Моделирование физических процессов в пакете Geant4

Алексей Жемчугов
ОИЯИ
E-mail: zhemchugov@jinr.ru

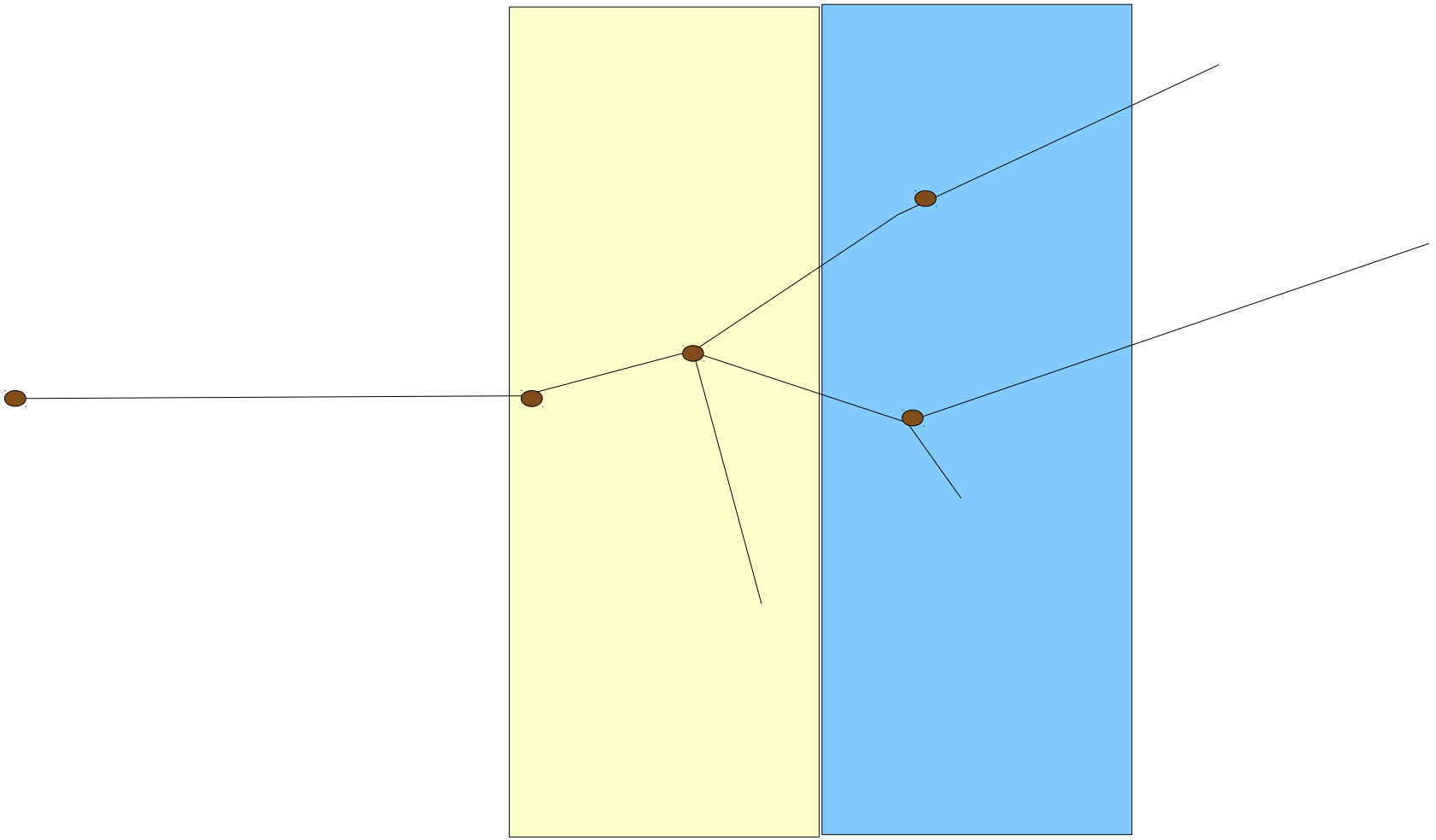
Иркутский государственный университет

28-29 апреля 2015

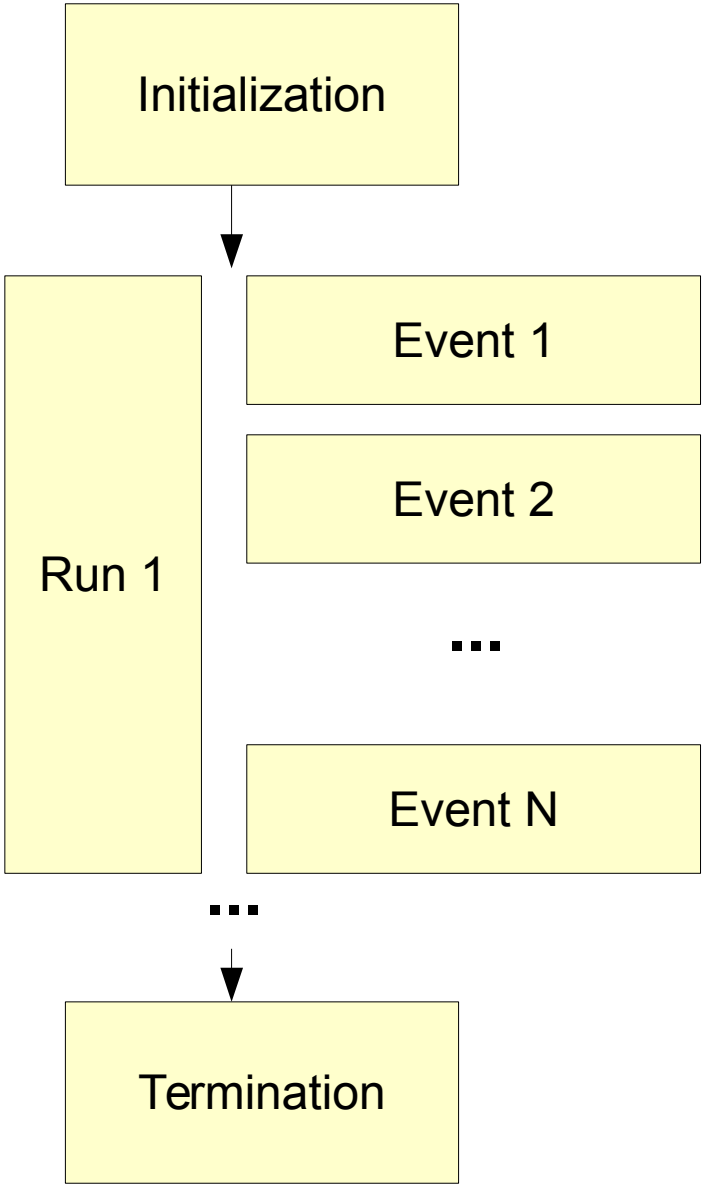
Трекинг

- Событие (event) - совокупность всех частиц, включая образованные при одном запуске генератора первичной вершины и рожденные ими вторичные частицы
- Каждая частица моделируется до одного из четырех исходов:
  - исчезновение в результате взаимодействия (распад, неупругое рассеяние)
  - достижение самой внешней границы моделируемого объема
  - уменьшение кинетической энергии до нуля
  - остановка по требованию пользователя
- Каждая частица перемещается по геометрии детектора шагами (step). Совокупность шагов от рождения до остановки частицы образует трек.

Трекинг



Цикл моделирования



Необходимо описать:

- Распределение вещества в детекторе и поля
- Генератор события
- Список физических процессов, учитываемых в моделировании

Лекция 2

Кроме того, по желанию:

- Чувствительные элементы и способы моделирования отклика
- Способы визуализации
- Графический интерфейс
- Сохранение результатов
- Пользовательские расширения

Лекция 3

Что нужно, чтобы написать простую программу моделирования на Geant4?

- Установленный Geant4
- Текстовый редактор
- Средства разработки: g++, cmake, make (или cmake, VisualC++ в Windows)

Требуется написать основную программу и два пользовательских класса: описание геометрии модели и генератор событий.

Простой пример: (simple.zip на http://geant4.jinr.ru)

hep@S5500BC:~$ ls simple/

CMakeLists.txt

simple.cc

include/SimpleGeometry.hh

src/SimpleGeometry.cc

include/SimpleParticleSource.hh

src/SimpleParticleSource.cc

Сборка и запуск

```
hep@S5500BC:~$ source /opt/hep/setup.sh
hep@S5500BC:~$ unzip simple.zip
hep@S5500BC:~$ cd simple
hep@S5500BC:~$ mkdir build
hep@S5500BC:~$ cd build
hep@S5500BC:~$ cmake ..
hep@S5500BC:~$ make
hep@S5500BC:~$ ./simple.exe
```

А что, собственно, было смоделировано?

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;
}
```

Классы, описывающие материалы

- **G4Isotope**

  описывает свойства атома: атомное число, количество нуклонов, молярную массу и т.д.

- **G4Element**

  описывает свойства элемента: эффективное атомное число, эффективную молярную массу, число изотопов

- **G4Material**

  описывает макроскопические свойства вещества:

  плотность, состояние, температуру, давление, радиационную длину, длину свободного пробега и т.д

Создание нового элемента

```
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);
```

Описание простых материалов

```
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);
```

Описание материалов по химической формуле

```
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);
```

Описание материала через массовые доли компонент

```
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);
```

Библиотека материалов Geant4

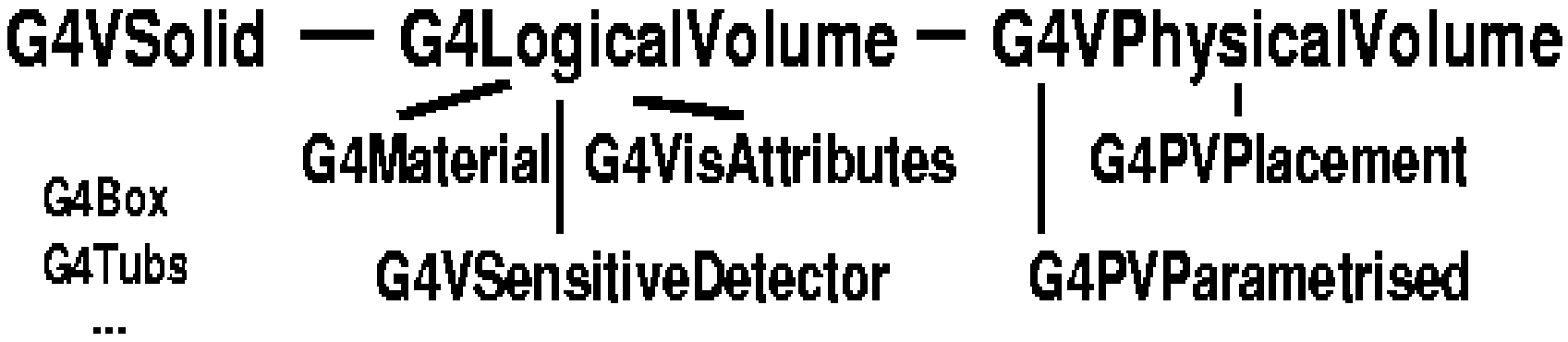
```
#include "G4NistManager.hh"

.....
  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");
```

Описание объема

Каждый объем описывается в три этапа

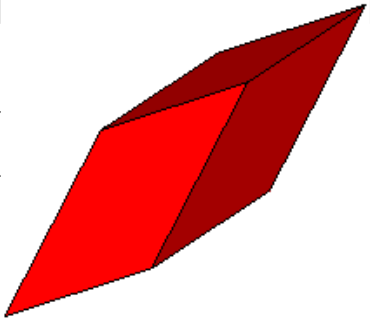
- форма (G4VSolid)
- логический объем (G4LogicalVolume)
- физический объем (G4VPhysicalVolume)

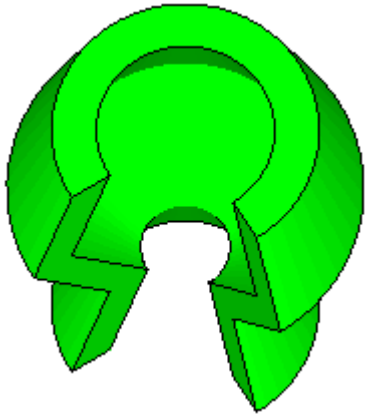


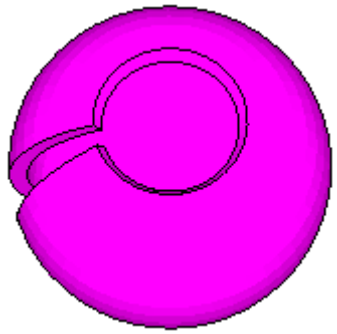
Формы

- Простые формы (CGS – Constructed Solid Geometry
  G4Box, G4Tubs, G4Cons, G4Trd, ...
- Специальные формы
  G4polycone, G4Polyhedra, G4Hype, ...
- Определяемые поверхностью
  (BREP-Boundary REPresented)

G4BREPSolidPolycone, G4BSplineSurface, ...

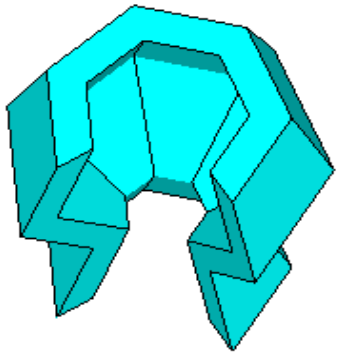


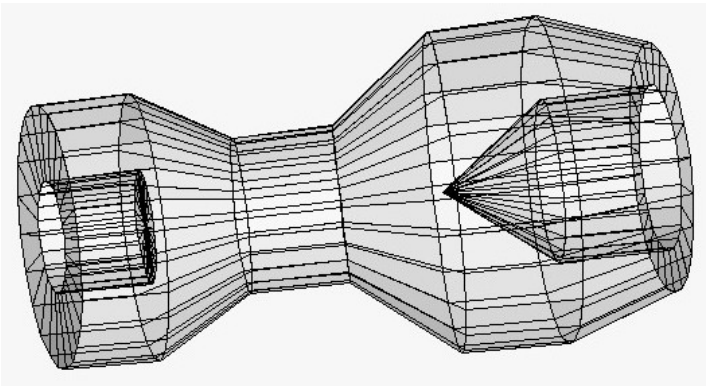




- Булевы формы

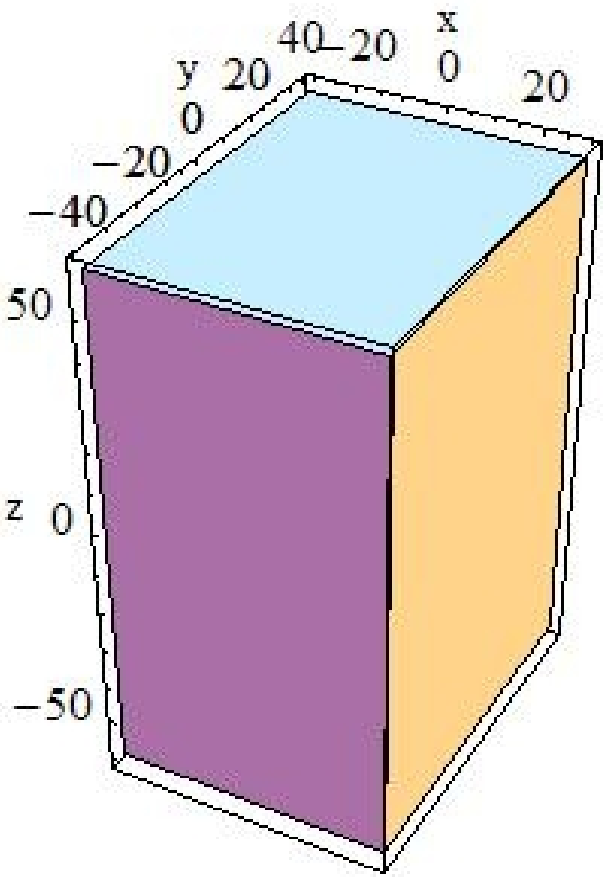
G4UnionSolid, G4SubtractionSolid, ...





Параллелепипед

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

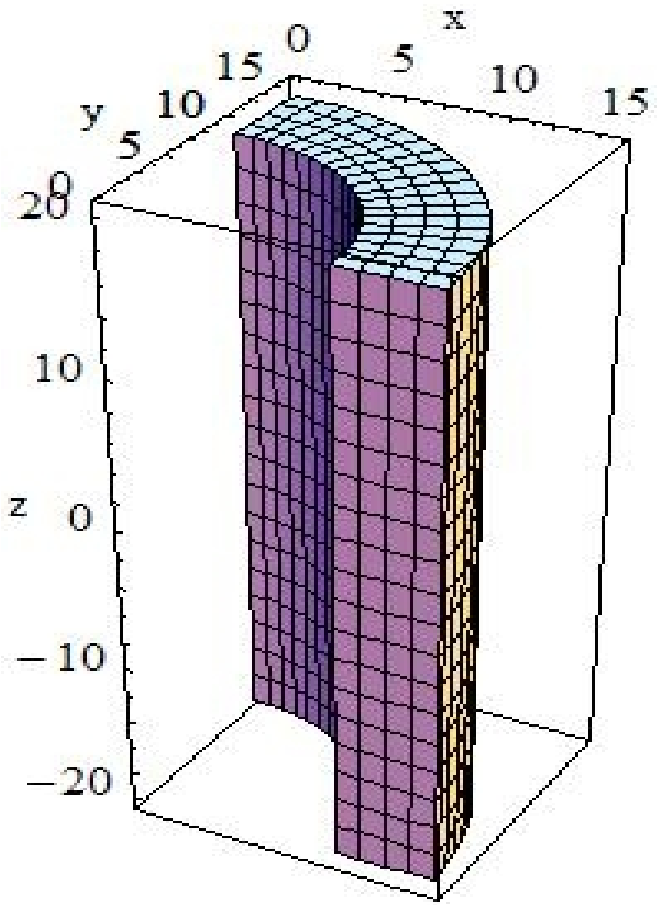


Пример:

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

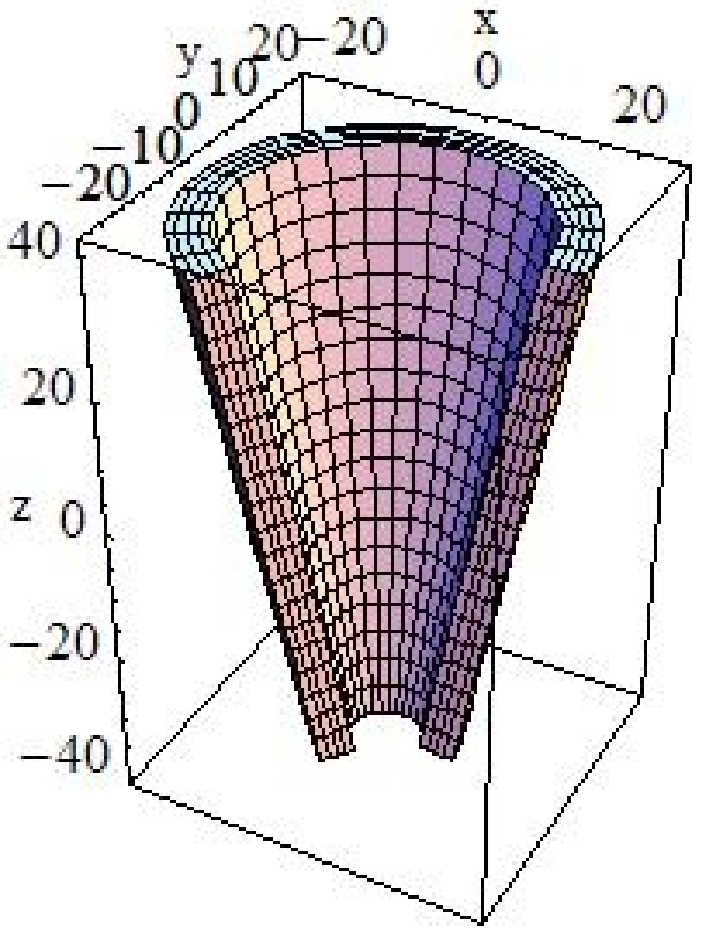
Цилиндр

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



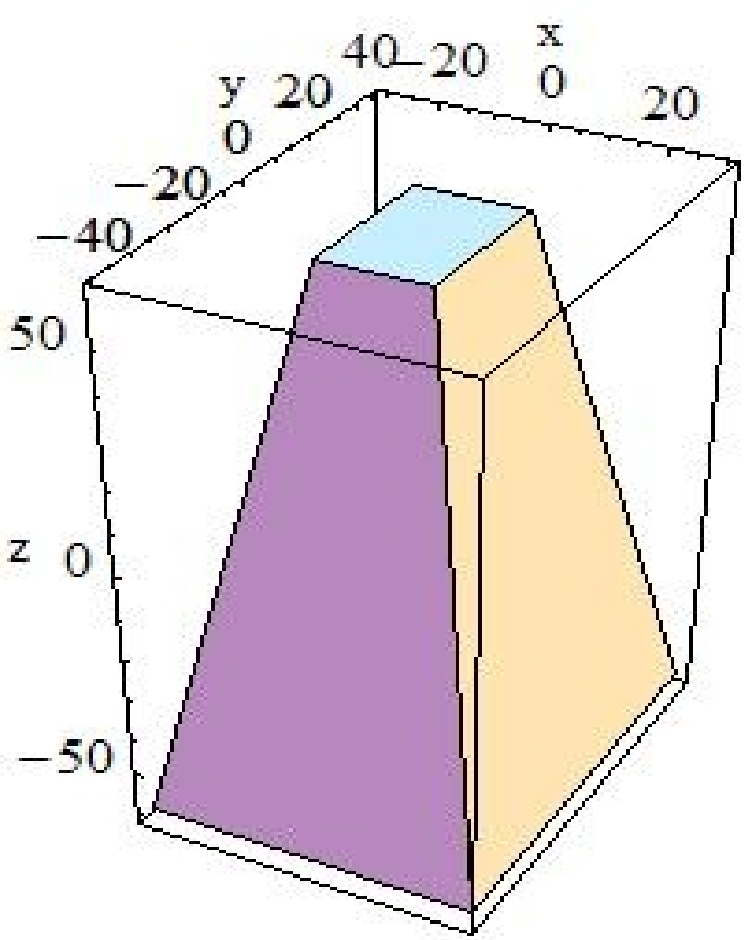
Конус

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



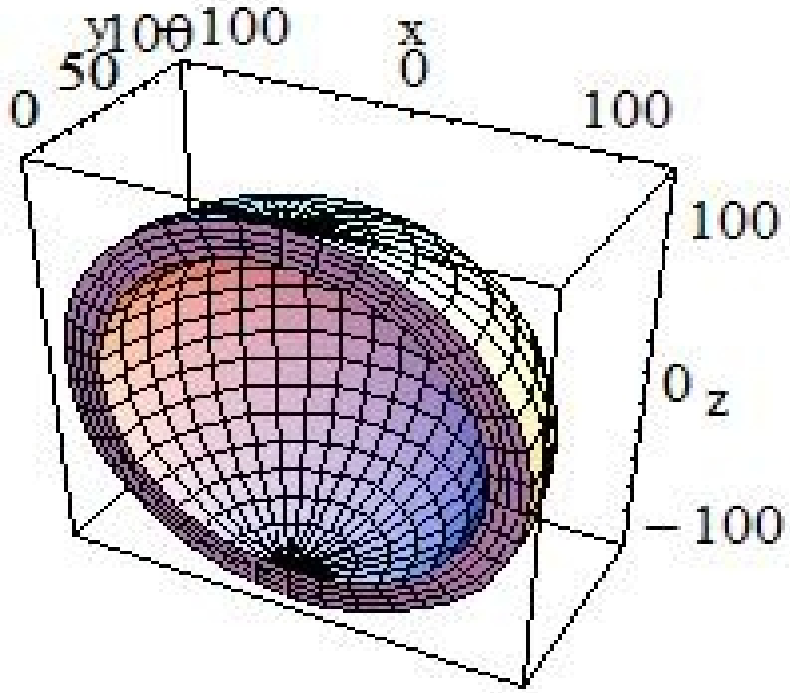
Пирамида

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



Сфера

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



Логический объем

- Строится на основе формы

- Кроме геометрических параметров, содержит описание материала, заполняющего объем, свойства визуализации объема и описание детектирующей способности

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

Физический объем

- Строится на основе логического объема
- Описывает положение объема в пространстве
- Позволяет одновременно описать серию одинаковых объемов (G4Replica) или параметризовать объем (G4VParametrised)

Вложенность объемов

- Все объемы должны быть вложены один в другой
  Перекрытие объемов не допускается!
- В любой модели существует только один “самый верхний объем” (экспериментальный зал), в который “вкладываются” все остальные
- Дочерний объем позиционируется в локальной системе координат, связанной с родительским объемом. Положение любого объекта (объема, частицы и т.д.) одновременно вычисляется как в глобальной координатной системе, связанной с экспериментальным залом, так и в локальной, связанной с объемом, в котором объект в данный момент находится

Проверка перекрытия объемов

```
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!
```

G4VPlacement

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

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;
}
```

Описание элементарных частиц

Каждая частица — это отдельный класс C++ (кроме ионов)

Каждая частица описывается в три этапа:

- **G4ParticleDefinition** - описание постоянных свойств частицы (масса, заряд, название ...)
- **G4DynamicParticle** – описание свойств, изменяющихся при взаимодействии с веществом (энергия, импульс)
- **G4Track** – описание движения частицы в пространстве

Методы G4ParticleDefinition

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

Основные коды PDG

22 - гамма-квант

11 - электрон

-11 - позитрон

2212 - протон

2112 - нейтрон

+-100ZZZAAAI - ион

например

1000010020 - дейтрон

1000010030 - тритон

1000020040 - альфа

1000020030 - He3

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

Класс - генератор событий

- Должен быть наследником **G4VUserPrimaryGeneratorAction**
- Должен содержать объект-наследник класса G4VPrimaryGenerator (например G4ParticleGun)
- Должен содержать описание метода *generatePrimaries*(), в котором происходит вызов метода G4VPrimaryGenerator::generatePrimaryVertex(), создающего первичную вершину
- В одном событии вершина может создаваться при участии нескольких объектов-наследников G4VPrimaryGenerator одновременно

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)

Интерактивное управление генератором – команды /gun/

- /gun/List показать список частиц
- /gun/particle задать тип частицы
- /gun/direction установить направление вылета
- /gun/energy **установить кинетическую энергию**
- /gun/position установить координаты вершины
- /gun/time установить начальное время
- /gun/polarization задать поляризацию
- /gun/number задать число первичных частиц
- /gun/ion задать свойства иона

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);
}
```

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

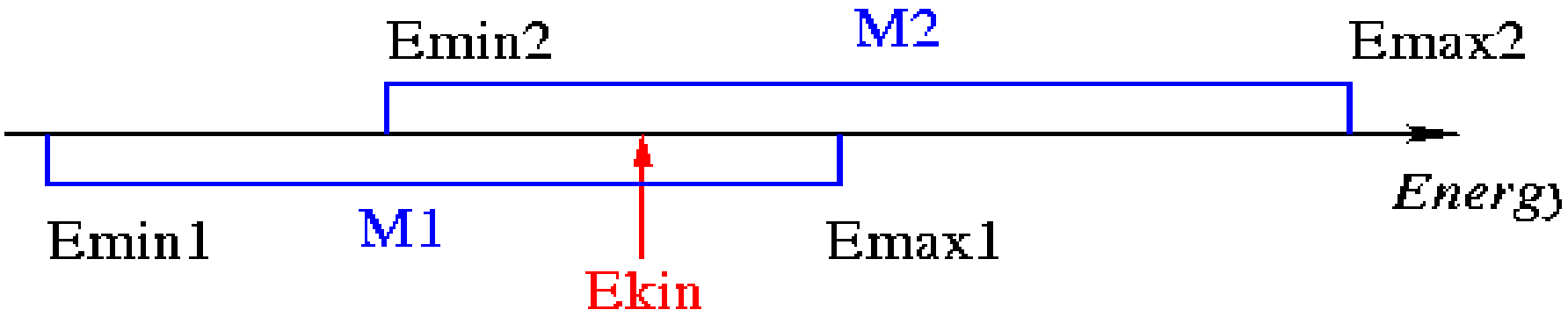
При перекрытии моделей используется следующий алгоритм:

- 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

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

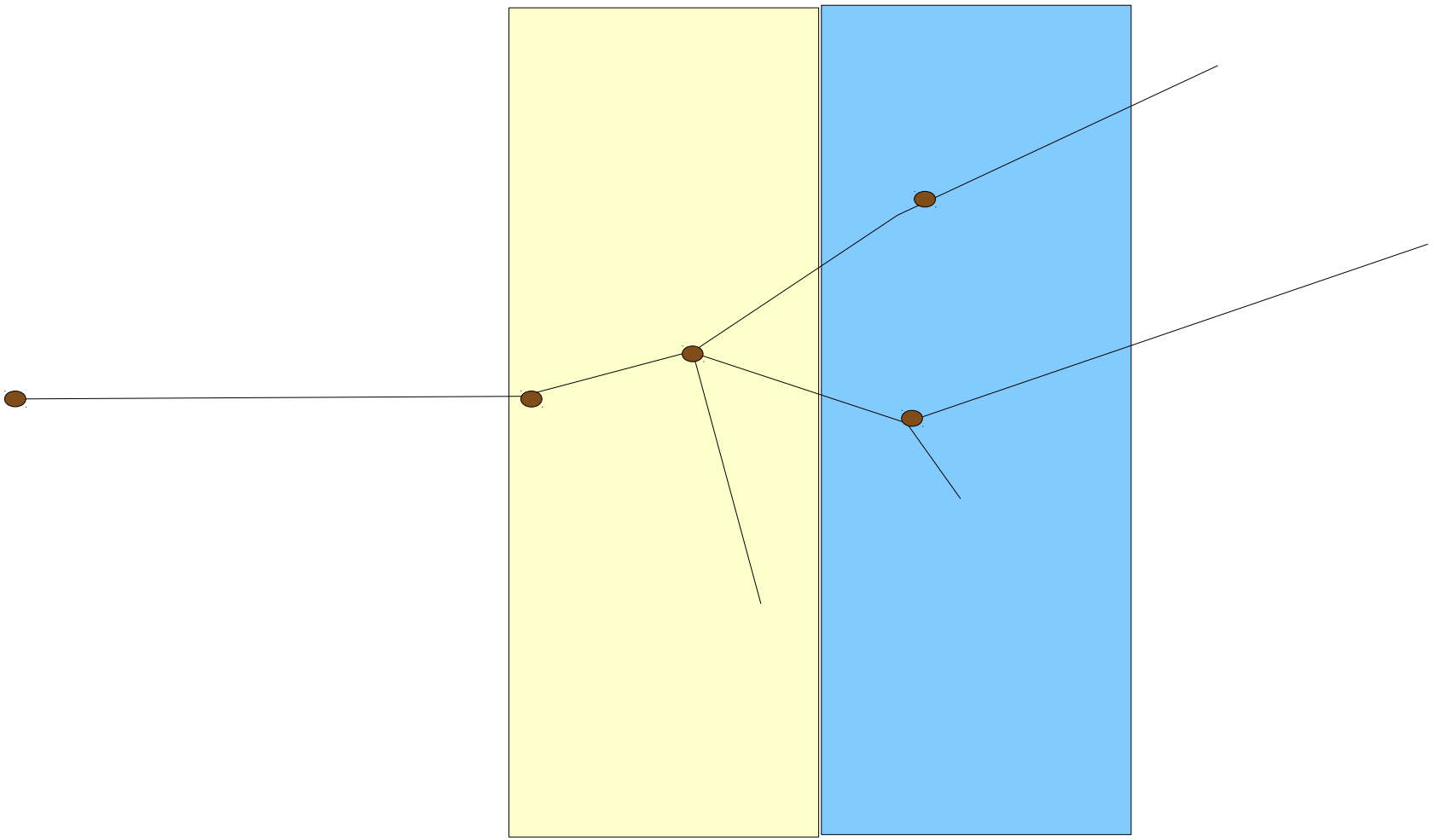
Дискретные и непрерывные процессы

•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



Пороговые значения (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.

Стандартные списки процессов

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

Пояснения

QGSP -кварк-глюонная струнная модель + модель компаунд-ядра

QGSC - кварк-глюонная струнная модель + CHIPS

FTFP - FRITIOF-модель +модель компаунд-ядра

FTFC - FRITIOF-модель +CHIPS

LHEP - адронные взаимодействия на основе параметризации (GHEISHA)

BERT - модель внутриядерного каскада Бертини

BIC - модель бинарного внутриядерного каскада

HP - моделирование взаимодействий нейтронов с повышенной точностью

QBBC - QGSC+BIC(протоны)+BERT(пионы)