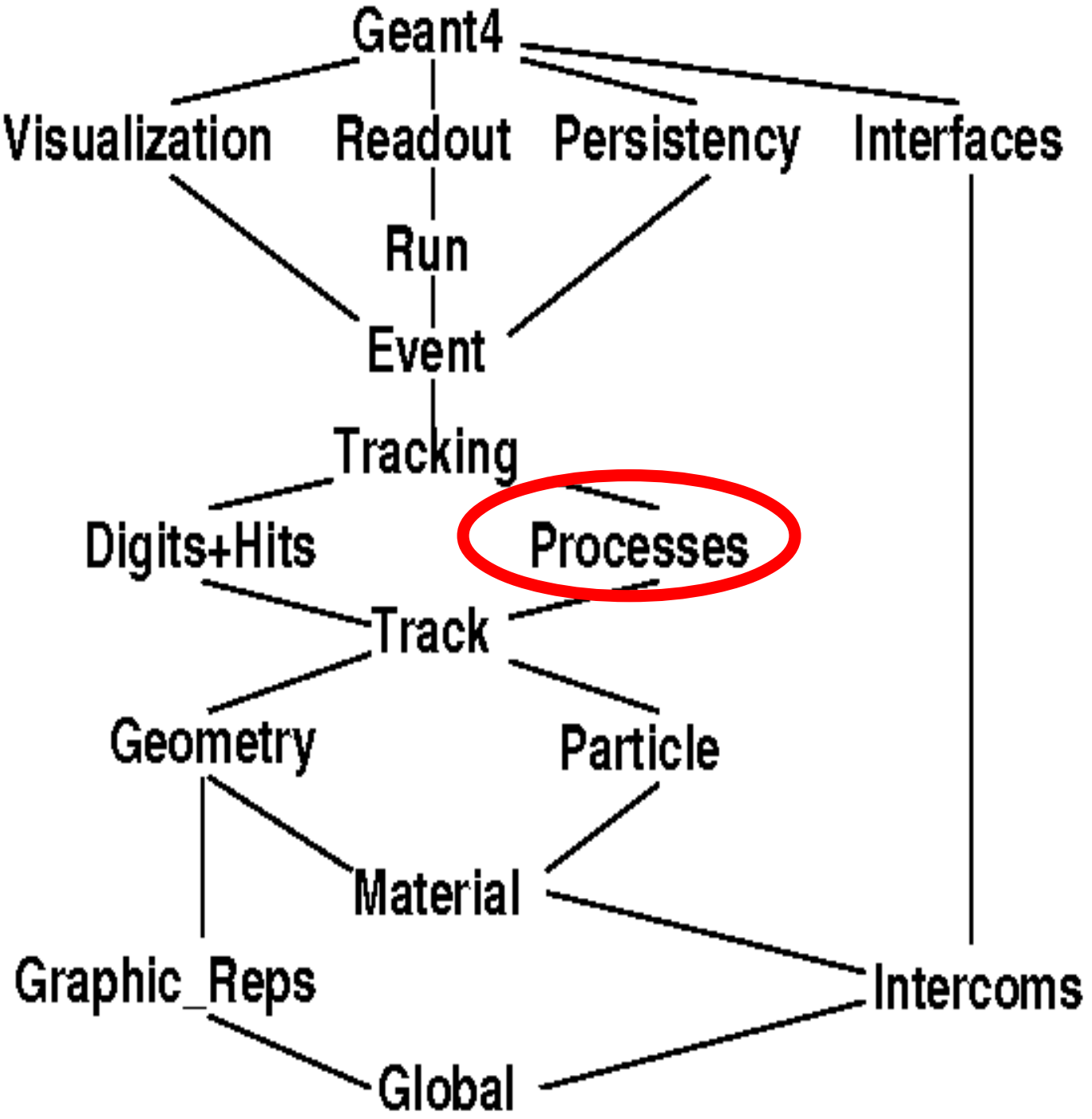
Моделирование частиц и физических процессов в веществе

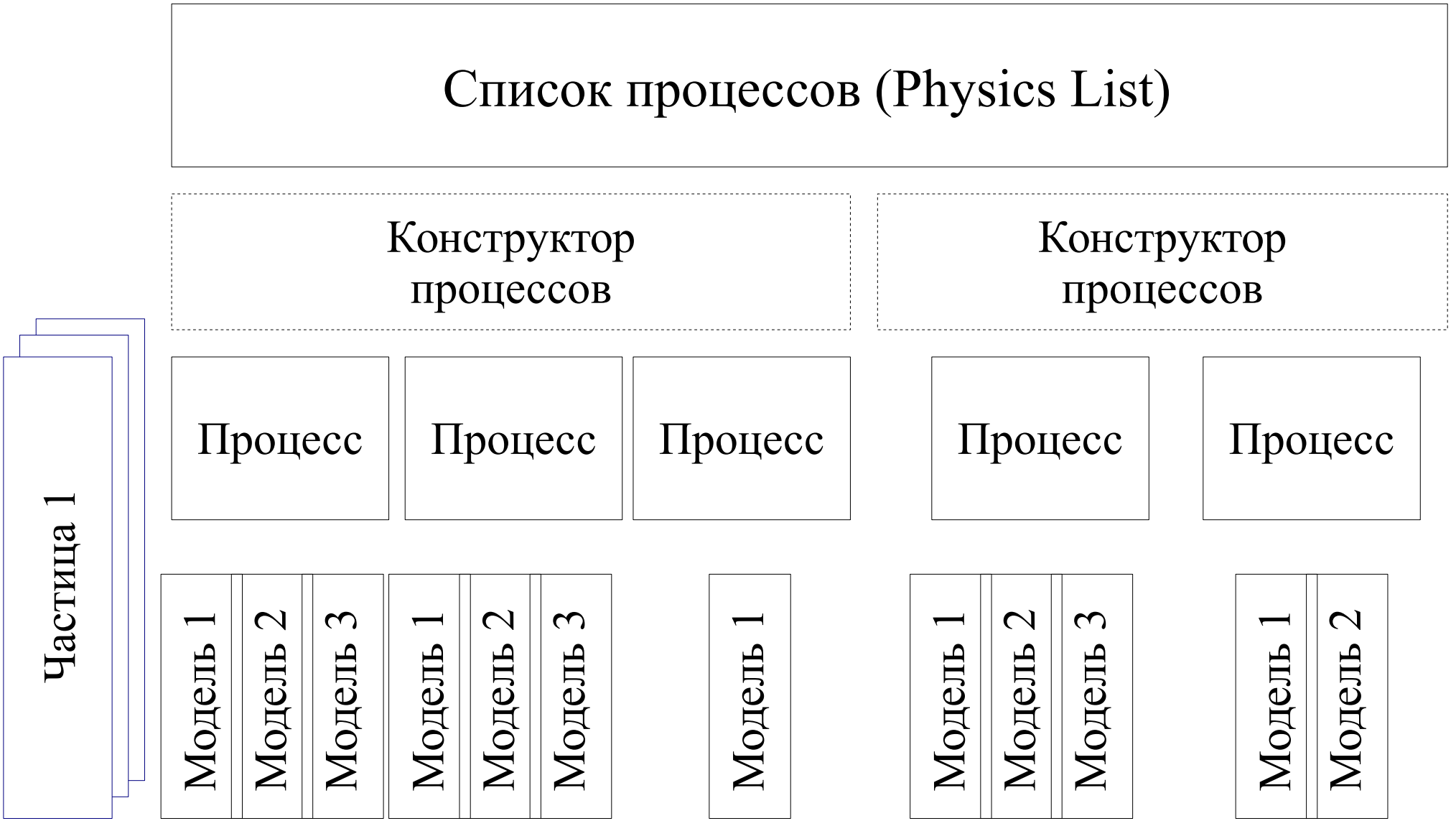

Geant4
Visualization
Readout
Persistency
Interfaces
Run
Event
Tracking
Digits+Hits
Processes
Track
Geometry
Particle
Material
Graphic_Reps
Intercoms
Global

Основные понятия

- 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
  - отражение на границе двух сред
  - ...
- ***параметризация и «быстрое» моделирование***

Использование нескольких моделей в одном процессе

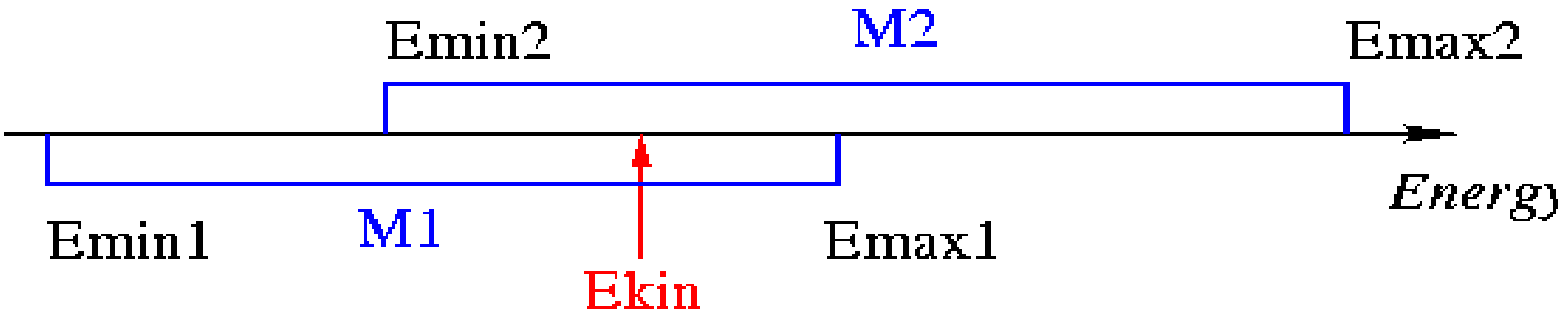
При перекрытии моделей используется следующий алгоритм:

- 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

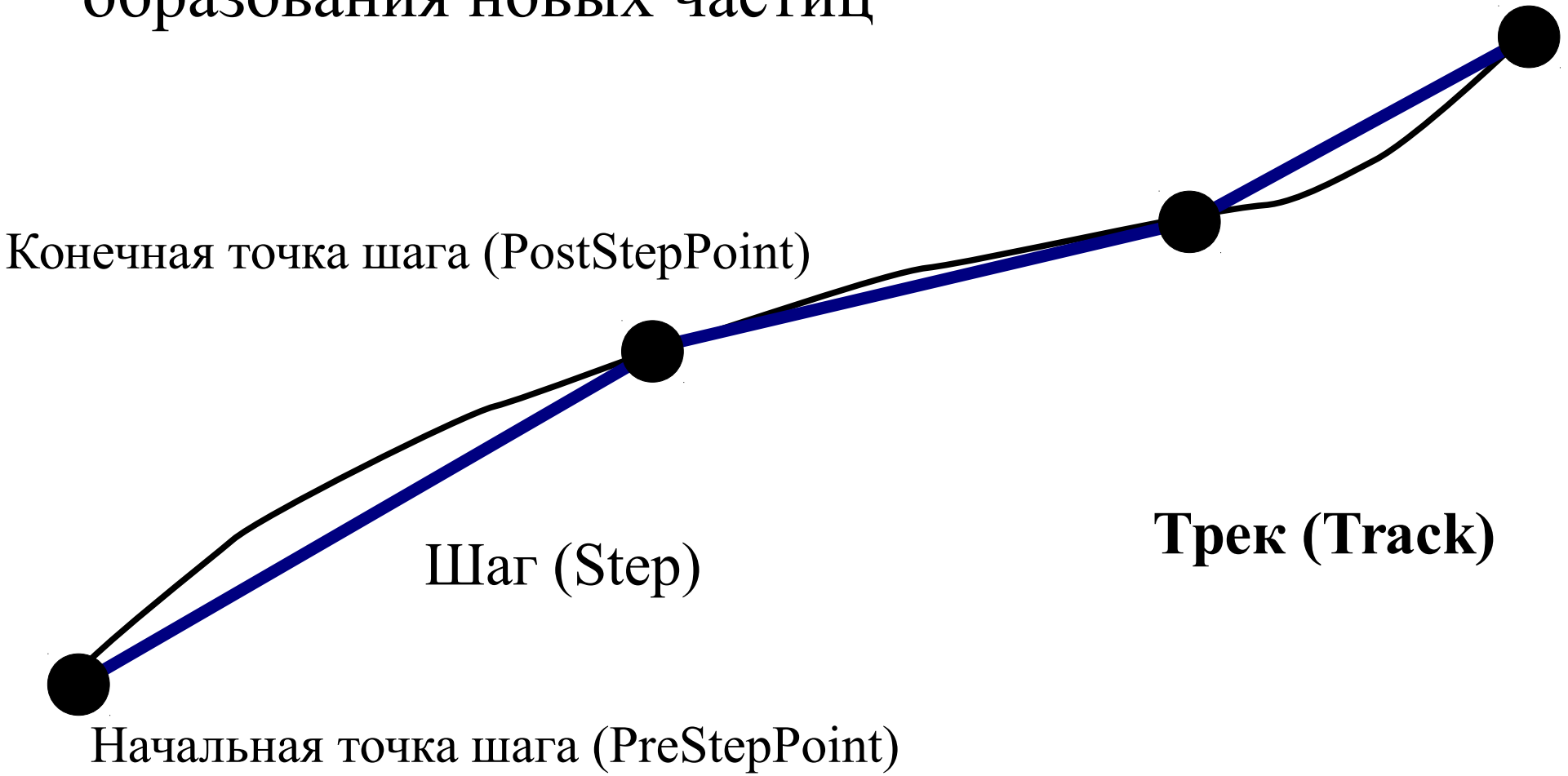


Разыгрывается случайное число R в интервале [0,1]. Модель M1 выбирается, если выполняется условие

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

Транспортировка

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



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.

Причины появления частиц с энергией ниже порога

- Если рождение вторичной частицы с энергией ниже порога позволяет произвести срабатывание в ближайшем чувствительном объеме

- Рождение пар гамма-квантом – позитрон рождается даже с энергией ниже порога (с последующей аннигиляцией)

Описание отдельного процесса

Класс с описанием процесса должен быть наследником класса **G4VProcess**

Для каждого процесса необходимо описать

3 метода DoIt()

- собственно моделирование взаимодействия

3 метода GPIL(GetPhysicalInteractionLength)

- расчет длины следующего шага на основе сечения процесса

virtual G4bool IsApplicable(const G4ParticleDefinition&)

- возвращает true, если процесс применим к данной частице

virtual void PreparePhysicsTable(const G4ParticleDefinition&)

virtual void BuildPhysicsTable(const G4ParticleDefinition&)

virtual void StartTracking()

virtual void EndTracking()

DoIt()

- **AlongStepDoIt()**

  вызывается на каждом шаге (например, ионизация)

- **PostStepDoIt()**

  вызывается в конце шага, если шаг определялся данным процессом (например, неупругое рассеяние)

- **AtRestDoIt()**

  вызывается при остановке частицы, если остановка была вызвана данным процессом (например, распад)

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

G4ProcessManager

- Объект **G4ProcessManager** создается отдельно для каждой частицы
  - **разный набор процессов для каждой частицы**
  - **разный набор пороговых значений**
- Доступ к этому объекту:

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

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

Наборы физических процессов (PhysicsLists)

- Полное описание физических моделей частиц и процессов содержится в списке физических процессов – объекте-наследнике класса **`G4VUserPhysicsList`**

- При необходимости, можно описать свой набор физических процессов

- Существует набор «стандартных» - заранее описанных в Geant4 – списков физических процессов

Использование собственного списка процессов

```
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "ExN01DetectorConstruction.hh"
#include "ExN01PhysicsList.hh"
#include "ExN01PrimaryGeneratorAction.hh"
int main()
{
  // construct the default run manager
  G4RunManager* runManager = new G4RunManager;
  // set mandatory initialization classes
  runManager->SetUserInitialization(new ExN01DetectorConstruction);
  runManager->SetUserInitialization(new ExN01PhysicsList);
  // set mandatory user action class
  runManager->SetUserAction(new ExN01PrimaryGeneratorAction);
  // initialize G4 kernel
  runManager->initialize();
  // get the pointer to the UI manager and set verbosities
  G4UImanager* UI = G4UImanager::GetUIpointer();
  UI->ApplyCommand("/run/verbose 1");
  UI->ApplyCommand("/event/verbose 1");
  UI->ApplyCommand("/tracking/verbose 1");
  // start a run
  int numberOfEvent = 3;
  runManager->BeamOn(numberOfEvent);
  // job termination
  delete runManager;
  return 0;
}
```

Пользовательский класс описания набора процессов

- Должен быть наследником G4VUserPhysicsList
- Должен содержать описания функций
  - **ConstructParticle()**
  - **ConstructProcess()**
  - **SetCuts()**

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

class ExN01PhysicsList: public G4VUserPhysicsList
{
  public:
    ExN01PhysicsList();
    ~ExN01PhysicsList();

  protected:
    // Construct particle and physics process
    void ConstructParticle();
    void ConstructProcess();
    void SetCuts();

};
```

ConstructParticle()

```
void UserAppPhysicsList::ConstructParticle()
{
  G4Proton::ProtonDefinition();
  G4Geantino::GeantinoDefinition();

     .....
  G4Electron::ElectronDefinition();

}
```

В случае если требуется определить все частицы:

```
void UserAppPhysicsList::ConstructParticle()
{
  // Construct all leptons
  G4LeptonConstructor lConstructor;
   lConstructor.ConstructParticle();

 // Construct all mesons
  G4MesonConstructor mConstructor;
  mConstructor.ConstructParticle();

 // .......
}
```

```
void UserAppPhysicsList::ConstructProcess()
{
  // Define transportation process
  AddTransportation();
  // electromagnetic processes
  ConstructEM();
}
void MyPhysicsList::ConstructEM()
{
   //  Get the process manager for gamma
  G4ParticleDefinition* particle = G4Gamma::GammaDefinition();
  G4ProcessManager* pmanager = particle->GetProcessManager();

  // Construct processes for gamma
  G4PhotoElectricEffect * thePhotoElectricEffect = new G4PhotoElectricEffect();
  G4ComptonScattering * theComptonScattering = new G4ComptonScattering();
  G4GammaConversion* theGammaConversion = new G4GammaConversion();

  // Register processes to gamma's process manager
  pmanager->AddDiscreteProcess(thePhotoElectricEffect);
  pmanager->AddDiscreteProcess(theComptonScattering);
  pmanager->AddDiscreteProcess(theGammaConversion);
}
```

ConstructProcess()

Еще пример: описание распадов

```
#include "G4Decay.hh"
void UserAppPhysicsList::ConstructGeneral()
{
  // Add Decay Process
  G4Decay* theDecayProcess = new G4Decay();
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
    if (theDecayProcess->IsApplicable(*particle)) {
      pmanager ->AddProcess(theDecayProcess);
      // set ordering for PostStepDoIt and AtRestDoIt
      pmanager ->SetProcessOrdering(theDecayProcess, idxPostStep);
      pmanager ->SetProcessOrdering(theDecayProcess, idxAtRest);
    }
  }
}
```

SetCuts()

```
void UserAppPhysicsList::SetCuts()
{
  // set cut values for gamma at first and for
  // e- second and next for e+, because some processes
  // for e+/e- need cut values for gamma
  SetCutValue(cutForGamma, "gamma");
  SetCutValue(cutForElectron, "e-");
  SetCutValue(cutForElectron, "e+");
}
```

Пороговые значения по умолчанию

Можно установить одинаковые пороговые значения для всех частиц одновременно

```
void ExN04PhysicsList::SetCuts()
{
  //   the G4VUserPhysicsList::SetCutsWithDefault() method sets
  //   the default cut value for all particle types
  SetCutsWithDefault();
}
```

При этом пороговое значение равно значению переменной–члена класса G4VUserPhysicsList defaultCutValue = 1.0*mm;

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

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

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)

Пример использования стандартного списка процессов

```
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "ExN01DetectorConstruction.hh"
#include "QBBC.hh"
#include "ExN01PrimaryGeneratorAction.hh"
int main()
{
  // construct the default run manager
  G4RunManager* runManager = new G4RunManager;
  // set mandatory initialization classes
  runManager->SetUserInitialization(new ExN01DetectorConstruction);
  G4VModularPhysicsList* plist = new QBBC;
  runManager->SetUserInitialization(plist);
  // set mandatory user action class
  runManager->SetUserAction(new ExN01PrimaryGeneratorAction);
  // initialize G4 kernel
  runManager->initialize();
  // get the pointer to the UI manager and set verbosities
  G4UImanager* UI = G4UImanager::GetUIpointer();
  // start a run
  int numberOfEvent = 3;
  runManager->BeamOn(numberOfEvent);
  // job termination
  delete runManager;
  return 0;
}
```

Конструкторы процессов (Builders)

```
template<class T> TQGSP_BERT<T>::TQGSP_BERT(G4int ver):  T(){
....
  // EM Physics
  this->RegisterPhysics( new G4EmStandardPhysics(ver) );

  // Synchroton Radiation & GN Physics
  this->RegisterPhysics( new G4EmExtraPhysics(ver) );

  // Decays
  this->RegisterPhysics( new G4DecayPhysics(ver) );

   // Hadron Elastic scattering
  this->RegisterPhysics( new G4HadronElasticPhysics(ver) );

  // Hadron Physics
  this->RegisterPhysics( new G4HadronPhysicsQGSP_BERT(ver));
.....}
```

Имеющиеся конструкторы

Наследники класса G4VPhysicsConstructor

G4IonBinaryCascadePhysics
G4IonPhysics
G4IonQMDPhysics
G4IonINCLXXPhysics
G4HadronDElasticPhysics
G4HadronHElasticPhysics
G4ChargeExchangePhysics
G4HadronElasticPhysics
G4IonElasticPhysics
G4OpticalPhysics
G4EmLowEPPhysics
G4EmPenelopePhysics
G4EmLivermorePolarizedPhysics
G4EmLEPTSPhysics

G4EmLivermorePhysics
G4EmDNAPhysics
G4EmStandardPhysics
G4FastSimulationPhysics
G4ParallelWorldPhysics
G4GenericBiasingPhysics
G4StepLimiterPhysics
G4StoppingPhysics
G4EmExtraPhysics
G4SpinDecayPhysics
G4DecayPhysics
G4MuonicAtomDecayPhysics
G4UnknownDecayPhysics
G4RadioactiveDecayPhysics

+ конструкторы адронных процессов

Дополнение стандартного набора процессов

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

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*

```
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
```