

# Experiment Simulation

CERN  
School of  
Computing  
2004

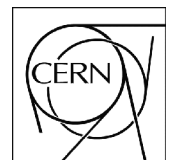


Vico  
Equense  
Italy

**"Always start your presentation with a joke, but be careful not to offend anyone! Don't mention religion, politics, race, age, money, technology, men, women, children, plants, animals, food...."**

**Martin Liendl**

SWX Swiss Exchange - IT Development  
(after having hacked at CERN)



# **PART I**

## Motivation Table of Contents

# What's this all about?

- Why do we need experiment simulation, if we are actually going to **do** the experiment??
- OK, we do simulation, but
  - why does everybody call it “Monte Carlo”?
  - what is this Geant4 “state of the art” simulation engine?
  - what are experiment specific requirements?
  - how are certain simulation requirements met by using Geant4?
- How is it done in a real example
  - some simulation related topics from the Compact Muon Solenoid (CMS) at the Large Hadron Collider (LHC) at CERN

# Emphasis on:

- Why do we need experiment simulation, if we are actually going to **do** the experiment??
- OK, we do simulation, but
  - why does everybody call it “Monte Carlo”?
  - what is this Geant4 “state of the art” simulation engine?
  - what are experiment specific requirements?
  - how are certain simulation requirements met by using Geant4?
- How is it done in a real example
  - some simulation related topics from the Compact Muon Solenoid (CMS) at the Large Hadron Collider (LHC) at CERN

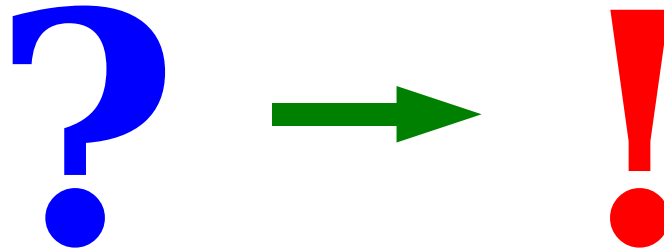
# Dis- / Ex- Claimer

- **For Physics students**
  - if you know already the principles of “Monte Carlo”:
    - see, how it's done in Geant4
  - if you know already basics of Geant4
    - enjoy Alberto's in depth lectures about physics in the special track
  - if you know Geant4 & its physics models in detail
    - see how experiment requirements (example: CMS at CERN) are coming in
  - if you know also this
    - well, the beach is not too far ...
- **For Computer Science students**
  - if you want to know why Physicists need a lot of computing power (of the Grid and more) even before they take data
    - see, why Geant4 is guilty (partly, at least)
  - if you want to know where lots of “non-physics” computing engineering needs to be done
    - see, why experiment simulation is a good playground
  - if you want to play Doom with your colleagues
    - well, the exercise room is free now ...

Remark on handouts and slides:

Your handouts don't contain all of the slides shown.  
Slides will be made available on the CSC web.

# Some reasons why we need simulation in high energy physics experiments



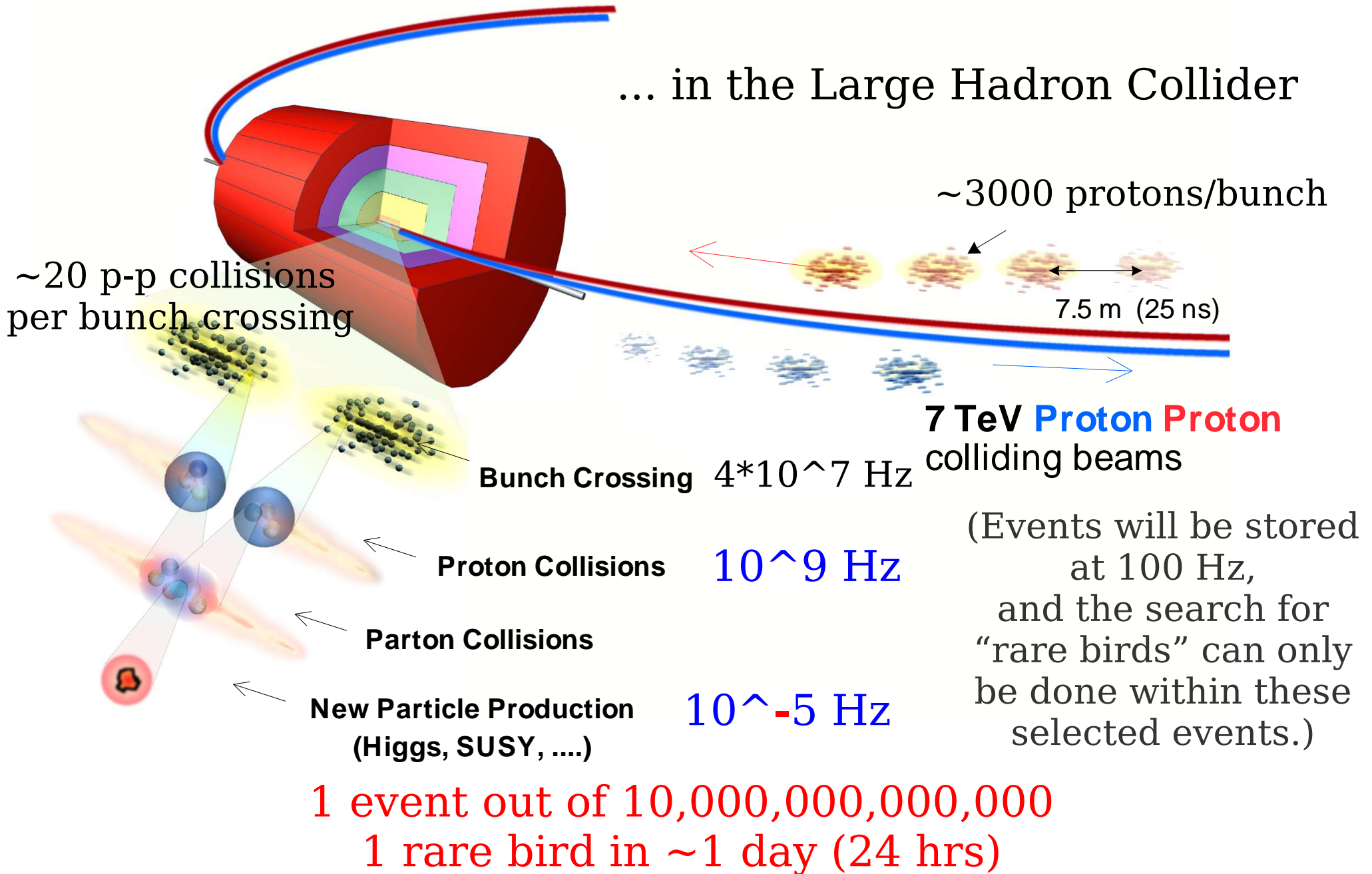
# Why Simulation?

- LHC experiments are enormous machines! CMS for example:
  - ~2000 collaborators world wide
  - First official papers of CMS: 1989/90!!  
Data taking (earliest): 2007!!
- Simulation is crucial for
  - the design of detector components:  
detector performance estimation, and measurement quality
  - the calculation/estimation of detector parameters (such as trigger rates, efficiencies) to be used in data taking and analysis
  - evaluation of a gigantic amount of experiment related software (reconstruction algorithms, data management, ..)!
  - data analysis in order to understand measured facts with simulation prediction



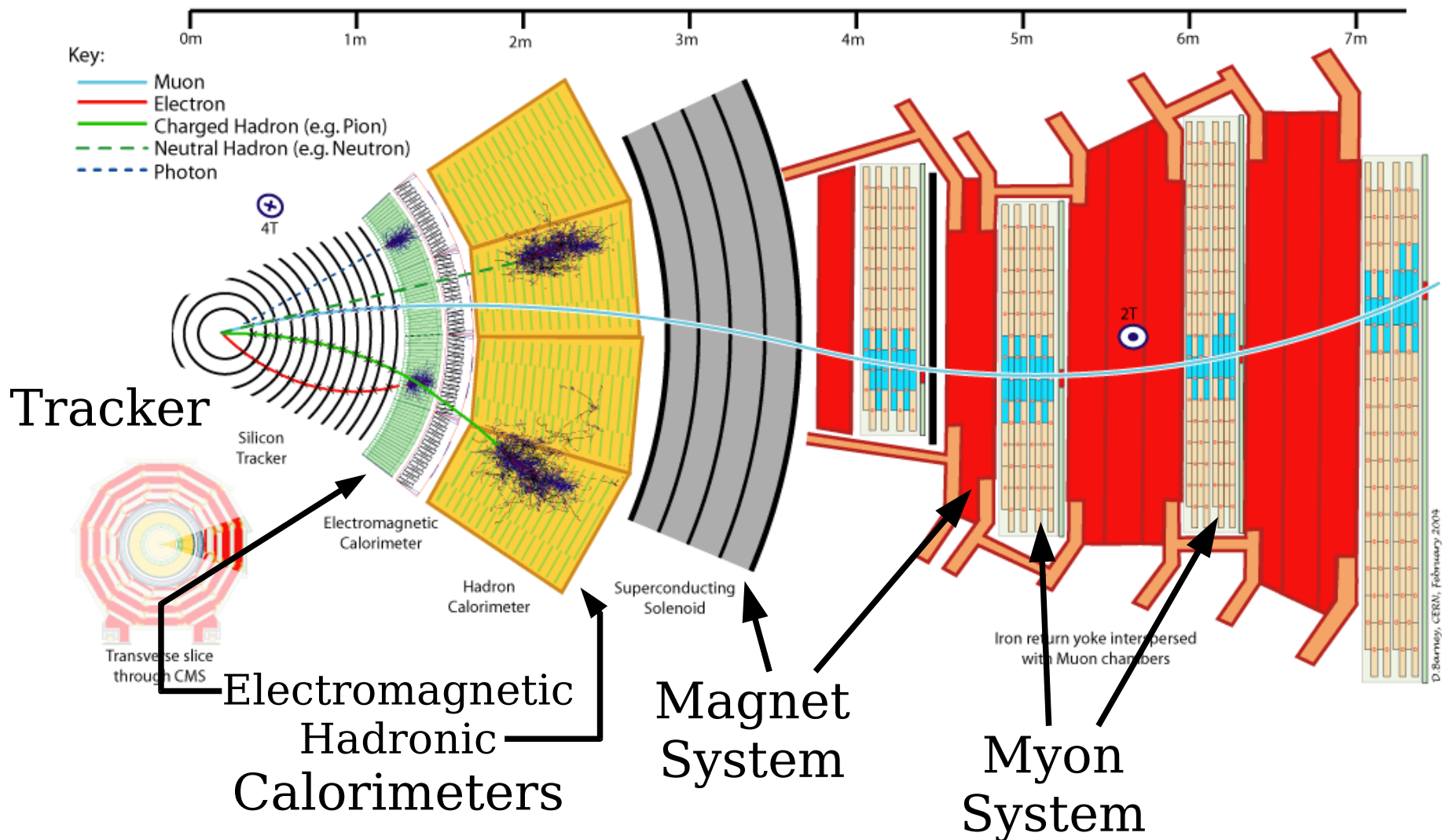
# Rare & Exotic Birds ...

... in the Large Hadron Collider

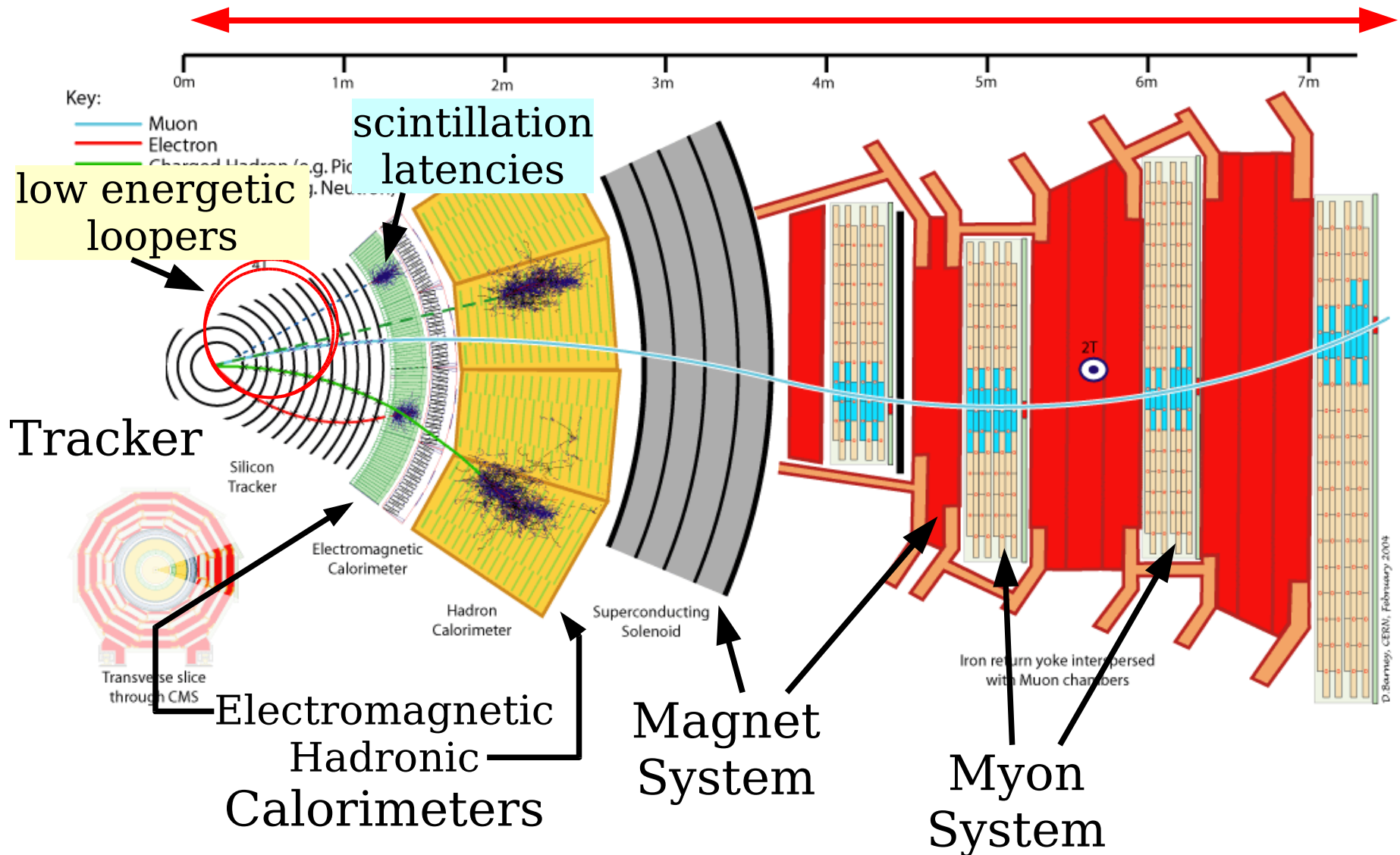


# The measurement process ...

parton collision -> "stable" particles, signature ->  
interaction of "stable" particles with detector (well known physics)

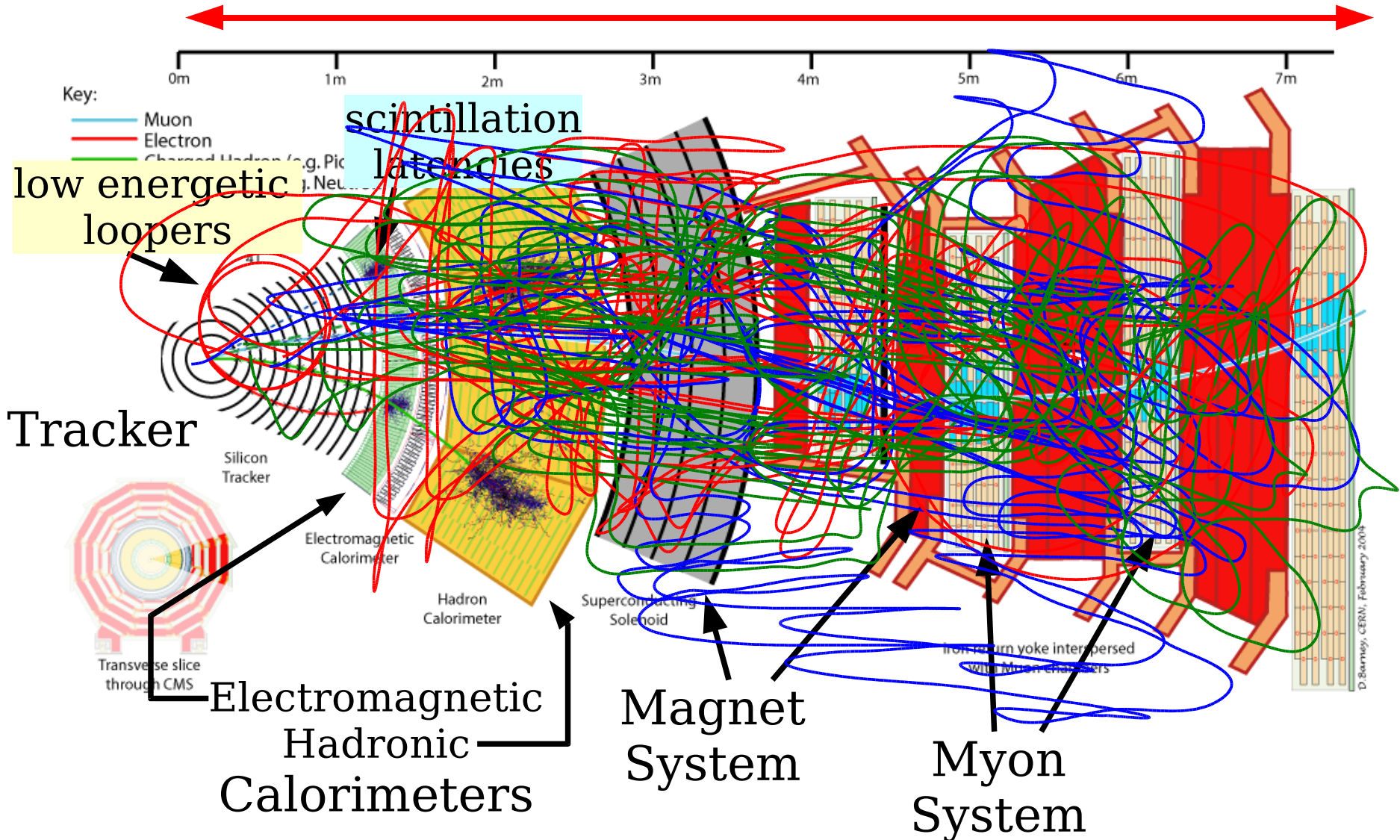


~7.5m = distance a particle travels in 25ns!



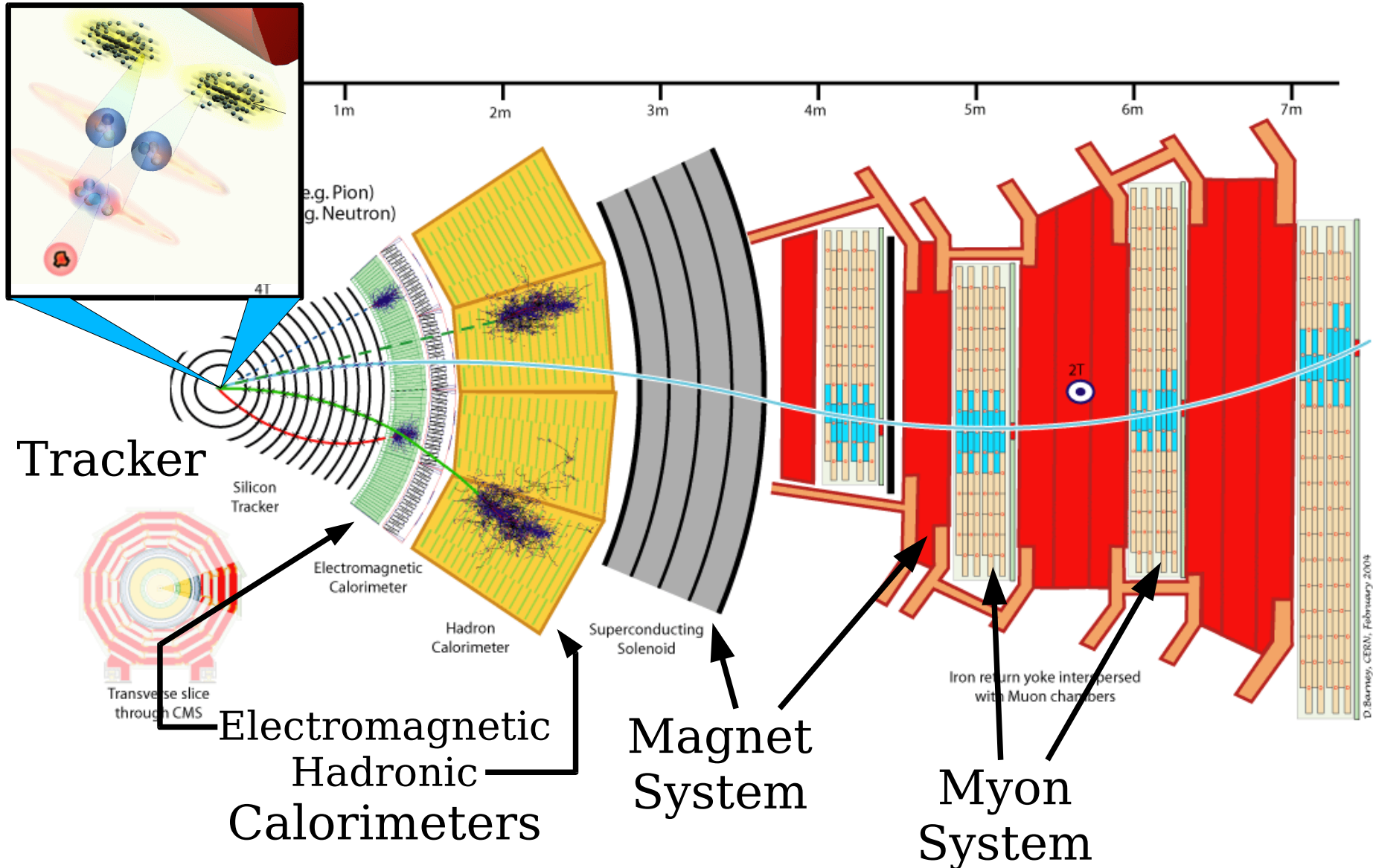
overlay of ~20 collisions per bunch crossing  
overlay of subsequent bunch crossings

~7.5m = distance a particle travels in 25ns!

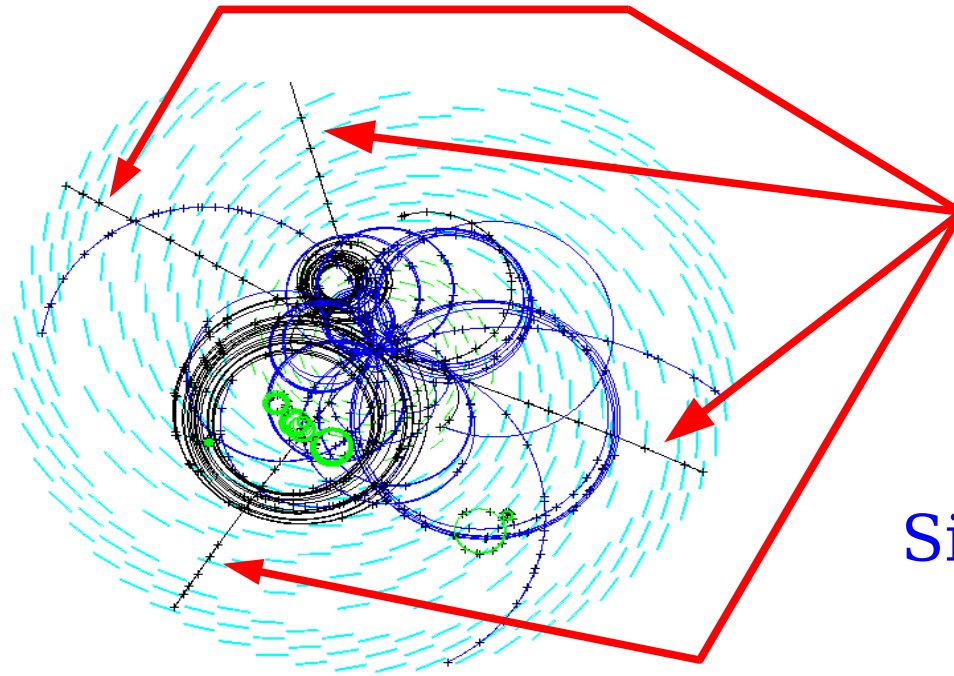


Event generators

# Particle through matter simulators



# Signatures



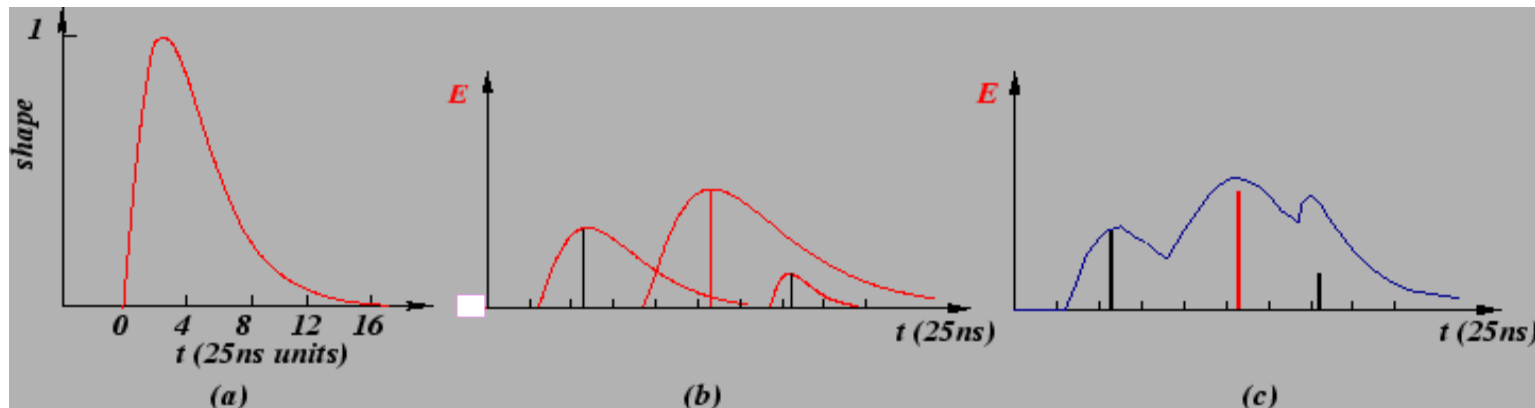
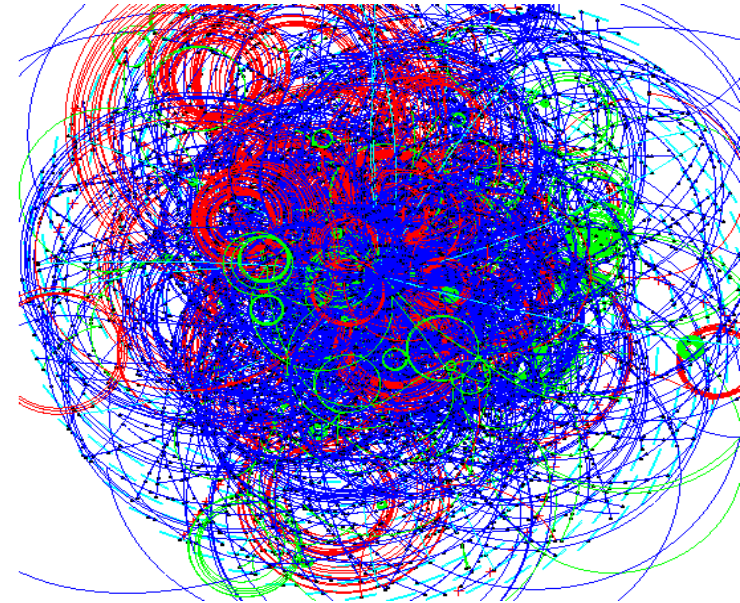
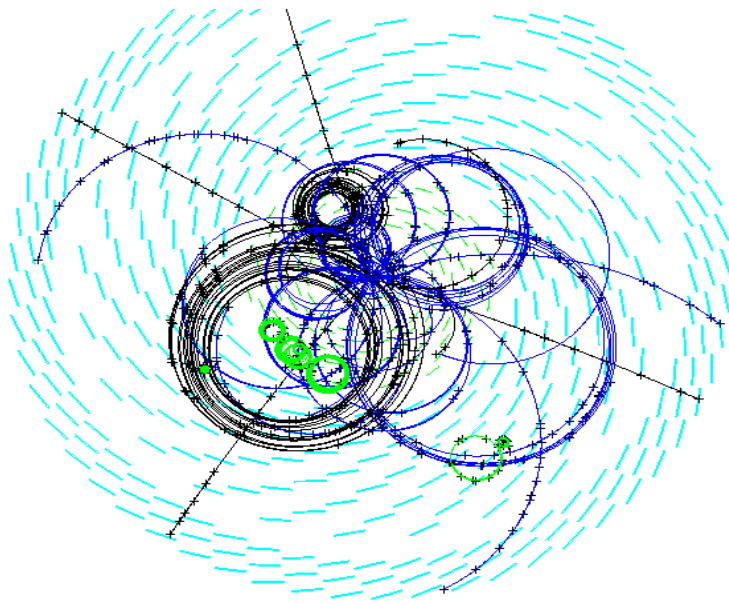
= what the Triggers are  
looking for

4 muons in the Tracker

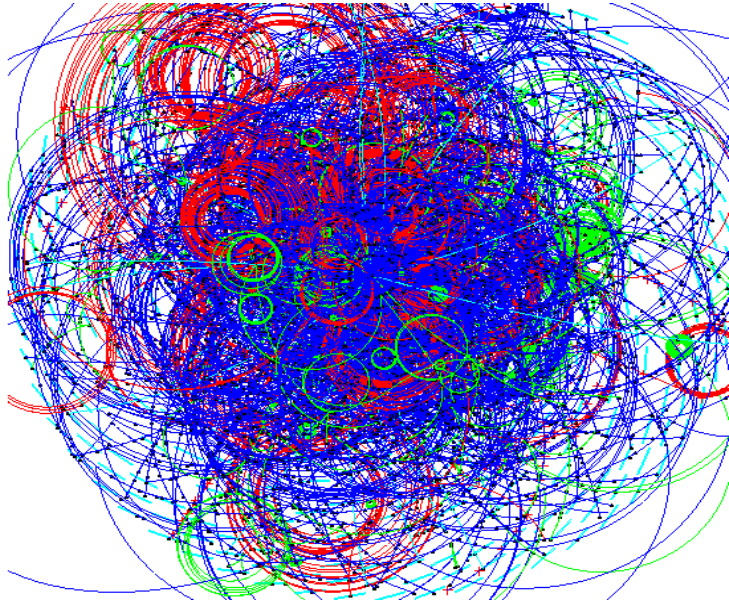
Signature of a Higgs decay

# What happens in parallel ...

= what you read out after the trigger says "yes"



# Measurement Process



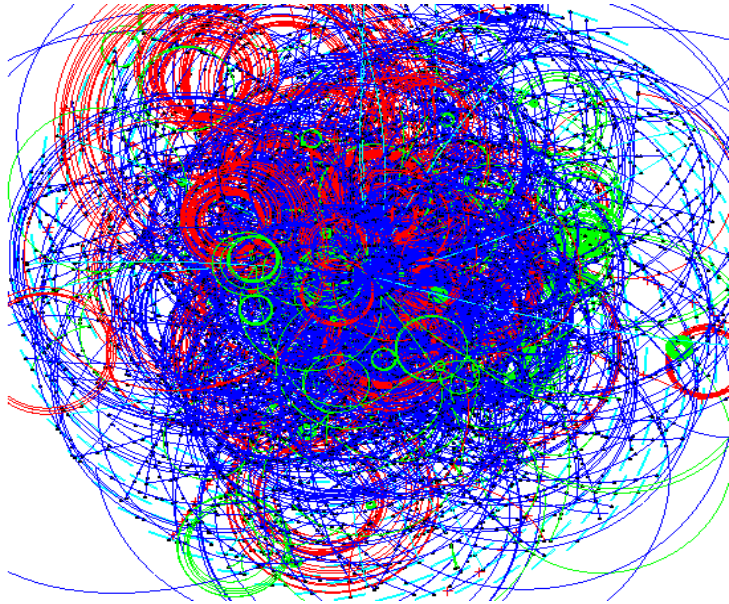
Trigger finds signals compatible with interesting signatures

“Snapshot” is saved (sensitive channels are read out and data is saved)

Reconstruction taking conditions into account; save rec-objects



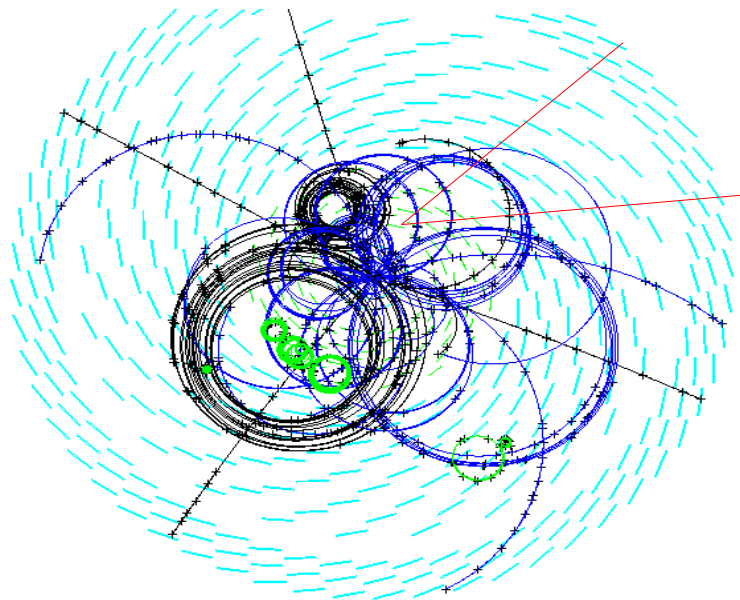
# Measurement Process



Trigger finds signals compatible with interesting signatures

“Snapshot” is saved (sensitive channels are read out and data is saved)

Reconstruction taking conditions into account; save rec-objects



**Cuts** are applied to select the events that probably belong to the physics channel to be studied

- some events are compatible but belong to different channels (**background** events, **cuts** once more)
- **efficiencies** of the measurement are taken into account

# Measurement Process

Trigger finds signals compatible with interesting signatures

“Snapshot” is saved (sensitive channels are read out and data

**Cuts = selective, controlled ignorance of data**

**usually based on physical and statistical reasoning, some times very personal (N.N.'s cuts were used to ...)**

- some events are compatible but belong to different channels (background events, cuts once more)
- efficiencies of the measurement are taken into account

# Measurement Process

What are “interesting signatures” of yet unmeasured physics events?

Mixing of signals, min. bias, background?

What will my detector be able to measure?

- if I change part A to B
- if I modify trigger tables
- ...

How good are my reco-algorithms?

Trigger finds signals compatible with interesting signatures

“Snapshot” is saved (sensitive channels are read out and data is saved)

Reconstruction taking conditions into account; save rec-objects

**Cuts** are applied to select the events that probably belong to the physics channel to be studied

- some events are compatible but belong to different channels (**background** events, **cuts** once more)
- **efficiencies** of the measurement are taken into account

# Measurement Process

What are “interesting signatures” of yet unmeasured physics events?

Mixing of signals, min. bias, background?

What will my detector be able to measure?

- if I change part A to B
- if I modify trigger tables
- ...

How good are my reco-algorithms?

Simulation of collisions of high energetic particles:  
**“Monte Carlo”  
Event Generators**

Simulation of particles passing macroscopic matter;  
Simulation of detector response signals, trigger acceptances, ...  
experiment specific:

**Experiment Simulation  
“Monte Carlo”**

# Some important aspects requiring experiment simulation

**Acceptance**

**Efficiency**

**Background**

To rule them all – the experimenter's art!



# Acceptance

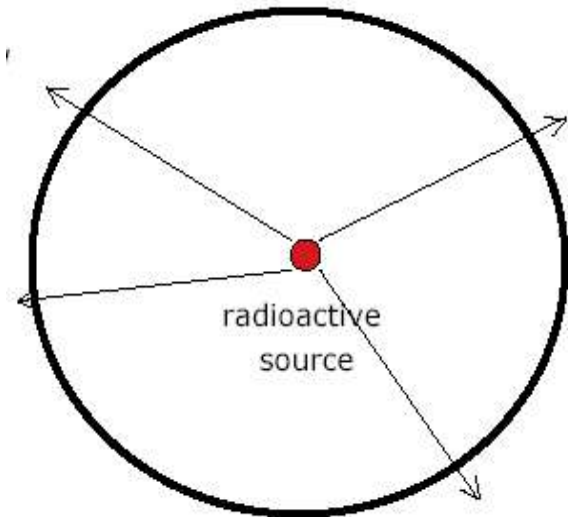
we measure

$N_{\text{measured}}$

we want to know

$N_{\text{true}}$

$$N_{\text{measured}} = \underbrace{\frac{\text{sensitive area}}{\text{total area}}}_{\text{acceptance}} N_{\text{true}}$$



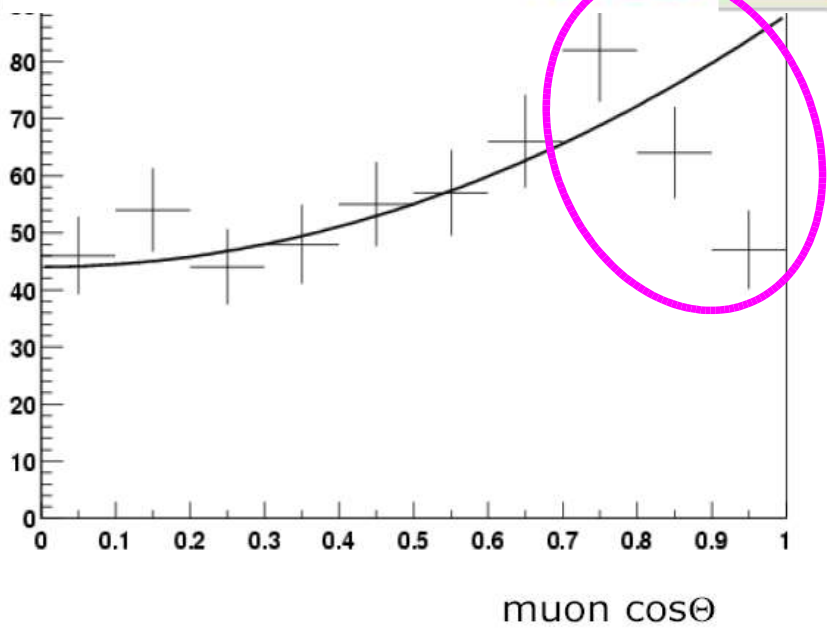
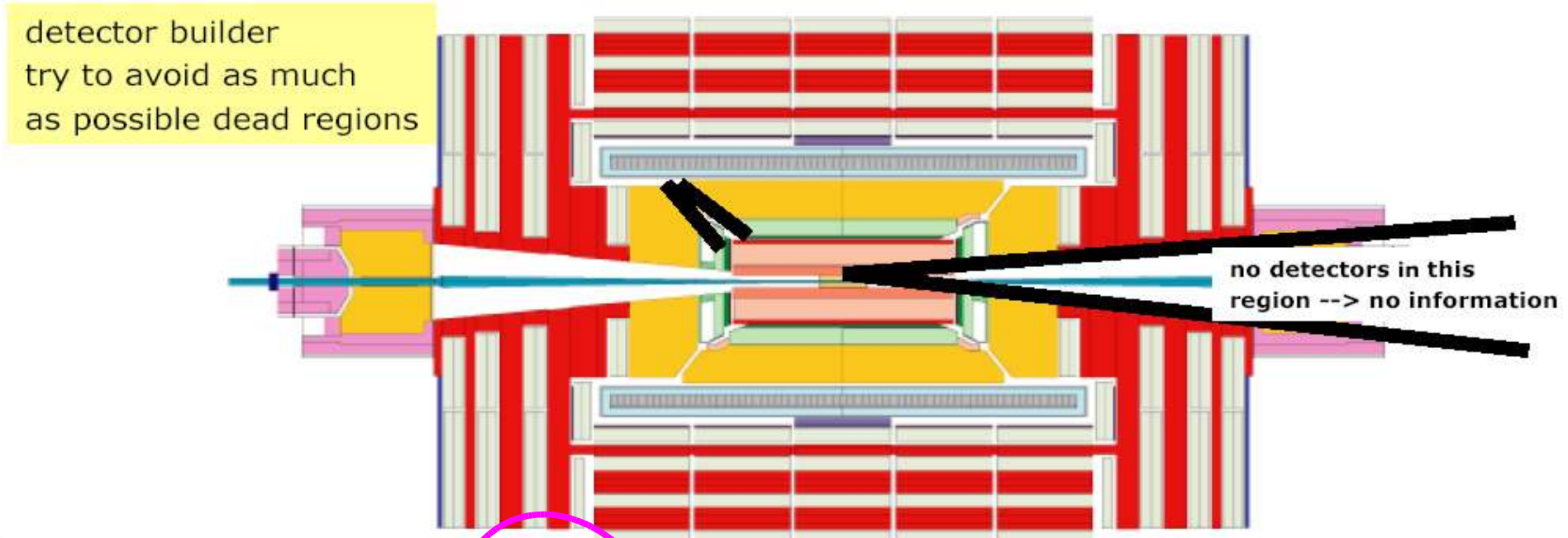
Example shown: symmetrical, fully covering detector and isotropic physics -> simple expression for acceptance.

More complex setups usually need Monte Carlo techniques!

# Detector acceptance corrections

$$N_{\text{measured}} = \frac{\text{sensitive area}}{\text{total area}} N_{\text{true}}$$

we measure
we want to know



$i \dots \cos(\theta)$

$$N_{\text{obs}}(i) = C(i) * N_{\text{true}}(i)$$

$$\frac{N_{\text{meas-MC}}(i)}{N_{\text{gen-MC}}(i)}$$

Acceptance correction factor obtained from Monte Carlo experiment simulation

# Measurement of cross sections

**Cross section  $\sigma$ :** Lorentz invariant measure of the probability of interactions in a two-particle initial state

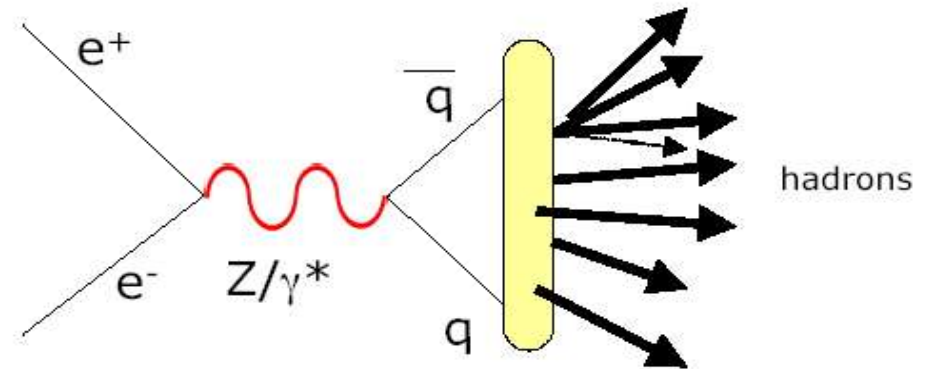
cross section to be determined

Ideally: 
$$N_{\text{sel}} = \sigma_{\text{signal}} \cdot L_{\text{int}}$$

number of measured and selected events of the interesting interaction (signal)

integrated beam luminosity

Example: Z decay in hadrons



$N_{\text{sel}}$ : accept event if a certain amount of charged tracks are observed



# Measurement of cross sections

Ideally:  $N_{\text{sel}} = \sigma_{\text{signal}} \cdot L_{\text{int}}$

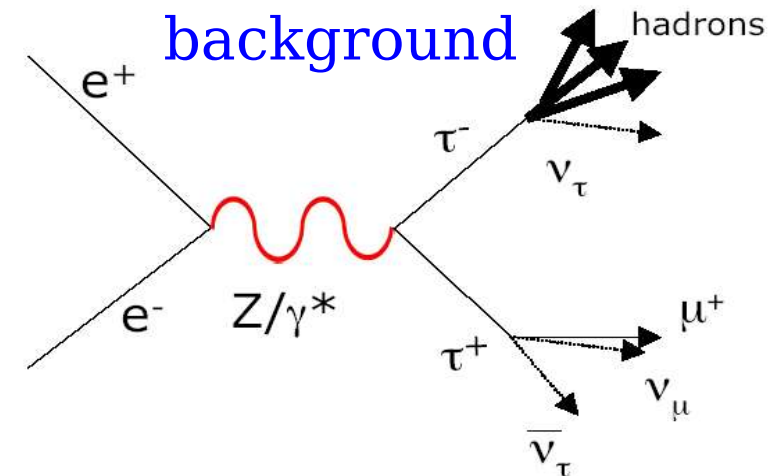
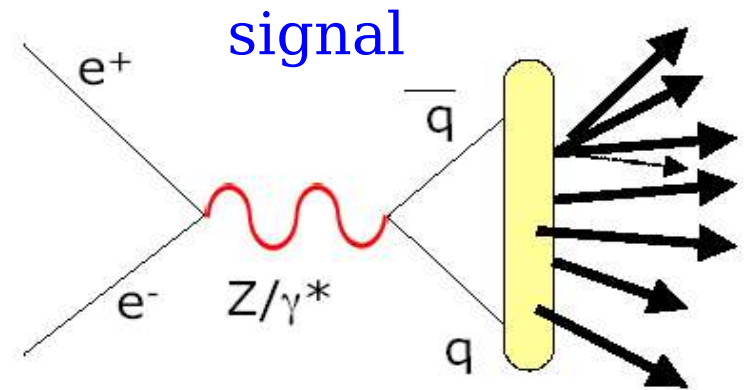
Really:  $N_{\text{sel}} = \sigma_{\text{signal}} \cdot L_{\text{int}} \cdot \epsilon + N_{\text{bckg}}$

selection efficiency

$$\epsilon = \frac{\text{selected signal events}}{\text{all signal events}}$$

background

$$N_{\text{bckg}} = \text{selected non-signal events}$$



$N_{\text{sel}}$ : accept event if a certain amount of charged tracks are observed

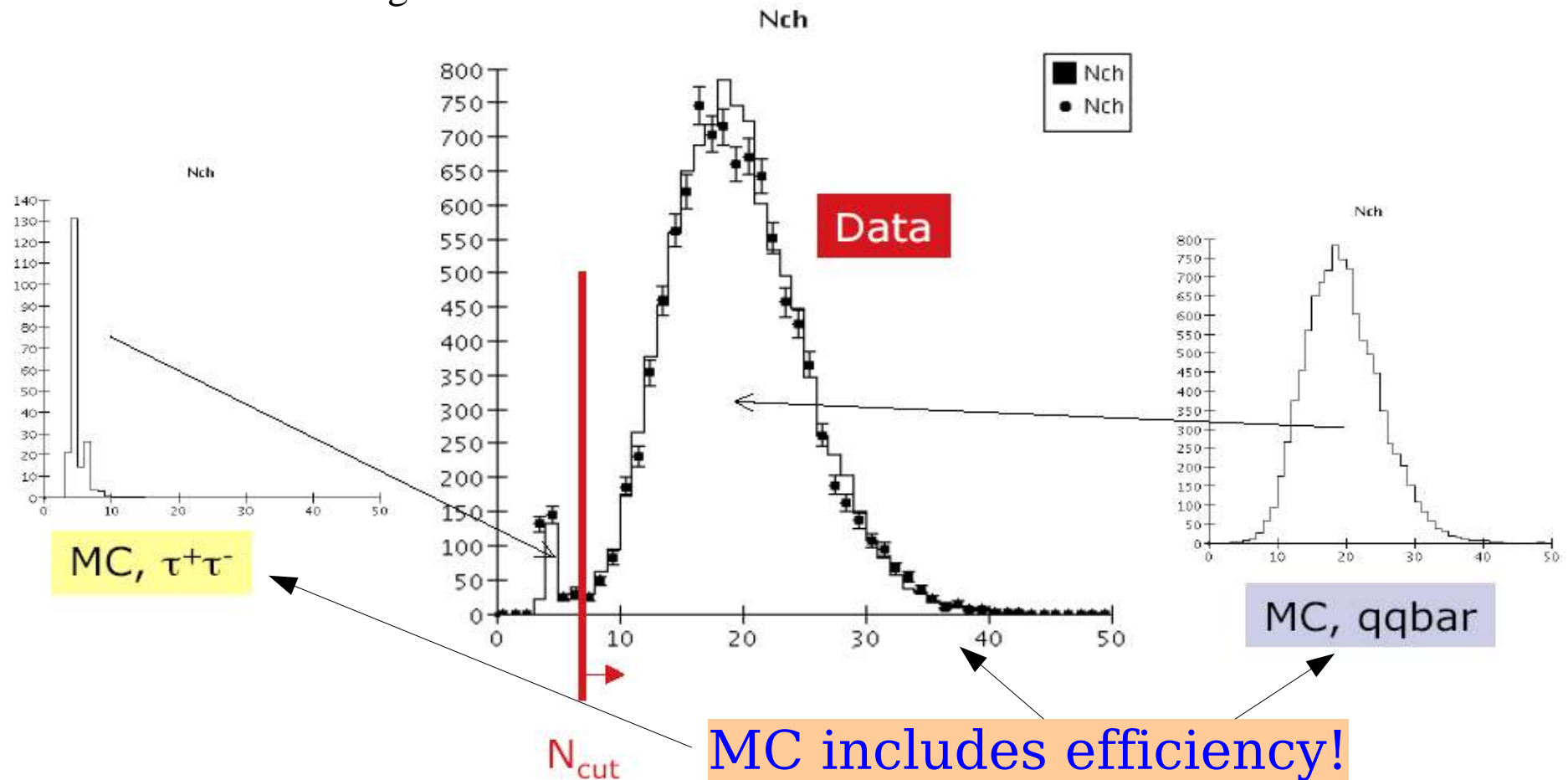
# Measurement of cross sections

selection  
efficiency

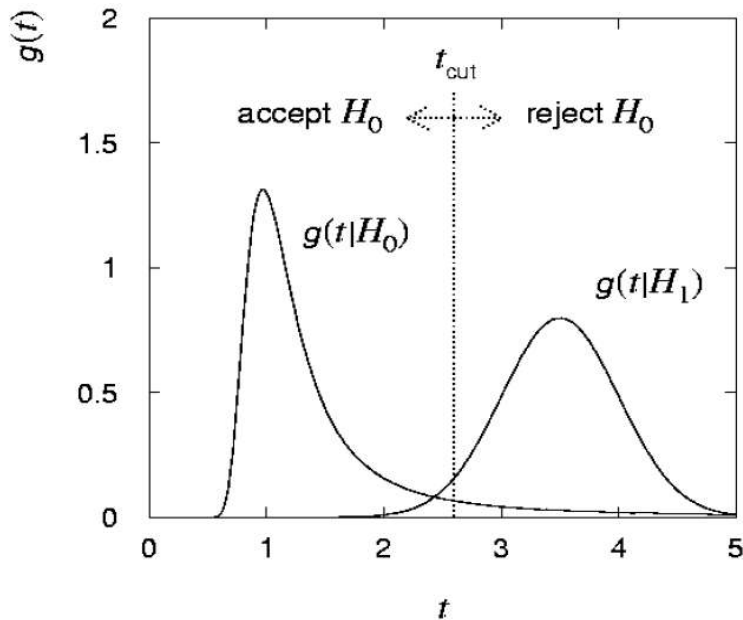
$$\epsilon = \frac{\text{selected Monte Carlo signal events}}{\text{all Monte Carlo signal events}}$$

background

$$N_{\text{bckg}} = \text{selected non-signal Monte Carlo events}$$



# Measurement of cross sections



based on the distributions obtained by simulation, apply statistical methods to set the cut.

## Consequences:

- biased/wrong models in your simulations: systematic errors
- not sufficient simulation statistics: larger uncertainties in your result

# Simulation overview

- Event generators
  - signal channels
  - background channels
  - experiment independent
  - physics dependent
- Experiment simulation
  - interaction of particles with matter
  - electronic signal response of detector elements
  - many other experiment specific dependencies

Event Generators like  
FLUKA, Herwig, ...  
  
not in these lectures

Focus on one simulation toolkit only: Geant4

# Experiment simulation overview

- Experiment simulation
  - interaction of particles with bulk matter
    - depends on physics of particles interacting with macroscopic matter
    - generic tools to simulate
    - we will present Geant4
  - electronic signal response of detector elements
    - conversion of physics interaction in matter into electronic signals
    - experiment specific
    - but generic tools, like Geant4, can be supportive
  - many other experiment specific dependencies
    - geometrical layout of the detector, materials used
    - magnetic field
    - data formats

# Outline of the following lectures

- Overview of Geant4
- Units (in Geant4)
- Detector description
  - materials
  - geometry
  - how it's done in Geant4
- Tracking in Geant4
  - stepping without physics
  - external fields
- Basics of “Monte Carlo”
  - where & why random numbers
- Physics in Geant4
  - principles
  - influence on tracking
  - physics lists
- Setting up a simulation
  - user initializations
  - user actions
  - hits, digitization
- Simulation in CMS
  - applications, use of Geant4
  - illustrative example
  - (G)UI aspects

# PART II

## Overview of Geant4

Basic Detector Description:

Units

Materials

Solids

Logical Volumes

# Overview of Geant4

“Geant4 is a **toolkit** for simulating the passage of particles through matter. It includes a complete range of functionality including **tracking**, **geometry**, **physics models** and **hits**.”<sup>[1]</sup>

[1] NIM A506 (2003), 250-303

**GEANT** comes from **GE**ometry **ANd** **T**racking.

The history of GEANT goes back to the 1970s and can be followed through CERN.

<http://www.cern.ch/geant4>

Geant4 is the C++ successor of the FORTRAN Geant3 (and much more).

1994–1998: R&D phase (~100 scientists from >10 experiments world wide;  
first production release in 1998; today's release: 6.2  
=> 10 years of work!!

Areas of application: high energy physics, medicine, space



# Quantities, Units

*Is this length in mm or cm?*

*Is the energy in Joule or eV or GeV?*

Is the speed in km/h or fractions of  $c$ ?

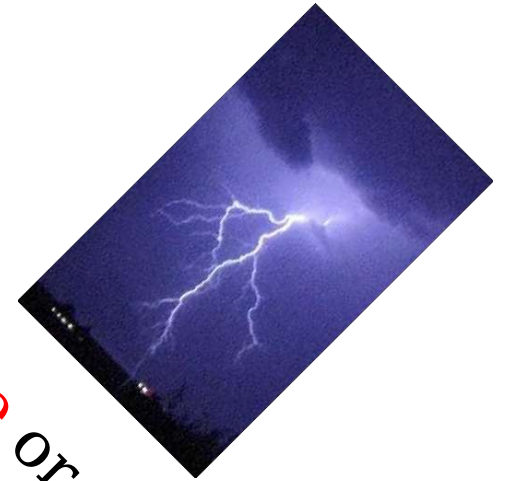
# Quantities, Units



Is this length in mm or cm?

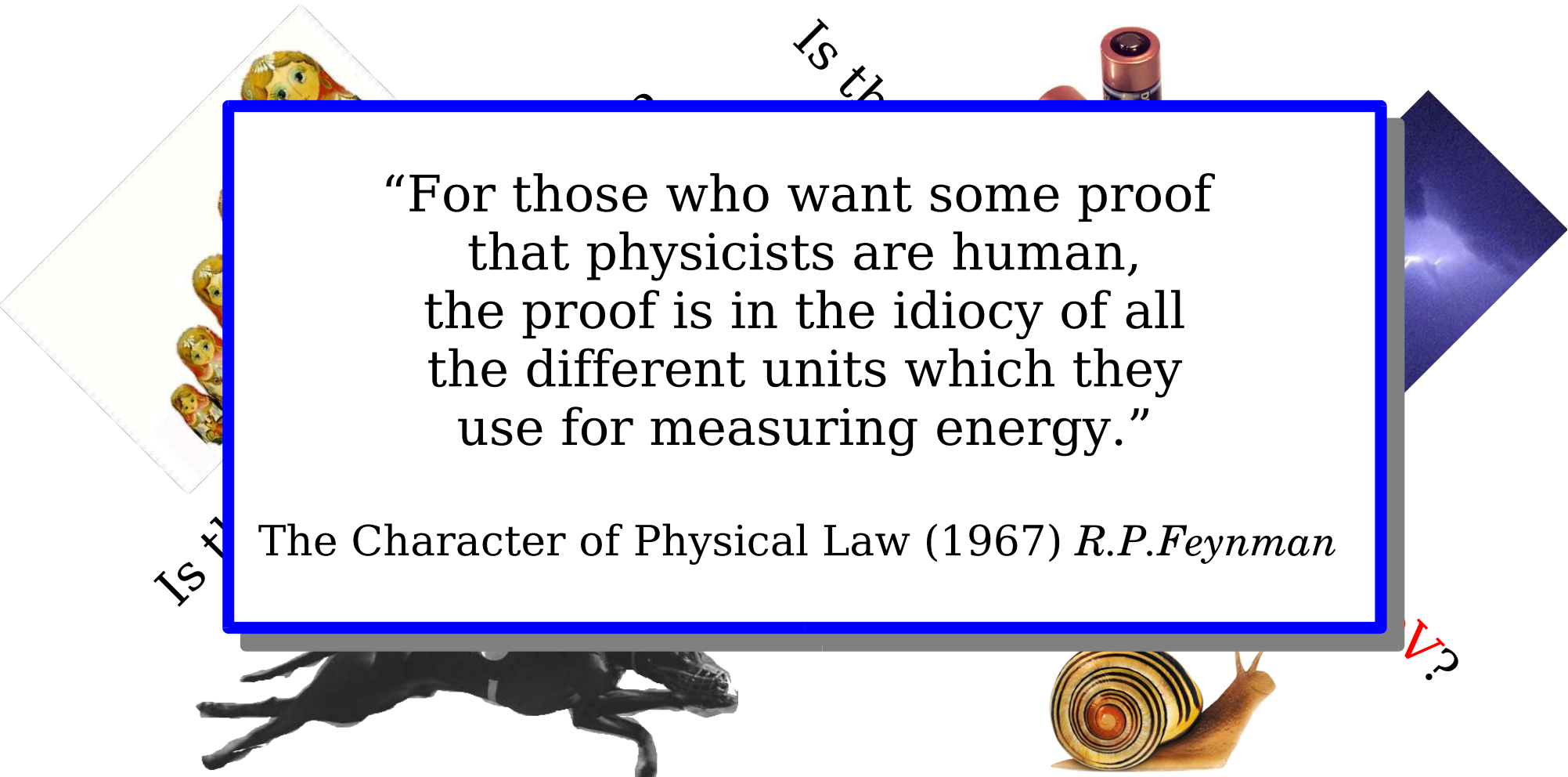


Is the energy in Joule or eV or GeV?



Is the speed in km/h or fractions of  $c$ ?

# Quantities, Units



“For those who want some proof that physicists are human, the proof is in the idiocy of all the different units which they use for measuring energy.”

The Character of Physical Law (1967) *R.P.Feynman*

Is the **speed** in **km/h** or **fractions of  $c$** ?

# Quantities, Units

Experiment simulation consists of many simulation packages. Each one may or may not be based on a common simulation package like Geant4. Each one may or may not have a consistent use of unit definitions.

# Quantities, Units

Experiment simulation consists of many simulation packages. Each one may or may not be based on a common simulation package like Geant4. Each one may or may not have a consistent use of unit definitions.

Geant4 consistently uses **CLHEP  
Units/SystemsOfUnits.h**

# Quantities, Units

Experiment simulation consists of many simulation packages. Each one may or may not be based on a common simulation package like Geant4. Each one may or may not have a consistent use of unit definitions.

Geant4 consistently uses **CLHEP  
Units/SystemsOfUnits.h**

## Simple Rules:

- all quantities are represented by a **double**.
- to pass/create a physical quantity, multiply the numerical value by its unit
- to read/use a physical unit, divide the numerical value by the unit you want to have

# Quantities, Units

## Attention:

Different quantities are not separately typed! E.g. energies and lengths share the same type: a double. Thus, one can easily & wrongly convert incompatible quantities into each other!!

any simulation be based on a int4. Each one use of unit

HEP  
h



## Simple Rules:

- all quantities are represented by a **double**.
- to pass/create a physical quantity, multiply the numerical value by its unit
- to read/use a physical unit, divide the numerical value by the unit you want to have

# CLHEP/Units/SystemOfUnits.h Example

Code:

```
double energy = 1.*GeV;]
cout << "Energy of 1 GeV:" << endl;
cout << " in ev      : " << energy/eV << endl
      << " in Joule: " << energy/joule << endl;

double h = 60.*60.*s;
double velocity = 20.*km/h;
cout << "Velocity of 20 km/h:" << endl
      << " internal : " << velocity << endl
      << " km/h      : " << velocity/(km/h) << endl
      << " m/sec     : " << velocity/(m/s) << endl;
```

Result:

```
Energy of 1 GeV:
 in ev      : 1e+09
 in Joule: 1.60218e-10

Velocity of 20 km/h:
 internal : 5.55556e-06
 km/h      : 20
 m/sec     : 5.55556
```

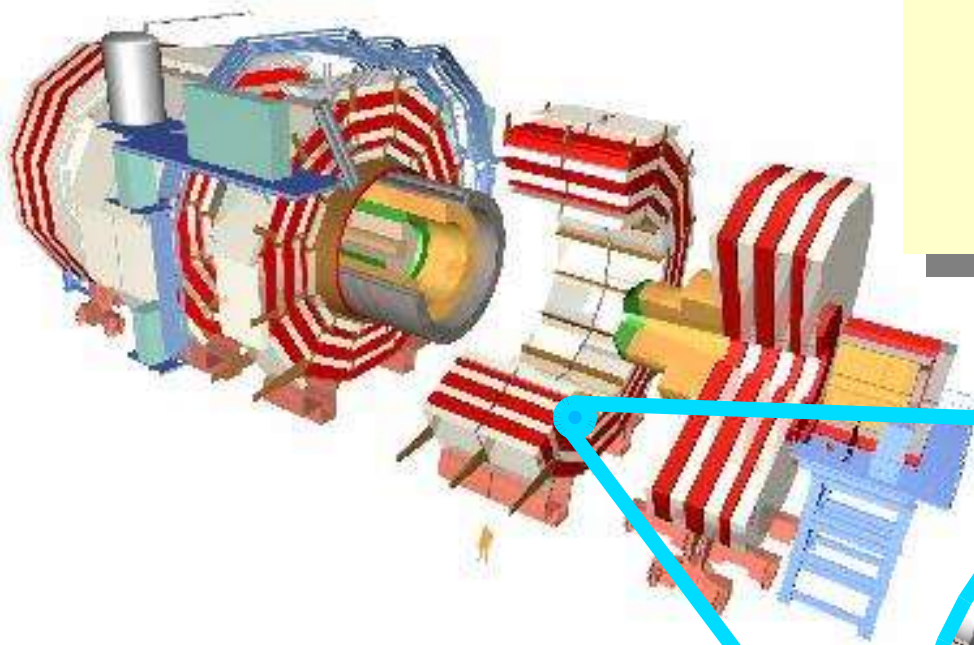
## Simple Rules:

- all quantities are represented by a **double**.
- to pass/create a physical quantity, multiply the numerical value by its unit
- to read/use a physical unit, divide the numerical value by the unit you want to have



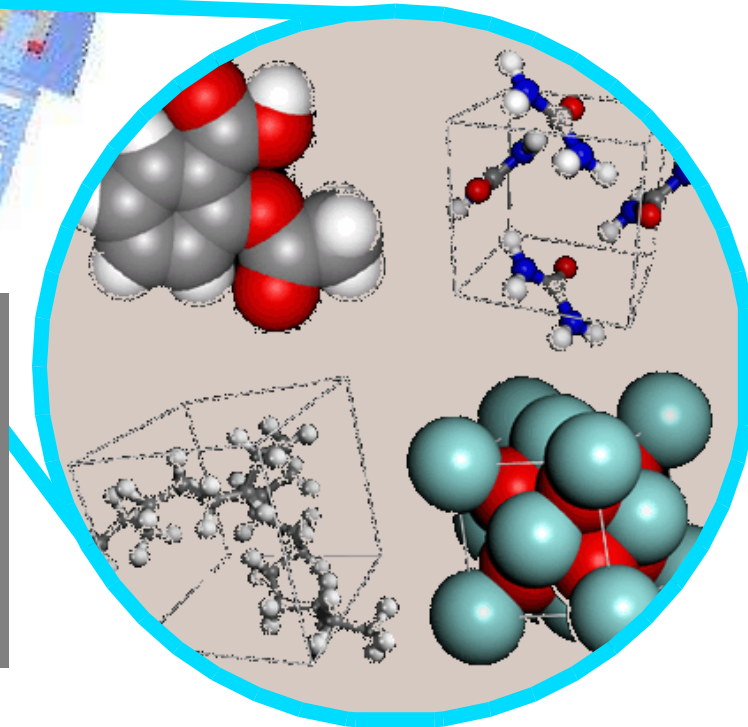
# Properties of Materials

A detector is made of  
**Materials**  
(like anything else  
in the world)



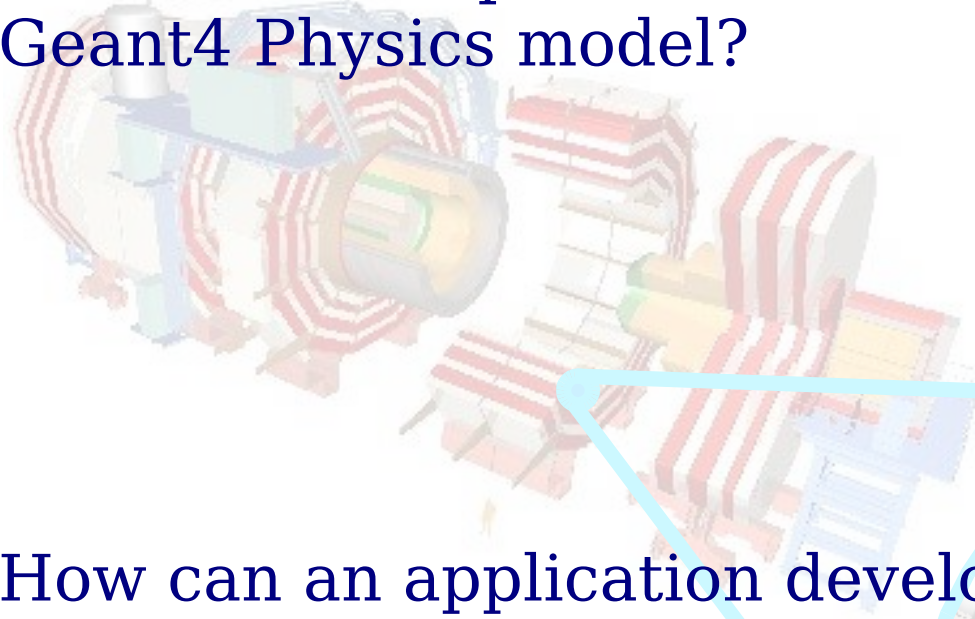
## Materials have properties:

- mechanical (construction)
- for **particle detection**
- for other external influences: magnetic field, ...

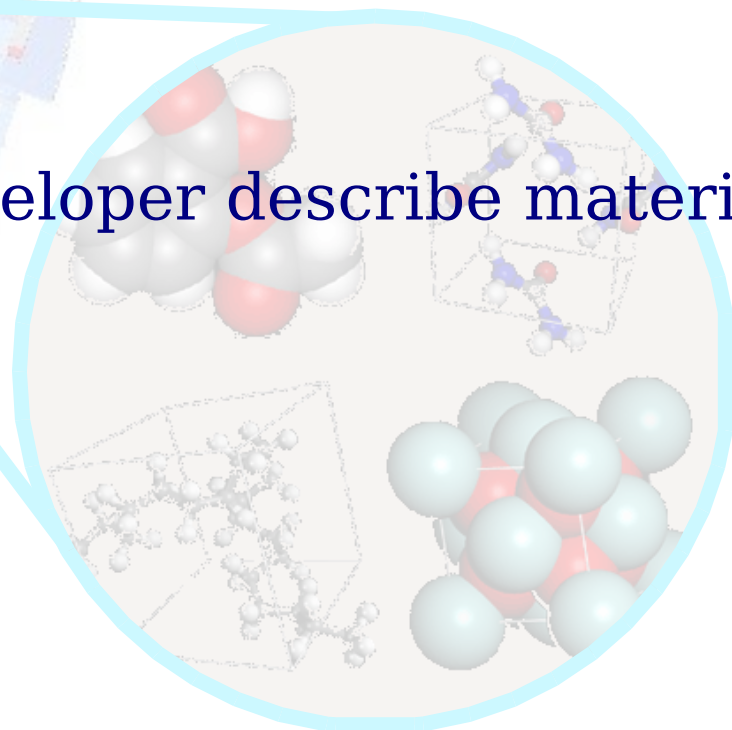


# Materials for Geant4 based Simulations

- What is the required material description for the Geant4 Physics model?

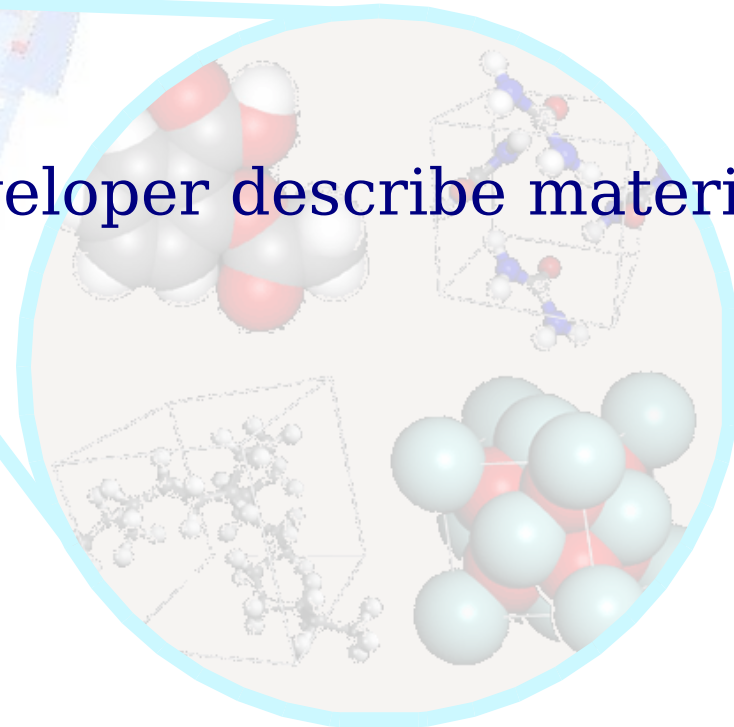


- How can an application developer describe materials comfortably/naturally?



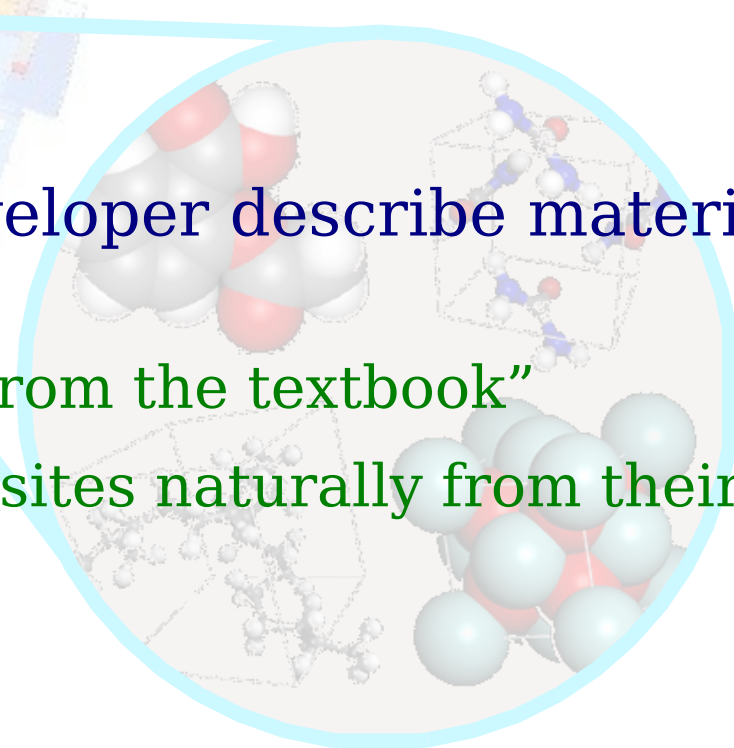
# Materials for Geant4 based Simulations

- What is the required material description for the Geant4 Physics model?
  - Particles interact with materials
  - Physical properties influence the way these interactions occur
- How can an application developer describe materials comfortably/naturally?



# Materials for Geant4 based Simulations

- What is the required material description for the Geant4 Physics model?
  - Particles interact with materials
  - Physical properties influence the way these interactions occur
- How can an application developer describe materials comfortably/naturally?
  - Describe them as “known from the textbook”
  - Compose mixtures / composites naturally from their “ingredients”



# Materials in Geant4 are **static** entities

- Materials are not affected by time
  - Gas does not diffuse, liquids don't flow
  - No thermodynamics: no heat transfer, no state changes (water <-> ice), ...

# Materials in Geant4 are **static** entities

- Materials are not affected by time
  - Gas does not diffuse, liquids don't flow
  - No thermodynamics: no heat transfer, no state changes (water  $\leftrightarrow$  ice), ...

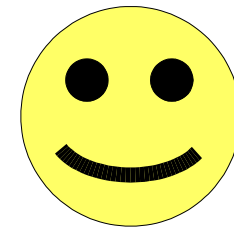
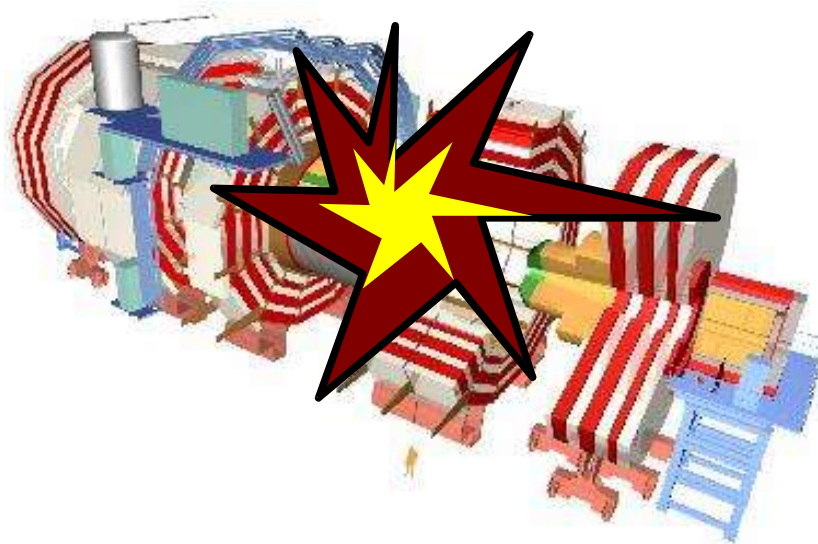


# Materials in Geant4 are **static** entities

- Materials are not affected by time
  - Gas does not diffuse, liquids don't flow
  - No thermodynamics: no heat transfer, no state changes (water <-> ice), ...
- Particle interaction does not harm!
  - No irradiation effects or radiation damages

# Materials in Geant4 are **static** entities

- Materials are not affected by time
  - Gas does not diffuse, liquids don't flow
  - No thermodynamics: no heat transfer, no state changes (water <-> ice), ...
- Particle interaction does not harm!
  - No irradiation effects or radiation damages





# Materials in Geant4 are **static** entities

- Materials are not affected by time
  - Gas does not diffuse, liquids don't flow
  - No thermodynamics: no heat transfer, no state changes (water <-> ice), ...
- Particle interaction does not harm!
  - No irradiation effects or radiation damages
- Not affected by external influences. Materials don't change physical properties under
  - Mechanical influences (stresses, pressures)
  - External electromagnetic fields, i.e. materials don't get magnetized, electrostatically charged (-> see later: description of external fields)

# But what to do, if you need to simulate these things?

- Materials are not affected by time
  - Gas does not diffuse, liquids don't flow
  - No thermodynamics: no heat transfer, no state changes (water  $\leftrightarrow$  ice), ...
- Particle interaction does not harm!

You have to use a different simulation package

# But what to do, if you need to simulate these things?

You have to use a different simulation package

- No irradiation effects or radiation damages
- Not affected by external influences. Materials don't change physical properties under

... and/or use Geant4 to calculate the doses/energies ...

# But what to do, if you need to simulate these things?

You have to use a different simulation package,  
and tell the results to Geant4

(e.g. Iron becomes magnetic in an external B-field,  
that chamber contains contains Xenon of 3 bar, ...)

change physical properties under

- Mechanical influences (stresses, pressures)
- External electromagnetic fields, i.e. materials don't get magnetized, electrostatically charged (-> see later: description of external fields)

# So, what are G4 Materials?

- All materials are represented as objects of a single class: **G4Material**
- `G4Material` is a concrete class that
  - provides all necessary accessor methods for G4
  - represents a homogeneous material made of elements, compounds, or mixtures of other materials
  - Internally, Geant4 decomposes every material into its constituting elements
- Users can access any material through a static registry:

```
static const G4MaterialTable* G4Material::GetMaterialTable();
```
- There are several ways to define a material in a user-friendly way through helper classes.

# There's more than one way to do it ...

## Material “from scratch” made of a single element:

```
G4double z;  
  
// tell the density (mass per volume)  
G4double density = 1.390*g/cm3;  
  
// tell the mass of one mole (numerically ~ atomic weight)  
G4double a = 39.95*g/mole;  
  
G4Material* lAr =  
    new G4Material("liquidArgon",z=18.,a,density);
```

/\* Note:

- \* Always create materials on the heap using the new operator.
- \* Be extremely careful with the specification of physical units,
- \* as Geant4 does not provide many self-consistency checks,
- \* and the values directly or indirectly enter cross-section
- \* calculations in the various physics processes employed!!!

\*/

# There's more than one way to do it ...

## Material from compounds:

A compound is made of several elements  
(composition by **number of atoms**)

```
G4String symbol; G4double y; G4int ncomp, natoms;
a = 1.01*g/mole; // mass per mole
G4Element* elH =
    new G4Element("Hydrogen", symbol="H", z=1., a);
a = 16.00*g/mole;
G4Element* elO =
    new G4Element("Oxygen", symbol="O", z=8., a);
density = 1.000*g/cm3; // tell the materials density
G4Material* H2O = // tell the number of componets
    new G4Material("Water", density, ncomp=2);
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

# There's more than one way to do it ...

## Material by elements by fraction of mass

```
a = 14.01*g/mole;
G4Element* elN =
    new G4Element(name="Nitrogen",symbol="N",z= 7.,a);
a = 16.00*g/mole;
G4Element* elO =
    new G4Element(name="Oxygen",symbol="O",z= 8.,a);
density = 1.290*mg/cm3;
G4Material* Air =
    new G4Material(name="Air",density,ncomponents=2);
Air->AddElement(elN, 70.0*perCent);
Air->AddElement(elO, 30.0*perCent);
```



# There's more than one way to do it ...

## Material by mixing (elements & materials)

```
G4Element* elC = ... ; // define "carbon" element
G4Material* SiO2 = ... ; // define "quartz" material
G4Material* H2O = ... ; // define "water" material

density = 0.200*g/cm3;
G4Material* Aerog =
    new G4Material("Aerogel",density,ncomponents=3);
Aerog->AddMaterial(SiO2,fractionmass=62.5*perCent);
Aerog->AddMaterial(H2O ,fractionmass=37.4*perCent);
Aerog->AddElement (elC ,fractionmass= 0.1*perCent);
```

# Summary: Materials

- Materials are static entities
  - ... and what that means ...
- The concrete class G4Material is the main interface to material related information
- Materials are stored by name in a static table
- Materials are created/defined directly and/or via helper classes G4Element, G4Isotope
- And remember:

# Summary: Materials

- Materials are static entities
  - ... and what that means ...
- The concrete class G4Material is the main interface to material related information
- Materials are stored by name in a static table
- Materials are created/defined directly and/or via helper classes G4Element, G4Isotope
- And remember:

There's more than one way to do it ...

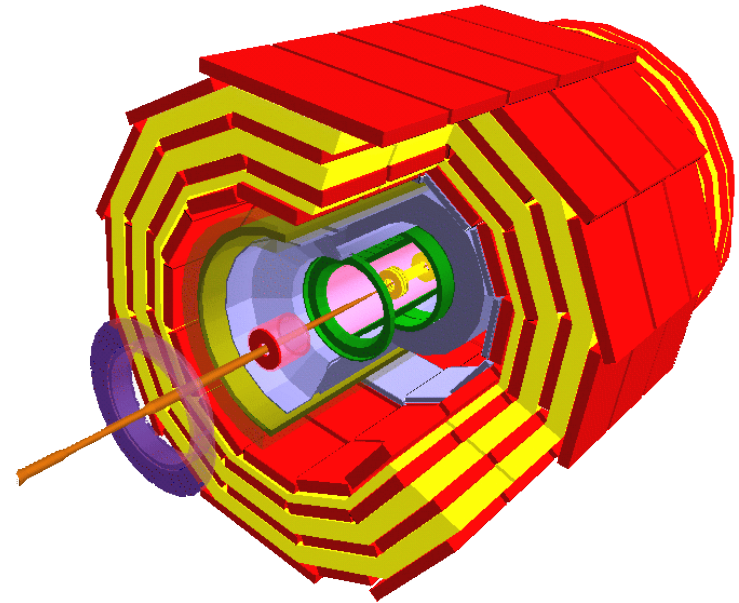
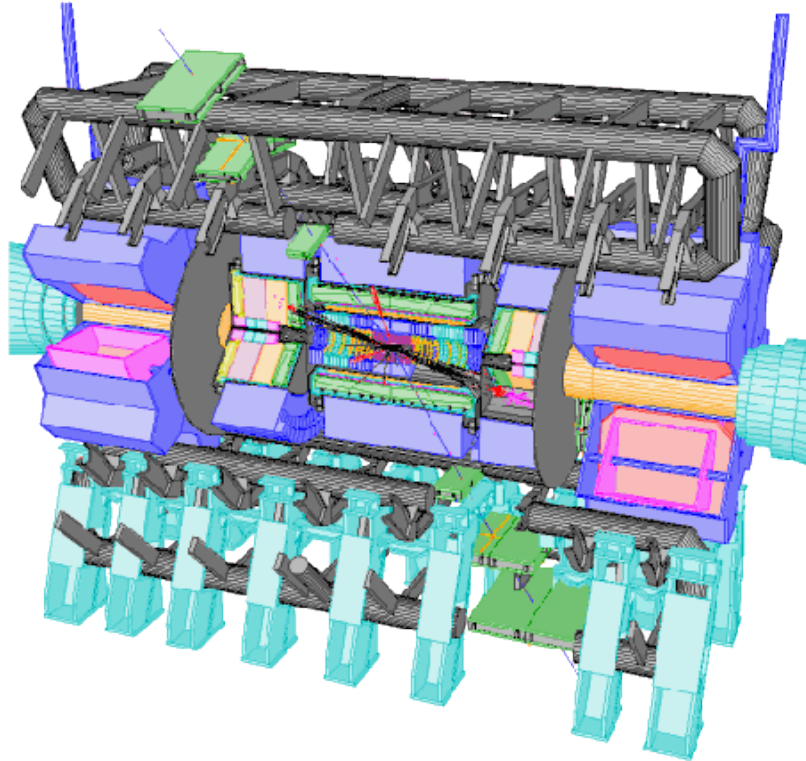
There's more than one way to do it ...

There's more than one way to do it ...

There's more than one way to do it ...

# Geometry

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 on Geometry

- Must be “realistic” for the Physicist
  - 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 ...

# General Requirements on Geometry

- Must conform to several constraints coming from the Geant4 particle-tracking sub-system
  - 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 ...

# General Requirements on Geometry

- **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”

# General Requirements on Geometry



The following focuses on the **Geant4 geometry** model.

However, bear in mind that for a successful experiment simulation all three issues are essential!

Later in the lectures, we will come back to these issues in the scope of CMS.



# Geant4 Geometry Model

- An instance of the Geant4 geometry model (the “geometry”) represents the detector
- Simulated particles are tracked through the geometry
  - Ray-tracing capability of Geant4
- Properties of the geometry affect the interactions of particles and thus the way their path is calculated
- Besides geometrical properties, other information relevant for simulation is attached to the geometry
  - Geometry is a natural choice for these properties
- In order to be a valid Geant4 geometry several constraints must be fulfilled

# Properties of the Geometry Model

- **Hierarchical** model
- Concept of “**logical volume**” / “**volume**”
  - **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** in the inside
    - Data extraction during simulation: **sensitivity, cuts**
- **Extensible** model
  - Add your custom made shapes!

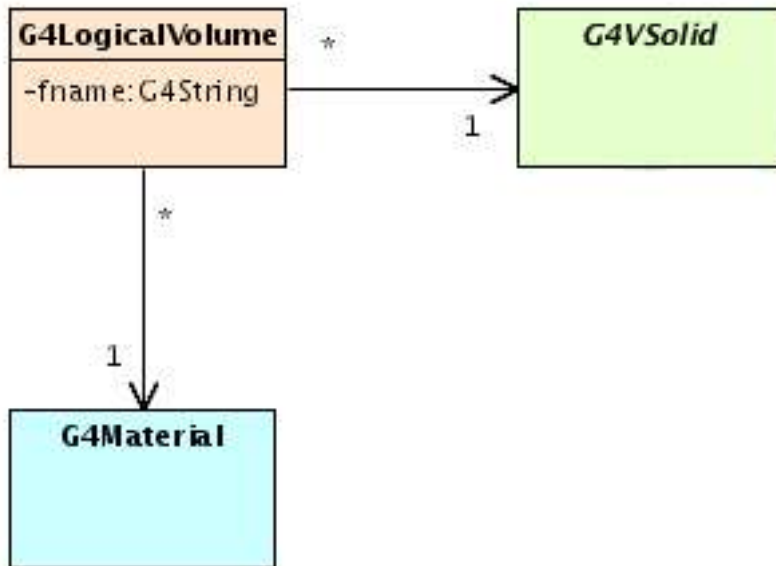
- Concept of “logical volume” / “volume”

- Mandatory properties:

- Has a **shape**

- Has a **material**

```
class G4LogicalVolume
class G4VSolid
class G4Material
```



- Concept of “logical volume” / “volume”

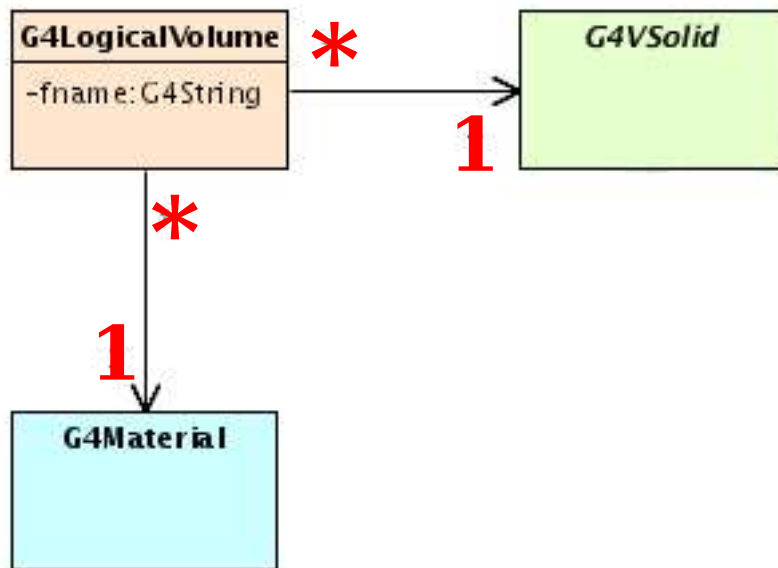
- Mandatory properties:

- Has a **shape**

- Has a **material**

```
class G4LogicalVolume
class G4VSolid
class G4Material
```

**Note:** solids and materials can be shared among volumes!



- Concept of “logical volume” / “volume”

- Mandatory properties:

- Has a **shape**

- Has a **material**

```
class G4LogicalVolume
class G4VSolid
class G4Material
```

**Note:** solids and materials can be shared among volumes!

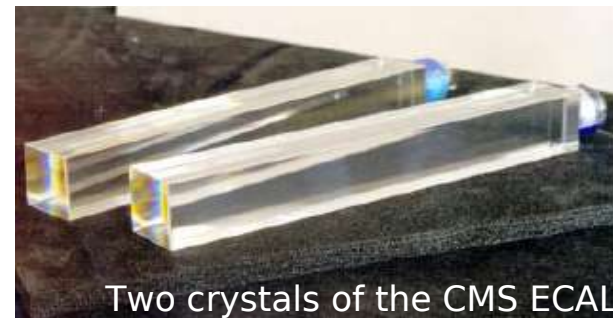
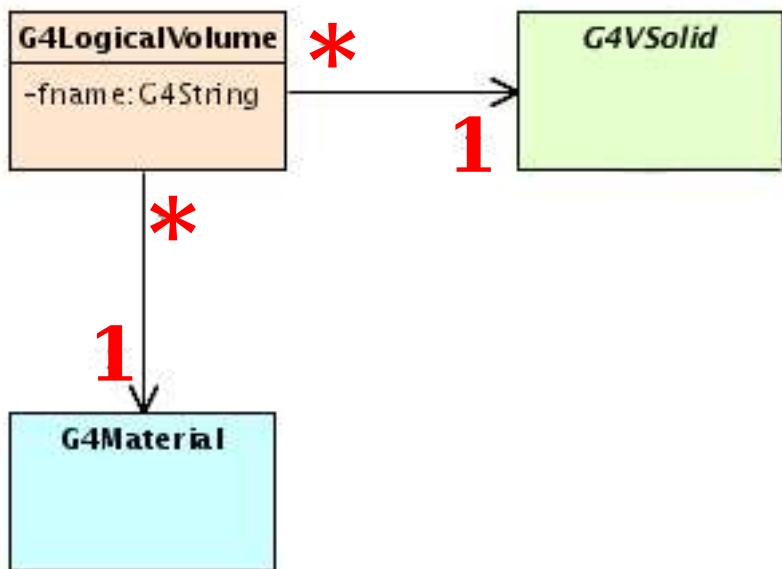
Example:

**CMS Electromagnetic Calorimeter**

~80.000 crystals which come in

~30 shapes

and are all made of lead-tungstate ( $\text{PbWO}_4$ ).



Two crystals of the CMS ECAL

- Concept of “**logical volume**” / “**volume**”

- **Mandatory** properties:

- Has a **shape**

- Has a **material**

```
class G4LogicalVolume
class G4VSolid
class G4Material
```

**Note:** solids and materials can be shared among volumes!

**Example:**

**CMS Electromagnetic Calorimeter**

~80.000 crystals which come in

~30 shapes

and are all made of lead-tungstate ( $\text{PbWO}_4$ ).

=>

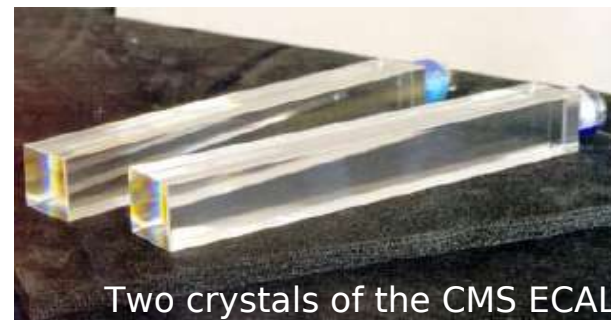
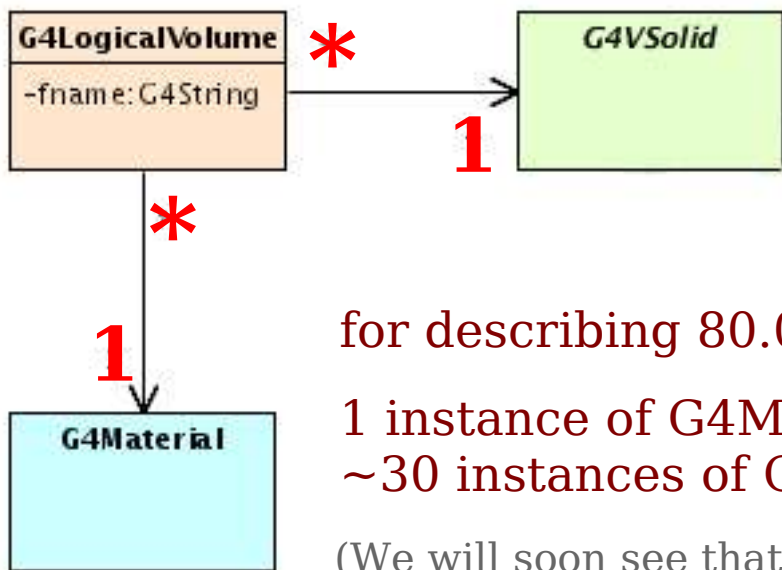
for describing 80.000 crystal volumes we need only:

1 instance of G4Material describing lead-tungstate

~30 instances of G4VSolid describing the shapes

(We will soon see that we don't necessarily need 80.000 instances of G4LogicalVolume

to describe all crystals because of the hierarchical character of the geometry model.)



Two crystals of the CMS ECAL

- Concept of “**logical volume**” / “**volume**”

- Constructor:

```
G4LogicalVolume(  
    G4VSolid* s,  
    G4Material* m,  
    const G4String& name,  
    G4FieldManager* fm = 0,  
    G4VSensitiveDetector* sd=0,  
    G4UserLimits* ul=0,  
    G4bool optimise=true)
```

The (mandatory) const G4String& argument is the name of the volume. It's not required to be unique, but it's a good & common practice to choose unique names ...

# Public interface of G4LogicalVolume ..

name	Type	Parameters
AddDaughter (md)	inline void	(G4VPhysicalVolume *)
BecomeEnvelopeForFastSimulation (md)	inline void	(G4FastSimulationManager *)
ClearDaughters (md)	inline void	()
ClearEnvelopeForFastSimulation (md)	void	(G4LogicalVolume *)
~G4LogicalVolume (md)	int	()
<b>G4LogicalVolume (md)</b>	<b>int</b>	<b>(G4VSolid *, G4Material *, const G4String &amp;, G4Fi</b>
GetBiasWeight (md)	inline G4double	()
GetDaughter (md)	inline G4VPhysicalVolume *	(const G4int)
GetFastSimulationManager (md)	inline G4FastSimulationManager *	()
GetFieldManager (md)	inline G4FieldManager *	()
GetMaterial (md)	inline G4Material *	()
GetMaterialCutsCouple (md)	inline const G4MaterialCutsCouple *	()
GetName (md)	inline G4String	()
GetNoDaughters (md)	inline G4int	()
GetRegion (md)	inline G4Region *	()
GetSensitiveDetector (md)	inline G4VSensitiveDetector *	()
GetSmartless (md)	inline G4double	()
GetSolid (md)	inline G4VSolid *	()
GetUserLimits (md)	inline G4UserLimits *	()
GetVisAttributes (md)	inline const G4VisAttributes *	()
GetVoxelHeader (md)	inline G4SmartVoxelHeader *	()
IsAncestor (md)	G4bool	(const G4VPhysicalVolume *)
IsDaughter (md)	inline G4bool	(const G4VPhysicalVolume *)
IsRegion (md)	inline G4bool	()
IsRootRegion (md)	inline G4bool	()
IsToOptimise (md)	inline G4bool	()
operator==(md)	int	(const G4LogicalVolume &)
PropagateRegion (md)	inline void	()
RemoveDaughter (md)	inline void	(const G4VPhysicalVolume *)
SetBiasWeight (md)	inline void	(G4double)
SetFieldManager (md)	void	(G4FieldManager *, G4bool)
SetMaterial (md)	inline void	(G4Material *)
SetMaterialCutsCouple (md)	inline void	(G4MaterialCutsCouple *)
SetName (md)	inline void	(const G4String &)
SetOptimisation (md)	inline void	(G4bool)
SetRegion (md)	inline void	(G4Region *)
SetRegionRootFlag (md)	inline void	(G4bool)
SetSensitiveDetector (md)	inline void	(G4VSensitiveDetector *)
SetSmartless (md)	inline void	(G4double)
SetSolid (md)	inline void	(G4VSolid *)
SetUserLimits (md)	inline void	(G4UserLimits *)
SetVisAttributes (md)	inline void	(const G4VisAttributes &)
SetVisAttributes (md)	inline void	(const G4VisAttributes *)
SetVoxelHeader (md)	inline void	(G4SmartVoxelHeader *)



# Public interface of G4LogicalVolume ...

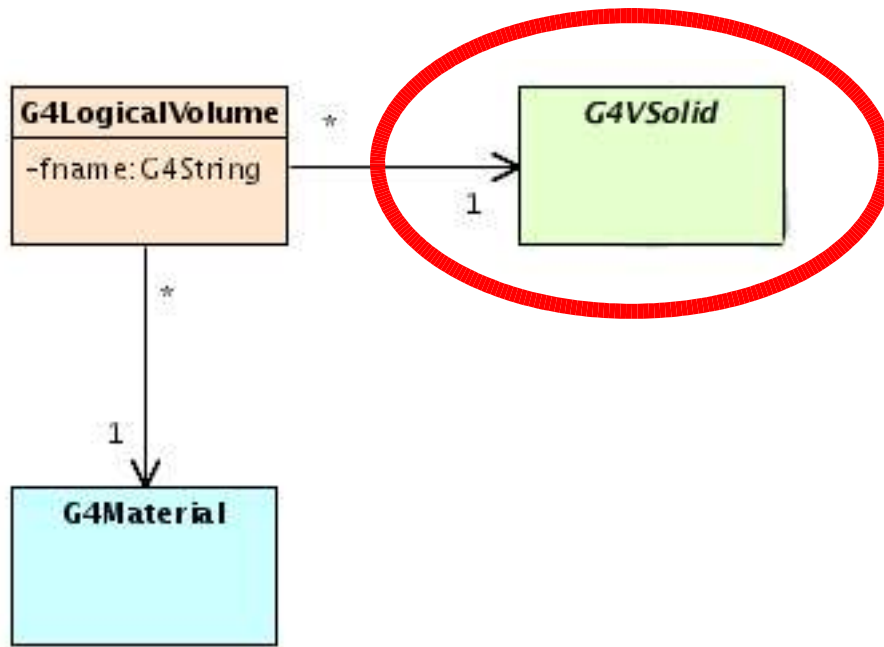
name	Type	Parameters
AddDaughter(md)	inline void	(G4VPhysicalVolume *)
BecomeEnvelopeForFastSimulation(md)	inline void	(G4FastSimulationManager *)
ClearDaughters(md)	inline void	()

## Don't be scared!

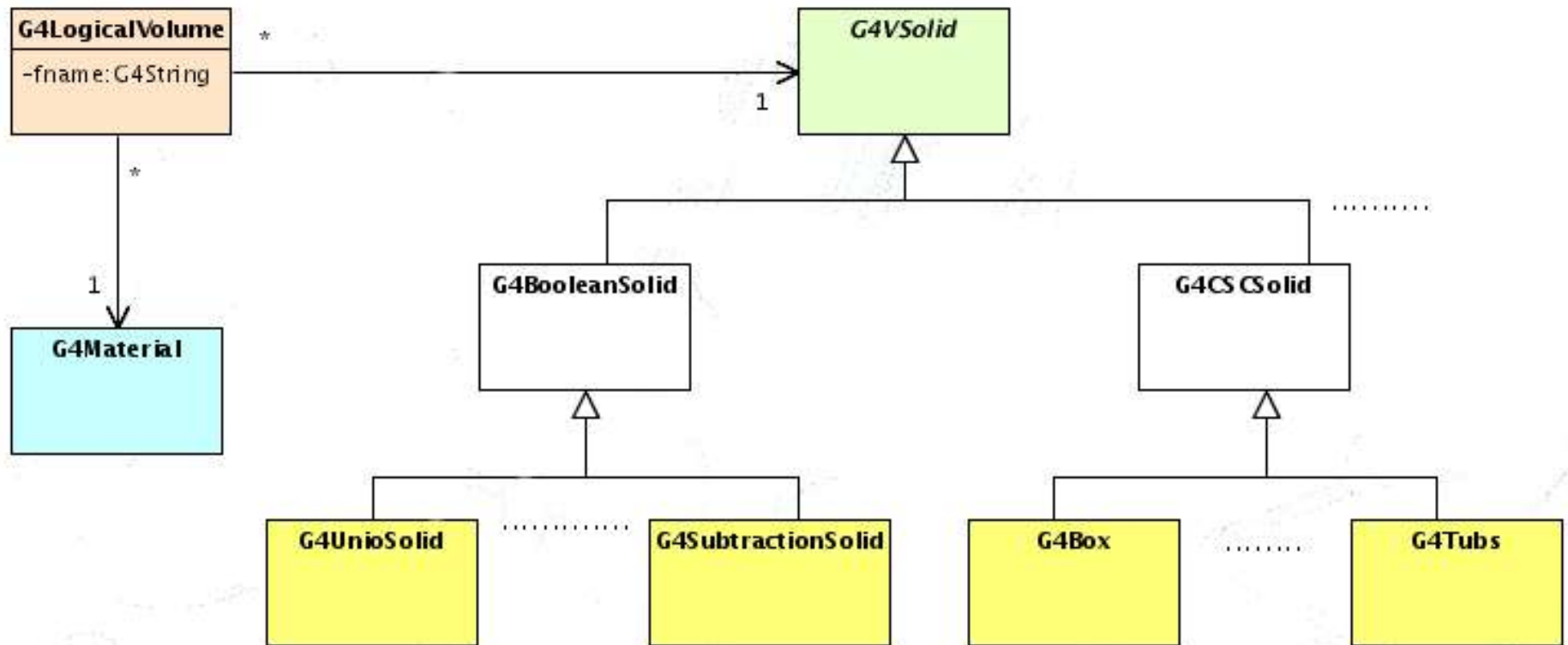
- Many G4-classes have a rich public interface
- Most methods have obvious signatures, e.g. G4Material\* GetMaterial() const;
- Unfortunately, all over Geant4 a clear distinction between user-methods and methods internally used by the Geant4 kernel only is lacking ...
- Read the official G4-documentation!
- ... and don't be scared!

SetUserLimits(md)	inline void	(G4UserLimits *)
SetVisAttributes(md)	inline void	(const G4VisAttributes &)
SetVisAttributes(md)	inline void	(const G4VisAttributes *)
SetVoxelHeader(md)	inline void	(G4SmartVoxelHeader *)

# Solids



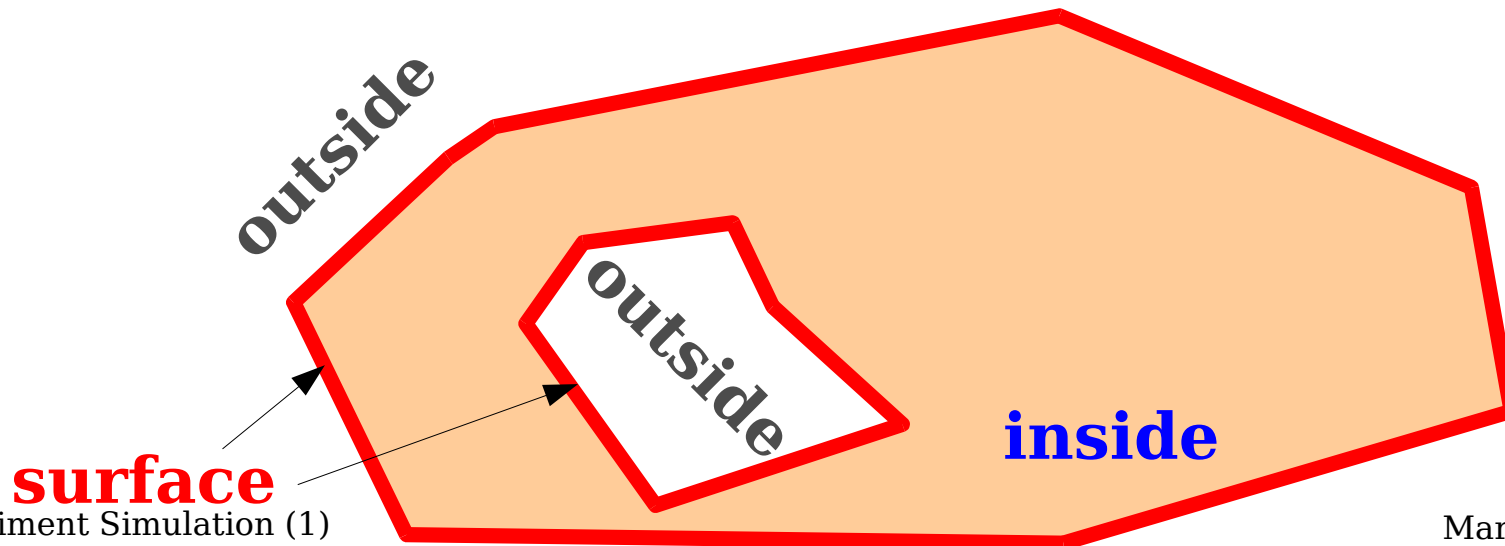
# Solids



- Geant4 ships with a large set of solid classes
- The G4 kernel uses solids only via their abstract interface **G4VSolid**

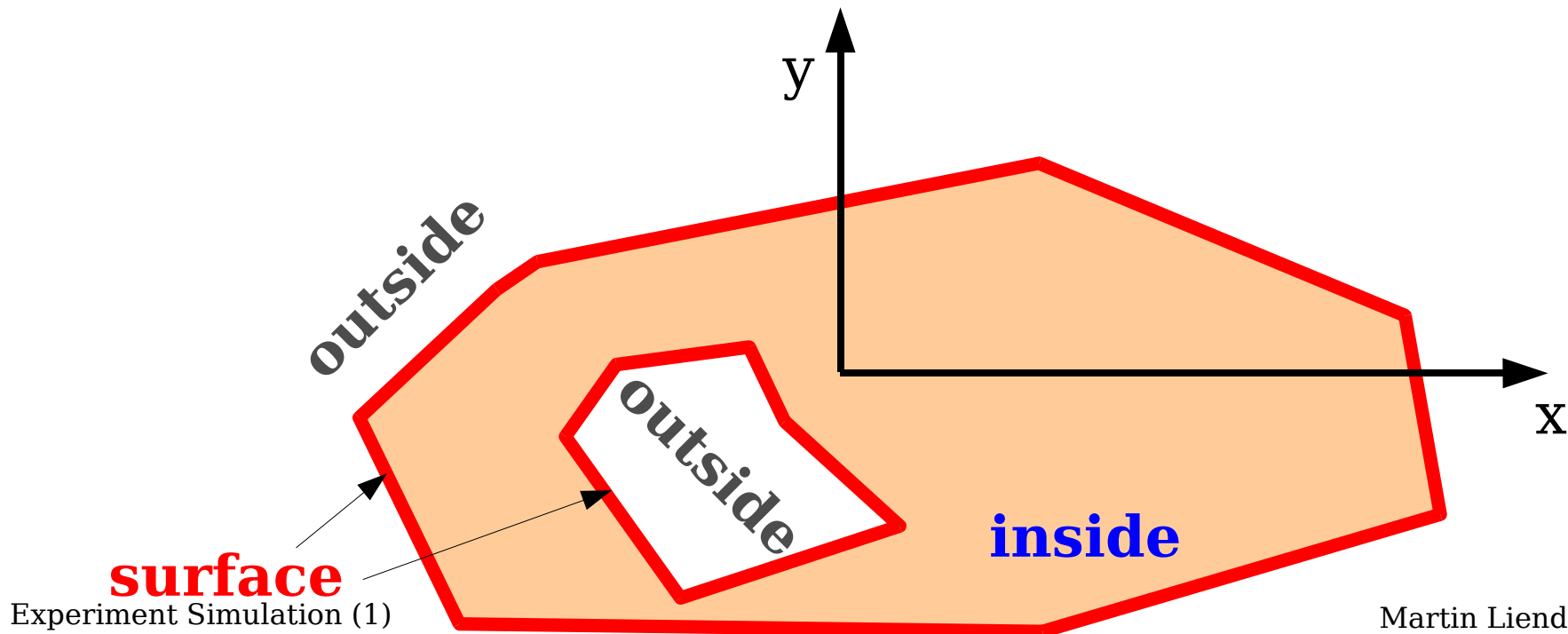
# Properties of Solids

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



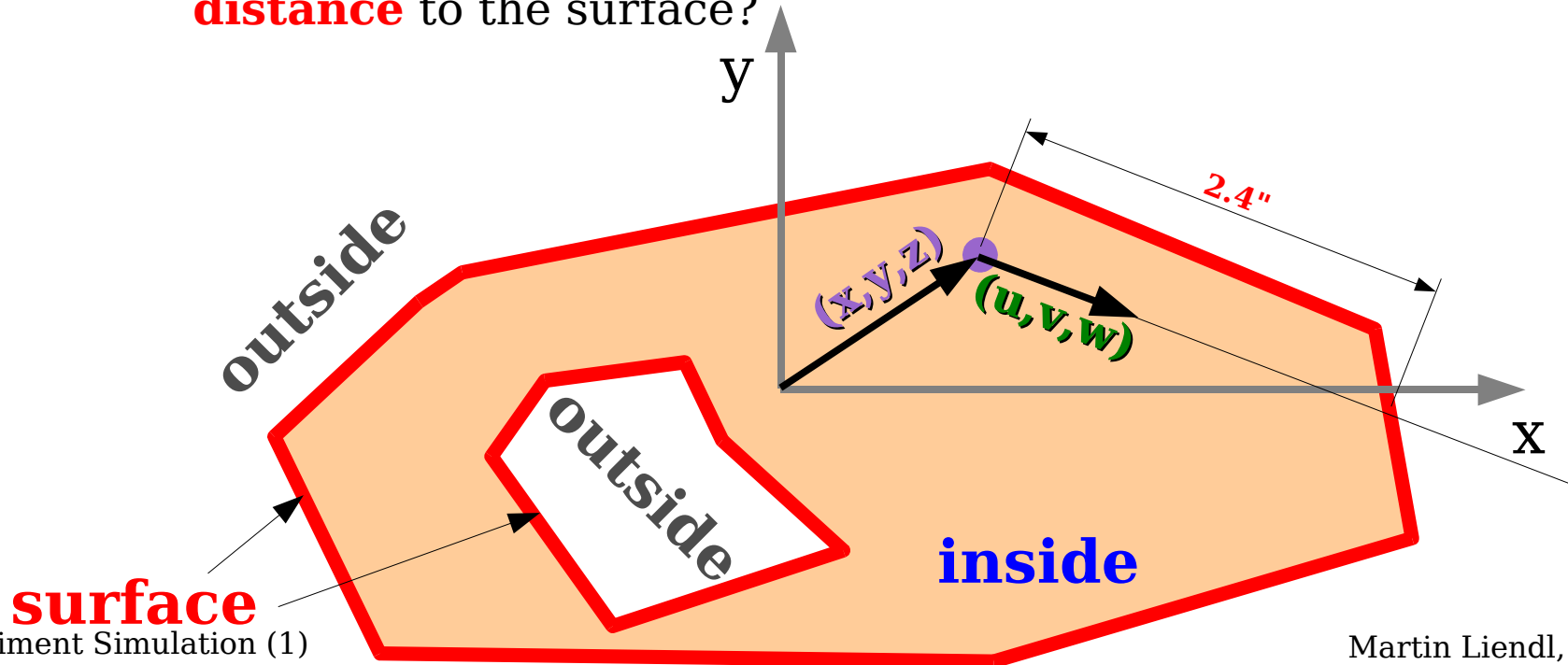
# Properties of Solids

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



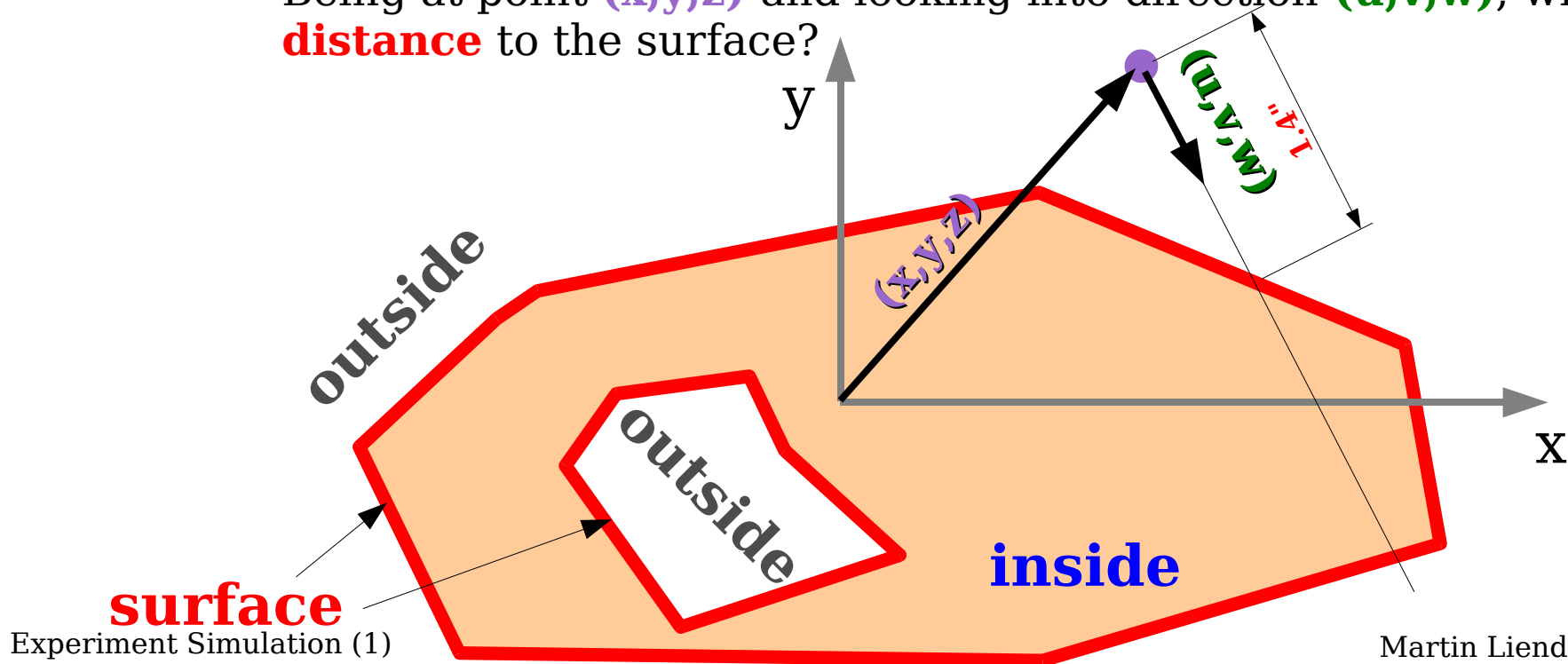
# Properties of Solids

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

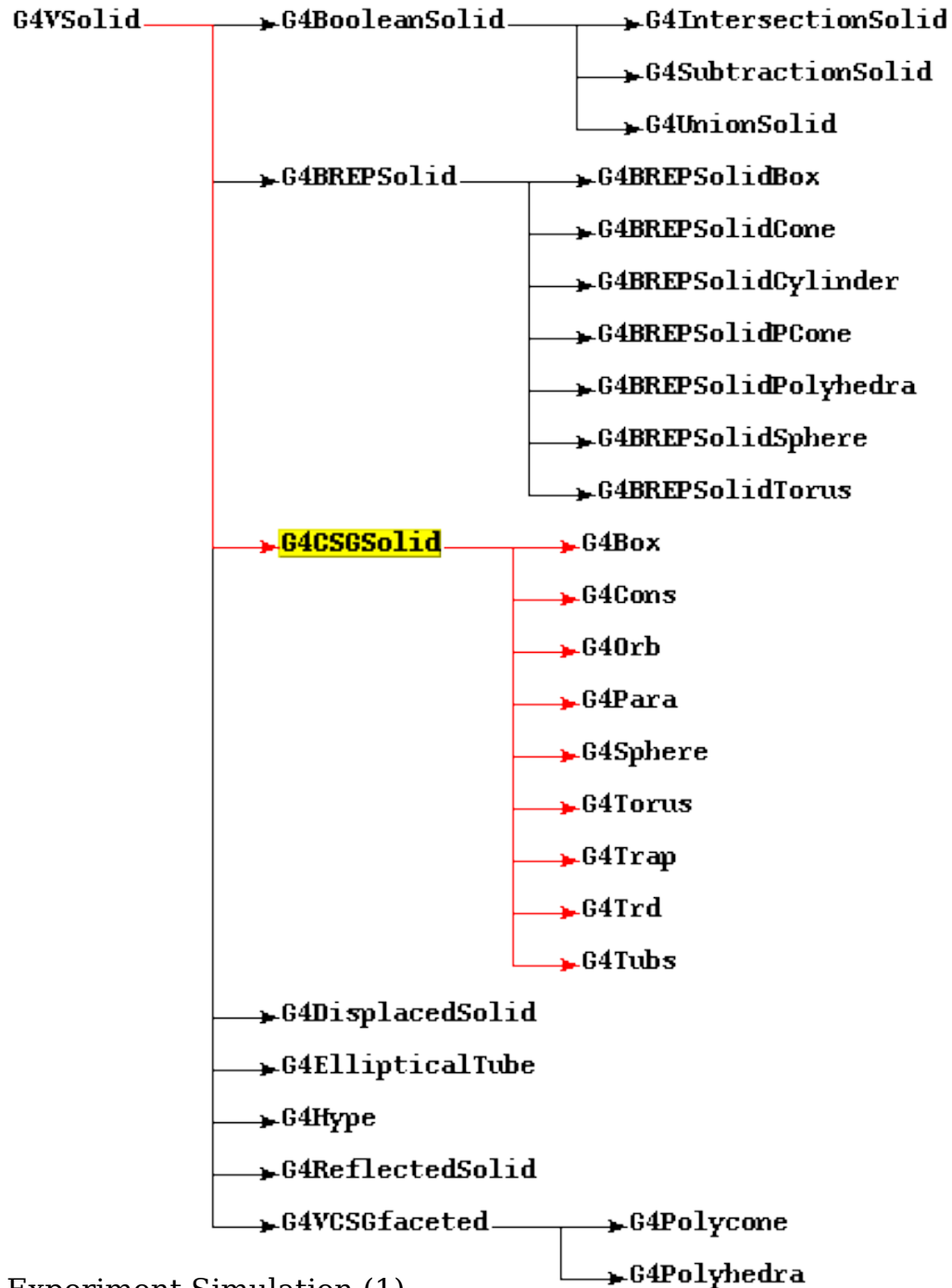


# Properties of Solids

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

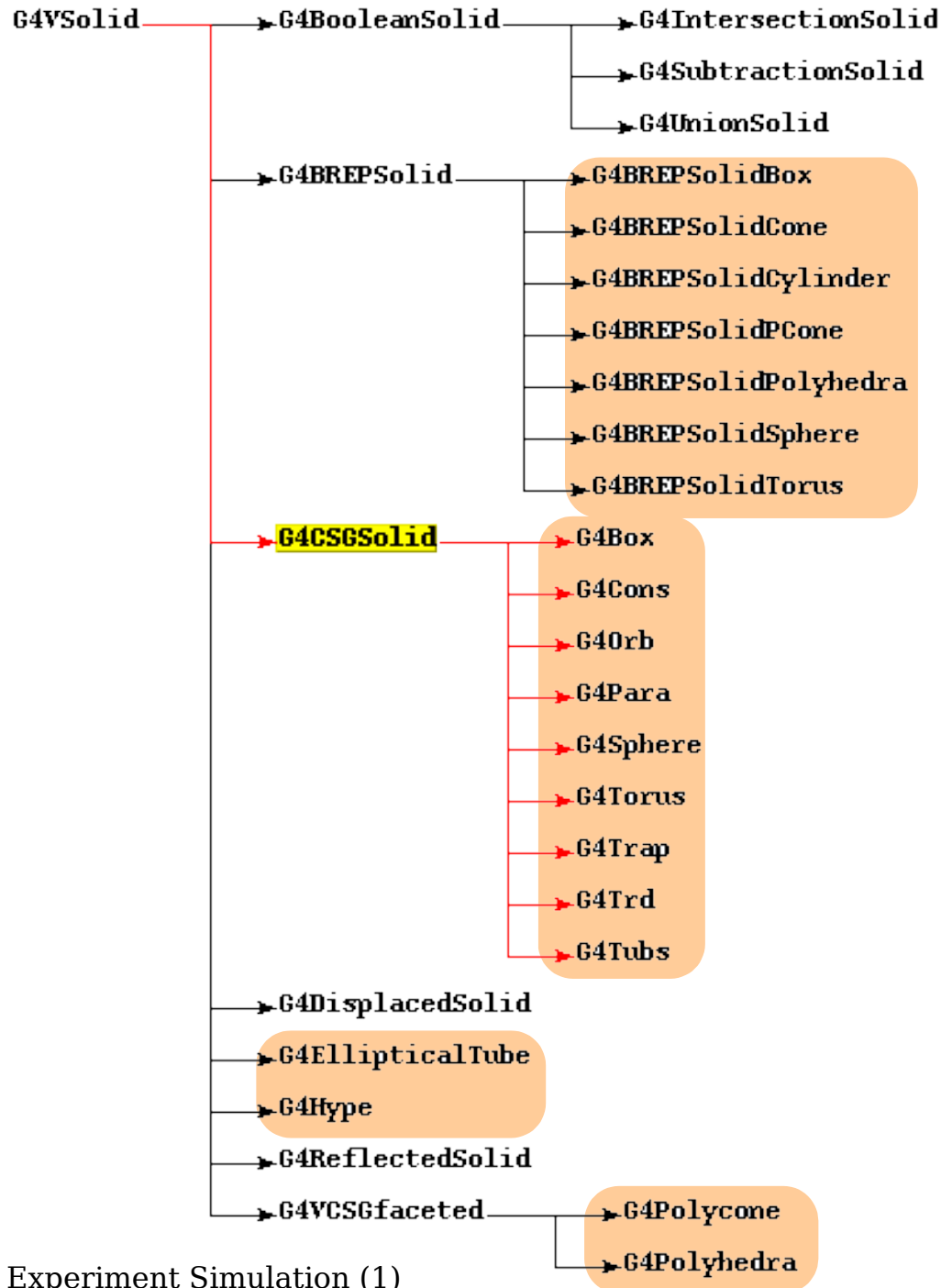


# Inheritance Hierarchy





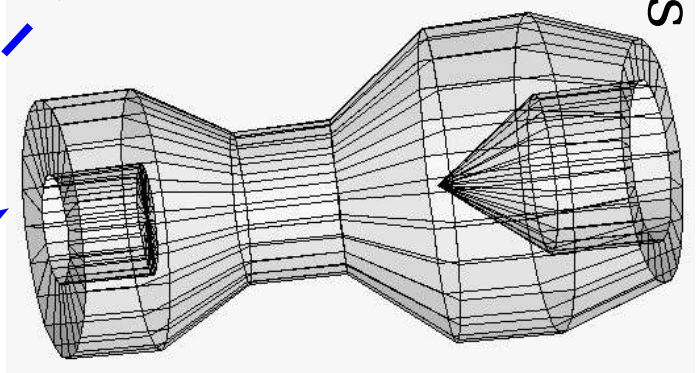
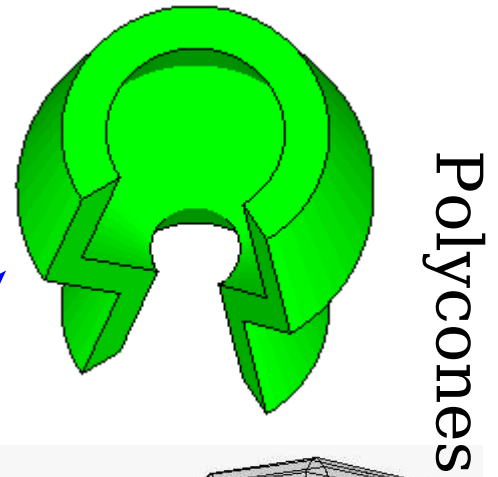
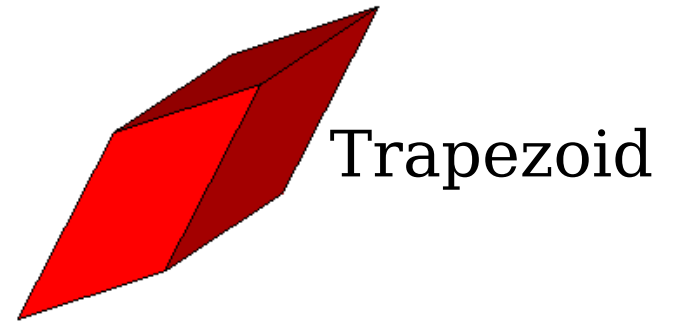
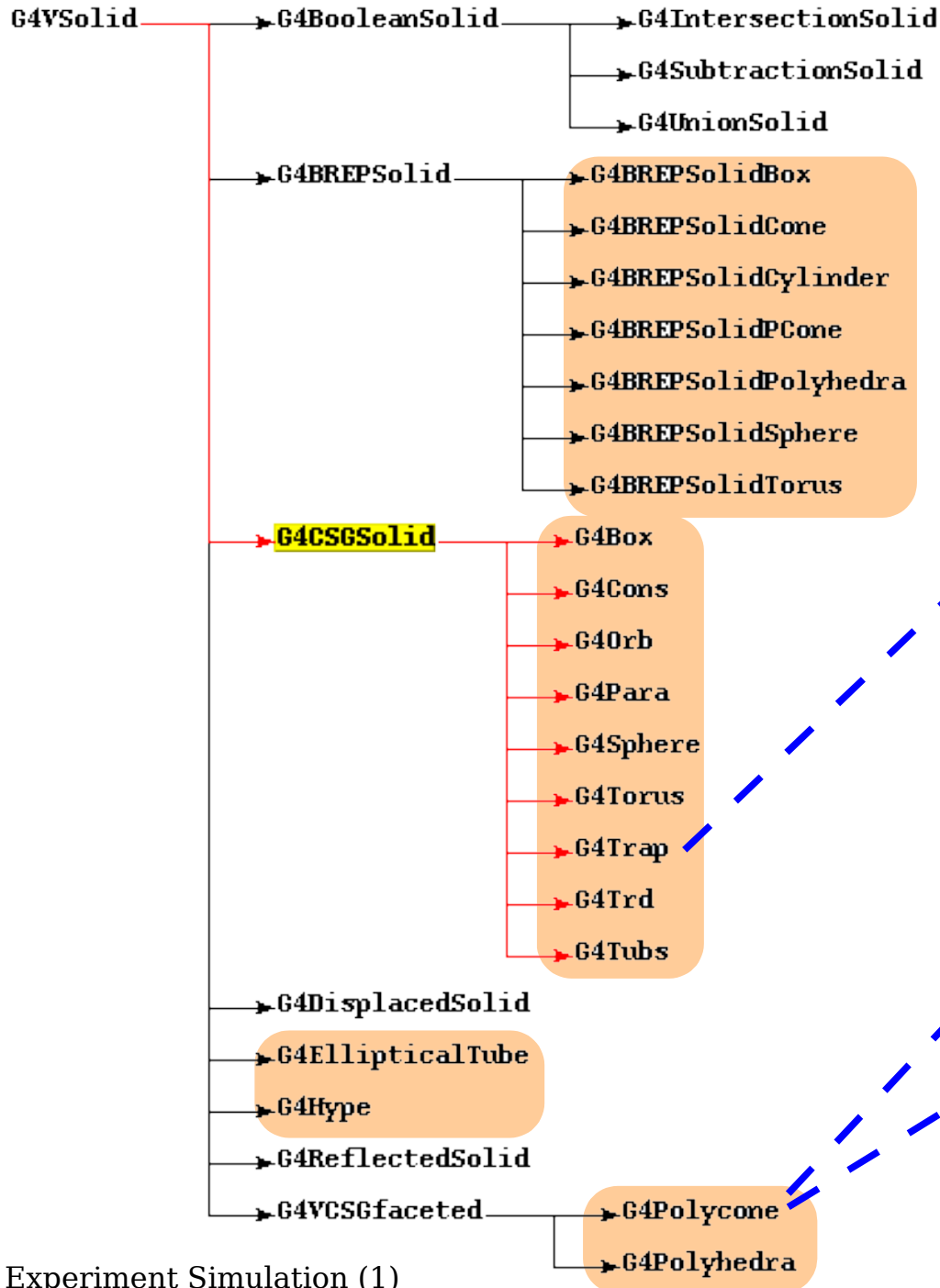
# “Atomic” Solids



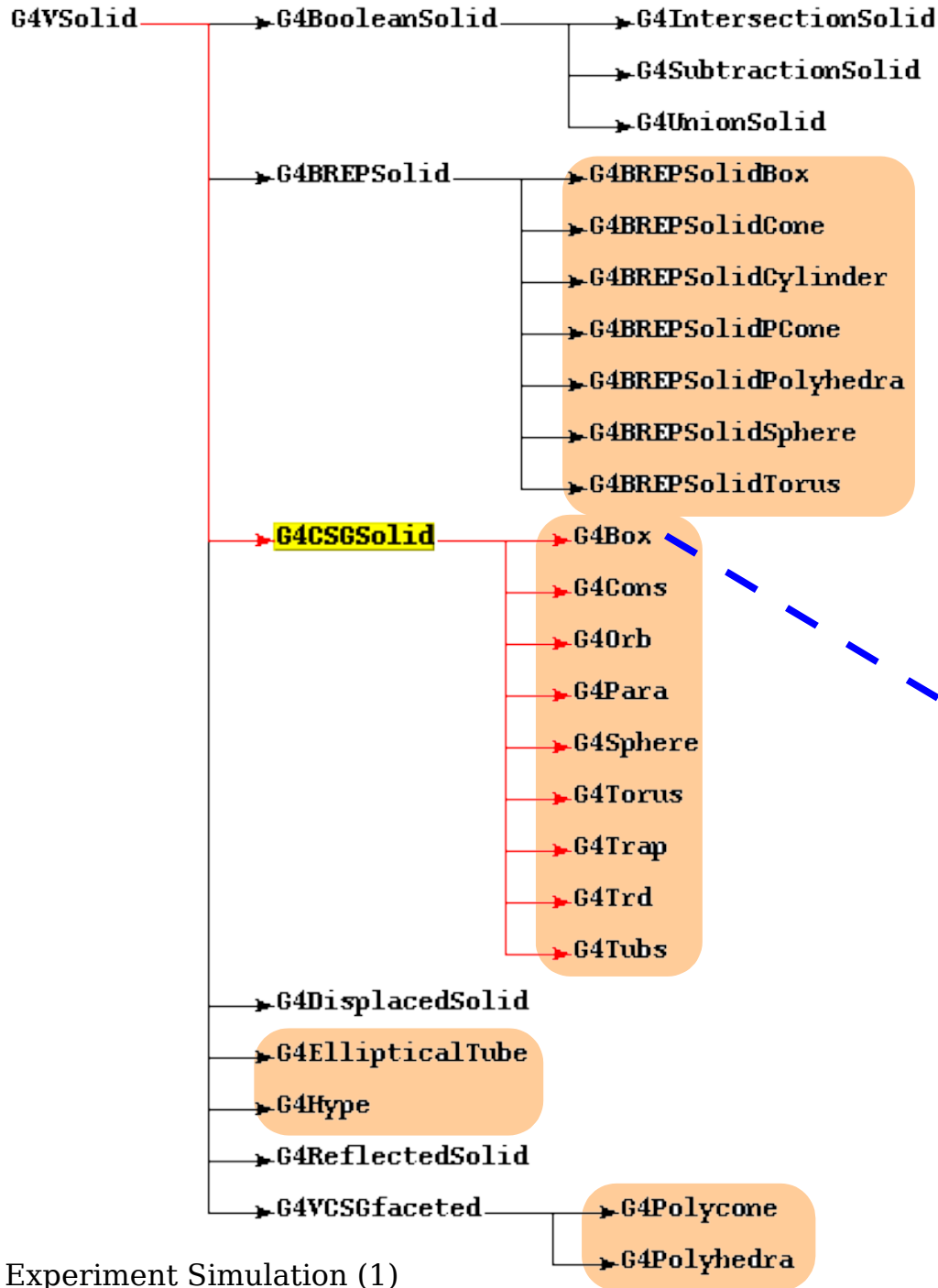
One instance of one of these classes defines a solid which is treated as a whole (atomic) by the Geant4 kernel.

Constructors take the parameters for the dimensions of the solids.

# "Atomic" Solids



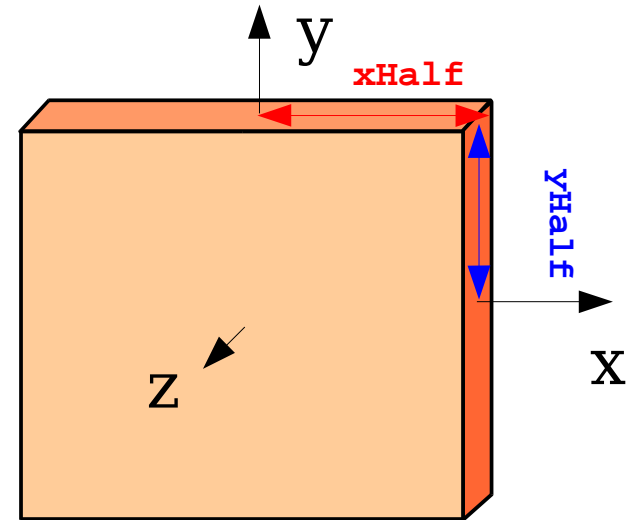
# "Atomic" Solids



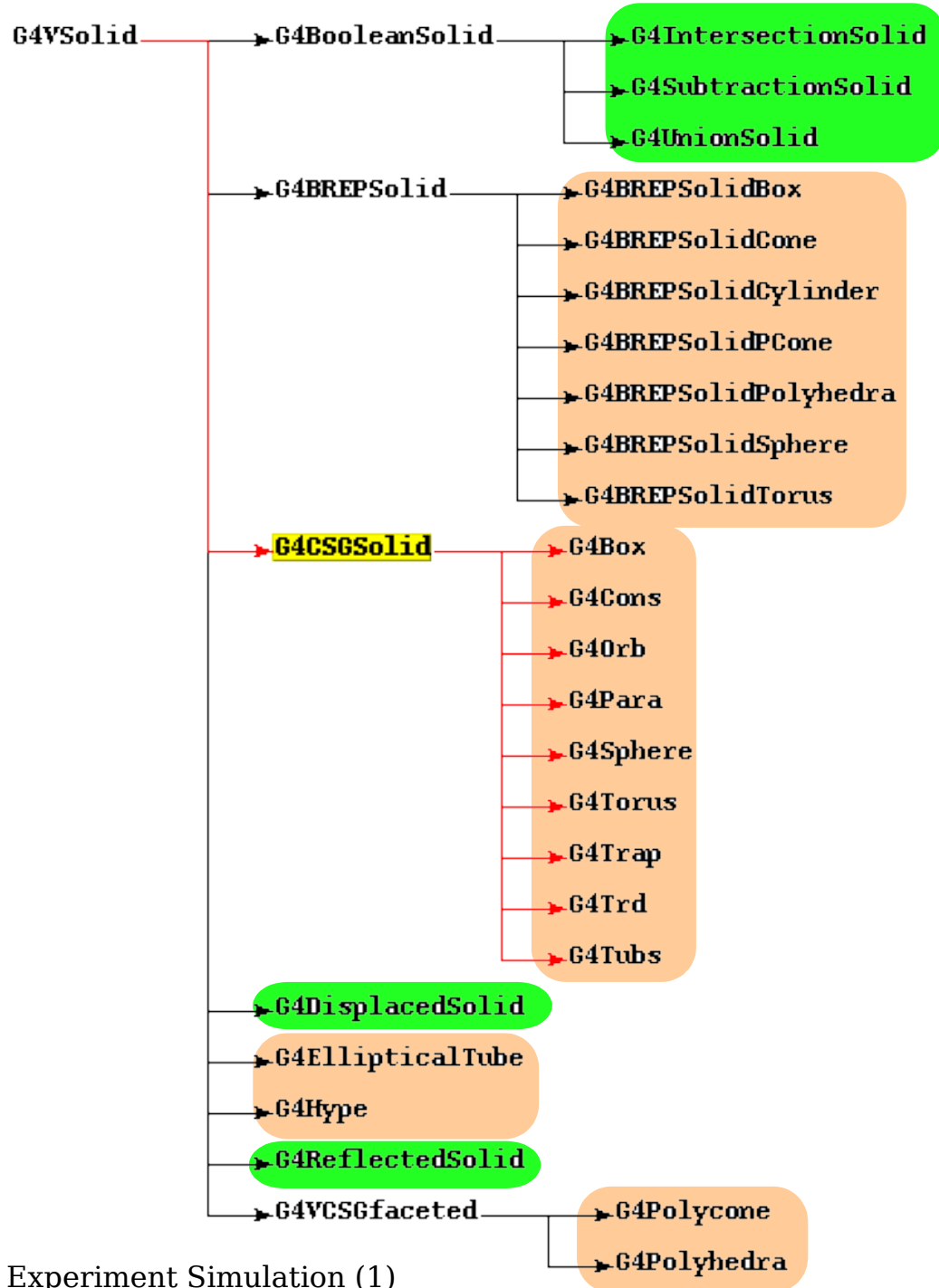
```

G4double xHalf,yHalf,zHalf;

G4VSolid* solid =
  new G4Box("Wafer",
           xHalf=3.2*cm,
           yHalf=2.7*cm,
           zHalf=1.1*mm);
  
```



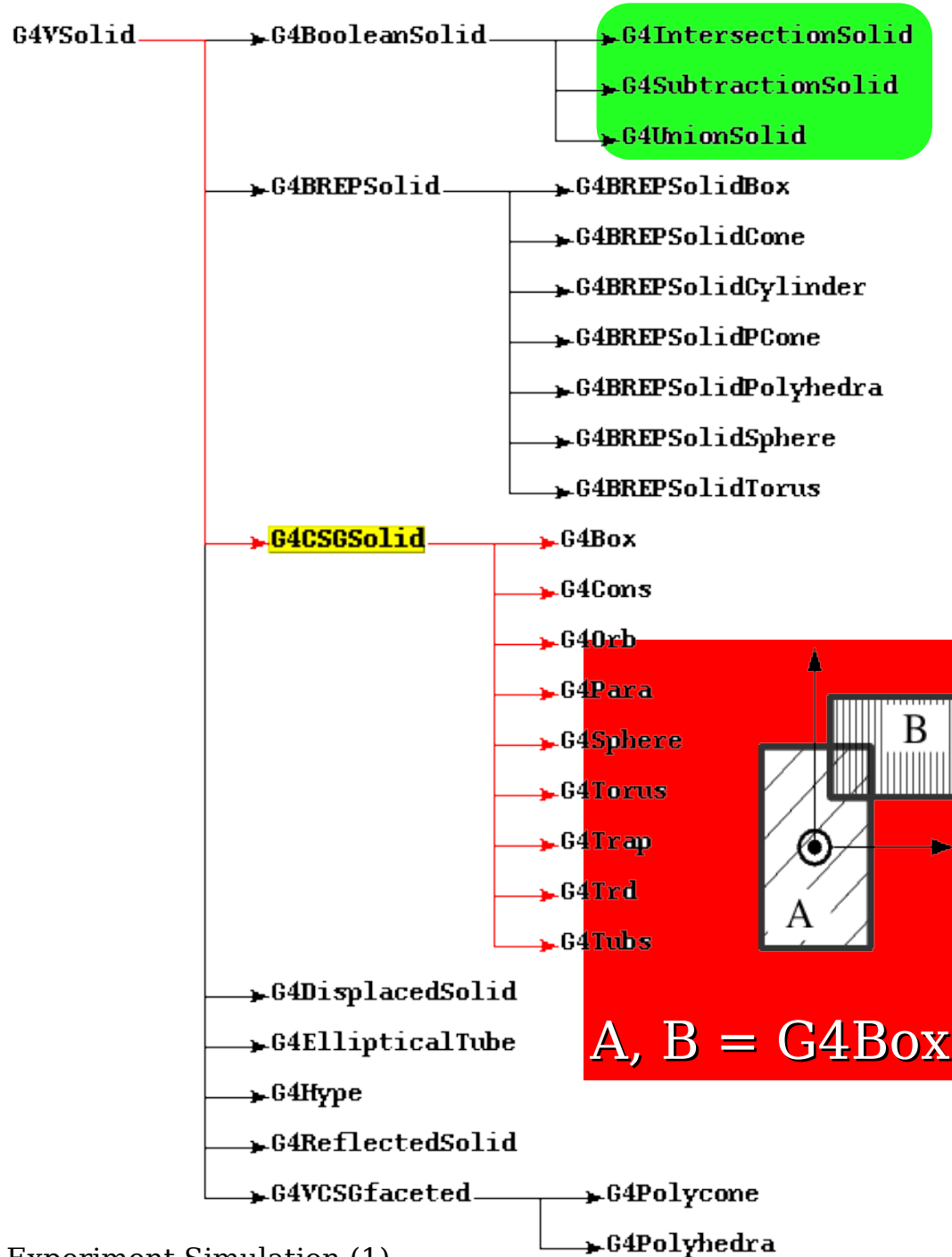
# Composite Solids



One instance **S** of a composite solid refers to at least one other instance of a solid **S'**

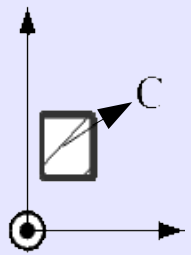
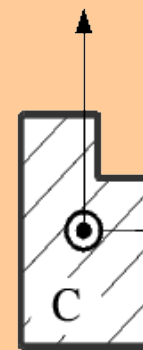
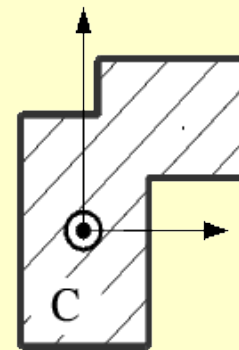
**S** defines certain operations on **S'**

# Boolean Solids



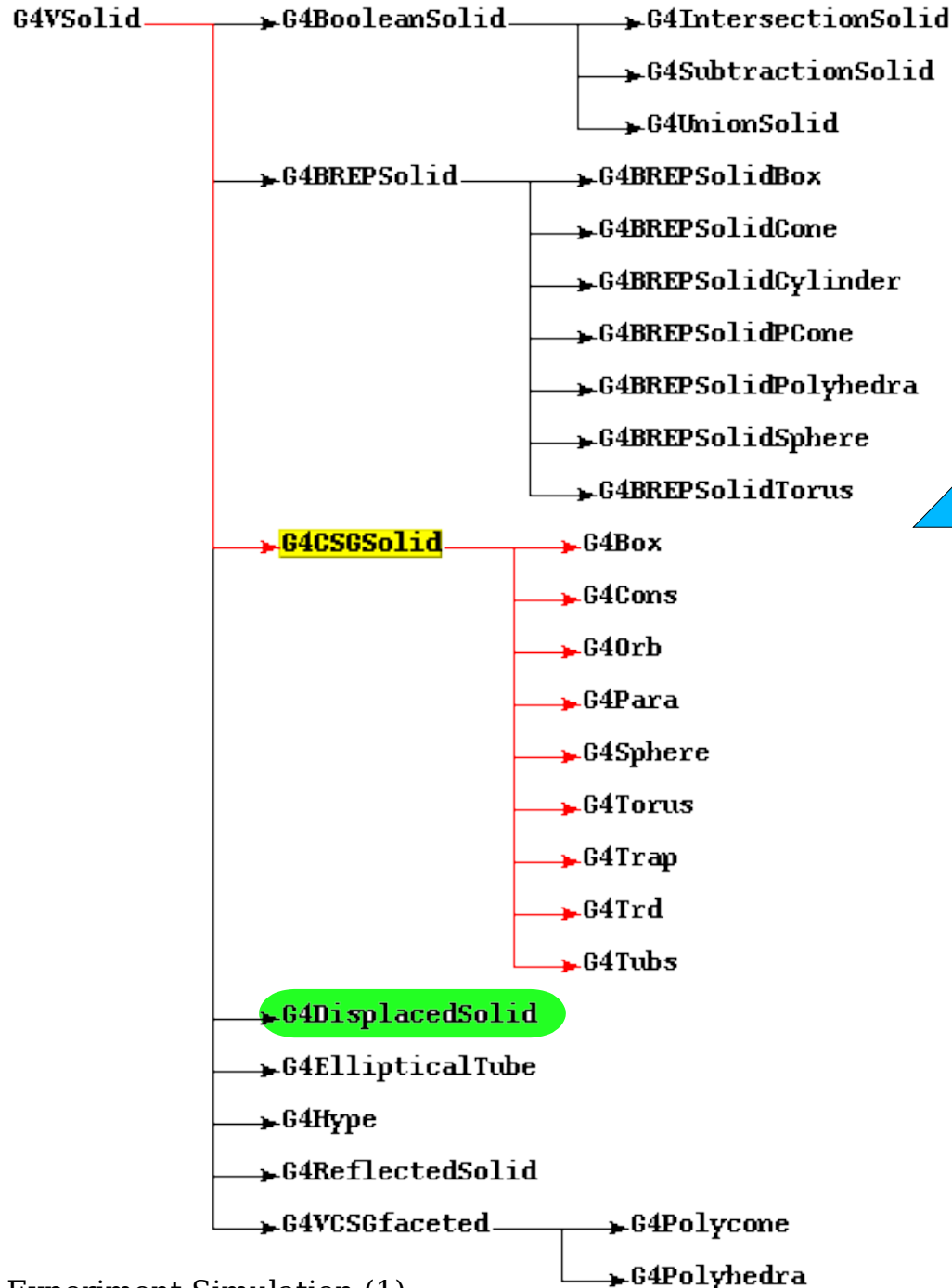
$$C = G4UnionSolid(A,B)$$

$$C = G4SubtractionSolid(A,B)$$

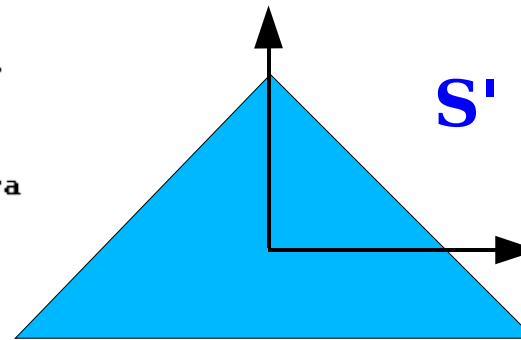


$$C = G4IntersectionSolid(A,B)$$

# Displaced Solid

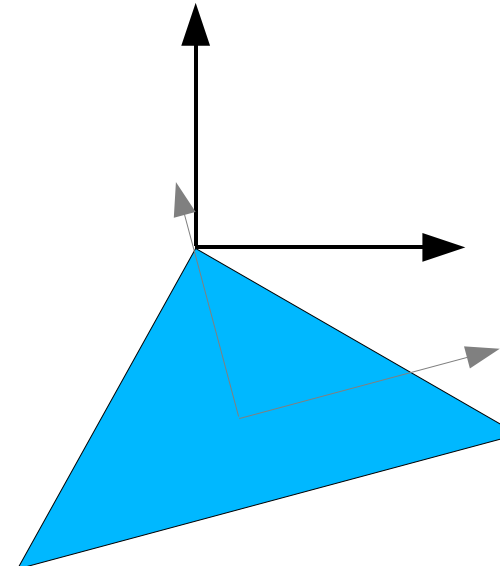


Displaces the frame of reference

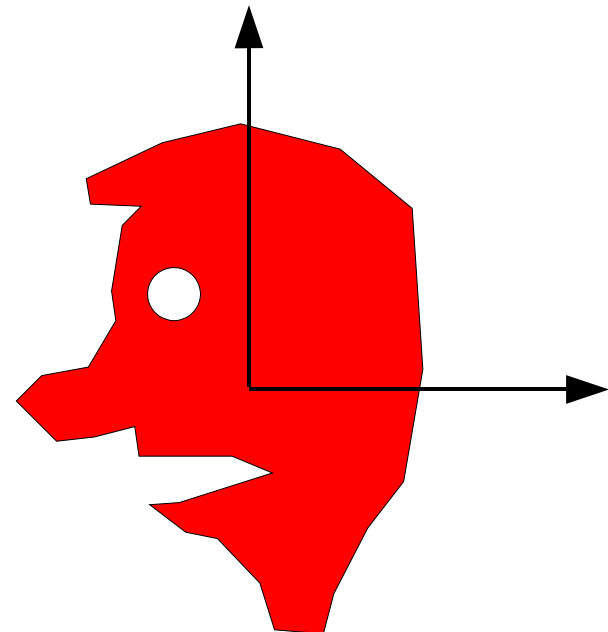
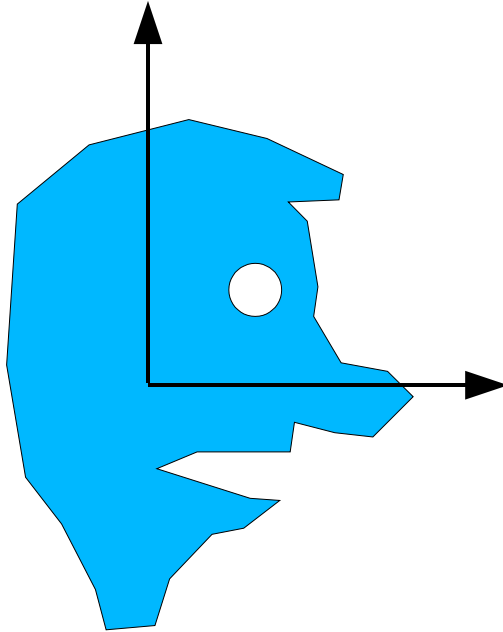
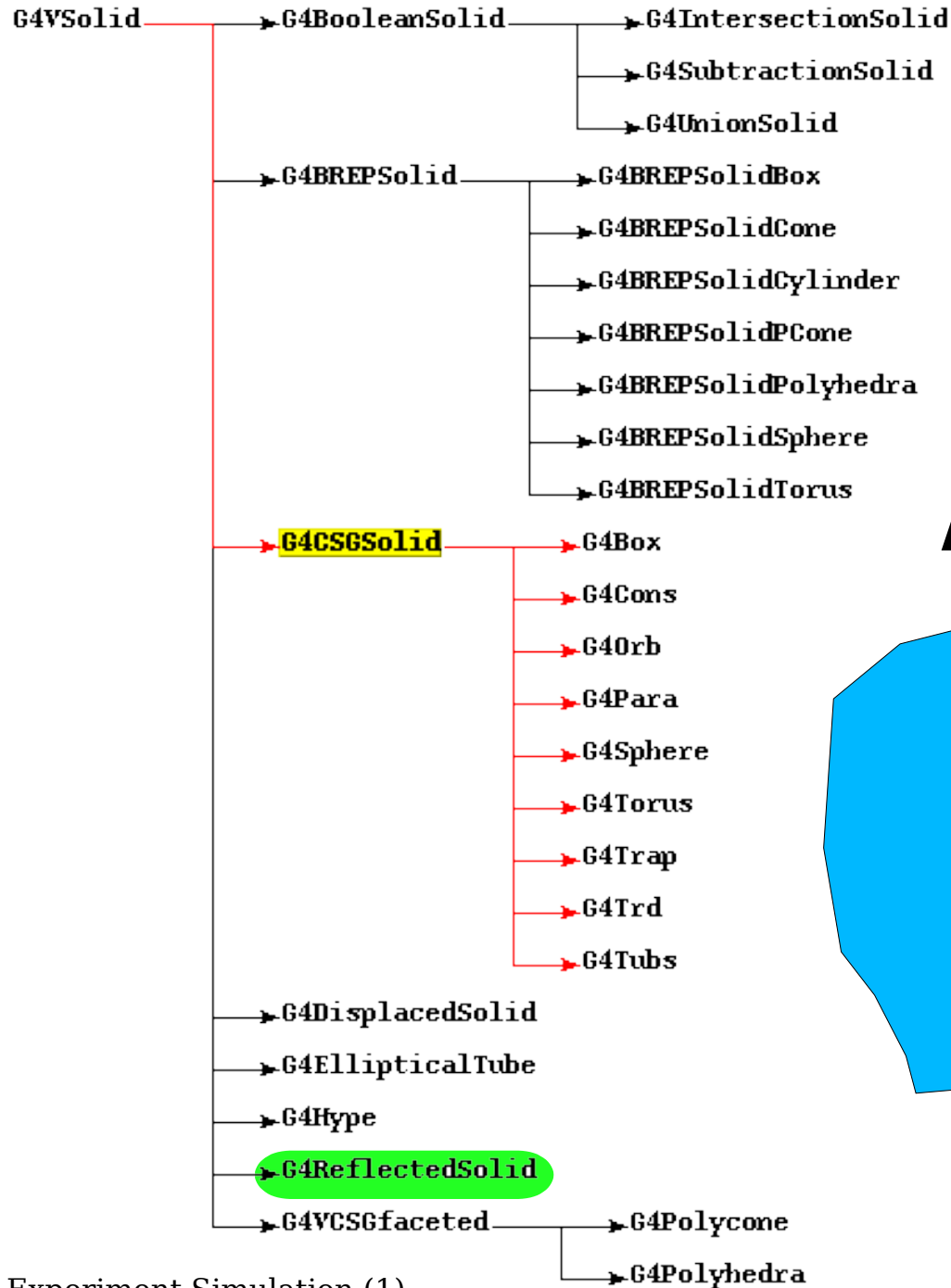


$S' = \text{G4Trap}$

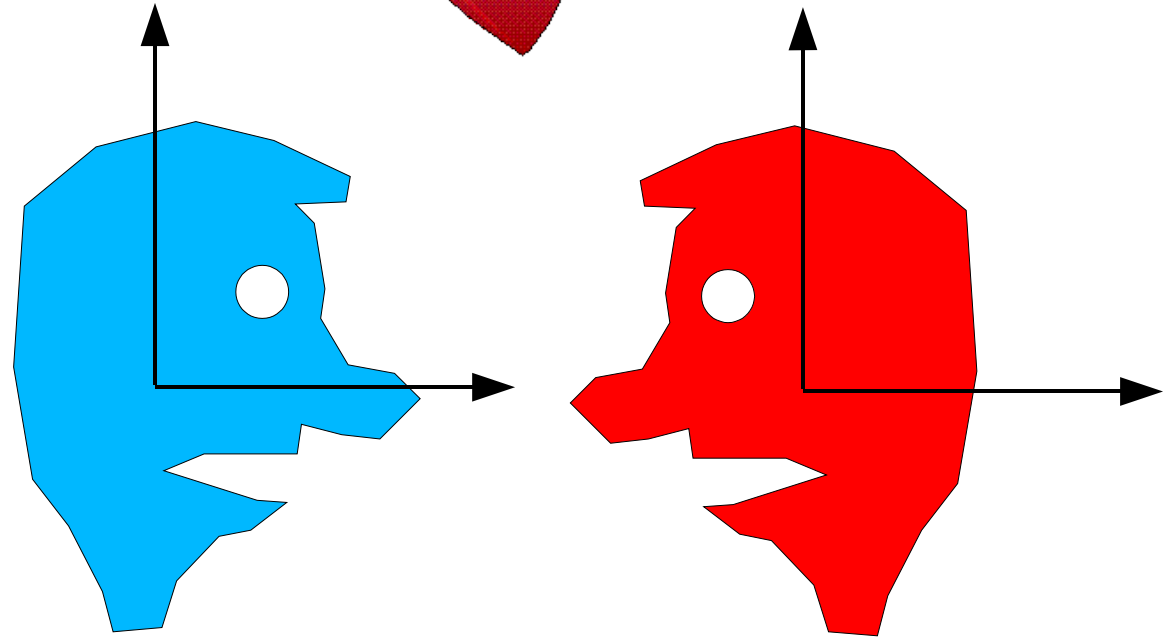
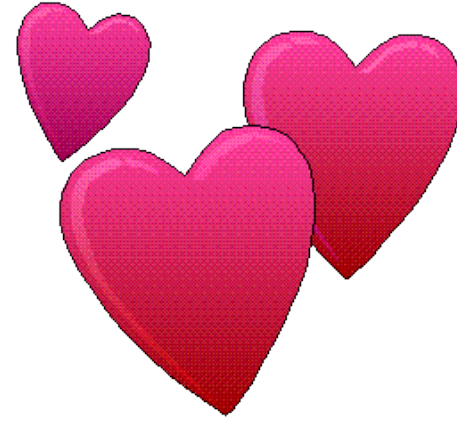
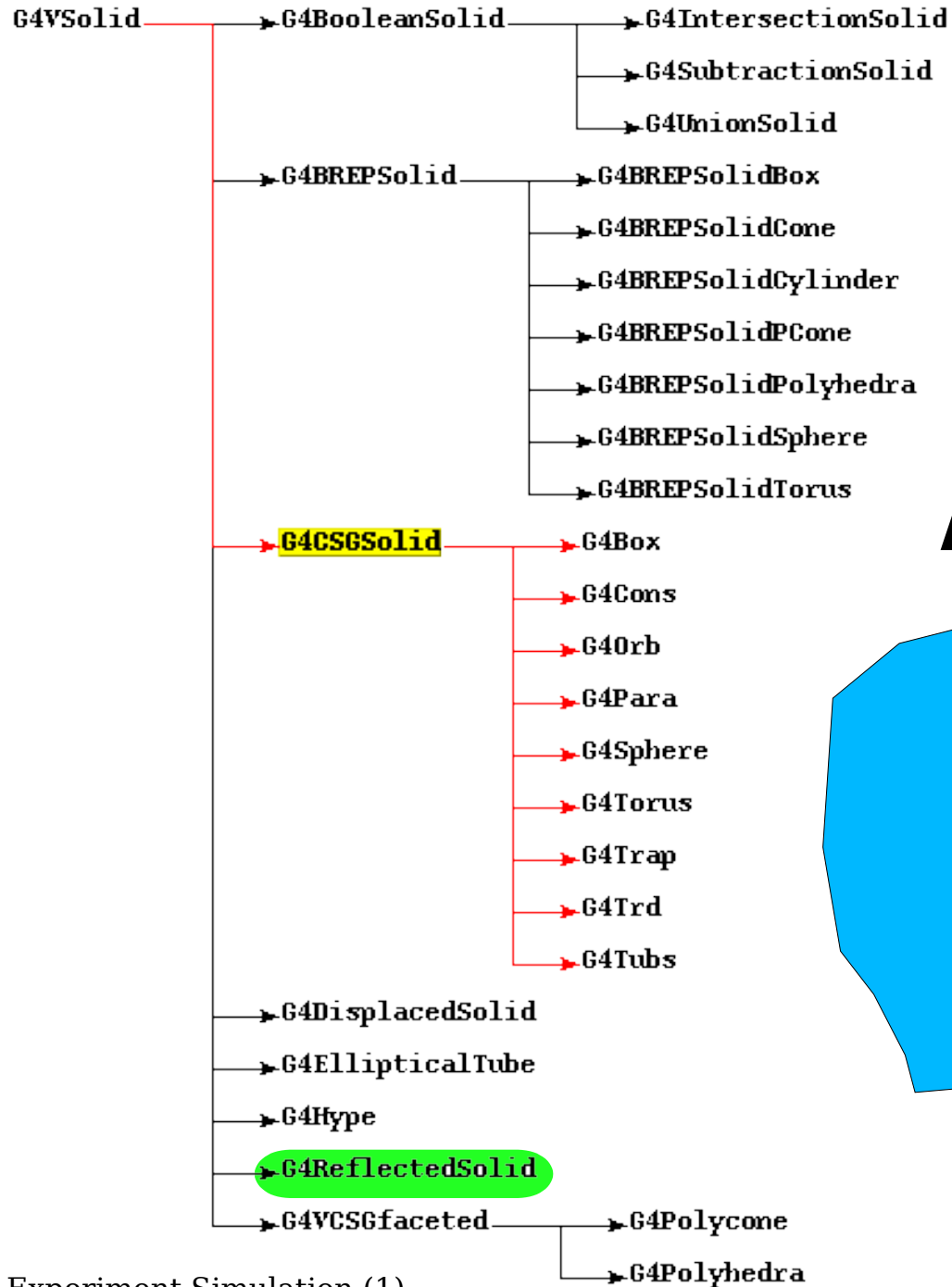
$S = \text{G4DisplacedSolid}(S', T, R)$



# Reflected Solid

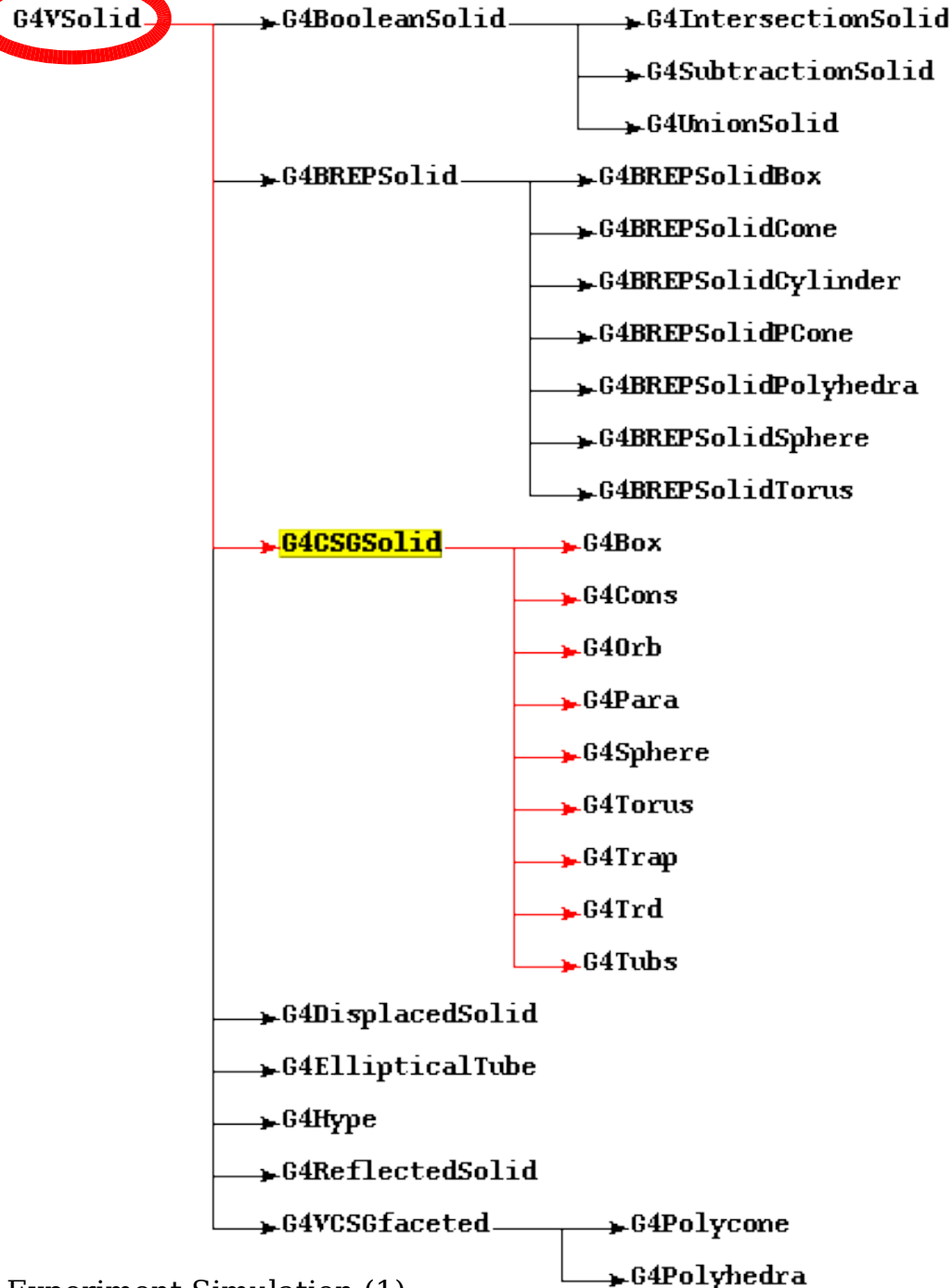


# Reflected Solid





# Do it yourself!



By implementing the G4VSolid interface, you can extend the list of solids by your own solids!

smallprint:

... but looking into the G4 code for existing solids, you'll soon realize, that it's not so easy ...

# Recap ...

What we have up to now:

- **Physical Unit**
  - multiply, divide
- **Material**
  - simple, composites
- **Solid**
  - frame of reference, inside, outside, surface
  - atomic, composite
- **Logical Volume**
  - points to a mandatory solid and a mandatory material

Example:



How can we describe a complex detector using these ingredients?