GEANT4: Simulation of Physical Processes

Mikhail Demichev, Alexey Zhemchougov

JINR (Dubna), MIPT (Moscow)
E-mail:
zhemchugov@jinr.ru
mdemichev@jinr.ru

Summer School
Bolshye Koty - 2016

Plan of the Lectures

- Lection 1
  - Why and when do we need simulation?
  - Monte-Carlo method. Available software
  - The basics of Geant4, the minimalistic simulation code
  - Definition of materials and geometry
  - Definition of particle source
  - Tasks for the students
- Lection 2
  - An extended simulation program.
  - Physical models, procesess and physical lists
  - User actions and user interface
  - Output of your simulation, ROOT trees and histograms

What is meant by «simulation»?

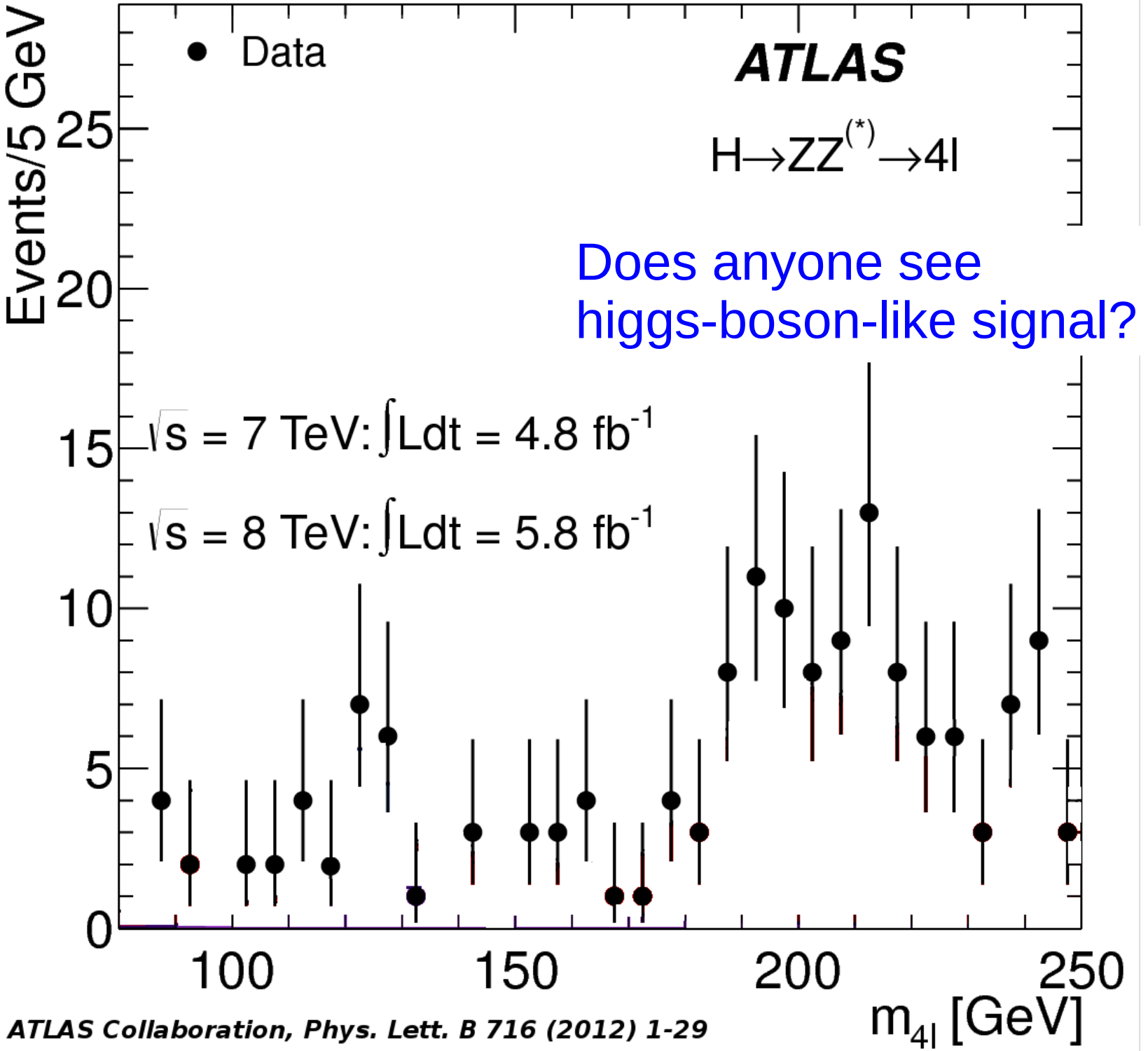
We simulate physical processes and detector response in an experiment being planned or conducted.

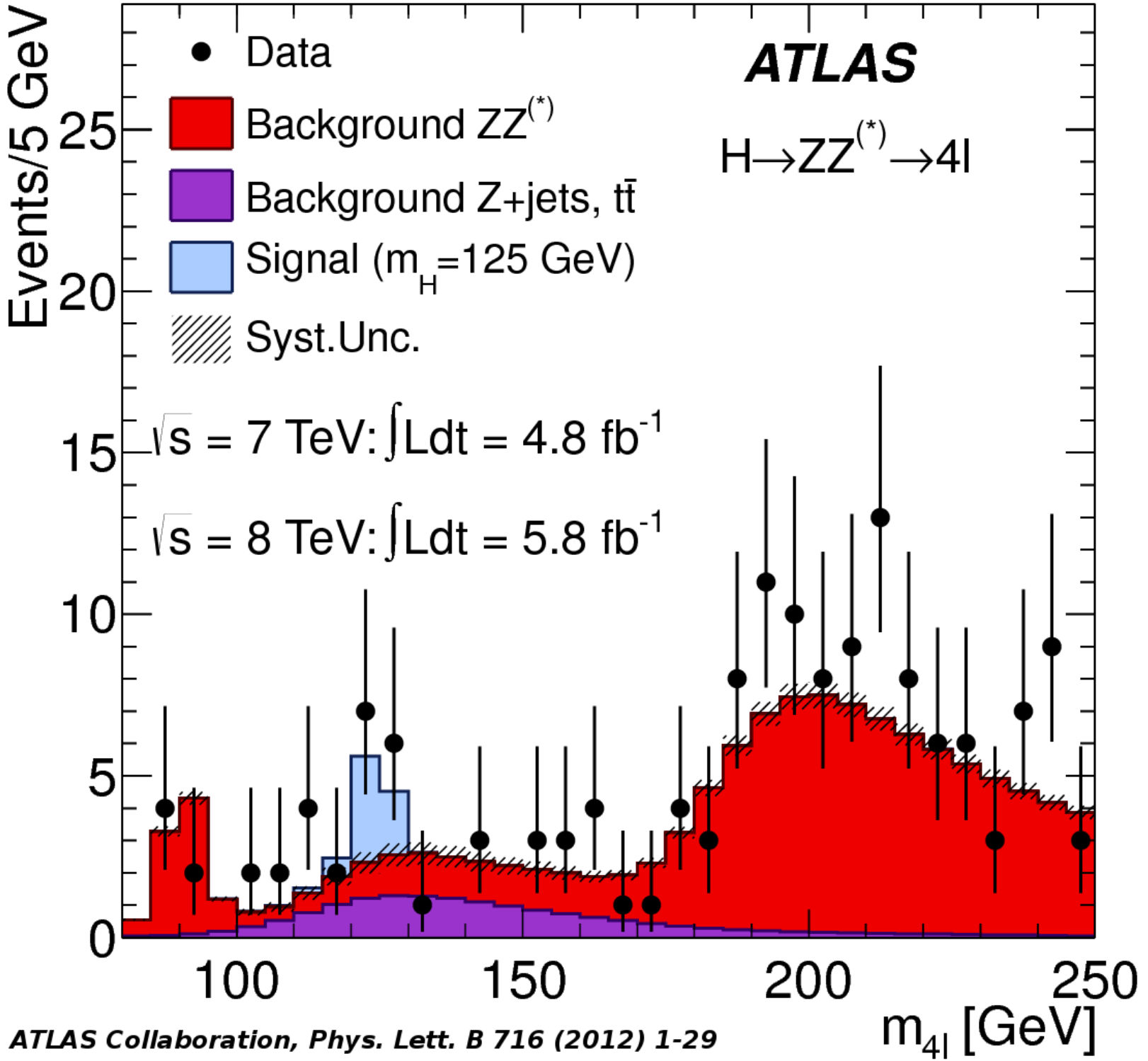
- A question:

  If we can use the computer to simulate all the physical processes so well can we abandon conducting physical experiments at all as they require much money and efforts?

- Another question:

  If we have already received experimental results then why should we do a simulation for them?


Data
ATLAS
H→ZZ(*)→4l
Does anyone see
higgs-boson-like signal?
√s = 7 TeV: ∫Ldt = 4.8 fb-1
√s = 8 TeV: ∫Ldt = 5.8 fb-1
Events/5 GeV
0
5
10
15
20
25
100
150
200
250
m4l [GeV]
ATLAS Collaboration, Phys. Lett. B 716 (2012) 1-29



ATLAS Collaboration, Phys. Lett. B 716 (2012) 1-29

Simulation goals

- Planning of experiment
  - Detector construction optimisation
  - Event reconstruction algorithms tuning and optimisation
  - Calculation of expected signal and backgrounds distributions
  - Estimation of measurement accuracy
- Analysis of experimental data
  - Analysis flow optimisation
  - Acceptance of experimental setup
  - Estimation of background contributions to the signal
  - Estimation of systematic errors
  - Comparison of theoretical prediction with experimental results

Monte-Carlo Method

Monte-Carlo method is a numerical method to solve applied mathematical problems by simulation of random variables and statistical analysis of their characteristics.

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION

Number 247 SEPTEMBER 1949 *Volume 44*

THE MONTE CARLO METHOD

NICHOLAS METROPOLIS AND S. ULAM
Los Alamos Laboratory

We shall present here the motivation and a general description of a method dealing with a class of problems in mathematical physics. The method is, essentially, a statistical approach to the study of differential equations, or more generally, of integro-differential equations that occur in various branches of the natural sciences.

This method was developed at the time of «Manhattan project»:

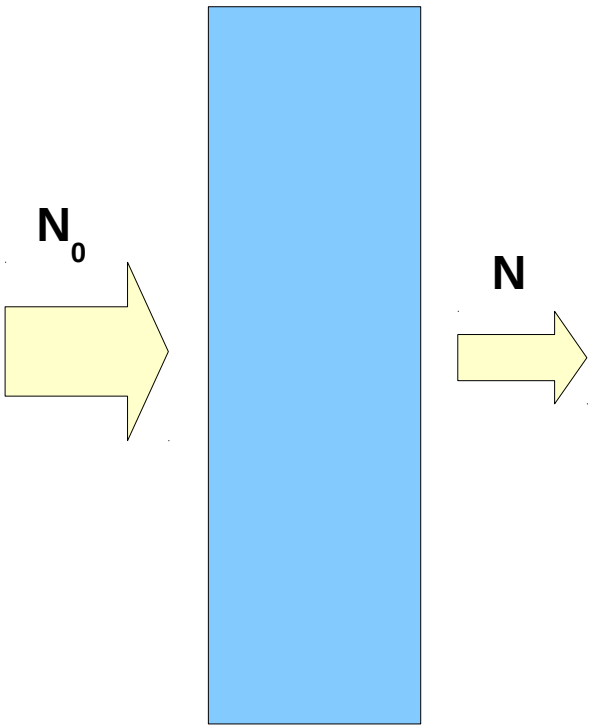
- **S.M. Ulam, J. von Neumann, "On combination of stochastic and deterministic processes". Bull. Amer. Math. Soc. 53 1120 (1947)**
- **S.M. Ulam, N. Metropolis, "The Monte-Carlo method", J. Amer. Statist. Assoc. 1949 , 44 Vol 247, 335-341**

Monte-Carlo Method

- The mathematical essence of Monte-Carlo method:

  An effective way to calculate finite multidimentional integrals

- An example:



Radiation loss in matter: Too simple?!

$dN = -\mu N\, dx$ **$N = N_0 \exp(-\mu x)$**

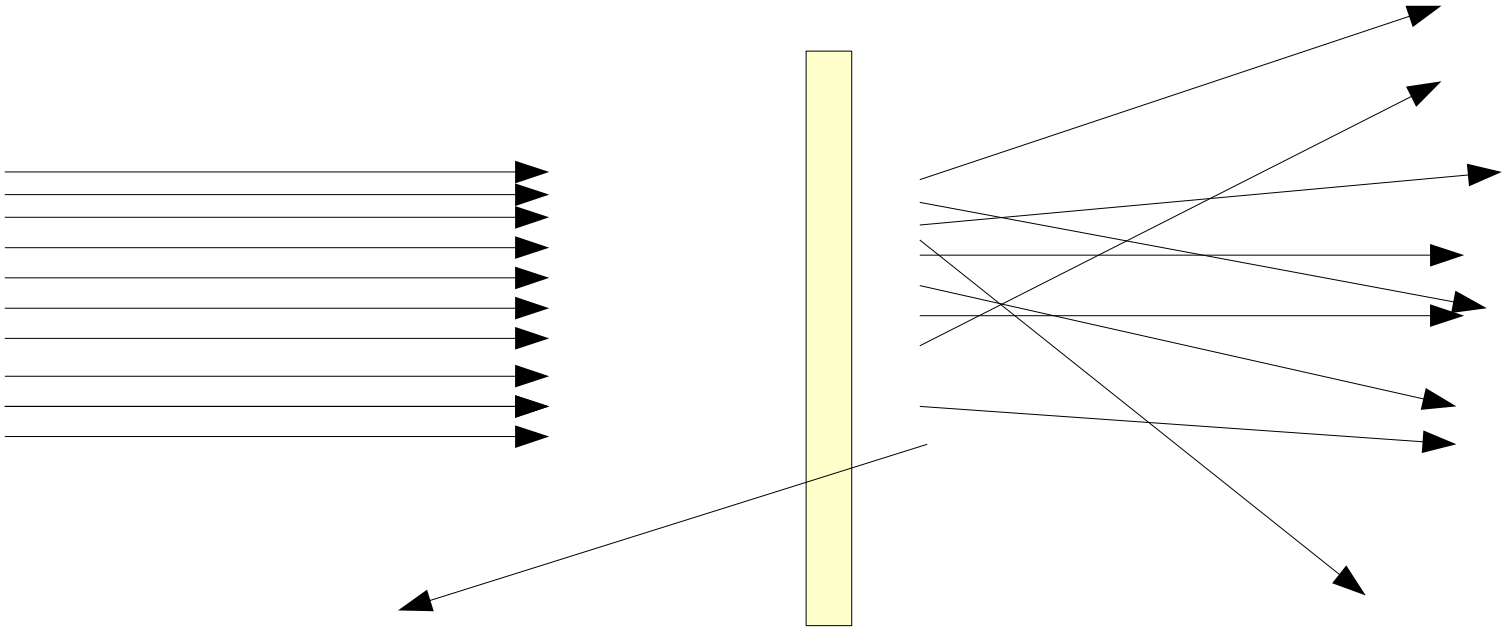
What if μ depends on (x,y,z,E) **??**

What if N depends on (E, $\bar{r}$) **???**

Let`s introduce scattering! **????**

What if μ is not analytical **?????**

Another example: Rutherford scattering



$$\frac{d\sigma}{d\Omega} = \left(\frac{Z_1 Z_2 e^2}{2mv^2}\right)^2 \frac{1}{\sin^4 \frac{\Theta}{2}}$$

~ Probability of scattering by angle Θ

Simulation steps:

1. Generate N-particle beam
2. For each particle toss a coin to get random scattering angle Θ according to this distribution
3. For each particle toss a coin to get random asimuthal anlge φ uniformly in [0,2π]
4. Simulation of scattering is done!

- Question to students: Which way shall we toss a coin then?

Simulation Software

A great deal of software packages are available to simulate physical processes with elementary particles using Monte-Carlo method:

- Event Generators
- Specialized Simulation Software
- General Purpose Software

Specialized Software

- Optimized for their primary task
- Specific physical models
- Usually the possible geometries of experimental setup are limited
- Examples:
  - Extensive atmosferic showers (ШАЛ): CORSIKA, AIRES (*)
  - Simulation of low energy particle path in matter

    (ion implantation e.t.c.): SRIM, MARLOWE
  - Radiation protection: MARS, SHIELD, PHITS
  - Radiation treatment simulation: EGSnrc, BEAMnrc, PEREGRINE
  - The very specific set of software for nuclear power plants…

(*) - some info in the backup slides

General Purpose Software

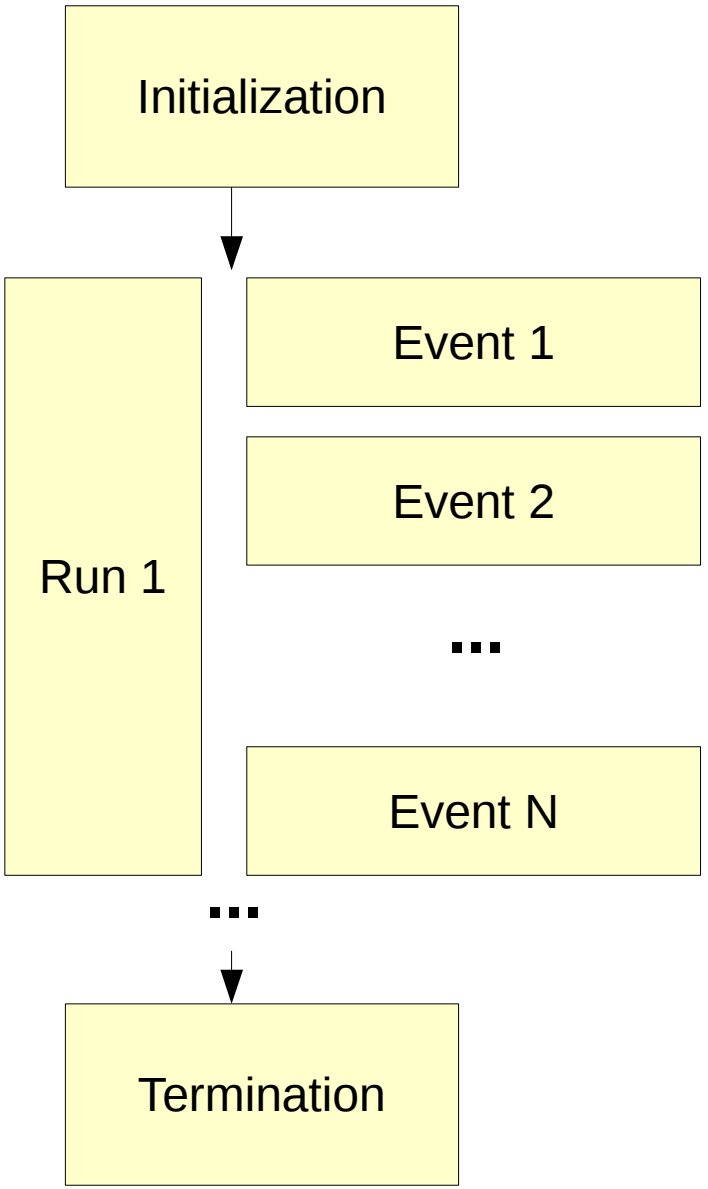
- Wide spectrum of physical models for all energy ranges
- Rich toolset to develop geometry of experimental setup
- Suitable for detector response simulation
- Internal event generators are primitive so the input events are usually simulated by some external software.
- Examples: MCNP, FLUKA, Geant3, Geant4
  - MCNP, FLUKA - «Black Boxes» with a text input configuration file
  - Geant3, Geant4 - «LEGO constructors» giving you a set of libraries and functions to use them in your own simulation code.

GEANT4

- **G**eometry **AN**d **T**racking
- Written in object-oriented language (C++) with full functionality of GEANT3
- The very first version appeared at 1995
- The first real practical application in BaBar experiment
- The main simulation tool in LHC experiments

  (excluding ALICE) since 2004
- The growng usage in particle physics, astronautics and radiation medicine.

GEANT4 Simulation Program Flow



Minimalistic working example:

- Distribution of the materials
- Primary particle generator
- A list of particles, physical processes and models

Can be extended with:

- Sensitive detectors
- Distribution of fields (M or/and E)
- Visualisation
- User actions (Run/Event/Step)
- User commands to G4
- User tools to save simulation output into file (hists, ntuples)

Information and References

The official G4 site: http://cern.ch/geant4

- Source code and/or pre-compiled binary libraries
- User`s guide
- Developer`s guide
- Physics reference manual
- *LOTS* of G4 usage examples (inside the source code)

Our lectures and 2 very basic examples *http://geant4.jinr.ru*

- Simple.zip main(), FTFP_BERT, Geometry, ParticleGun
- SimpleHE.zip **+** Visualisation, Cherenkov photons, UserActions, ROOT histograms

Main reference papers:

S. Agostinelli et al.,
Geant4: a simulation toolkit, NIM A 506 (2003) 250-303
J. Allison et al.,
Geant4 developments and applications
IEEE Trans. Nucl. Sci. 53 No. 1 (2006) 270-278

Tasks for students:

Task 1: «Radiation Treatment Dose Distributions»

A «patient» is a cube of **water** with mass **100 kg** placed in **vacuum**

It undergoes radiation treatment with a medical pencil beam:

A) **proton** with kinetic energy **200 MeV**

B) **gamma** source with energy **7 MeV**

- Find number of protons and gammas required for the «patient» to receive equal full dose (deposited energy)
- Draw the distribution of the deposited energy along beam axis
- Draw the distribution of energy losses of incoming particle per individual simulation step. Ignore zero losses. Compare A) and B)
- Draw the function of mean proton energy loss at simulation step divided by steplength vs. current proton energy
- **Hint:** Divide the «patient» into equal slices **OR** use StepLimiter

Tasks for students:

Task 2: «A Handful of Neutrons»

The pointlike neutron source can emit neutrons of one of the several energies: 1e-3 eV, 1 eV, 1 keV, 1 MeV

At the distance 2m from the source there is detector in the shape of a cylinder h=1m, D=30cm.

- Calculate neutron registration efficiency normalizing it by the detector aperture (solid angle) in the cases of:

  A) Detector material is polistyrol

  B) Detector material is He^3 at $T=20^oC$ and 5 Atm pressure

  C) Case B) + the detector is screened by iron square plate of 1cm thickness and $1m^2$ area placed 20cm in front of the detector

  D) Case B) + material of the screen is paraffin

  E) Case D) + the screen was *by mistake* put 20cm behind !!!

- **Hint:** Count neutron as registered if after interaction with detector`s material the total energy release in full detector is >10 keV

Tasks for students:

Task 3: «A Study on Radiation Protection»

- In a square S=6m*6m room with 30 cm **concrete** walls, ceiling and floor, there is an inner 3m*30 cm **wall in the middle**. The inner wall divides the room into «treatment part» to the right and «entrance part» to the left. The entrance part has a **steel** door 1m*10 cm. The treatment part has a positions to put a «patient» either in lower **A** or in upper **B** quarters.
- The «patient» **P** is a 70 kg ball of **water**. It is irradiated 20 cm from top by a 7 MeV gamma source.
- Another unwise «spectator» person **S** who is also 70 kg ball of **water** stands close to the steel door but outside of the room in the corridor during the treatment of «patient» **P**.
- Compare the ratio of the doses of persons **S** and **P** in case of the «patient» is placed in **A** or **B** positions.
- **Hint:** Place a fake counting volume in the room in front of the door

Basic G4 features: Types

Basic variable types

- For greater compatibility basic types are redefined

  G4int, G4long, G4float, G4double,

  G4bool, G4complex, G4String

- The good style is to use these redefined types

Basic G4 features: I/O

Input/Output inside G4 code

OK, you might use C-style printf() and C++-style cout

The good style is to use the redefined I/O streams of Geant4:

G4cout << *“test”* << *G4endl;*

G4cerr << *“error”* << *G4endl;*

- User interface can be more elaborate than just a command line
- Output stream will always be syncronised with the program flow as all messages go through the same I/O buffers

Basic G4 features: CLHEP

CLHEP: Class Library for High Energy Physics

- **G4ThreeVector**
  - Just a vector (x,y,z) in 3-dimensional space and operations
- **G4LorenzVector**
  - Lorentian 4-component vector (x,y,z,t) and operations
- **G4RotationMatrix**
  - 3x3 matrix, used for rotation in euclidian space
- **G4LorenzRotation**
  - 4X4 matrix, used for rotation in minkovsky space
- Geometrical objects and their transformations
  - **G4Plane3D, G4Transform3D, G4Normal3D, G4Point3D, G4Vector3D**

http://cern.ch/clhep

Basic G4 features: System of Units

• Each physical quantity that is not dimensionless must be **multiplied** by the corresponding unit to be entered inside Geant4

length = 10.0 * cm;

kinetic_energy = 5.0 * GeV;

• To get your result back measured in particular units one must **divide** his value by corresponding unit

G4cout << eDep / MeV << " [MeV]" << G4endl;

• The most usable units are inside Geant4 (CLHEP). But you can define and add your own units

SimpleGeometry: How to make a material?

- **G4Isotope**

  Defines atomic properties: atomic number, nucleons number, molar mass e.t.c.

- **G4Element**

  Defines element properties: effective atomic number, effective molar mass, number of isotopes

- **G4Material**

  Defines macroscopic properties of material:

  density, state, temperature, pressure, radiation length e.t.c.

How to make a new element?

```
a = 1.01*g/mole;
G4Element* elH
      = new G4Element(name="Hydrogen",symbol="H" , z= 1., a);
a = 12.01*g/mole;
G4Element* elC
      = new G4Element(name="Carbon"   ,symbol="C" , z= 6., a);
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);
```

Creating simple materials

```
G4double density = 2.700*g/cm3;
G4double a = 26.98*g/mole;
G4Material* Al = new G4Material(name="Aluminum", z=13.,
    a, density);

G4double density = 1.390*g/cm3;
G4double a = 39.95*g/mole;
G4Material* lAr = new G4Material(name="liquidArgon",
    z=18., a, density);
```

Materials by their chemical formula

```
G4double density = 1.000*g/cm3;
G4Material* H2O = new G4Material(name="Water",
   density, ncomponents=2);
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);

G4double density = 1.032*g/cm3;
G4Material* Sci = new G4Material(name="Scintillator",
   density, ncomponents=2);
Sci->AddElement(elC, natoms=9);
Sci->AddElement(elH, natoms=10);
```

Materials by their fraction masses (массовые доли)

```
G4double density = 1.290*mg/cm3;
G4Material* Air =
    new G4Material (name="Air"  , density,
    ncomponents=2);

Air->AddElement(elN, fractionmass=0.7);
Air->AddElement(elO, fractionmass=0.3);
```

Note: fraction mass is of double type!

Geant4 materials library

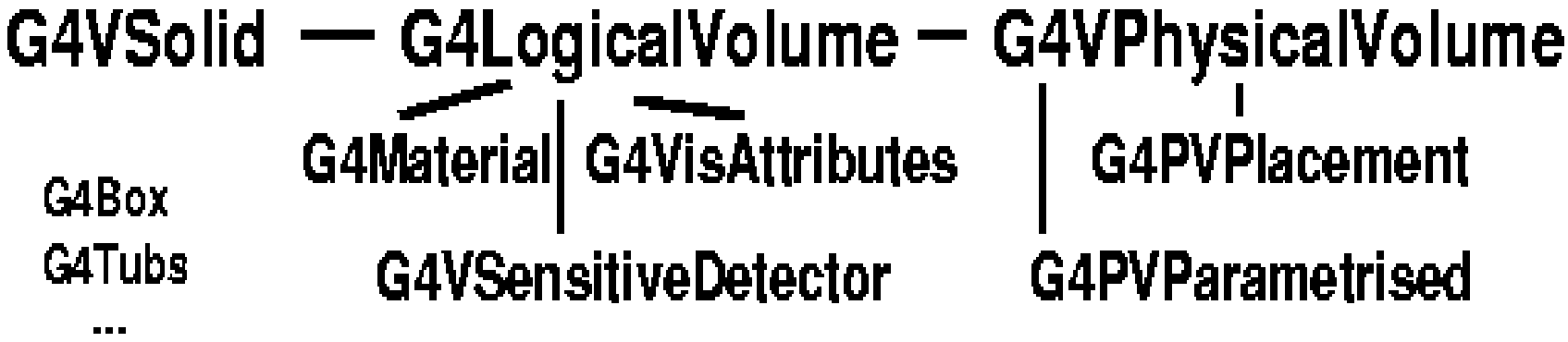
```
#include "G4NistManager.hh"

.....
  G4NistManager* man = G4NistManager::Instance();
  // define elements
  G4Element* elAl = man->FindOrBuildElement("Al");
  // define pure NIST materials
  G4Material* Al = man->FindOrBuildMaterial("G4_Al");
  G4Material* Cu = man->FindOrBuildMaterial("G4_Cu");
  // define NIST materials
  G4Material* H2O  = man->FindOrBuildMaterial("G4_WATER");
  G4Material* Sci  = man->
    FindOrBuildMaterial("G4_PLASTIC_SC_VINYLTOLUENE");
```

SimpleGeometry: Volume Definition

Each volume (physical detector element) is defined in 3 consequent steps

- Shape (G4VSolid) — just a geometrical shape
- Logical volume (G4LogicalVolume) — shape filled with selected material
- Physical Volume (G4VPhysicalVolume) — placed in space



Shapes

- Simple shapes (CGS – Constructed Solid Geometry)

  G4Box, G4Tubs, G4Cons, G4Trd, ...

- Extended shapes

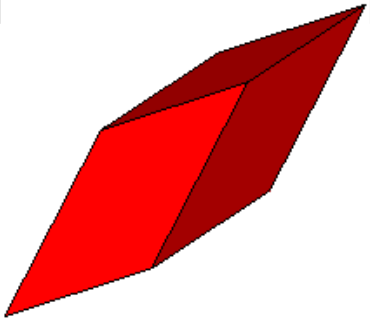
  G4polycone, G4Polyhedra, G4Hype, ...

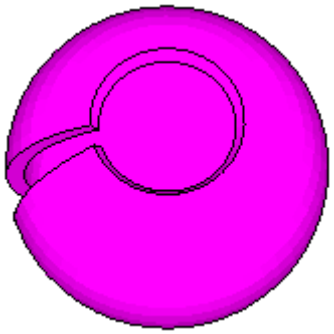
- BREP-Boundary REPresented

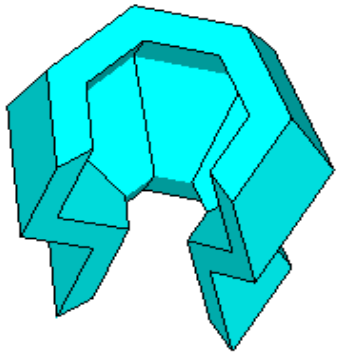
  G4BREPSolidPolycone, G4BSplineSurfa

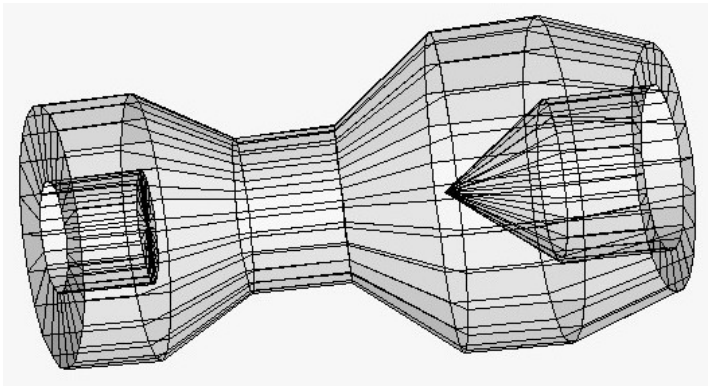
- Boolean

  G4UnionSolid, G4SubtractionSolid, ...

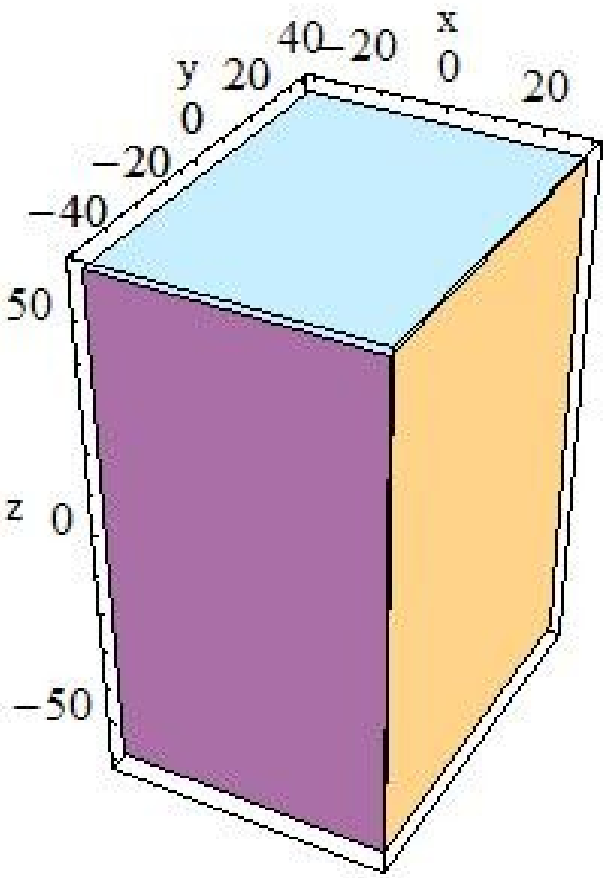








A Box




```
G4VSolid* scint_solid = new
    G4Box(const G4String& pName,
                G4double  pX,
                G4double  pY,
                G4double  pZ);
```

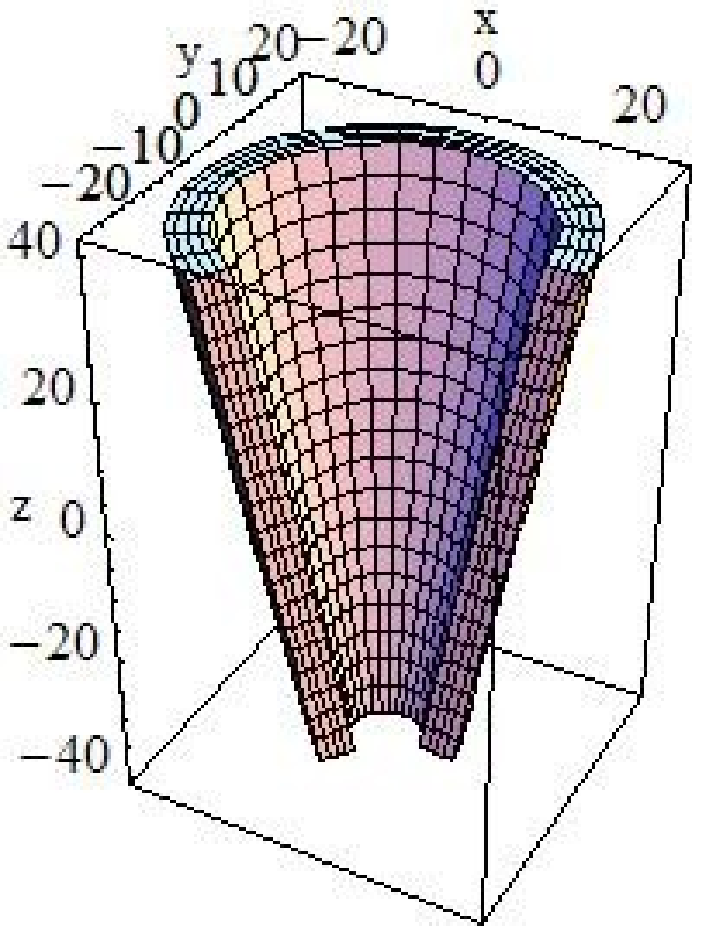
An example of constructor call:

(notice passing the 0.5*side_size value)

```
G4Box* aBox = new G4Box("BoxA", 20.0*cm, 40.0*cm, 60.0*cm);
```

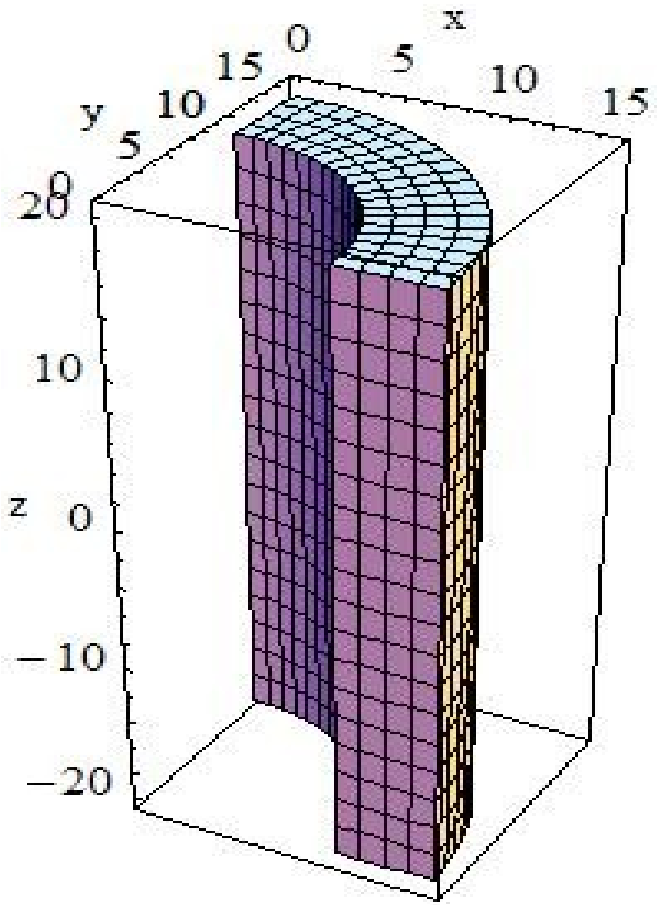
Cone

```
G4VSolid* scint_solid = new
 G4Cons(const G4String& pName,
        G4double  pRmin1,
        G4double  pRmax1,
        G4double  pRmin2,
        G4double  pRmax2,
        G4double  pDz,
        G4double  pSPhi,
        G4double  pDPhi)
```



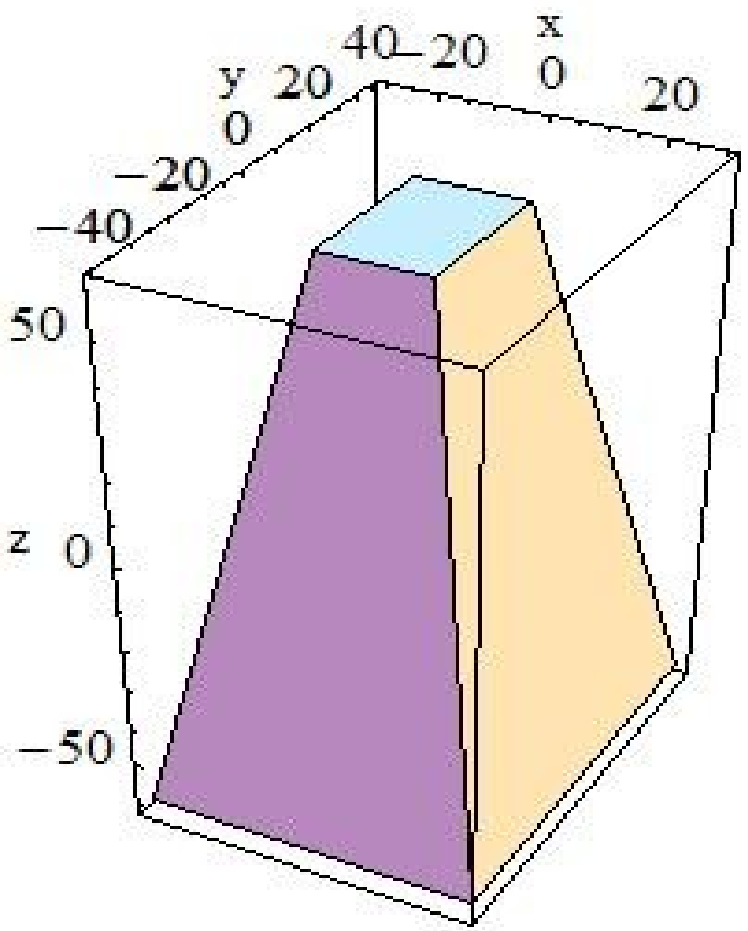
Cylinder

```
G4VSolid* calor_solid = new
    G4Tubs(const G4String& pName,
           G4double  pRMin,
           G4double  pRMax,
           G4double  pDz,    - again 0.5*h
           G4double  pSPhi,
           G4double  pDPhi)
```



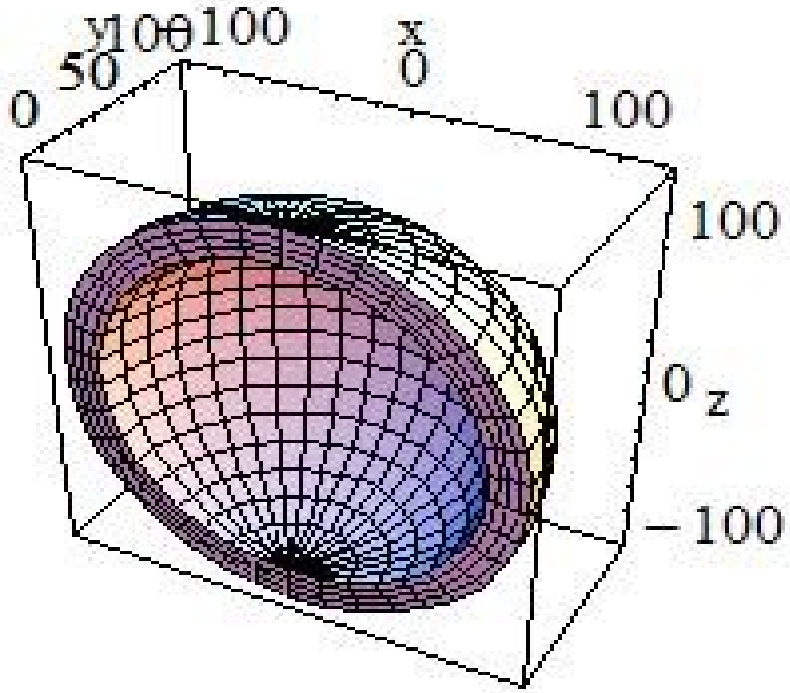
Pyramid

```
G4VSolid* aSolid = new
 G4Trd(const G4String& pName,
             G4double  dx1,
             G4double  dx2,
             G4double  dy1,
             G4double  dy2,
             G4double  dz)
```



Sphere

```
G4VSolid* aSolid = new
  G4Sphere(const G4String& pName,
           G4double  pRmin,
           G4double  pRmax,
           G4double  pSPhi,
           G4double  pDPhi,
           G4double  pSTheta,
           G4double  pDTheta )
```



SimpleGeometry: Logical Volume

- Requires the shape to be filled with material

- Also can contain visualisation properties for the volume and the reference to sensitive detector

```
G4LogicalVolume* aLogical =
            new  G4LogicalVolume( G4VSolid* pSolid,
                G4Material*             pMaterial,
                const G4String&         Name,
                G4FieldManager*         pFieldMgr=0,
                G4VSensitiveDetector*   pSDetector=0,
                G4UserLimits*           pULimits=0,
                G4bool                  optimise=true );
```

SimpleGeometry: Physical Volume

- Requires logical volume to be already defined

- Defined by positioning of the logical volume in space (G4PVPlacement)

- Allows to place a series of identical objects at different spacepoints (G4Replica) or parametrised (G4VParametrised) for instance with increasing size

G4VPlacement

```
G4PVPlacement(  G4RotationMatrix*  pRot,
                const G4ThreeVector&     translate,
                G4LogicalVolume*   pCurrentLogical,
                const G4String&          pName,
                G4LogicalVolume*   pMotherLogical,
                G4bool               pMany,
                G4int               pCopyNo,
                G4bool               pSurfChk=false )
```

Inheritance of the volumes

- All the volumes must be put one fully inside the other

  no volume overlap is allowed as it gives ambiguity for material

- The 'top' by hierarchy volume is unique. It is so called «World» or «Experimental Hall». Other volumes are in it.
- Daughter volume (inner) is positioned the coordinate system of the mother volume. The origin of the coordinates is in the center of mother volume. The position of every particle is calculated both in global coordinates and in the coordinates of the volume the particle is currenly in.

Checking the geometry overlap

```
Idle> /geometry/test/run
Checking overlaps for volume Conus ... OK!
Checking overlaps for volume subcyllv ... OK!
Checking overlaps for volume Z ...
-------- WWWW ------- G4Exception-START -------- WWWW -------
*** G4Exception : GeomVol1002
      issued by : G4PVPlacement::CheckOverlaps()
Overlap with volume already placed !
          Overlap is detected for volume Z
          with subcyllv2 volume's
          local point (-33.3068,-39.803,24.5), overlapping by at least: 500 um
NOTE: Reached maximum fixed number -1- of overlaps reports for this volume !
*** This is just a warning message. ***
-------- WWWW -------- G4Exception-END --------- WWWW -------
Checking overlaps for volume Conus1 ... OK!
```

SimpleParticleSource

Each elementary particle is a separate C++ class (except for ions)

Each particle is defined in 3 consequent steps:

- **G4ParticleDefinition** — constant particle properties (name, mass, charge, e.t.c)
  - We obtain ParticleDefinitions from PhysicsList
- **G4DynamicParticle** — only properties that can change in interaction with matter (energy, momentum)
- **G4Track** – a motion of a particle in space (time, position)

G4ParticleDefinition 'Get' Methods

G4String	GetParticleName()	название
G4double	GetPDGMass()	масса
G4double	GetPDGWidth()	ширина распада
G4double	GetPDGCharge()	заряд
G4double	GetPDGSpin()	спин
G4int	GetPDGiParity()	четность
G4int	GetPDGiConjugation()	зарядовое сопряжение
G4double	GetPDGIsospin()	изоспин
G4double	GetPDGIsospin3()	I_3
G4int	GetPDGiGParity()	G-четность
G4String	GetParticleType()	описание частицы
G4String	GetParticleSubType()	краткое описание частицы
G4int	GetLeptonNumber()	лептонный заряд
G4int	GetBaryonNumber()	барионный заряд
G4int	GetPDGEncoding()	код частицы согласно PDG
G4int	GetAntiPDGEncoding()	код соотв. античастицы

Useful PDG codes

22 - gamma

11 - electron

-11 - positron

2212 - proton

2112 - neutron

+-100ZZZAAAI — ion, for example

1000010020 - deuteron

1000010030 - triton

1000020040 - alpha

1000020030 - He3

SimpleParticleSource:

- Must be itself inherited from **G4VUserPrimaryGeneratorAction**
- Must also contain an object inherited from G4VPrimaryGenerator (mainly just the **G4ParticleGun**)
- Must implement *GeneratePrimaries*() method, with a call to **G4VPrimaryGenerator::generatePrimaryVertex()**, which inserts a primary particle(s) into an empty Event.
- The primary vertex can be build with calling several different objects inherited from G4VPrimaryGenerator.

SimpleParticleSource: G4ParticleGun

- Produces one or several primary particles with given positions, momenta and other properties
- Has interactive user command interface
- Class Methods:

void SetParticleDefinition(G4ParticleDefinition*)
void SetParticleMomentum(G4ParticleMomentum)
void SetParticleMomentumDirection(G4ThreeVector)
void SetParticleEnergy(G4double)
void SetParticleTime(G4double)
void SetParticlePosition(G4ThreeVector)
void SetParticlePolarization(G4ThreeVector)
void SetNumberOfParticles(G4int)

Command line interface to G4ParticleGun /gun/*

- /gun/List display list of all known particles
- /gun/particle setup current particle
- /gun/direction setup direction of outgoing particles
- /gun/energy **setup kinetic energy**
- /gun/position setup vertex coordinates in 'World' axis
- /gun/time setup initial time
- /gun/polarization setup polarisation
- /gun/number setup number of primary particles
- /gun/ion setup ion properties

What is required to write a simple simulation program in G4?

- Geant4 properly installed and set up on your notebook
- Your favourite text editor to read and write the code
- Code development tools: **g++**, **cmake**, **make** (or cmake, VisualC++ in Windows)

Also from Virtual Box 'G4ubuntu32' @

/home/user/G4Examples/geant4.jinr.ru/simple

It contains the main(){} function in *simple.cc* and two classes for detector geometry and primary vertex generator. Each class is divided into the header *'*.hh'* file with class definitions and the body *'*.cc'* file with class functions implemented.

user@G4ubuntu32:~$ ls simple/

CMakeLists.txt

simple.cc

include/SimpleGeometry.hh

src/SimpleGeometry.cc

include/SimpleParticleSource.hh

src/SimpleParticleSource.cc

Set up, compile and on we go!

user@G4ubuntu32:~$ ***source /opt/g4_setup.sh***

user@G4ubuntu32:~$ ***cd G4Examples/geant4.jinr.ru/simple***

user@G4ubuntu32:~$ ***mkdir build***

user@G4ubuntu32:~$ ***cd build***

user@G4ubuntu32:~$ ***cmake ..***

user@G4ubuntu32:~$ ***make***

user@G4ubuntu32:~$ ***./simple.exe***

Where comes the result of simulation?

Text output can be piped into log-file to store it and examine later.

./simple.exe &> out.log; less out.log

Set up, compile and on we go!

```
****************************************************************************************************
* G4Track Information:   Particle = proton,   Track ID = 1,   Parent ID = 0
****************************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0        0        0     -100     1e+03        0        0         0       World initStep
    1        0        0      -50     1e+03 1.49e-24       50        50   LeadPlate Transportation
    2    0.724   -0.376     13.6       911     87.8     63.6       114   LeadPlate hIoni
    :----- List of 2ndaries - #SpawnInStep=  1(Rest= 0,Along= 0,Post= 1), #SpawnTotal=  1----------
    :     0.724    -0.376      13.6      1.35                  e-
    :-------------------------------------------------------------- EndOf2ndaries Info----------
    3     1.48   -0.405     44.5       870     41.2     30.9       145   LeadPlate CoulombScat
    4     1.61   -0.629       50       862     8.03     5.53       150       World Transportation
    5     17.5    -20.1      500       862  1.4e-23      451       601  OutOfWorld Transportation

****************************************************************************************************
* G4Track Information:   Particle = e-,   Track ID = 2,   Parent ID = 1
****************************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0    0.724   -0.376     13.6      1.35        0        0         0   LeadPlate initStep
    1    0.705   -0.413     13.7      0.36    0.985    0.886     0.886   LeadPlate eBrem
    2    0.717   -0.416     13.7         0     0.36    0.194      1.08   LeadPlate eIoni
```

How we achieved such a great success? :-)
The answer is inside the code...

simple.cc

```
#include "SimpleGeometry.hh"
#include "SimpleParticleSource.hh"
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "FTFP_BERT.hh"

int main()
{
  // 1. Create RunManager instance
  G4RunManager * runManager = new G4RunManager;
  // 2. Add detector construction
  runManager->SetUserInitialization(new SimpleGeometry());
  // 3. Add physics list
  runManager->SetUserInitialization(new FTFP_BERT);
  // 4. Add primary particle source
  runManager->SetUserAction(new SimpleParticleSource());
  // 5. Initialize G4 kernel
  runManager->Initialize();
  // 6*. Send a command to G4 kernel: «Display verbose output on Tracking»
  G4UImanager::GetUIpointer()->ApplyCommand("/tracking/verbose 2");
  // 7. Send a command to G4 kernel: «Start a RUN with 10 events»
  runManager->BeamOn(10);
  // 8. Cleanup and quit
  delete runManager;
  return 0;
}
```

SimpleGeometry.hh

```
#ifndef SimpleGeometry_h
#define SimpleGeometry_h 1

#include "G4VUserDetectorConstruction.hh"
#include "globals.hh"

class G4VPhysicalVolume;

class SimpleGeometry : public G4VUserDetectorConstruction
{
  public:
    SimpleGeometry();
    virtual ~SimpleGeometry();
  public:
    virtual G4VPhysicalVolume* Construct();
};

#endif
```

SimpleGeometry.cc

```
#include "SimpleGeometry.hh"
#include "G4RunManager.hh"
#include "G4NistManager.hh"
#include "G4Box.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4SystemOfUnits.hh"
using namespace CLHEP;

SimpleGeometry::SimpleGeometry()  : G4VUserDetectorConstruction() { }

SimpleGeometry::~SimpleGeometry() { }

G4VPhysicalVolume* SimpleGeometry::Construct()
{
  G4bool checkOverlaps = true;

  G4NistManager* nist = G4NistManager::Instance();
  G4Material* Lead = nist->FindOrBuildMaterial("G4_Pb");
  G4Material* Vacuum = nist->FindOrBuildMaterial("G4_Galactic");
  // World
  G4Box* world = new G4Box("World", 50*cm, 50*cm, 50*cm);
  G4LogicalVolume* worldlv =  new G4LogicalVolume(world, Vacuum, "World");
  G4VPhysicalVolume* physWorld = new G4PVPlacement(0, G4ThreeVector(), worldlv, "World", 0, false, 0, checkOverlaps);

  // Lead plate
  G4Box* plate = new G4Box("LeadPlate", 10*cm, 10*cm, 5*cm);
  G4LogicalVolume* platelv =  new G4LogicalVolume(plate, Lead, "LeadPlate");

  new G4PVPlacement(0, G4ThreeVector(0.,0.,0.), platelv, "LeadPlate", worldlv, false, 0, checkOverlaps);

  return physWorld;
}
```

SimpleParticleSource.hh

```
#ifndef SimpleParticleSource_h
#define SimpleParticleSource_h 1
#include "G4VUserPrimaryGeneratorAction.hh"
#include "G4ParticleGun.hh"
#include "globals.hh"

class G4ParticleGun;
class G4Event;

class SimpleParticleSource : public G4VUserPrimaryGeneratorAction
{
public:
  SimpleParticleSource();
  virtual ~SimpleParticleSource();

  virtual void GeneratePrimaries(G4Event*);

private:
  G4ParticleGun*  fParticleGun;
};
#endif
```

SimpleParticleSource.cc

```
#include "SimpleParticleSource.hh"
#include "G4ParticleGun.hh"
#include "G4Proton.hh"
#include "G4ParticleDefinition.hh"
#include "G4SystemOfUnits.hh"
using namespace std;

SimpleParticleSource::SimpleParticleSource() : G4VUserPrimaryGeneratorAction()
{
  fParticleGun = new G4ParticleGun(1);
  fParticleGun->SetParticleDefinition(G4Proton::ProtonDefinition());
  fParticleGun->SetParticleEnergy(1*GeV);
  fParticleGun->SetParticlePosition(G4ThreeVector(0.,0.,-10.0*cm));
  fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
}

SimpleParticleSource::~SimpleParticleSource() {}

void SimpleParticleSource::GeneratePrimaries(G4Event* evt)
{
  fParticleGun->GeneratePrimaryVertex(evt);
}
```

Thank You!

- Compile, run, study and modify our two «simple» examples
- Try examples included in G4 code. Have a closer look at *examples/extended/electromagnetic/TestEm0* which prints cross sections.
- Students tasks for G4 simulation: pick one of the following
  - «Radiation treatment doses»
  - «Capturing neutrons»
  - «Radiation protection»

- Next time:
  - An extended simulation program.
  - Physical models, processes and physical lists
  - User actions and user interface
  - Output of your simulation, ROOT trees and histograms

I Want to Have Some More of it ! :-)

ADDITIONAL SLIDES:

GEANT4 System Requirements

Main system platforms and compilers

- Linux + gcc
- Windows + VisualC++
- MacOSX + gcc

External packages

- CLHEP (since 4.9.5 built-in in Geant4)
- cmake
- Xerces-C (optional, if you use GDML)
- OpenGL, Qt and other graphocal libs (optional, required for visualization)

Requires ~ 1.2 GB disk space (full setup)

Requires minimum 128 MB memory to run examples

- **To be built from source requires ~ 3GB disk space, ~ 1GB of memory and many «development» packages installed (with headers to the system libs)**

What is inside virtual mashine images?

1. Debian
2. Ubuntu
3. Lubuntu

geant4 - Geant4 версия 4.10.01.p01

root - ROOT версия 6.02.05

setup.sh - скрипт задания переменных окружения:

$ source /opt/hep/setup.sh

Some fun with a Simple Example:

Make a few changes to the «simple example» so you can get more familiar with example`s code and GEANT4 in general.

- Ex.1 Add another block of lead of the same size but separated from the first by 5 cm in Z-axis. Do a geometry check.
- Ex.2 Add new material — water. Change lead to water in the first block.
- Ex.3 Change particle type from proton to muon. Hint: you can use command interface to ParticleGun. How much energy does muon deposit on the average in water and in lead?

EAS Simulation

- Количество частиц в ливне может достигать 10^{10}-10^{15}
- Развитие ливня зависит от свойств атмосферы, магнитного поля Земли и т.д.
- Требуются модели физических взаимодействий ориентированные на сверхвысокие энергии
- Требуется смоделировать развитие ЭМ и адронного каскадов частиц, черенковское излучение
- Расчет с помощью универсальных программ требует огромных затрат времени ЦПУ

EAS Simulation: CORSIKA, AIRES

- Специализированные программы для моделирования ШАЛ
  - оптимизация скорости расчетов за счет введения статистических весов (моделирование 1000 частиц с близкими свойствами заменяется моделированием одной частицы с весом 1000)
  - встроено описание атмосферы и магнитного поля Земли
  - большой набор адронных моделей для сверхвысоких энергий (DPMJET, SYBILL, QGS). Непонятно, как проверить их правильность? Одно из решений - посчитать с разными моделями и сравнить ответ.
  - в результате работы программ получается список частиц на поверхности детектора (без моделирования его отклика)
- CORSIKA https://web.ikp.kit.edu/corsika/
- AIRES http://www2.fisica.unlp.edu.ar/auger/aires/