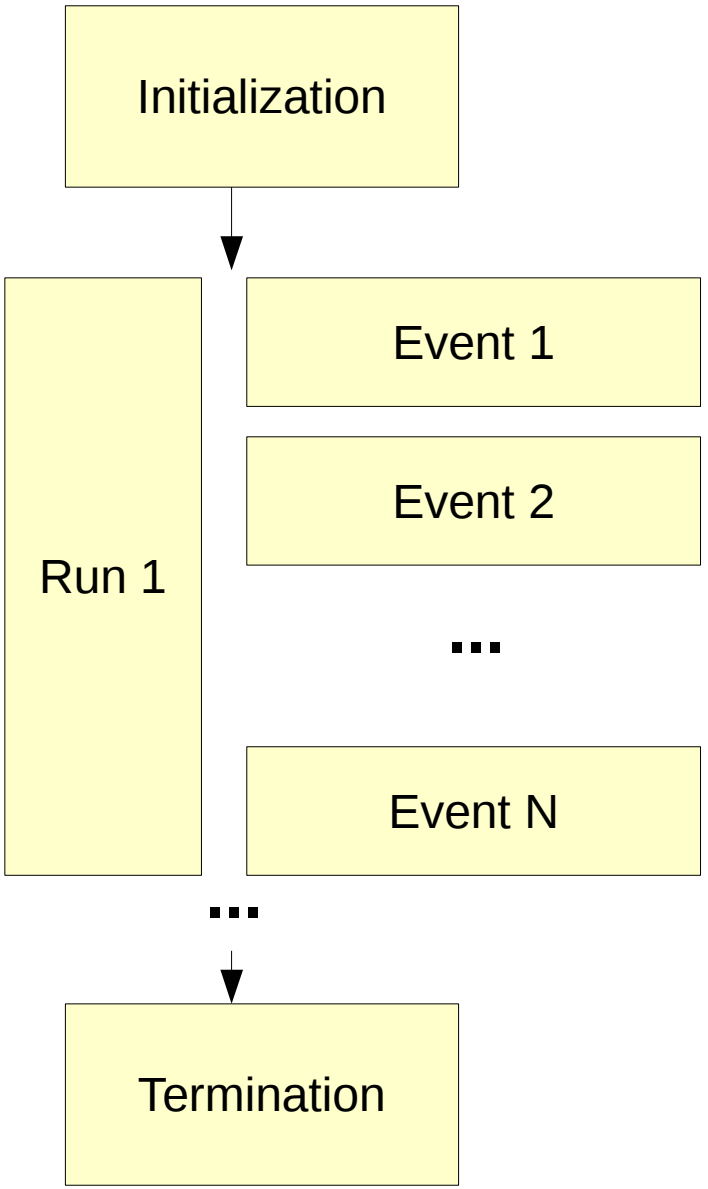
Geant4 Simulation of Physical Processes part 1: A really simple example

Алексей Жемчугов, Михаил Демичев
ОИЯИ
E-mail: zhemchugov@jinr.ru
mdemichev@jinr.ru

Большие Коты

4-12 июня 2015

Simulation Loop



Minimum working example:

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

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)

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)

An example is provided from (simple.zip @ http://geant4.jinr.ru)

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()
{
  G4RunManager * runManager = new G4RunManager;
  // Detector construction
  runManager->SetUserInitialization(new SimpleGeometry());

  // Physics list
  runManager->SetUserInitialization(new FTFP_BERT);

  // Primary generator action
  runManager->SetUserAction(new SimpleParticleSource());

  // Initialize G4 kernel
  runManager->Initialize();
  G4UImanager::GetUIpointer()->ApplyCommand("/tracking/verbose 2");

  runManager->BeamOn(10);

  delete runManager;
  return 0;
}
```

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

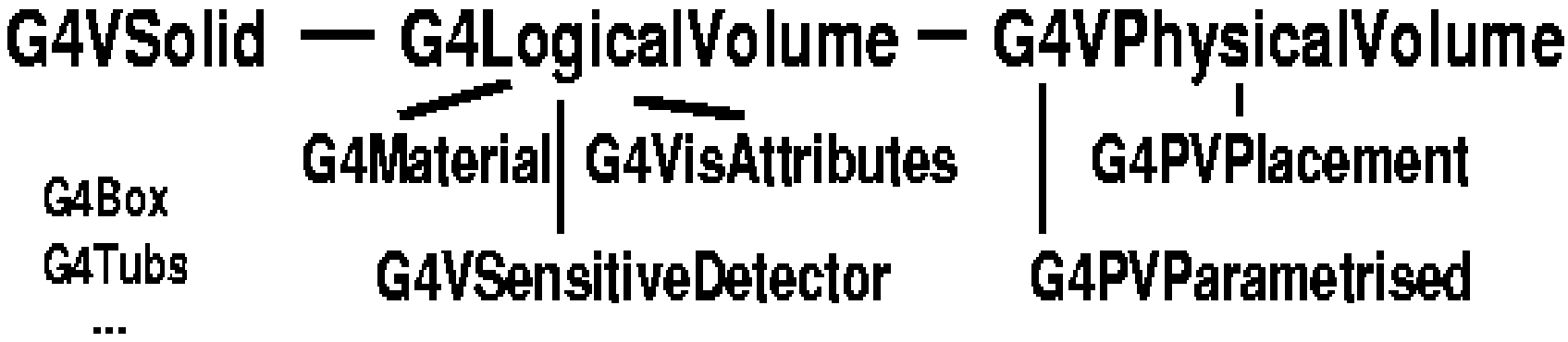
```
#include "G4NistManager.hh"

.....
  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

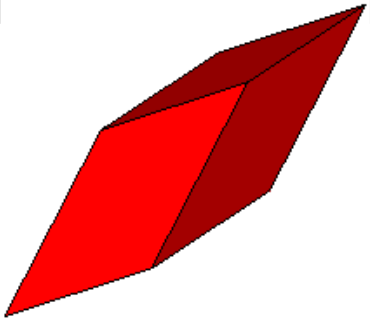
  G4polycone, G4Polyhedra, G4Hype, ...

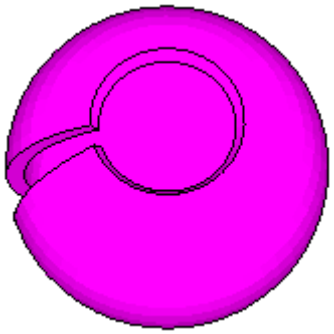
- BREP-Boundary REPresented

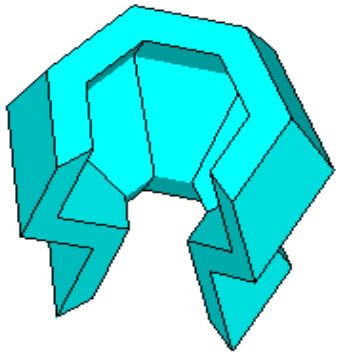
  G4BREPSolidPolycone, G4BSplineSurfa

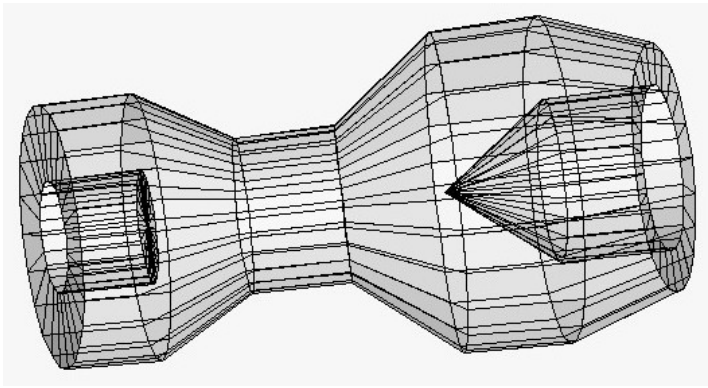
- Boolean

  G4UnionSolid, G4SubtractionSolid, ...

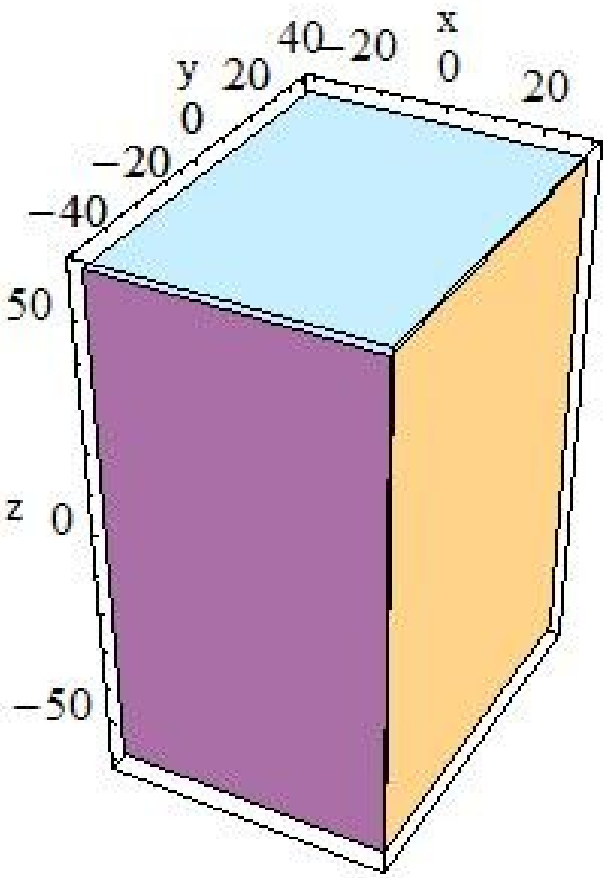








A Box




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

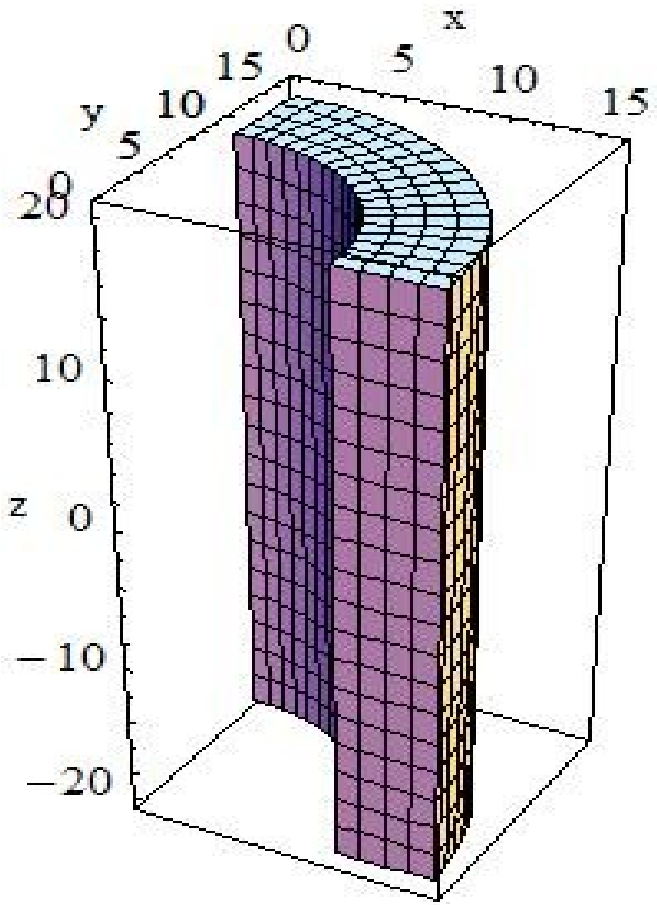
An example of constructor call:

(notice passing the 0.5*side_size value)

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

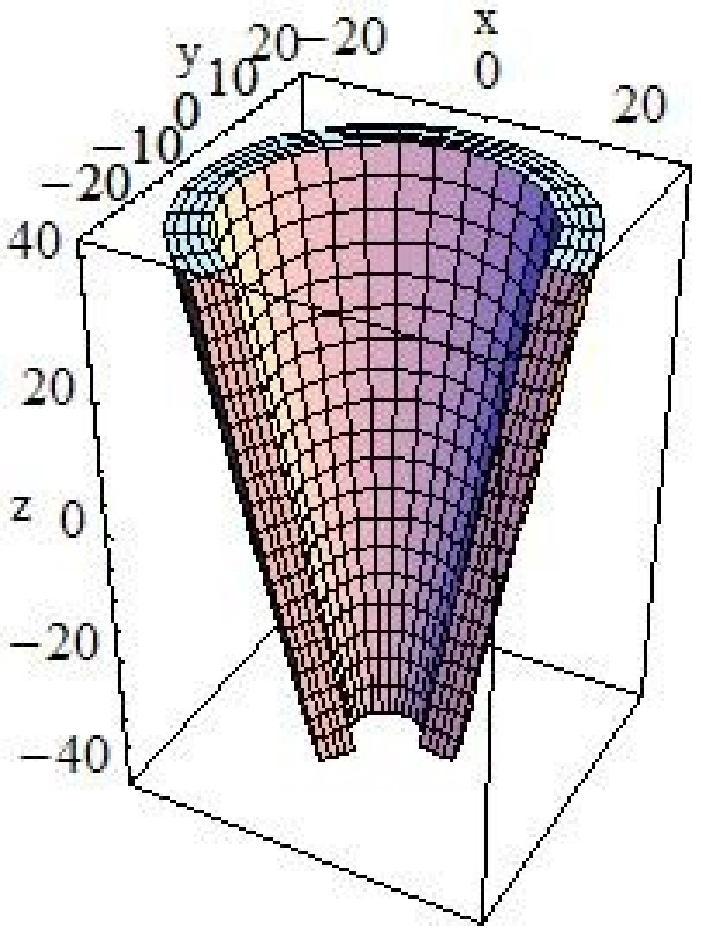
Cylinder

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



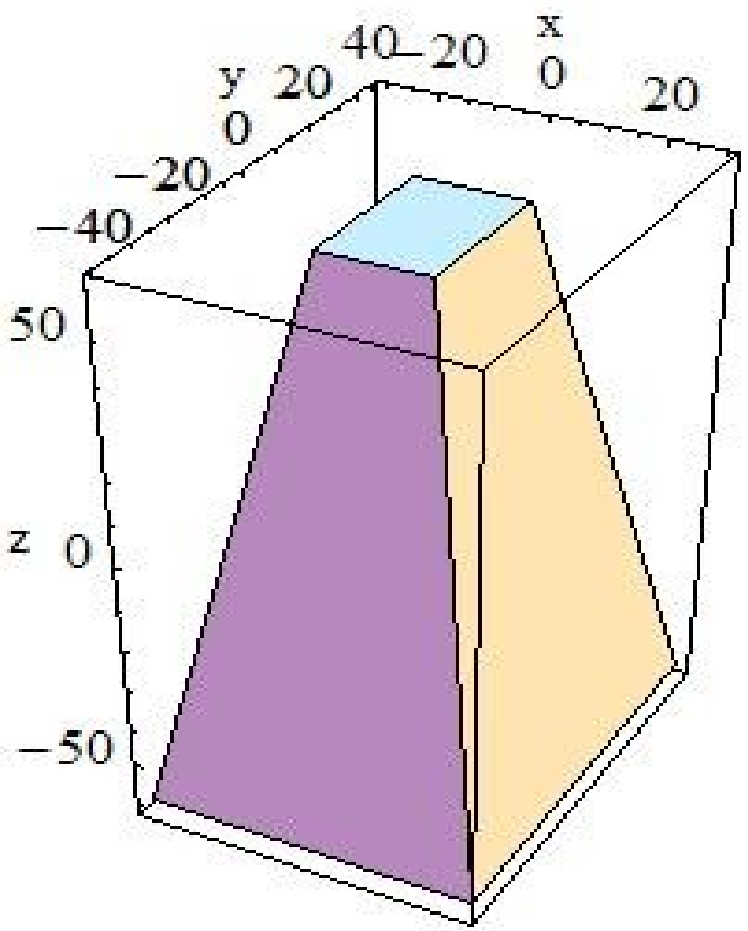
Cone

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



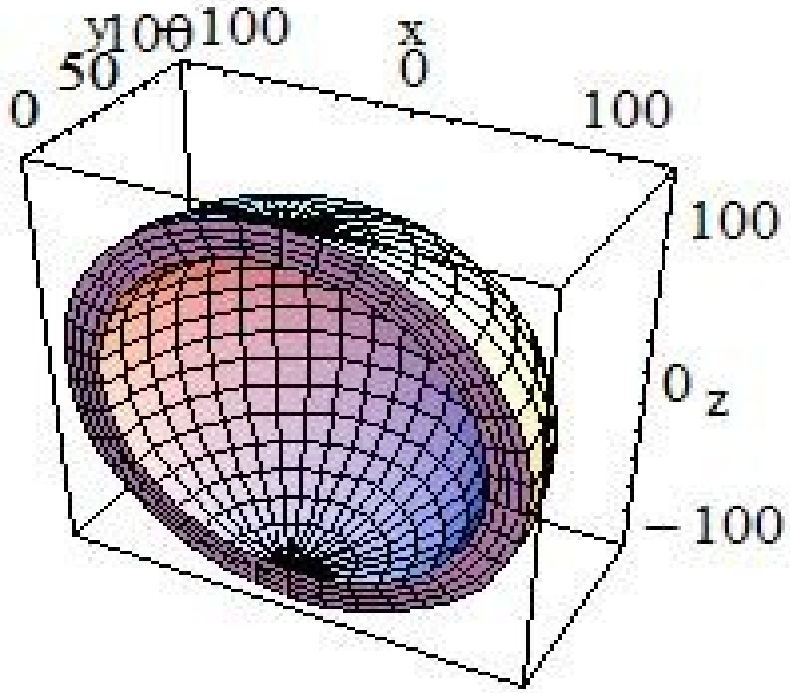
Pyramid

```
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 defined
- A position of the 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!
```

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

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

- Производит первичную вершину из одной или нескольких частиц с заданными импульсом и начальным положением в пространстве
- Может управляться интерактивно
- Методы:

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

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);
}
```

Simple Example: Have fun! :-)

- Ex.1 Add another block of lead of the same size but separated from the first by 5 cm in Z-axis. Check geometry.
- 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?

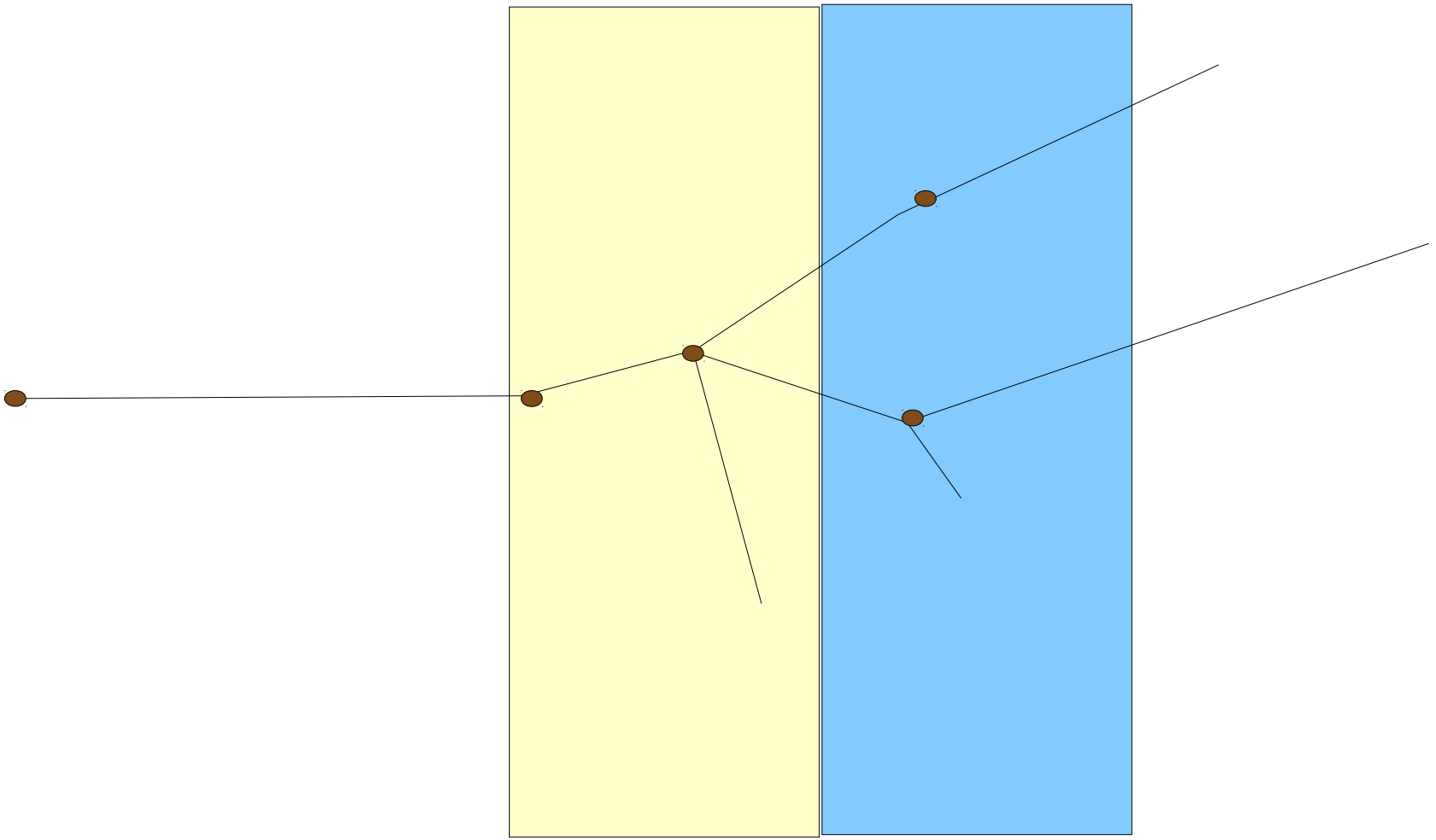
Simple Example:

ADDITIONAL SLIDES:

Tracking

- **An event** (Событие) is a collection of particles from
  - A single shot of primary vertex generator — primary particles
  - Secondaries which come as interaction of primaries with the material objects of our detector
- Each partilcle is simulated until either:
  - it decays or disappears in the act of inelastic scattering
  - it reaches the outer wall of our «World» volume
  - it stops completely, Ekin = 0
  - it is stopped by the good/evil design of the user
- Each particle moves through geometry **step**-by-step (шаг). A **track** is a collection of all the steps from particle generation until the stop of particle simulation.

Tracking



Категории частиц

- **Частицы, участвующие в трекинге**
  - стабильные частицы (протон, электрон, фотон ...)
  - долгоживущие (>10^{-14} с) частицы (пион, мюон ...)
  - короткоживущие частицы, распад которых моделируется в Geant4 (π^0 ...)
  - K-мезоны
  - оптические фотоны
  - geantino
- **Ядра атомов**
  - легкие ядра (дейтрон, альфа-частица, ядро трития)
  - тяжелые ионы
- **Короткоживущие частицы**
  - кварки
  - глюоны
  - мезонные и барионные резонансы

в трекинге не участвуют
появляются только в некоторых моделях физических процессов

Процессы и модели

- Модель (Model) — описание отдельного типа взаимодействия в определенном диапазоне энергий, и в определенном регионе (G4Region)
- Процесс (Process) — описание отдельного типа физического взаимодействия частицы во всем диапазоне энергий. Может включать одну или несколько моделей

  Пример: неупругое рассеяние протонов (ProtonInelastic)

  - высокие энергии (>6 ГэВ) – кварк-глюонная струнная модель
  - средние энергии (1-9 ГэВ)– внутриядерный каскад Бертини
  - низкие энергии (0-1.5 ГэВ)– модель компаунд-ядра
- Список (набор) моделей (Physics List) — совокупность всех процессов, заданных для всех частиц, определяющая моделирование физических взаимодействий в Geant4


Список процессов (Physics List)
Конструктор процессов
Конструктор процессов
Частица 1
Процесс
Процесс
Процесс
Процесс
Процесс
Модель 1
Модель 2
Модель 3
Модель 1
Модель 2
Модель 3
Модель 1
Модель 1
Модель 2
Модель 3
Модель 1
Модель 2

Категории процессов

- ***электромагнитные взаимодействия***
  - ионизация
  - комптоновское рассеяние
  - многократное рассеяние
  - тормозное излучение
  - ...
- ***адронные взаимодействия***
  - упругое и неупругое рассеяние
  - захват, деление
- ***транспортировка***
- ***распады***
- ***оптические***
  - рассеяние Рэлея, черенковское излучение, сцинтилляция, переизлучение в светосмещающих волокнах ...
- ***параметризация и «быстрое» моделирование***

Использование нескольких моделей в одном процессе

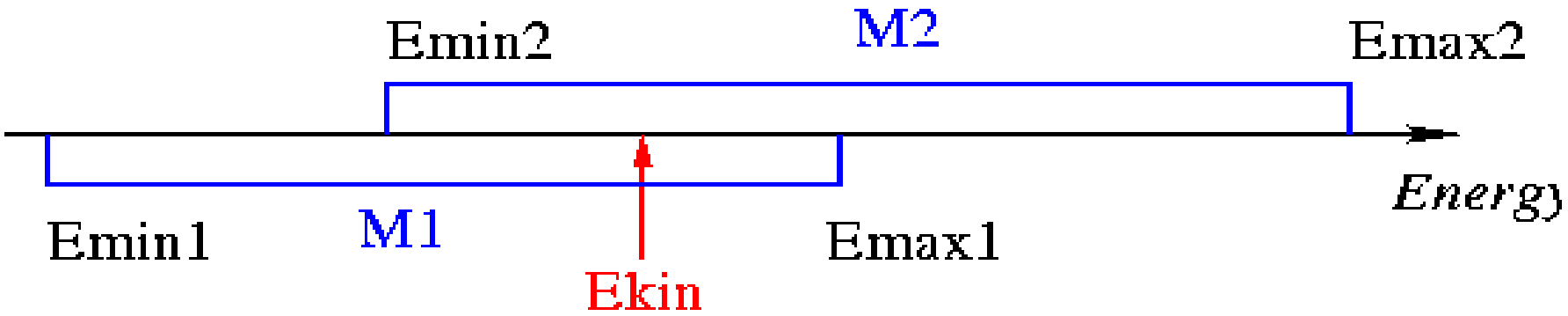
При перекрытии моделей используется следующий алгоритм:

- если данной энергии соответствует более двух моделей, или диапазоны энергий моделей перекрываются полностью, вырабатывается исключение

- в случае частичного перекрытия двух моделей, модель выбирается случайно, но более вероятен выбор модели, предел применимости которой лежит дальше от данной энергии частицы

Иллюстрация:

Есть модели M1 и M2 и частица с энергией Ekin



Разыгрывается случайное число R в интервале [0,1]. Модель M1 выбирается, если выполняется условие

$$\frac{E_{max1}-E_{kin}}{E_{max1}-E_{min2}} < R$$

Дискретные и непрерывные процессы

•Результат непрерывных процессов вычисляется в соответствии с длиной шага, а свойства данной частицы изменяются в конечной точке согласно суммарному эффекту

Пример: ионизация, многократное рассеяние

•Результат дискретных процессов вычисляется в конечной точке шага.

Пример: распад, упругое и неупругое рассеяние

Geant4 list of physical processes

CHIPS

FTF_BIC FTFP_BERT_EMV FTFP_BERT_EMX

FTFP_BERT FTFP_BERT_TRV

LBE

LHEP_EMV LHEP

QBBC QGS_BIC QGSC_BERT QGSC_CHIPS QGSP_BERT_CHIPS QGSP_BERT_EMV QGSP_BERT_EMX QGSP_BERT_HP

QGSP_BERT QGSP_BERT_NOLEP QGSP_BERT_TRV

QGSP_BIC_EMY QGSP_BIC_HP QGSP_BIC QGSP_FTFP_BERT QGSP QGSP_INCL_ABLA QGSP_QEL

Shielding

Comments

QGSP - quark-gluon string model and compound nuclei model
QGSC - quark-gluon string model + CHIPS
FTFP - FRITIOF-model + compound nuclei
FTFC - FRITIOF-model + CHIPS
LHEP - parametrised hadronic interactions (GHEISHA)
BERT - Bertini intranuclear cascade model
BIC - binary intranuclear cascade model
HP - high precision neutron simulation, requires neutron low energy datasets
QBBC - QGSC+BIC(protons)+BERT(pions)

Secondaries generation thresholds (Cuts)

- Каждый процесс имеет свои собственные ограничения на энергию вторичных частиц, им производимых.
- Движение всех вторичных частиц моделируется в Geant4 до нуля энергии
- Каждая частиц имеет пороговое значение (в единицах длины), которое пересчитывается в энергию для каждого материала , и может быть использовано процессом
- Ниже порога, энергия родительской частицы тоже уменьшается, но не идет на рождение (выбивание) новой, а считается выделенной в объеме.