

# Физические процессы

Обязательной частью реализации математической модели с использованием Geant4 является построение списка используемых физических процессов.

**По каким законам происходит моделирование?**

## Основные понятия

- Модель (Model) – описание отдельного типа физического взаимодействия частицы в определённом диапазоне энергий.
- Процесс (Process) – описание отдельного типа физического взаимодействия частицы во всём диапазоне энергий.

### 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 моделей (Physics List) – набор всех процессов, заданных для каждой частицы, участвующей в моделировании физических взаимодействий в среде Geant4.

# Физические процессы

## Категории процессов

- электромагнитные взаимодействия
  - ионизация (G4eIonisation)
  - комптоновское рассеяние (G4ComptonScattering)
  - многократное рассеяние (G4MultipleScattering)
  - тормозное излучение (G4eBremsstrahlung)
  - ...
- адронные взаимодействия
  - захват (G4HadronCaptureProcess)
  - деление (G4HadronFissionProcess)
  - упругое рассеяние (G4HadronElasticProcess)
  - неупругое рассеяние (G4HadronInelasticProcess)
- фотоядерные и лептон-ядерные взаимодействия
- распады
  - распады лептонов, полуплептонные распады, радиоактивные распады ядер
  - электромагнитные распады ( $\pi^0$ ,  $\Sigma^0$ )

- оптические
  - рассеяние Рэлея
  - отражение на границе двух сред
  - ...
- параметризация
  - модели, основанные на параметризации экспериментальных данных (распространение электромагнитных ливней)
- транспортировка

Все классы процессов в Geant4 являются наследниками абстрактного класса **G4VProcess**.

Для каждого процесса должны быть перегружены чисто виртуальные методы:

- ✓ AtRestDoIt
- ✓ AlongStepDoIt
- ✓ PostStepDoIt
- ✓ AtRestGetPhysicalInteractionLength
- ✓ AlongStepGetPhysicalInteractionLength
- ✓ PostStepGetPhysicalInteractionLength

# Физические процессы

## Транспортировка (G4Transportation)

Перемещение частицы в пространстве, без изменения ее свойств, выделения энергии и образования новых частиц



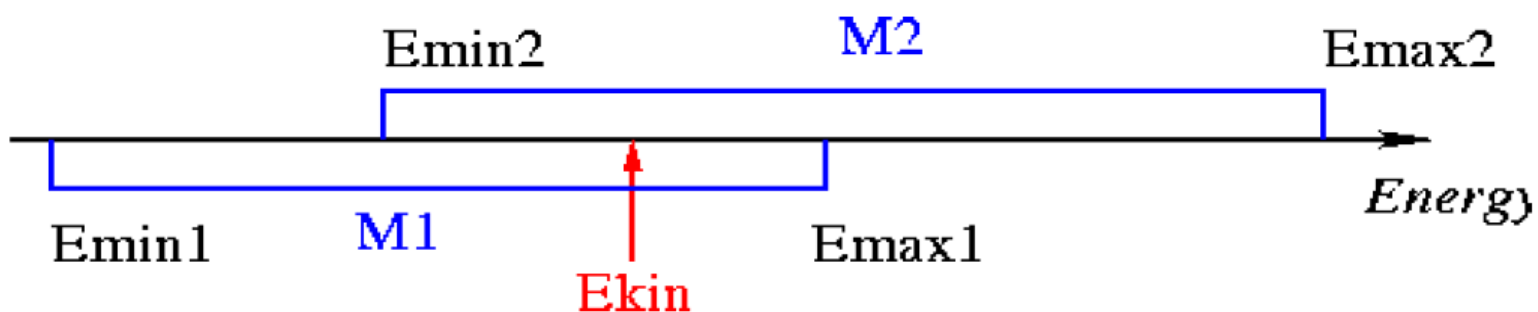
- Расчёт расстояния до следующей границы объёма
- Расчёт времени трекинга для каждой частицы

# Физические процессы

При перекрытии моделей используется следующий алгоритм:

- если данной энергии соответствует более двух моделей и диапазоны энергий моделей перекрываются полностью, вырабатывается исключение
- В случае частичного перекрытия двух моделей, модель выбирается случайно, но наиболее вероятен выбор модели, предел применимости которой лежит дальше от данной энергии частицы

Есть модели M1 и M2 и частица с энергией  $E_{kin}$



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

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

# Физические процессы

## 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.

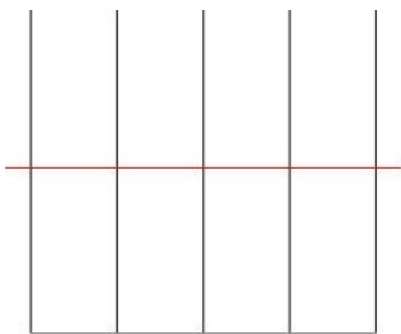
Пример: распад, упругое и неупругое рассеяние

# Физические процессы

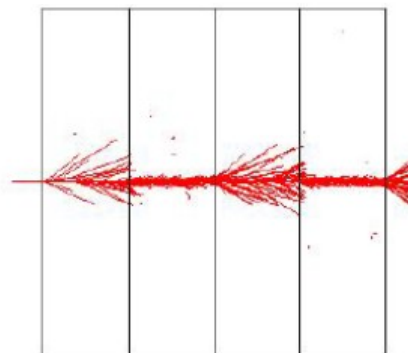
## Пороговая энергия.

- Каждый процесс рождения вторичных частиц имеет собственные ограничения на минимальную энергию этих частиц
- Движение всех вторичных частиц моделируется до нулевой энергии.
- Каждая частица имеет минимальное **пороговое значение остаточного пробега** (1 мм по умолчанию) для каждого материала, которое пересчитывается в энергию. Данное значение используется при моделировании процессов.
- Ниже порога энергия трассируемой частицы уменьшается за счёт непрерывных потерь энергии без генерации вторичных частиц).

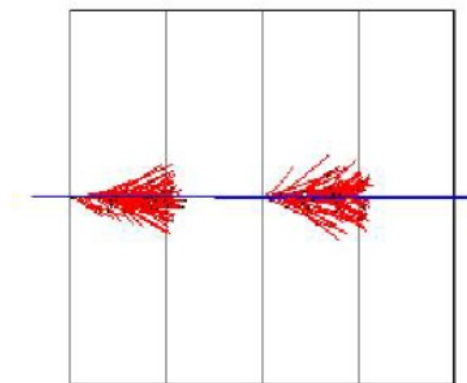
500 MeV p in LAr-Pb Sampling Calorimeter (жидкий аргон/свинец)



Cut = 2 MeV



Cut = 450 keV



Production range = 1.5 mm

Появление частиц с энергией ниже порога:

- Рождение вторичной частицы с энергией ниже порога позволяет получить энергосигнал в ближайшем чувствительном объёме.
- При рождении пар гамма-квантом позитрон с энергией ниже порога участвует в аннигиляции.

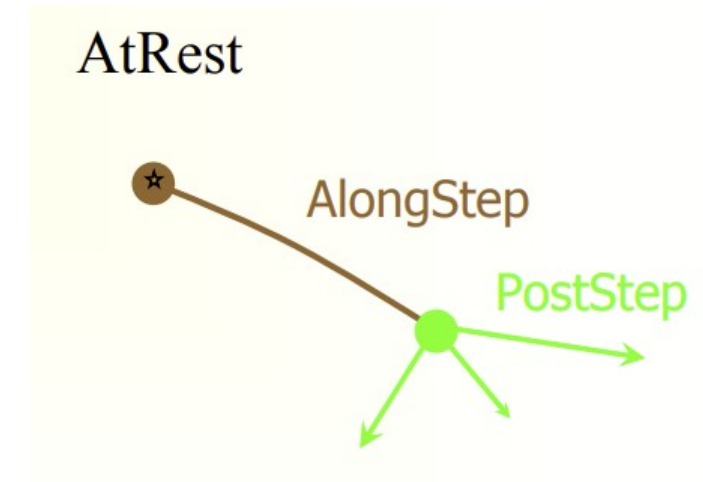
# Физические процессы

Моделирование каждого процесса даёт ответ на два вопроса:

1. Когда и где произойдёт взаимодействие?  
метод `GetPhysicalInteractionLength()` использует сечение процесса или время распада, для процесса транспортировки используется расстояние вдоль трека до границ ближайшего объёма.
2. Какие частицы образуются в результате взаимодействия или как меняется импульс первичной частицы?  
метод `DoIt()` использует соответствующую данному процессу физическую модель.

В зависимости от типа процесса взаимодействие первичной частицы может произойти в конкретной точке пространства (`PostStep`), быть пространственно распределённым вдоль текущего шага (`AlongStep`) или произойти в конкретный момент времени (`AtRest`).

Для сложных процессов реализуется комбинация всех трёх вариантов.



# Физические процессы

Оценка длины взаимодействия для данного процесса:

$$\left( \frac{1}{\sigma_i n} \right), \text{ где } \sigma_i - \text{сечение процесса, } n - \text{концентрация ядер мишени.}$$

Пример. Для случая взаимодействия гамма кванта с веществом при энергиях  $\sim$  МэВ длина взаимодействия для комптоновского рассеяния (большое сечение) будет значительно меньше, чем длина взаимодействия для рождения пар (меньшее сечение).

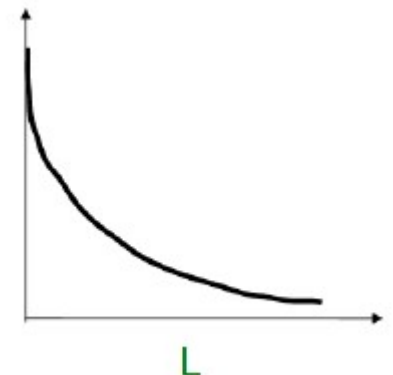
Длина первого шага  $L$  трекинга выбирается равной минимальной длине взаимодействия среди всех процессов с учётом процесса транспортировки (расстояние до границы объёма).

Производится расчёт непрерывных взаимодействий на длине шага, в конце шага изменяется значение кинетической энергии.

Проверяется достижение пороговой энергии в соответствии с минимальной величиной остаточного пробега.

Изменяется положение частицы.

Производится расчёт дискретных процессов методом Монте-Карло для распределения  $e^{-\sigma_i n L}$ , при необходимости создаётся список вторичных частиц.





# Физические процессы

## Взаимодействия гамма кванта с веществом.

Синий цвет – образование пар.

Красный цвет – Комптон эффект.

Чёрный цвет – транспортировка.

- **Step 1:**

- all lengths sampled

- Compton occurs

- **Step 2:**

- Compton re-sampled

- boundary is crossed

- **Step 3:**

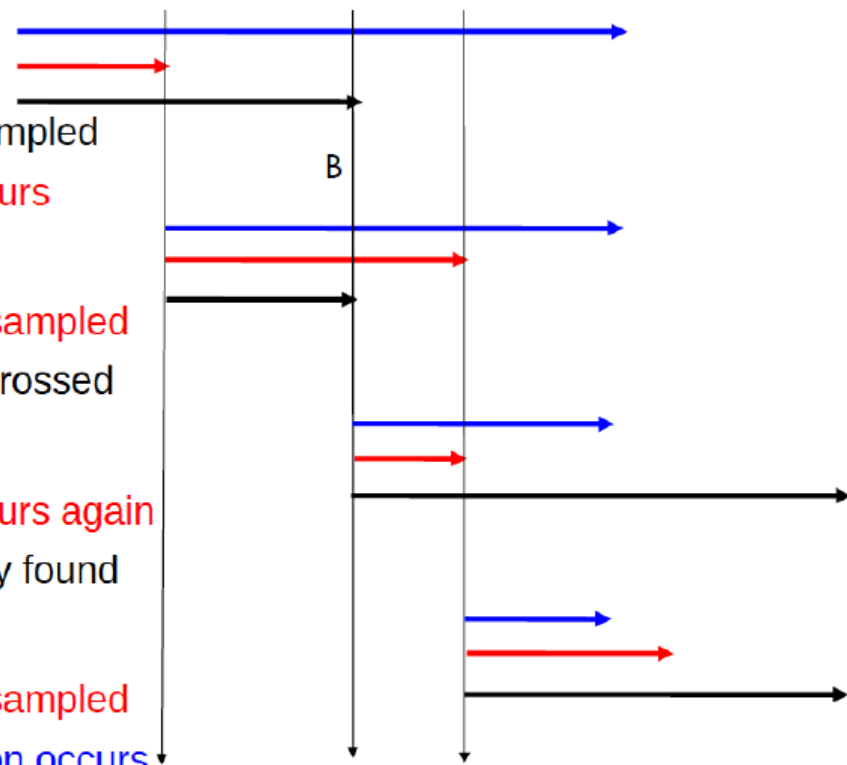
- Compton occurs again

- new boundary found

- **Step 4:**

- Compton re-sampled

- pair production occurs



При реализации процесса на данном шаге его длина взаимодействия на следующем шаге вычисляется заново. Для других нереализованных на данном шаге процессов из длины взаимодействия на следующем шаге вычитается значение длины взаимодействия на предыдущем шаге.

# Физические процессы

50 MeV e- entering LAr-Pb calorimeter

Процессы, участвующие в моделировании:

радиационные потери

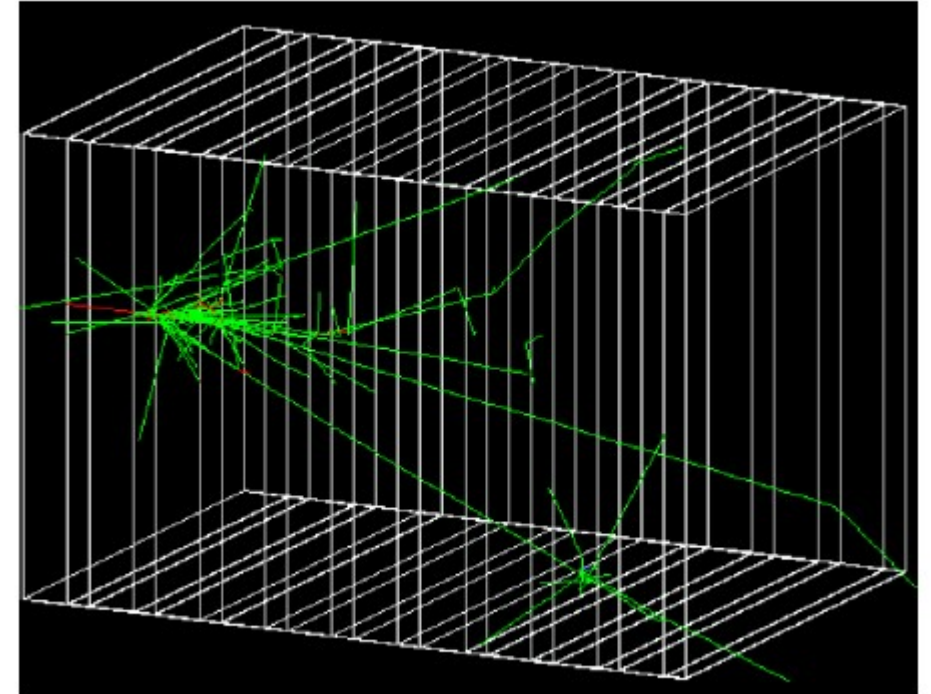
ионизационные потери

многократное рассеяние

аннигиляция позитрона

рождение пар

Комптоновское рассеяние



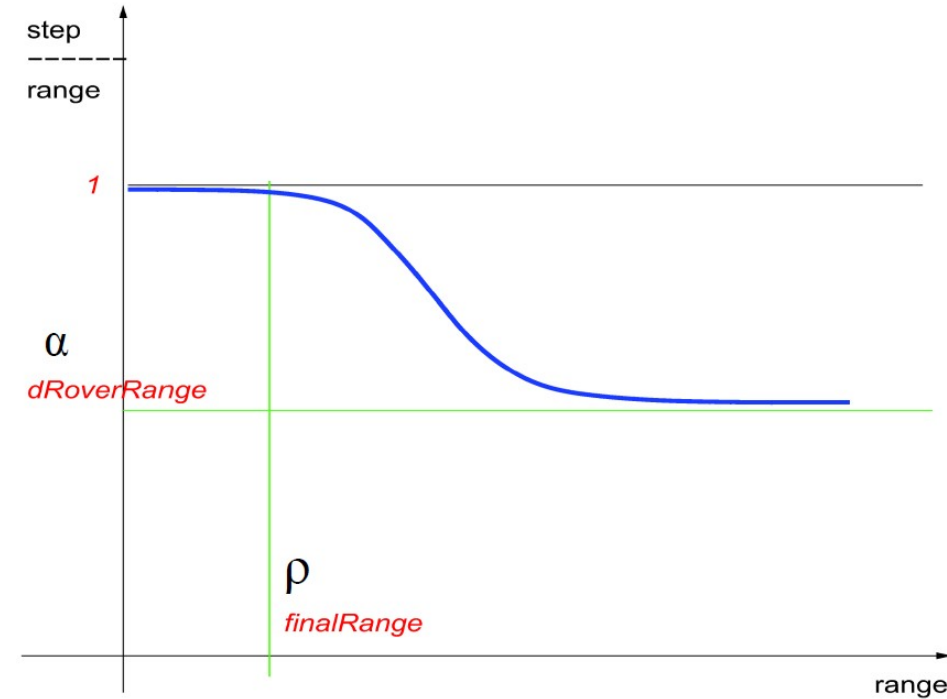
# Физические процессы

- Моделирование ионизации в Geant4 имеет непрерывную ( $dE/dx$ ) и дискретную ( $\delta$ -электроны, тормозное излучение) компоненты. Соотношение регулируется значением порога рождения вторичных частиц.
- При моделировании ионизационных потерь на шаге, принимаются во внимание флуктуации потерь
- Баланс «точность — скорость вычислений» достигается использованием функции шага:

$$\text{Длина шага} = \max(\rho, \alpha R(E) + \rho(1-\alpha)(2-\rho/R(E)))$$

где  $\rho$  — минимальная длина шага (1 мм по умолчанию)

$\alpha$  — параметр (по умолчанию 0,2),  $R(E)$  - пробег частицы для данной энергии.



Коррекция функции шага при создании набора процессов:

```
G4eIonisation* eIoni = new G4eIonisation();  
eIoni->SetStepFunction(0.2, 100*um);
```

в командной строке  
`/process/eLoss/StepFunction 0.2 100 um`

# Физические процессы

## Взаимодействия адронов

ядерные реакции

*деление, захват*

*мультифрагментация*

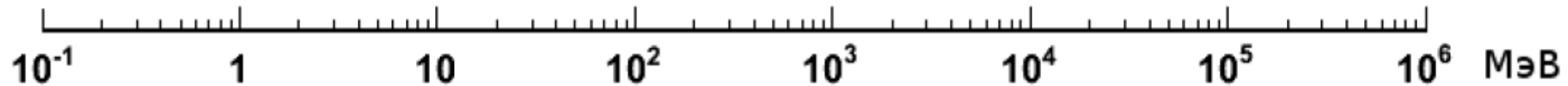
множественное рождение адронов

*внутриядерные*

*каскады*

*адронизация кварков и глюонов*

рассеяние на нуклонах



Невозможно описать все взаимодействия одной моделью

Вероятность взаимодействия, и вероятность  
образования конкретного конечного состояния  
рассчитываются независимо

# Физические процессы

Для каждого процесса наследника класса **G4VProcess** при моделировании траектории вызываются:

Метод **AlongStepDoIt** вызывается на каждом шаге при трекинге частицы для каждого  $i$ -ого процесса независимо от того, какой из процессов даёт минимальный шаг  $\left(\frac{1}{\sigma_i n}\right)_{MIN}$ , где  $\sigma_i$  - сечение процесса,  $n$  – концентрация ядер мишени.

**AlongStepGetPhysicalInteractionLength** – расчёт длины шага для данного процесса.

Метод **PostStepDoIt** вызывается в конце шага при трекинге частицы только для процесса, который даёт минимальный шаг  $\left(\frac{1}{\sigma_i n}\right)_{MIN}$ , т.е. величина шага определяется данным процессом, или в случае, если данный процесс принудительно вызывается.

Информация о треке обновляется после каждого вызова **PostStepDoIt**.

**PostStepGetPhysicalInteractionLength** – расчёт длины шага для данного процесса.

Метод **AtRestDoIt** вызывается при остановке частицы, которая была вызвана данным процессом (распад), или в случае, если данный процесс принудительно вызывается.

**AtRestGetPhysicalInteractionLength** – расчёт длины временного интервала на текущем шаге для данного процесса.

Обычно для данного процесса используется какой-либо один метод, но возможны более сложные случаи, когда используются несколько методов одновременно (ионизация и образование дельта-электронов).

# Физические процессы

Задание простых процессов через классы-наследники **G4VProcess**:

**G4VRestProcess** – для процессов с одним методом `AtRestDoIt` (захват нейтронов);

**G4VDiscreteProcess** – для процессов с одним методом `PostStepDoIt` (комptonовское рассеяние, неупругое взаимодействие адронов);

**G4VContinuousProcess** – для процессов с одним методом `AlongStepDoIt` (излучение Черенкова).

Задание более сложных процессов через классы-наследники **G4VProcess**:

**G4VContinuousDiscreteProcess** – для процессов с методами `AlongStepDoIt` и `PostStepDoIt` (транспортировка, ионизация: потери энергии, дельта-электроны);

**G4VRestDiscreteProcess** – для процессов с методами `AtRestDoIt` и `PostStepDoIt` (аннигиляция позитрона, распад);

**G4VRestContinuousProcess** – для процессов с методами `AtRestDoIt` и `AlongStepDoIt`;

**G4VRestContinuousDiscreteProcess** – для процессов с методами `AtRestDoIt`, `AlongStepDoIt` и `PostStepDoIt`.

# Физические процессы

Управление процессами реализуется в классе **G4ProcessManager**.

G4ProcessManager содержит список физических процессов и пороговых значений для каждого типа частиц.

Для каждой частицы, участвующей в моделировании, создаётся **СВОЙ** объект класса G4ProcessManager.

При трекинге каждой частицы G4ProcessManager моделирует физические процессы, в которых может участвовать данная частица.

Добавление процессов и порядок их моделирования: AddProcess(), SetProcessOrdering().

Добавление простых процессов: AddRestProcess(), AddContinuousProcess(), AddDiscreteProcess().

Активация/деактивация добавленных процессов: ActivateProcess(), InActivateProcess().

Доступ к объекту:

```
G4ParticleDefinition* particle = G4Proton::Proton();
```

```
G4ProcessManager* pmanager = particle->GetProcessManager();
```

# Физические процессы

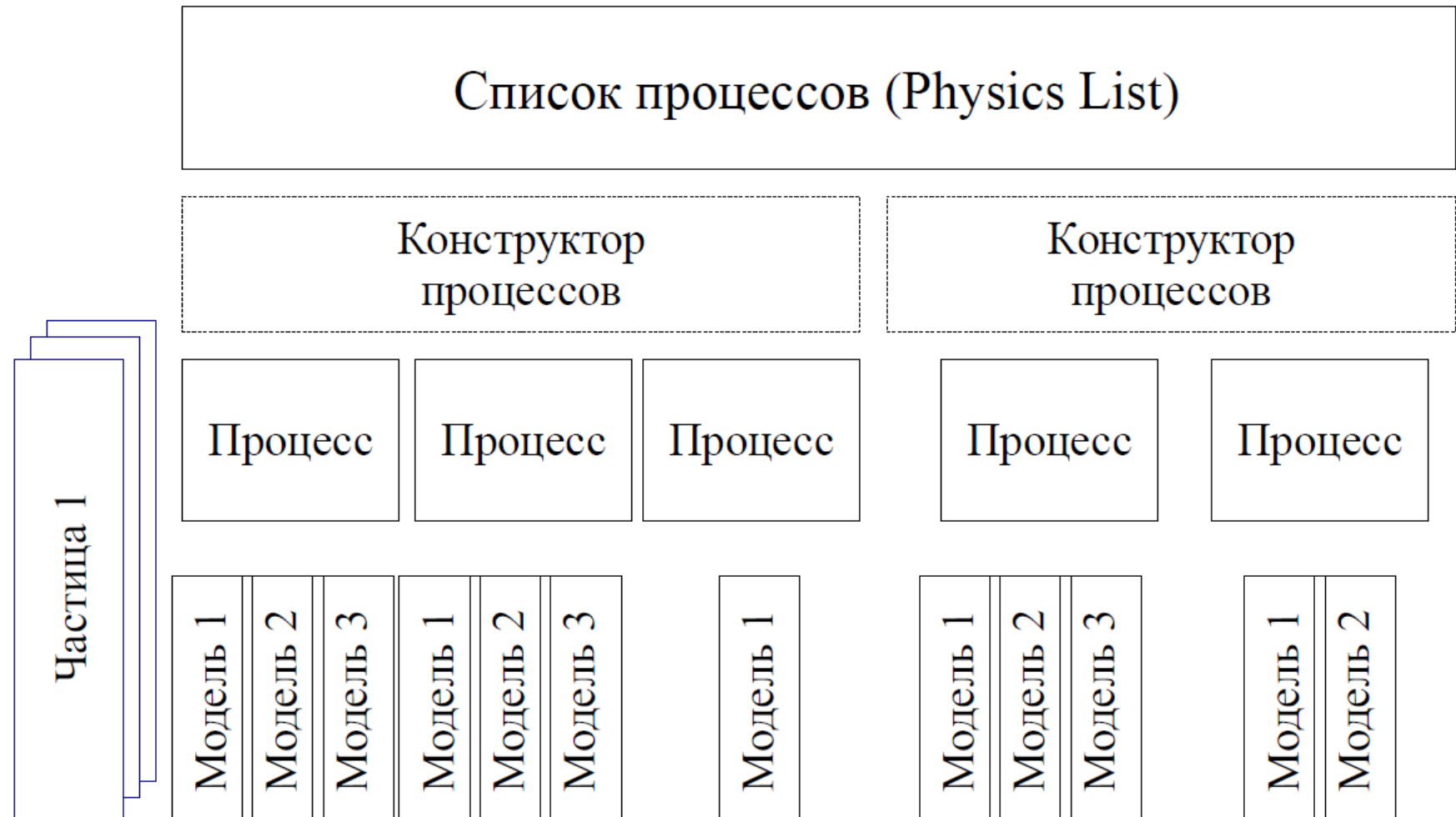
Полное описание всех физических моделей и процессов для данной задачи моделирования формирует **PhysicsList**.

PhysicsList содержится в объекте-наследнике класса **G4VUserPhysicsList**, в котором пользователь должен определить все типы частиц и процессы, участвующие в данном моделировании.

В объекте-наследнике класса **G4VUserPhysicsList** должны быть переопределены чисто виртуальные методы:

**ConstructParticle()** (описание частиц, участвующих в моделировании),  
**ConstructProcess()** (описание процессов для каждого типа частиц).

Также может быть переопределён виртуальный метод **SetCuts()** для установки минимального значения остаточного пробега





# Физические процессы

Заголовочный файл для объекта-наследника **G4VUserPhysicsList**

```
#include <G4VUserPhysicsList.hh>
#include "globals.hh"
```

```
class MyPhysicsList: public G4VUserPhysicsList
{
public:
    MyPhysicsList();
    ~MyPhysicsList();

protected:
    void ConstructParticle();
    void ConstructProcess();
    void SetCuts();
}
```

```
void MyPhysicsList::ConstructParticle()
{
    G4LeptonConstructor lConstructor;
    lConstructor.ConstructParticle();
    G4MesonConstructor mConstructor;
    mConstructor.ConstructParticle();
}
```

Варианты ConstructParticle()

```
void MyPhysicsList::ConstructParticle()
{
    G4Proton::ProtonDefinition();
    G4Geantino::GeantinoDefinition();
    ...
    G4Electron::ElectronDefinition();
}
```

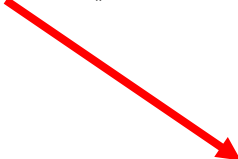
Конструкторы шести классов частиц:

- G4BosonConstructor
- G4LeptonConstructor
- G4MesonConstructor
- G4BaryonConstructor
- G4IonConstructor
- G4ShortlivedConstructor.

# Физические процессы

## Варианты ConstructProcess()

```
void MyPhysicsList::ConstructProcess()
{
    AddTransportation();
    ConstructEM();
}
```



```
void MyPhysicsList::ConstructEM()
{
    G4ParticleDefinition* particle = G4Gamma::GammaDefinition();
    G4ProcessManager* pmanager = particle->GetProcessManager();
```

```
G4PhotoElectricEffect * thePhotoElectricEffect = new G4PhotoElectricEffect();
G4ComptonScattering * theComptonScattering = new G4ComptonScattering();
G4GammaConversion* theGammaConversion = new G4GammaConversion();
```

```
pmanager->AddDiscreteProcess(thePhotoElectricEffect);
pmanager->AddDiscreteProcess(theComptonScattering);
pmanager->AddDiscreteProcess(theGammaConversion);
}
```

# Физические процессы

## Варианты ConstructProcess()

Подключение High Precision Models (HP) для нейтронов низких энергий (до 20 МэВ), а именно, подключение процессов упругого (G4HadronElasticProcess) и неупругого (G4NeutronInelasticProcess) взаимодействия нейтронов.

Добавление сечений процессов

Установка диапазона действия модели

Добавление специальных моделей для процессов

Привязка процессов к типу частиц (нейтронам)

```
void MyPhysicsList::ConstructProcess()
{
    auto theNeutronElasticProcess = new G4HadronElasticProcess;
    theNeutronElasticProcess->AddDataSet(new G4ParticleHPElasticData());
    theNeutronElasticProcess->AddDataSet(new G4ParticleHPThermalScatteringData);

    auto theNeutronElasticModel = new G4ParticleHPElastic;
    theNeutronElasticModel->SetMinEnergy(4.0*eV);
    theNeutronElasticProcess->RegisterMe(theNeutronElasticModel);

    theNeutronThermalElasticModel = new G4ParticleHPThermalScattering;
    theNeutronThermalElasticModel->SetMaxEnergy(4.0*eV);
    theNeutronElasticProcess->RegisterMe(theNeutronThermalElasticModel);

    G4ProcessManager* pmanager = G4Neutron::Neutron()->GetProcessManager();
    pmanager->AddDiscreteProcess(theNeutronElasticProcess);

    auto theNeutronInelasticProcess = new G4NeutronInelasticProcess();
    theNeutronInelasticProcess->AddDataSet(new G4ParticleHPInelasticData());

    auto theNeutronInelasticModel = new G4ParticleHPInelastic;
    theNeutronInelasticProcess->RegisterMe(theNeutronInelasticModel);
    pmanager->AddDiscreteProcess(theNeutronInelasticProcess);
}
```

The diagram illustrates the mapping between the Russian descriptions of the `ConstructProcess()` function and the corresponding C++ code lines. The descriptions on the left are connected to the code on the right by colored arrows:

- Подключение High Precision Models (HP) для нейтронов низких энергий (до 20 МэВ), а именно, подключение процессов упругого (G4HadronElasticProcess) и неупругого (G4NeutronInelasticProcess) взаимодействия нейтронов.** (Red arrow) points to the first three lines of code, which initialize the `G4HadronElasticProcess` and `G4NeutronInelasticProcess` objects.
- Добавление сечений процессов** (Red arrow) points to the lines where data sets are added to the elastic process: `theNeutronElasticProcess->AddDataSet(new G4ParticleHPElasticData());` and `theNeutronElasticProcess->AddDataSet(new G4ParticleHPThermalScatteringData);`.
- Установка диапазона действия модели** (Black arrow) points to the lines where the energy range is set for the elastic model: `theNeutronElasticModel->SetMinEnergy(4.0*eV);` and `theNeutronThermalElasticModel->SetMaxEnergy(4.0*eV);`.
- Добавление специальных моделей для процессов** (Green arrow) points to the lines where the models are registered to the processes: `theNeutronElasticProcess->RegisterMe(theNeutronElasticModel);` and `theNeutronInelasticProcess->RegisterMe(theNeutronInelasticModel);`.
- Привязка процессов к типу частиц (нейтронам)** (Blue arrow) points to the lines where the processes are added to the `G4ProcessManager`: `pmanager->AddDiscreteProcess(theNeutronElasticProcess);` and `pmanager->AddDiscreteProcess(theNeutronInelasticProcess);`.

# Физические процессы

## Варианты SetCuts()

```
void MyPhysicsList::SetCuts()
{
    SetCutValue(cutForGamma,"gamma");
    SetCutValue(cutForElectron,"e-");
    SetCutValue(cutForPositron,"e+");
}
```

Одинаковые пороговые значения для всех частиц

```
void MyPhysicsList::SetCuts()
{
    SetCutsWithDefault();
}
```

`defaultCutValue = 1.0*mm;`

# Физические процессы

## Важные замечания

- В Geant4 отсутствует проверка на то, что данный процесс уже добавлен. Добавляя процесс несколько раз для данной частицы, вы во столько же раз увеличиваете его вклад
- От правильного подбора моделей зависит результат моделирования. В зависимости от приближений, сделанных в модели, расхождение может быть значительным
- Если вы создаете свой набор процессов, по возможности, делайте его верификацию, используя экспериментальные данные

# Физические процессы

## Управление набором процессов из командной строки

`/process/list` — вывести список процессов

`/process/dump ProcessName` — вывести справку по процессу

`/process/inactivate ProcessName` - выключить процесс из набора

`/process/activate ProcessName` — включить процесс в набор, если процесс описан, но неактивен

# Физические процессы

команда

Idle> /process/list

Transportation, msc, hIoni, ionIoni  
eIoni, eBrem, annihil, phot  
compt, conv, hBrems, hPairProd  
muMsc, muIoni, muBrems, muPairProd  
CoulombScat, PhotonInelastic, ElectroNuclear, PositronNuclear  
Decay, hElastic, NeutronInelastic, nCapture  
nFission, ProtonInelastic, PionPlusInelastic, PionMinusInelastic  
KaonPlusInelastic, KaonMinusInelastic, KaonZeroLInelastic, KaonZeroSInelastic  
AntiProtonInelastic, AntiNeutronInelastic, LambdaInelastic, AntiLambdaInelastic  
SigmaMinusInelastic, AntiSigmaMinusInelastic,  
SigmaPlusInelastic, AntiSigmaPlusInelastic XiMinusInelastic, AntiXiMinusInelastic,  
XiZeroInelastic, AntiXiZeroInelastic OmegaMinusInelastic,  
AntiOmegaMinusInelastic, CHIPSNuclearCaptureAtRest, muMinusCaptureAtRest  
DeuteronInelastic, TritonInelastic, AlphaInelastic, nKiller



# Физические процессы

Существуют укомплектованные разработчиками Geant4 списки физических процессов

## Стандартные списки процессов

FTF\_BIC FTFP\_BERT\_ATL **FTFP\_BERT**  
FTFP\_BERT\_HP FTFP\_BERT\_TRV FTFP\_INCLXX  
FTFP\_INCLXX\_HP FTFQGSP\_BERT  
LBE NuBeam  
**QBBC** QGS\_BIC **QGSP\_BERT** QGSP\_BERT\_HP  
QGSP\_BIC\_AllHP QGSP\_BIC QGSP\_BIC\_HP  
QGSP\_FTFP\_BERT QGSP\_INCLXX  
QGSP\_INCLXX\_HP  
Shielding ShieldingLEND

CHiral Invariant Phase Spase (CHIPS)  
quark-gluon string model (FRITIOF or FTF)

## Пояснения


<b>QGSP</b>	-кварк-глюонная струнная модель + модель компаунд-ядра
<b>QGSC</b>	- кварк-глюонная струнная модель + CHIPS
<b>FTFP</b>	- FRITIOF-модель +модель компаунд-ядра
<b>FTFC</b>	- FRITIOF-модель +CHIPS
<b>LHEP</b>	- адронные взаимодействия на основе параметризации (GHEISHA)
<b>BERT</b>	- модель внутриядерного каскада Бертини
<b>BIC</b>	- модель бинарного внутриядерного каскада
<b>HP</b>	- моделирование взаимодействий нейтронов с повышенной точностью
<b>QBBC</b>	- QGSC+BIC(протоны)+BERT(пионы)



# Физические процессы

## Дополнение стандартного набора процессов


модульный физлист



```
G4VModularPhysicsList* plist = new QBBC;  
plist->RegisterPhysics(new G4OpticalPhysics);  
plist->RegisterPhysics(new MyPhysics);  
plist->SetDefaultCutValue(1.0*mm);  
runManager->SetUserInitialization(plist);
```

Loader.cc

```
G4VModularPhysicsList* physicsList = new QBBC;  
physicsList->RegisterPhysics(new G4StepLimiterPhysics());  
runManager->SetUserInitialization(physicsList);
```



Добавление процесса, обеспечивающего ограниченное значение шага. Значение шага может задаваться методом SetUserLimits() для логических объёмов.

Geometry.cc

```
fStepLimit = new G4UserLimits(maxStep,maxTrack,maxTime,minE,minRange);  
Pb_log->SetUserLimits(fStepLimit);
```