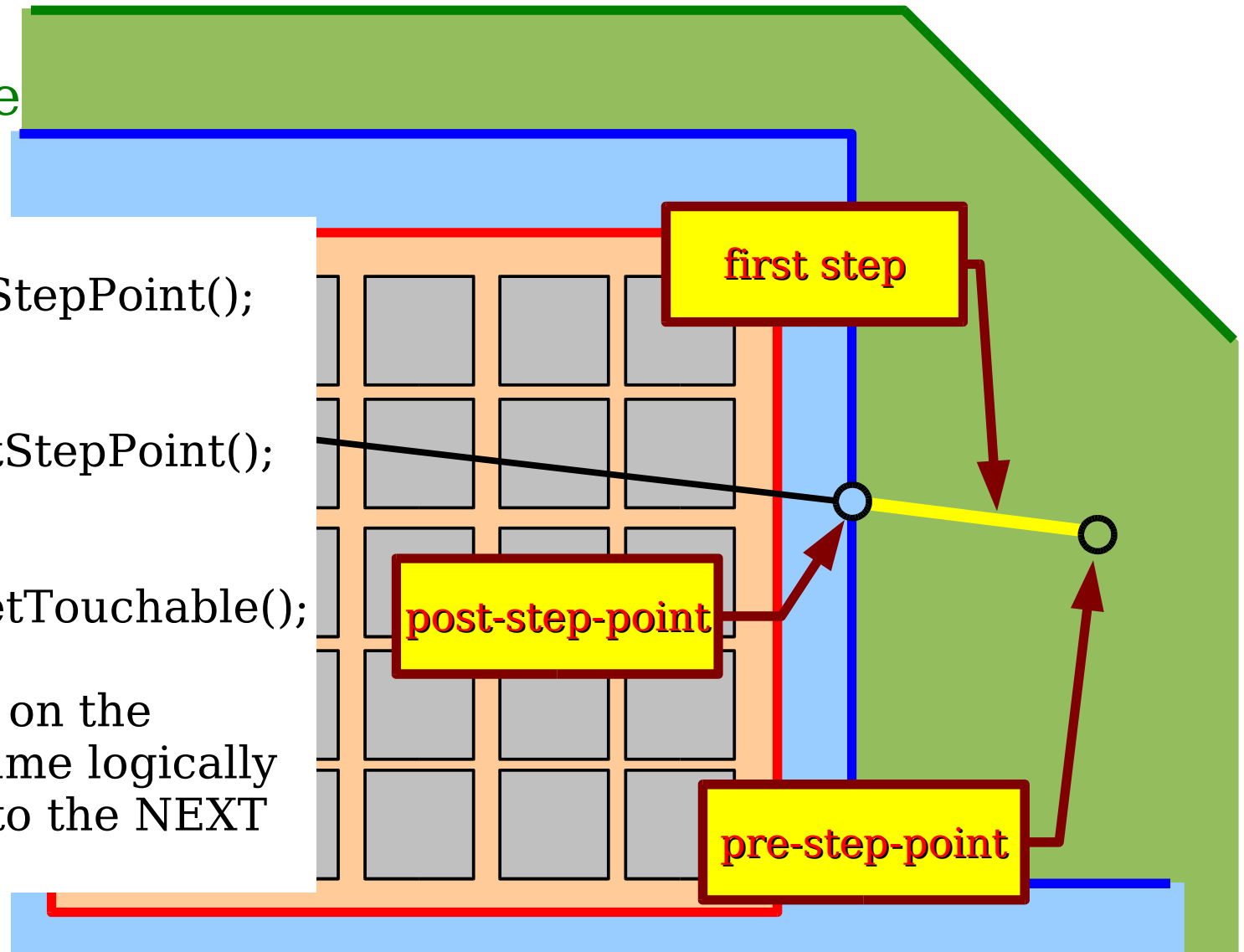
G4StepPoint *

G4Step::GetPostStepPoint();

G4VTouchable *

G4StepPoint::GetTouchable();

A PostStepPoint on the surface of a volume logically always belongs to the NEXT volume.



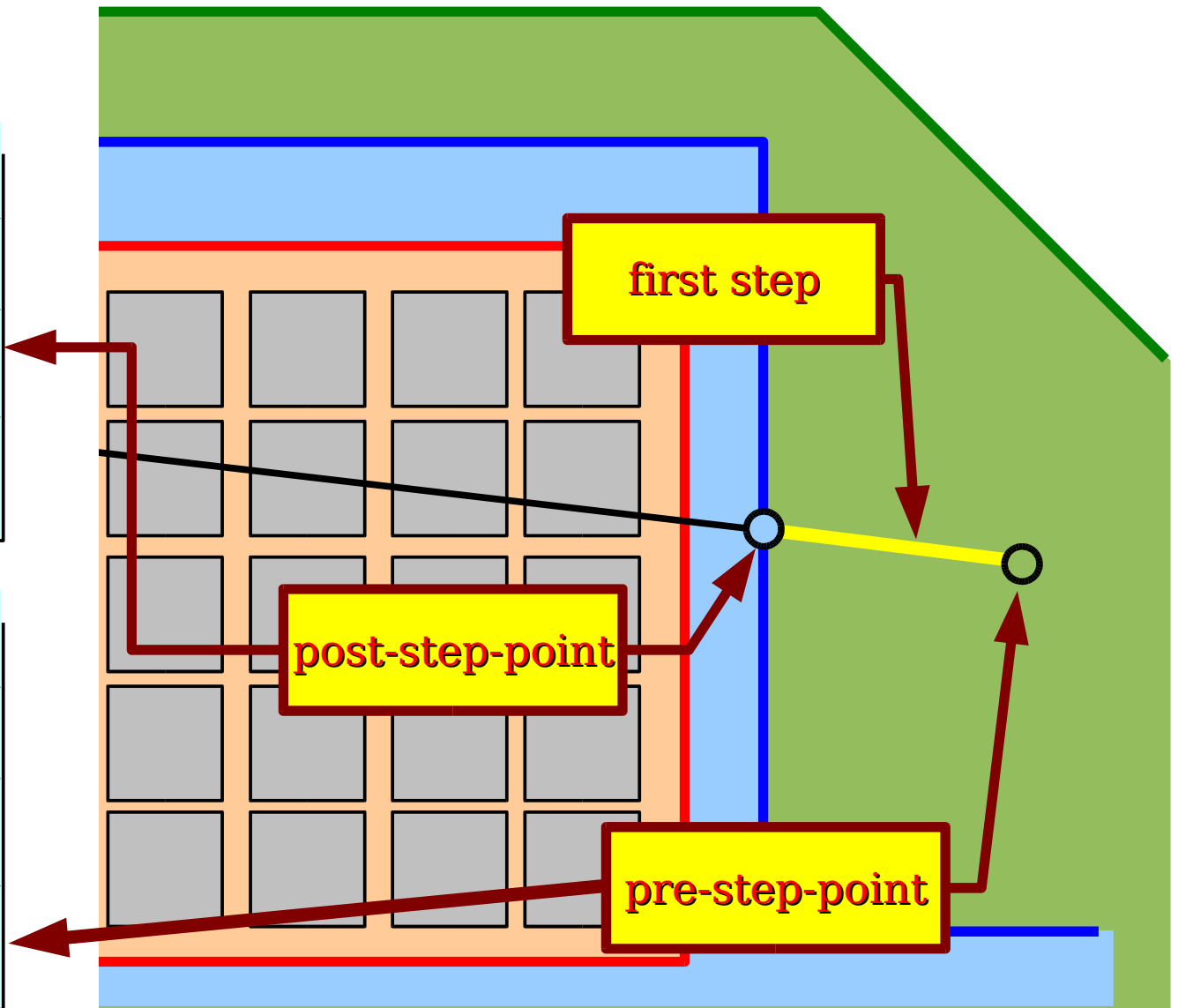
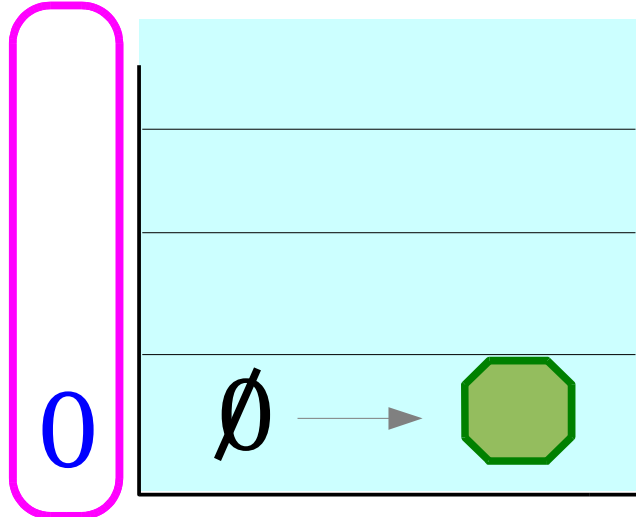
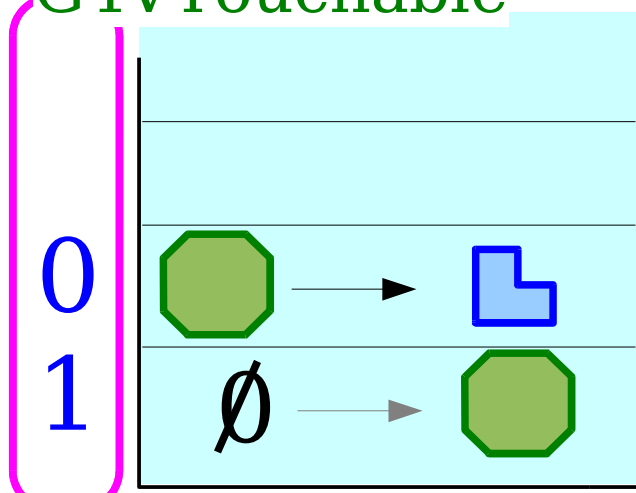
no external fields:

G4 classes:

G4Step

G4StepPoint

G4Touchable



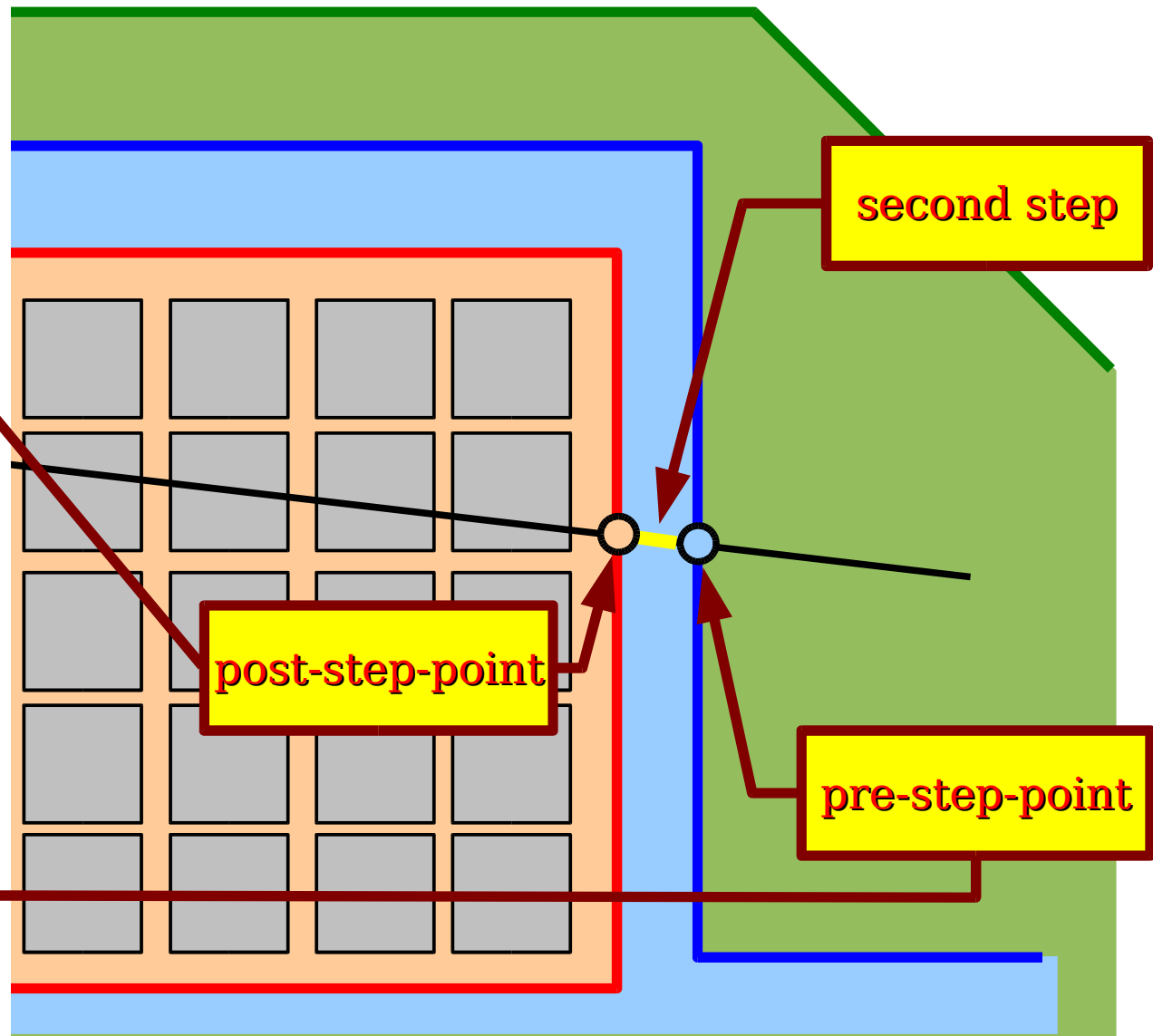
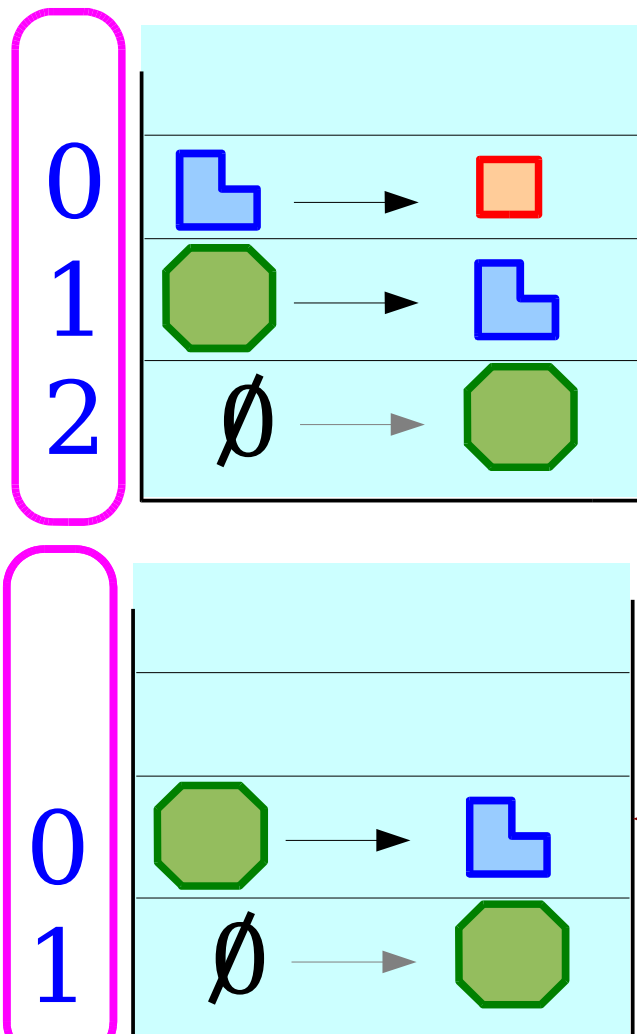
no external fields:

G4 classes:

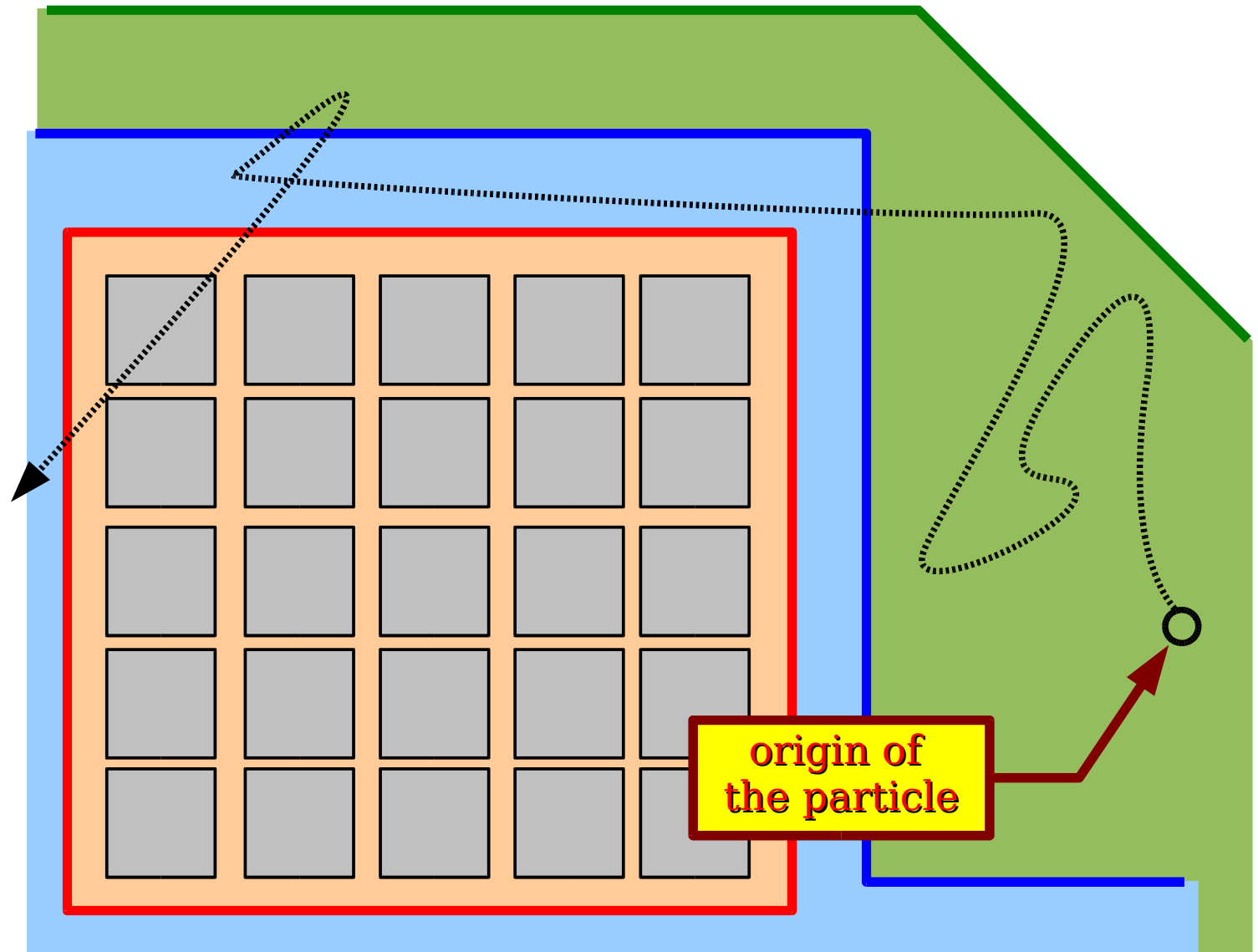
G4Step

G4StepPoint

G4VTouchable

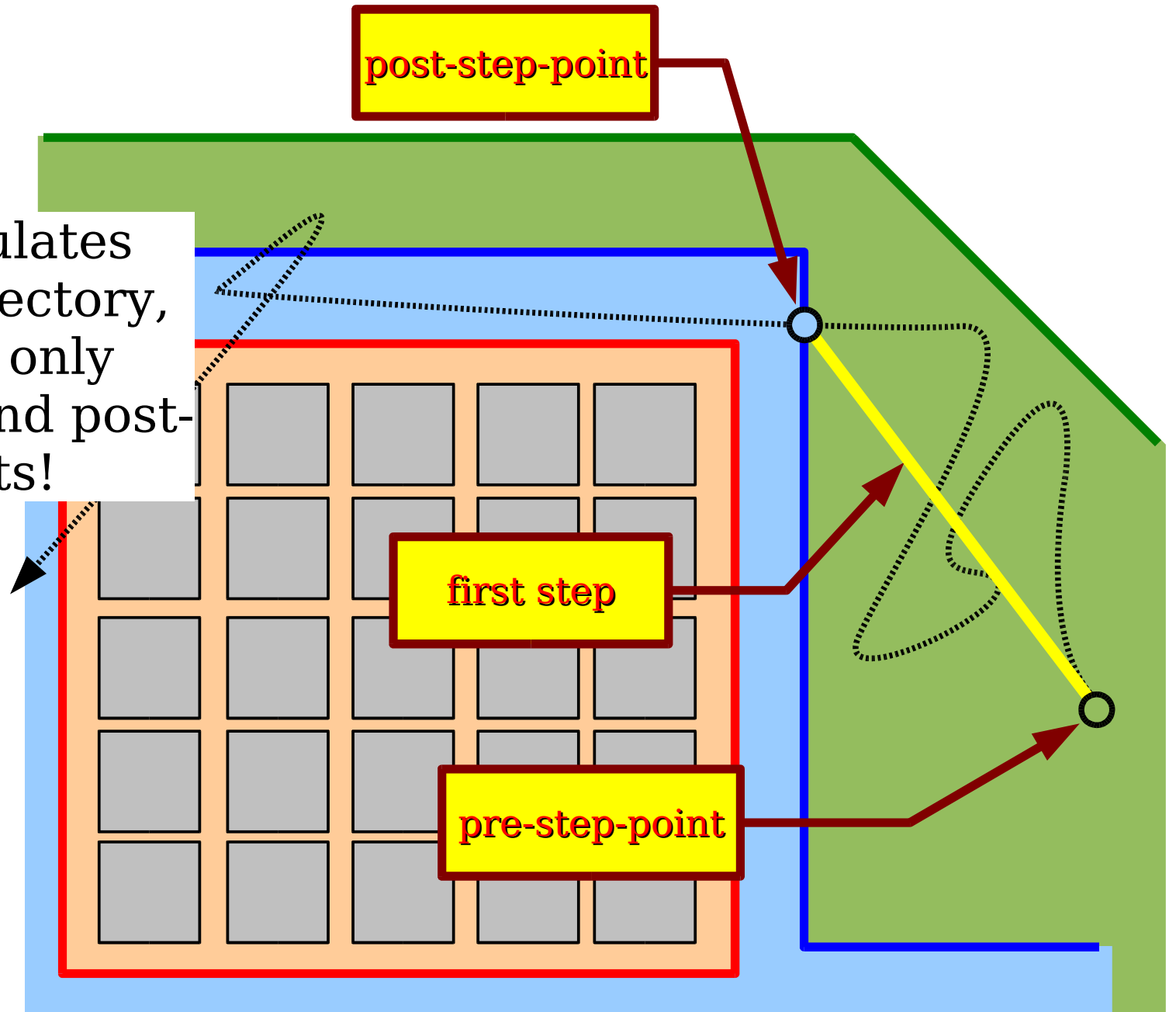


with external fields:
(a quite crazy field in this example ...)



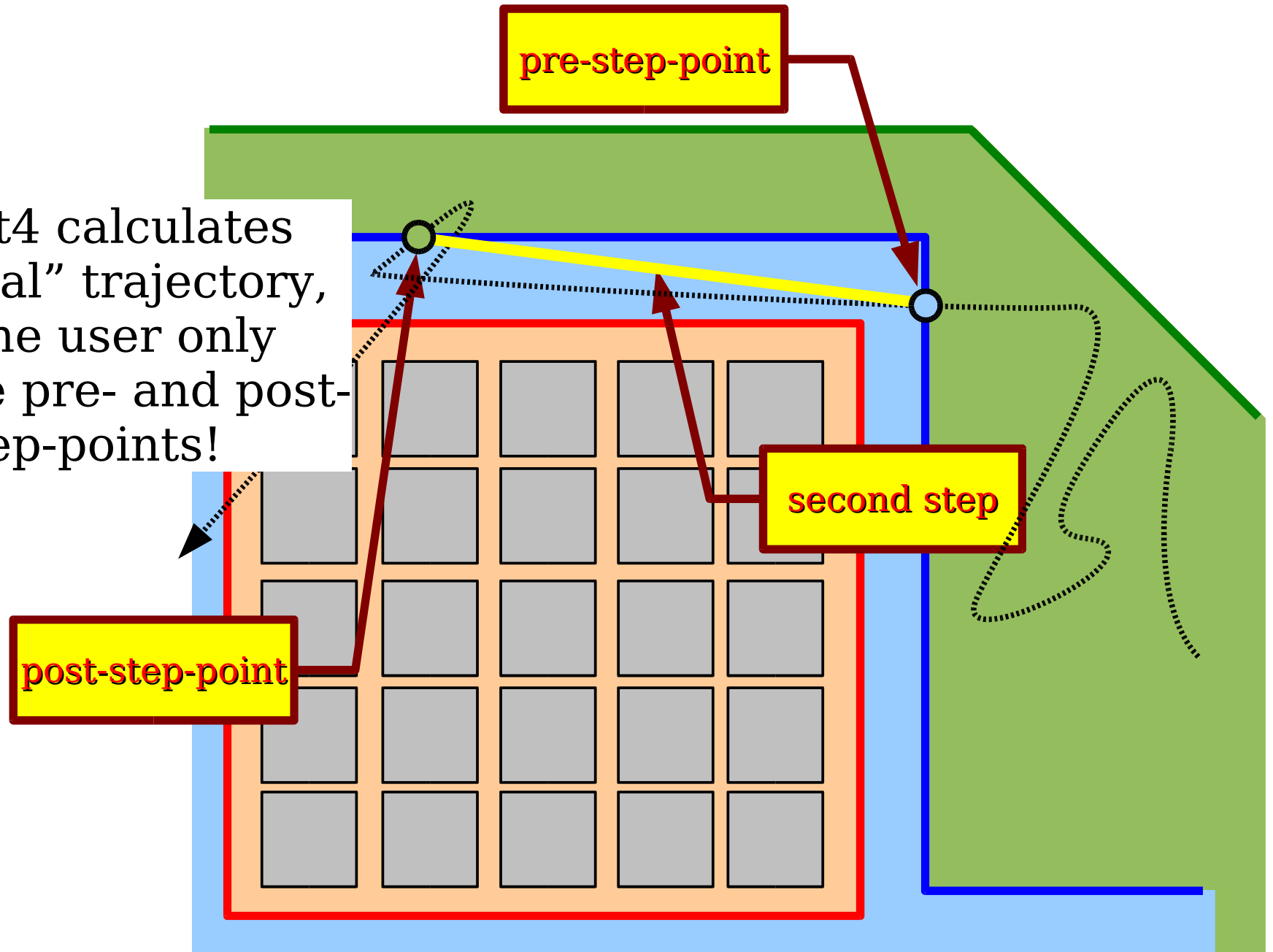
with external fields:

Geant4 calculates the “real” trajectory, but the user only sees the pre- and post-step-points!

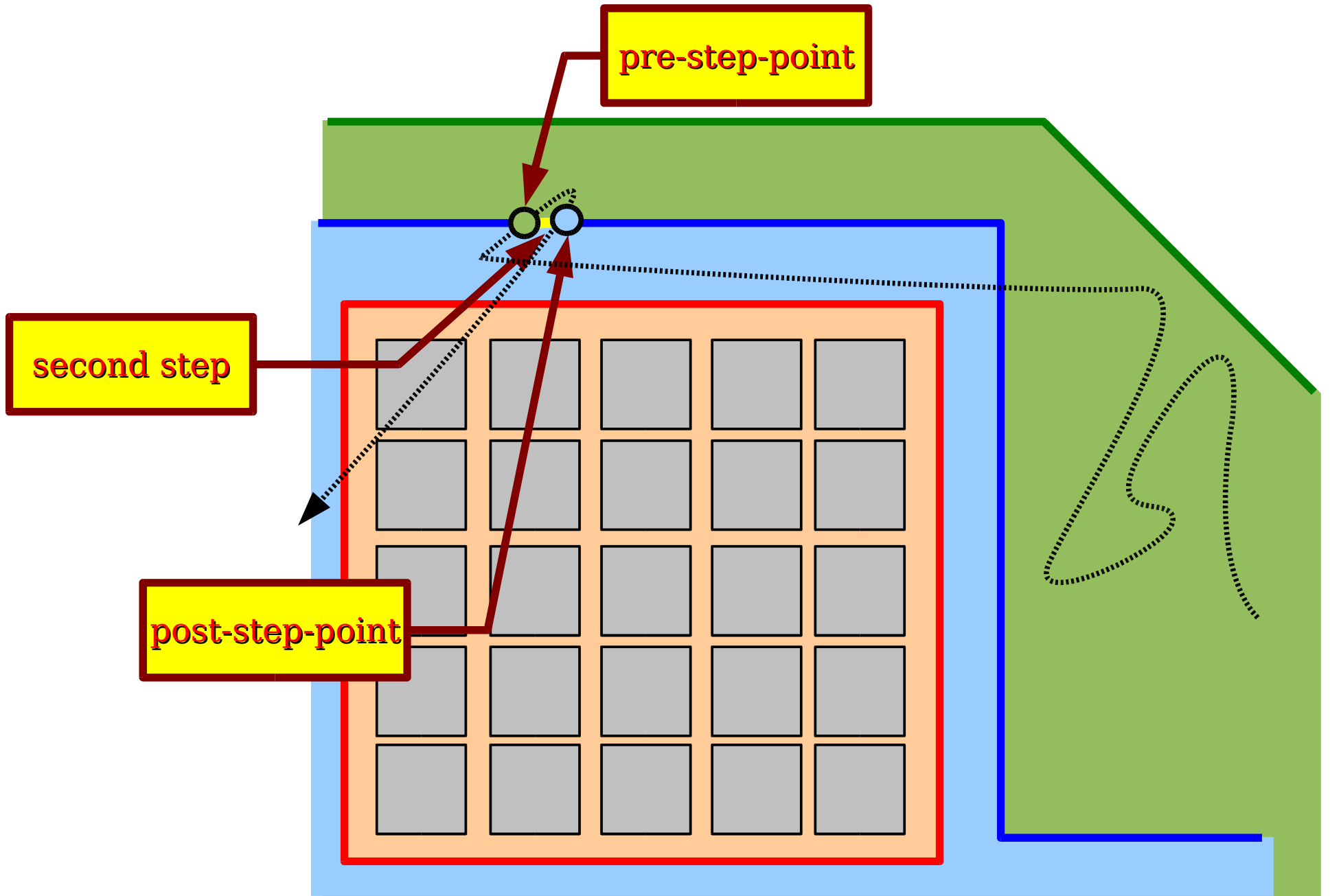


with external fields:

Geant4 calculates the “real” trajectory, but the user only sees the pre- and post-step-points!

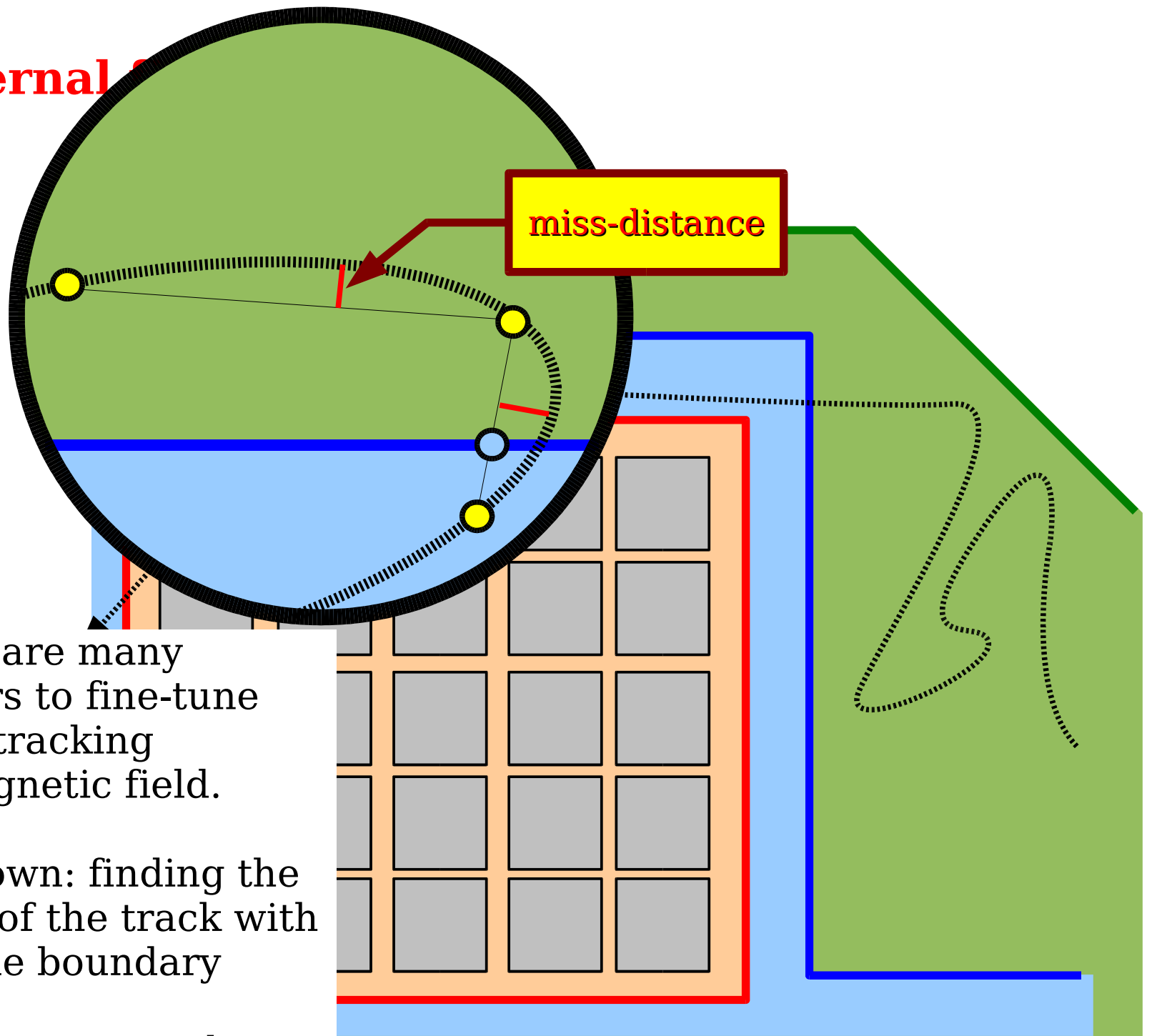


with external fields:



with external

zoom!



There are many parameters to fine-tune the tracking in a magnetic field.

Example shown: finding the intersection of the track with a volume boundary

accuracy vs. speed

External Fields

- To solve the equations of motions of a charged particle, the external field has to be known.
- In Geant4, external fields have to be described independently of anything else!!!
 - Materials and shapes are not “aware” of the Maxwell equations, i.e. they don't adjust the field values ...
 - G4 does not do any field calculations for you, i.e. given field sources, the field is not calculated depending on the geometry
- In HEP detectors: **static magnetic fields**
 - complex detector structures with magnetizable materials
 - complex fields, usually calculated with special field calculation software
 - **problem of “synchronization” of geometries ...**

Field Implementation

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) }
*(field+4) } electric
*(field+5) } component



Field Implementation - Granularity

- In addition to a single, global field, one can define a field per G4LogicalVolume.
- Co-ordinates are then measured in the frame of the corresponding solid.
- The field is (recursively) valid for all daughters of the G4LogicalVolume, unless a G4LogicalVolume of a daughter has its own field ...
- Symmetries of the field can thus be exploited
- The global field, if provided, is the default field.

Wrap Up, Wake Up!

- **What do we have up to now:**
 - Geometry (volumes with shapes, material, hierarchically placed)
 - Navigation of point-like, non-interacting particles
 - External fields
- **What we still don't know:**
 - Why randomness (Monte Carlo Method)
 - Influence on particle tracks by physics
 - User's duties (physics selection, primaries, ...)
 - Obtaining simulation results!
 - Managing data ...

PART IV

The Monte Carlo Method (a free historic sightseeing tour)

“People of zee wurl, relax!”

the infamous parrot “Sailor” in
Tom Robbins “Fierce Invalids Home From Hot Climates”

Relax: Some slides about **The Monte Carlo Method**



- The law of particle physics are based on quantum mechanics
 - Non deterministic in the sense that one can “only” calculate the probabilities of certain things to happen
 - Results are valid not for a single particle/interaction, but for an (statistical) ensemble
- An experimental measurement can be regarded as drawing a sample from the ensemble of possible outcomes (both, in reality and in simulation)
- When simulating the traversal of single particles through matter we have to account for the probabilistic nature of our assumptions:
 - We have to use random numbers to draw samples from distributions!!!
 - We use **the MONTE CARLO method!**

Monte Carlo

This wonderful method is indeed **named after:**



Ulam, Stanislaw
(1909-1986)



Metropolis, Nicholas Constantine
(1915-1999)

Experiment Simulation (2)

Metropolis Nicholas and Stanislaw
Ulam (1949).

The Monte Carlo method,
Journal of the American Statistical
Association,
44 (247), 335-341.



Neumann, John von
(1903-1957)

Monte Carlo



Ulam, Stanislaw
(1909-1986)

Statistical problems were pushing the construction of the first computers!!



Metropolis Nicholas and Stanislaw Ulam (1949).

The Monte Carlo method,
Journal of the American Statistical Association,
44 (247), 335-341.



Metropolis, Nicholas Constantine
(1915-1999)

Experiment Simulation (2)

ENIAC

Electronic Numeric Integrator and Calculator



Neumann, John von
(1903-1957)

Monte Carlo



Ulam, Stanislaw
(1909-1986)



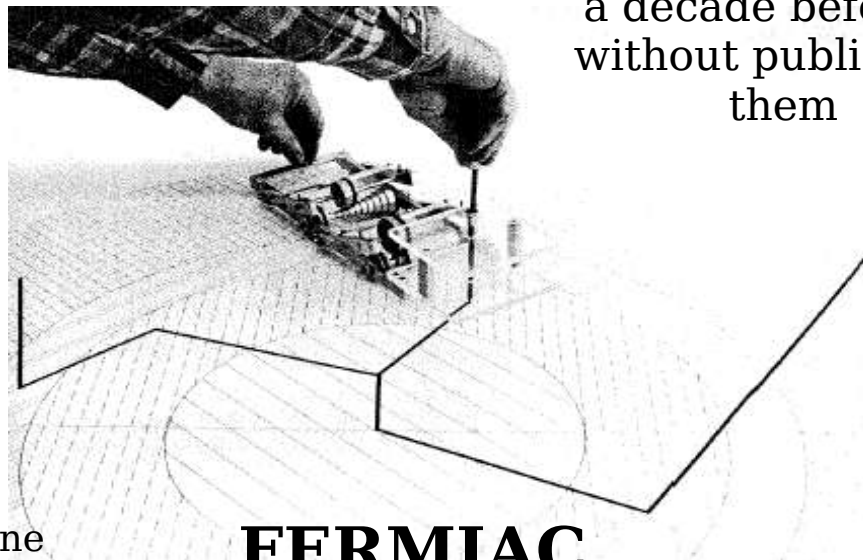
Fermi, Enrico
(1901-1954)



Fermi used
MC methods
without computers
a decade before ...
without publishing
them



Metropolis, Nicholas Constantine
(1915-1999)



FERMIAC



Neumann, John von
(1903-1957)

Monte Carlo



Ulam, Stanislaw
(1909-1986)



Metropolis, Nicholas Constantine
(1915-1999)

Experiment Simulation (2)

Anyone who
considers
arithmetic
methods
of
producing
random digits
is, of course, in a
state of
sin.



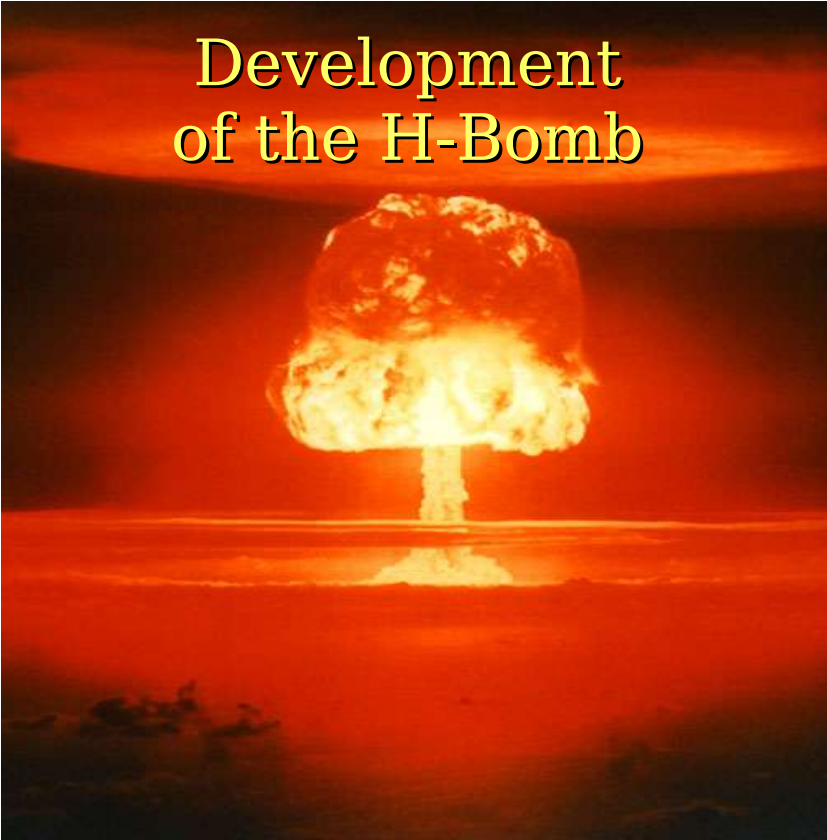
Neumann, John von
(1903-1957)

Monte Carlo

This wonderful method was indeed successfully applied during:



Ulam, Stanislaw
(1909-1986)



sin.



Neumann, John von
(1903-1957)



Metropolis, Nicholas Constantine
(1915-1999)

Monte Carlo

Letter: von Neuman to Ulam, May 1947

I am very glad that preparations for the random numbers work are to begin soon. In this connection, I would like to mention this: Assume that you have several random number distributions, each equidistributed in $0, 1$: $(x^i), (y^i), (z^i), \dots$. Assume that you want one with the distribution function (density) $f(\xi) d\xi : (\xi^i)$. One way to form it is to form the cumulative distribution function: $g(\xi) = \int_0^\xi f(\xi) d\xi$ to invert it $h(x) = \xi \Leftrightarrow x = g(\xi)$, and to form $\xi^i = h(x^i)$ with this $h(x)$, or some approximant polynomial. This is, as I see, the method that you have in mind.

An alternative, which works if ξ and all values of $f(\xi)$ lie in $0, 1$, is this: Scan pairs x^i, y^i and use or reject x^i, y^i according to whether $y^i \leq f(x^i)$ or not. In the first case, put $\xi^i = x^i$ in the second case form no ξ^i at that step.

- Two (nowadays standard) methods of generating random numbers of arbitrary distributions from $(0,1)$ -uniformly distributed random numbers:
- sampling from the inverse cumulative distribution function
 - rejection method using the density function

A slight generalization of b) is applied for multidimensional numerical integration!

Drawing Random Numbers

Solve only this “random” problem:

y ... random variable uniformly distributed on $[0,1]$

x ... random variable; $f(x)$... probability distribution of x

$F(Z) = \int_{-\infty}^Z f(x) \dots$ cumulative distribution function (prob. that $x \leq Z$)

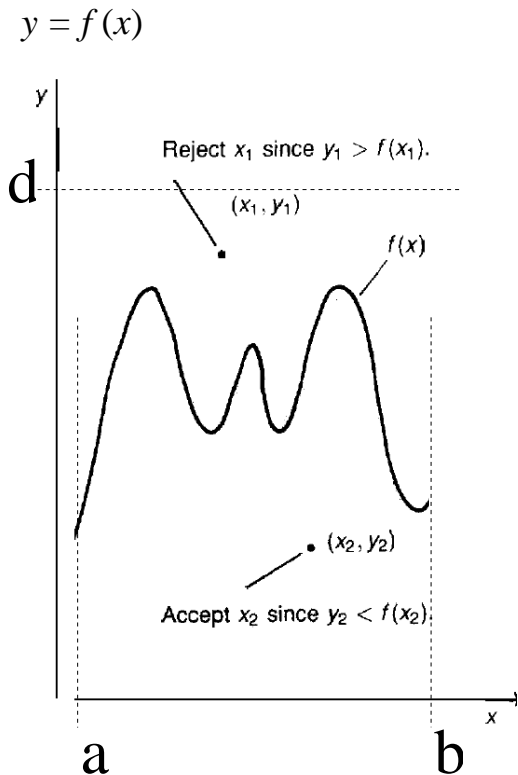
Inverse method: $x = F^{-1}(y)$
 $\Rightarrow x$ is $f(x)$ distributed

Accept / reject method:

x ... vector of $[a,b]$ uniformly distributed random numbers
 y ... vector of $[0,d]$ uniformly distributed random numbers

accept x_i for which $y_i < f(x_i)$,

\Rightarrow accepted x_i are $f(x)$ distributed!



Monte Carlo

This wonderful method in words:

The spirit of Monte Carlo is best conveyed by the example discussed in von Neumann's letter to Richtmyer. Consider a spherical core of fissionable material surrounded by a shell of tamper material. Assume some initial distribution of neutrons in space and in velocity but ignore radiative and hydrodynamic effects. The idea is to now follow the development of a large number of individual neutron chains as a consequence of scattering, absorption, fission, and escape.

At each stage a sequence of decisions has to be made based on statistical probabilities appropriate to the physical and geometric factors. The first two decisions occur at time $t = 0$, when a neutron is selected to have a certain velocity and a cer-

tain spatial position. The next decisions are the position of the first collision and the nature of that collision. If it is determined that a fission occurs, the number of emerging neutrons must be decided upon, and each of these neutrons is eventually followed in the same fashion as the first. If the collision is decreed to be a scattering, appropriate statistics are invoked to determine the new momentum of the neutron. When the neutron crosses a material boundary, the parameters and characteristics of the new medium are taken into account. Thus, a genealogical history of an individual neutron is developed. The process is repeated for other neutrons until a statistically valid picture is generated.

from N.Metropolis: The Beginning of the Monte Carlo Method, Los Alamos Science, Special Issue 1987

Monte Carlo

This wonderful method in words:

The spirit of Monte Carlo is best conveyed by the example discussed in von Neumann's letter to Richtmyer. Consider a spherical core of fissionable material surrounded by a shell of tamper material. Assume some initial distribution of neutrons in space and in velocity but ignore radiative and hydrodynamic effects. The idea is to now follow the development of a large number of individual neutron chains as a consequence of scattering, absorption, fission, and escape.

At each stage a sequence of decisions has to be made based on statistical probabilities appropriate to the physical and geometric factors. The first two decisions occur at time $t = 0$, when a neutron is selected to have a certain velocity and a cer-

tain spatial position. The next decisions are the position of the first collision and the nature of that collision. If it is determined that a fission occurs, the number of emerging neutrons must be decided upon, and each of these neutrons is eventually followed in the same fashion as the first. If the collision is decreed to be a scattering, appropriate statistics are invoked to determine the new momentum of the neutron. When the neutron crosses a material boundary, the parameters and characteristics of the new medium are taken into account. Thus, a genealogical history of an individual neutron is developed. The process is repeated for other neutrons until a statistically valid picture is generated.

from N.Metropolis: The Beginning of the Monte Carlo Method, Los Alamos Science, Special Issue 1987

Monte Carlo

This wonderful method in Geant4:

The spirit of Monte Carlo is best conveyed by the example discussed in von Neumann's letter to Richtmyer. Consider a spherical core of fissionable material surrounded by a shell of tamper material. Assume some initial distribution of neutrons in space and in velocity but ignore radiative and hydrodynamic effects. The idea is to now follow the development of a large number of individual neutron chains as a consequence of scattering, absorption, fission, and escape.

At each stage a sequence of decisions has to be made based on statistical probabilities appropriate to the physical and geometric factors. The first two decisions occur at time $t = 0$, when a neutron is selected to have a certain velocity and a cer-

tain spatial position. The next decisions are the position of the first collision and the nature of that collision. If it is determined that a fission occurs, the number of emerging neutrons must be decided upon, and each of these neutrons is eventually followed in the same fashion as the first. If the collision is decreed to be a scattering, appropriate statistics are invoked to determine the new momentum of the neutron. When the neutron crosses a material boundary, the parameters and characteristics of the new medium are taken into account. Thus, a genealogical history of an individual neutron is developed. The process is repeated for other neutrons until a statistically valid picture is generated.

Geometry, Materials

Event Generator
(not Geant4)

Stepping, Tracking

Physics Processes

from N.Metropolis: The Beginning of the Monte Carlo Method, Los Alamos Science, Special Issue 1987

So, what then is Geant4???

Copyright 2004 by Randy Glasbergen.
www.glasbergen.com



“I want you to find a bold and innovative way to do everything exactly the same way it’s been done for ~~25~~ years.”

~60 years!