Описание элементарных частиц

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

Что такое geantino?

Нейтральная частица, не имеющая физического аналог, не участвующая в физических взаимодействиях, и используемая для отладки транспортировки через объемы детектора.

Существует также заряженное geantino, которое кроме транспортирования через объемы может взаимодействовать с электромагнитным полем.

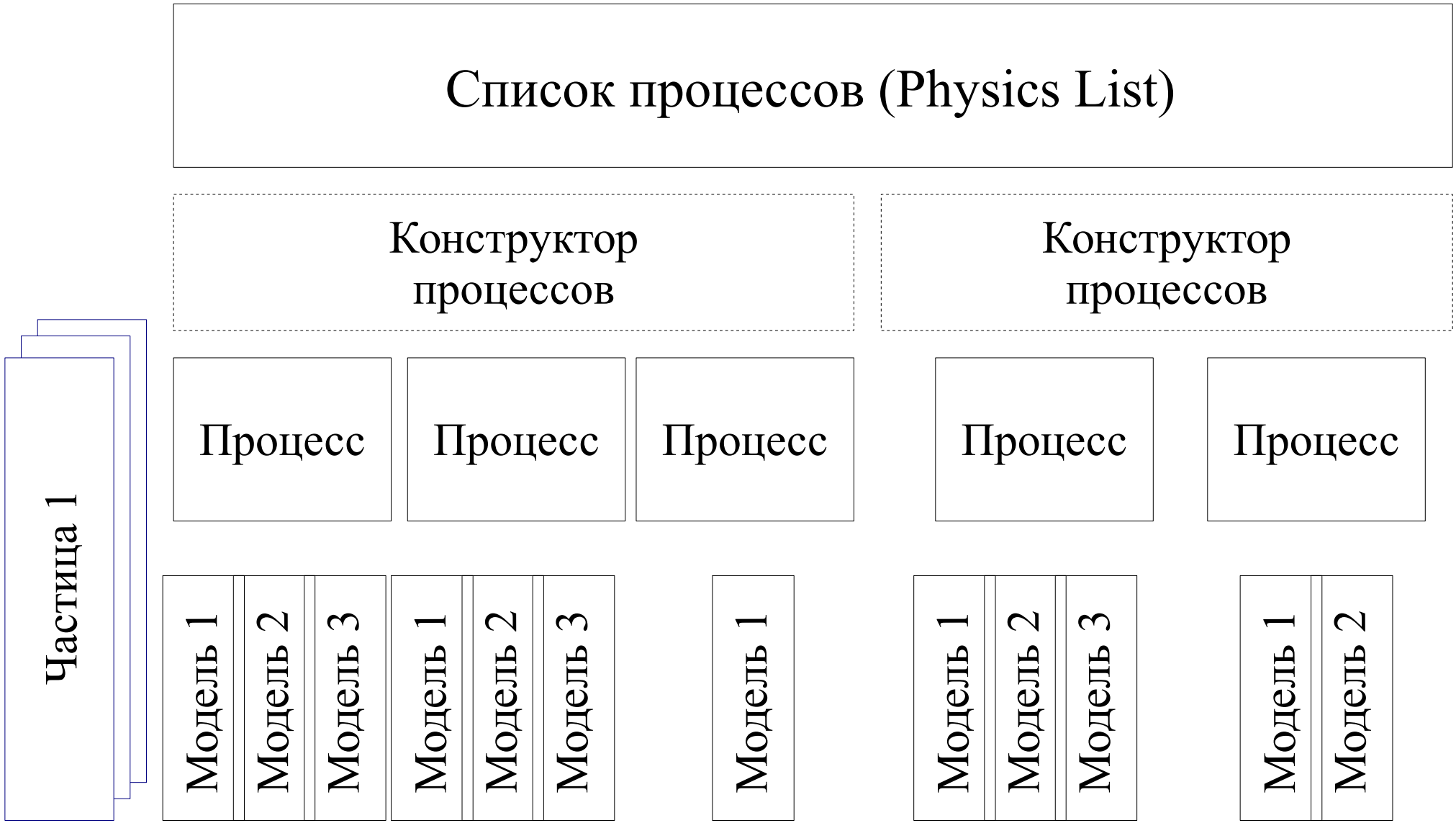
Моделирование частиц и физических процессов в веществе

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

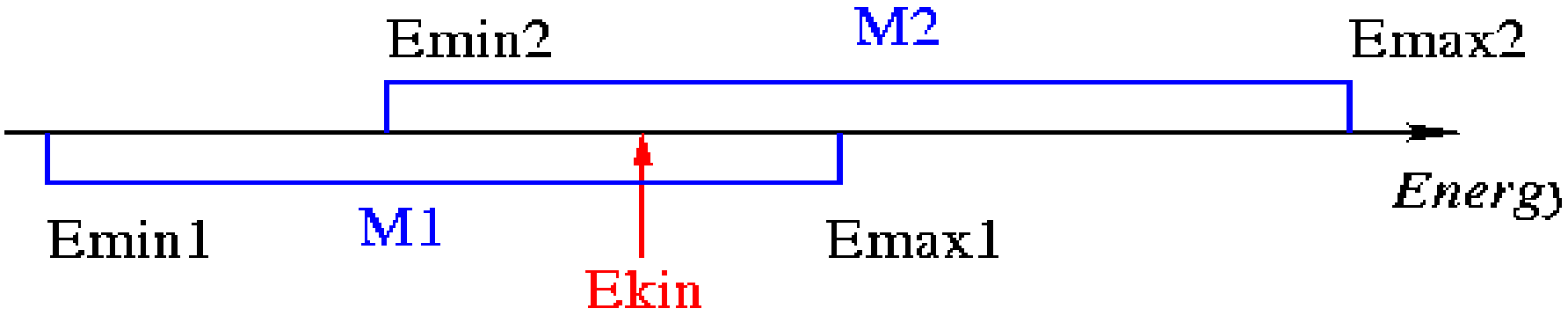
При перекрытии моделей используется следующий алгоритм:

• 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

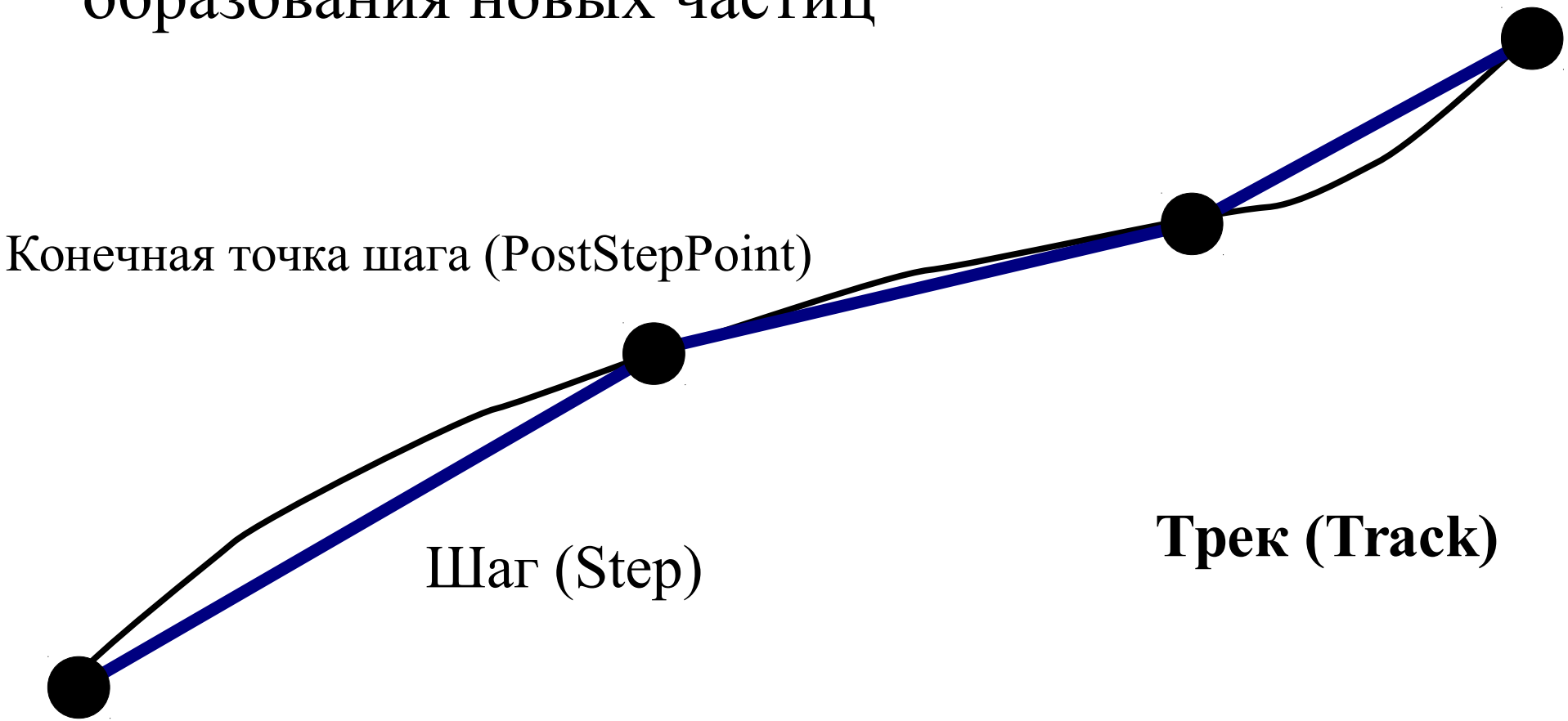


Разыгрывается случайное число R в интервале [0,1]. Модель M1 выбирается, если выполняется условие

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

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

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)

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 HadronPhysicsQGSP_BERT(ver));
.....}
```

Имеющиеся конструкторы

Наследники класса G4VPhysicsConstructor

G4ChargeExchangePhysics
G4DecayPhysics
G4EmDNAPhysics
G4EmExtraPhysics
G4EmLivermorePhysics
G4EmLivermorePolarizedPhysics
G4EmPenelopePhysics
G4EmStandardPhysics
G4EmStandardPhysics_option1
G4EmStandardPhysics_option2
G4EmStandardPhysics_option3
G4HadronDElasticPhysics
G4HadronElasticPhysics93
G4HadronElasticPhysics
G4HadronElasticPhysicsHP
G4HadronElasticPhysicsLHEP
G4HadronElasticPhysicsXS
G4HadronHElasticPhysics
G4HadronQElasticPhysics
G4IonBinaryCascadePhysics
G4IonInclAblaPhysics
G4IonPhysics
G4IonQMDPhysics
G4LHEPStoppingPhysics
G4OpticalPhysics
G4OpticalPhysicsMessenger
G4QAtomicPhysics
G4QCaptureAtRestPhysics
G4QElasticPhysics
G4QEmExtraPhysics
G4QIonPhysics
G4QNeutrinoPhysics
G4QPhotoNuclearPhysics
G4QStoppingPhysics
G4RadioactiveDecayPhysics

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
```

Электромагнитные процессы

- Основные электромагнитные процессы описаны в двух библиотеках: **Standard** и **LowEnergy**
- **Standard** — описание ЭМ процессов для решения задач ФВЭ (применимы до 100 ТэВ)
- **LowEnergy** — более точное описание процессов при низких энергиях (учет атомных эффектов и т. п.), применяется для решения прикладных задач (микродозиметрия, медицинские приложения и т. д.)
- Для обеих библиотек разработан единый интрефейс: все процессы являются наследниками одного из классов:

  G4VEnergyLoss
  G4VEmProcess
  G4VMultipleScattering

Кроме того, существуют дополнительные библиотеки:

- Polarisation
  - Процессы с поляризованными частицами
- Xray
  - Синхротронное излучение (G4SynchrotronRadiation)
  - Переходное излучение (G4TransitionRadiation)
  - Оптические процессы
- Фотон-ядерные и электрон-ядерные взаимодействия

«Стандартный» набор электромагнитных процессов

- **Фотонные процессы**
  - Комптон-эффект (G4ComptonScattering)
  - рождение пар (G4GammaConversion)
  - фотоэффект (G4PhotoElectricEffect)
  - рождение мюонных пар (G4GammaConversionToMuons)
- **Электрон-позитронные процессы**
  - ионизация и рождение дельта-электронов (G4eIonisation)
  - тормозное излучение (G4eBremsstrahlung)
  - аннигиляция позитрона (G4eplusAnnihilation)
  - аннигиляция позитрона в 2 мюона (G4AnnihiToMuPair)
  - аннигиляция позитрона в адроны (G4eeToHadrons)
- **Мюонные процессы**
  - ионизация и рождение дельта-электронов (G4MuIonisation)
  - тормозное излучение (G4MuBremsstrahlung)
  - рождение пар (G4MuPairProduction)
- **Адронные и ионные ЭМ процессы**
  - ионизация(G4hIonisation)
  - ионизация ядрами (G4ionEnergyLoss)
- **Многократное рассеяние**
  - для всех заряженных частиц (G4MultipleScattering)

Набор электромагнитных процессов для низких энергий

Фотонные и электрон-позитронные процессы (250 эВ — 1 ГэВ)

- Модель PENELOPE — теоретическое описание, основанное на экспериментальных данных
- Модель Livermore — параметризация экспериментальных данных
- Рассеяние Рэлея для фотонов (G4RayleighScattering)

Ионизация

- Тормозная способность задана по параметризации ICRU49 (по умолчанию), или по модели Ziegler
  Более подробно описано на http://www.ge.infn.it/geant4/lowE
- Ионизация тяжелыми частицами: G4BraggModel, G4BraggNoDeltaModel, G4BetheBlochNoDeltaModel)
- Ионизация в тонком слое (G4PAIModel)

Микродозиметрия (7 эВ — 10 МэВ) — проект Geant4-DNA

- Расчет эффектов в воде, сравнимых с энергией связи в моделкуле ДНК *(проект развивается)*

Подробнее: S.Chauvie, IEEE Trans. Nucl. Sci. 54 (2007) 2619

Geant4-MicroElec - моделирование электронов от 5 эВ в кремнии

Многократное рассеяние

- Вместо моделирования отдельного акта рассеяния, вычисляется среднее смещение частицы, соответствующее длине шага
- По умолчанию, применяется модель Урбана(L.Urban, CERN-OPEN-2006-077)
  - *Параметризация данных по рассеянию электрона, отдельно для центральной области углового распределения и отдельно для «хвостов»*
  - *Баланс «точность-быстродействие» оптимален для ФВЭ, но точность часто недостаточна при моделировании ЭМ-калориметров*
- Существуют две альтернативные теоретические модели
  - Модель Goudsmit-Saunderson *(O.Kadri et al., NIM B267 (2009) 3624)*

    наиболее точна для расчета рассеяния электронов. Примерно вдвое медленнее чем модель Урбана
  - Модель WentzelVI

    пригодна для расчета многократного рассеяния в разреженных средах

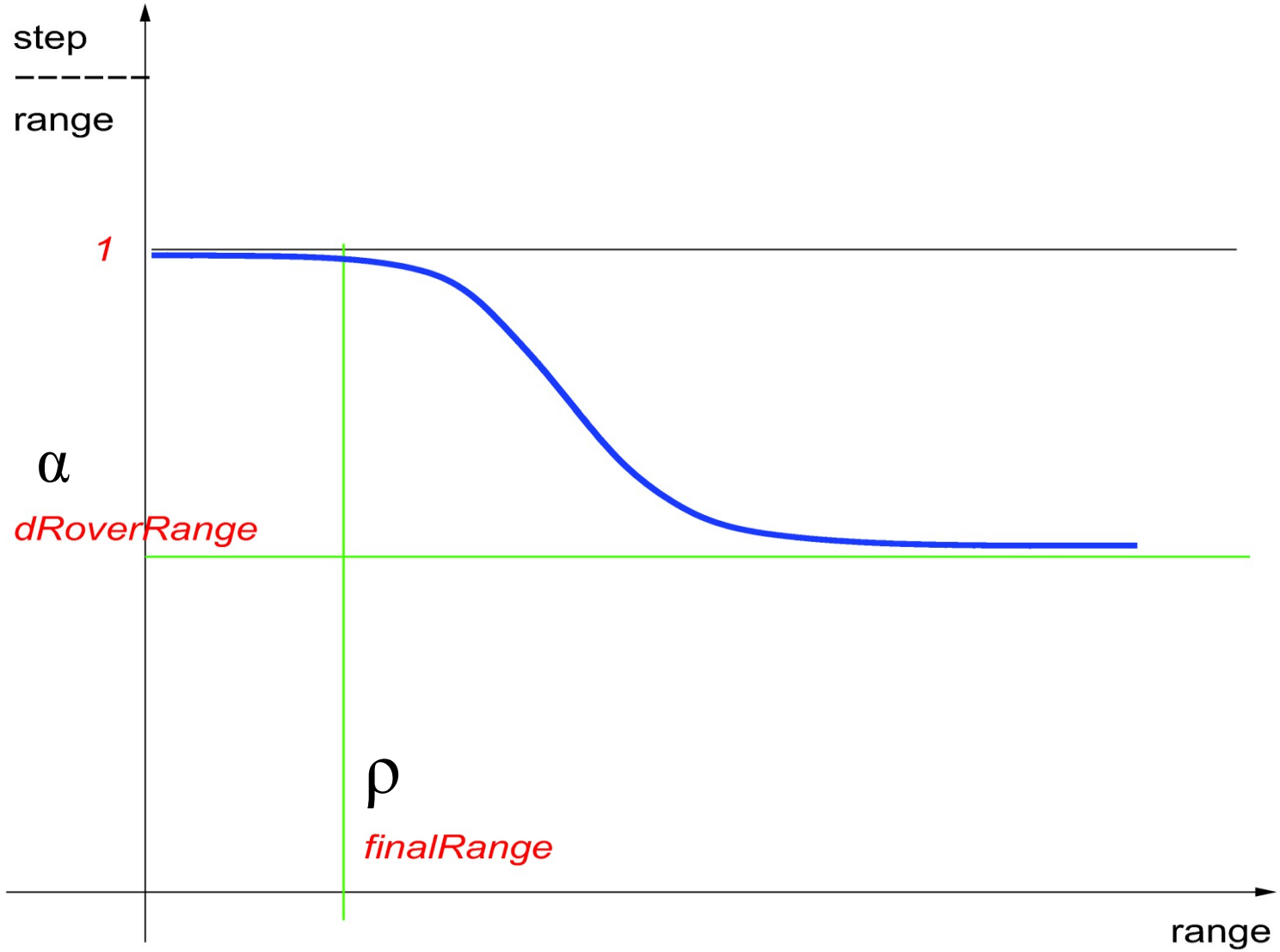
Ионизация

- Моделирование ионизации в Geant4 имеет непрерывную (dE/dx) и дискретную (δ-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 ρ — минимальная длина шага (1 мм по умолчанию)

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


step
range
1
α
dRoverRange
ρ
finalRange
range

Коррекция функции шага при создании набора процессов:

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

или в командной строке

```
/process/eLoss/StepFunction 0.2 100 um
```

Оптические процессы

- Фотон моделируется как оптический, если его длина волны много больше размера атома.
- В Geant4 оптический фотон и гамма-квант – две разные частицы.
- Оптические фотоны в Geant4 моделируются без учета сохранения энергии
- Оптические свойства вещества описываются в объекте G4MaterialPropertiesTable, который передается в G4Material

Оптические процессы

В Geant4 описаны процессы

- излучение Вавилова-Черенкова (G4Cerenkov)
- сцинтилляция (G4Scintillation)
- смещение длины волны (G4OpWLS)
- поглощение фотонов (G4OpAbsorption)
- рассеяние Рэлея (G4OpRayleigh)
- граничные эффекты (G4OpBoundaryProcess)

Конструкторы ЭМ процессов (основные для ФВЭ)

- **G4EmStandardPhysics** *(применяется по умолчанию)*

  Разработан для эксперимента ATLAS на LHC

- **G4EmStandardPhysics_option1** *(наборы с суффиксом _EMV)*

  Разработан для эксперимента CMS на LHC

  Более быстрые расчеты, из-за более простой оценки длины шага

  Не очень хорош для моделирования ЭМ самплинг-калориметров

- **G4EmStandardPhysics_option2** *(экспериментальный, применяется в QBBC)*

  Модель многократного рассеяния WentzelVI

Конструкторы ЭМ процессов (комбинированные ФВЭ+низкие энергии)

- **G4EmStandardPhysics_option3** *(наборы QGSP_BIC_EMY, Shielding)*

Многократное рассеяние: применяется модифицированная модель Урбана

Скорректированы параметры функции шага для ионизации (0.2, 100 мкм)

- **G4EmLivermorePhysics**

Многократное рассеяние: модель Godsmit-Saunderson

Для γ, e± ниже 1 GeV — модели Livermore, выше - Standard

- **G4EmPenelopePhysics**

Многократное рассеяние: модель Godsmit-Saunderson

Для γ, e± ниже 1 GeV — модели PENELOPE, выше - Standard

Конструкторы ЭМ процессов (дополнительные)

- **G4EmLivermorePolarizedPhysics**
  Процессы с поляризованными фотонами
- **G4EmExtraPhysics**
  Синхротронное излучение
  Фото- и электроядерные процессы
- **G4OpticalPhysics**
  оптические процессы
- **G4EmDNAPhysics**
  микродозиметрия

Взаимодействия адронов

Адронные процессы в Geant4

- Неупругое рассеяние (G4HadronInelasticProcess)
- Упругое рассеяние (G4HadronElasticProcess)
- Деление (G4HadronFissionProcess)
- Захват (G4HadronCaptureProcess)

Вероятность взаимодействия, и вероятность образования конкретного конечного состояния рассчитываются независимо

Взаимодействие нейтронов низких энергий

- Пределы применимости: 0.025 эВ - 20 МэВ
  - Упругое рассеяние (G4NeutronHPElasticData)
  - Захват (G4NeutronHPCaptureData)
  - Деление (G4NeutronHPFissionData)
  - Неупругое рассеяние(G4NeutronHPInelasticData)
- Используются таблицы экспериментально измеренных сечений G4NDL на основе таблиц оцененных нейтронных данных ENDF/B-VII (417 изотопов)
- Конструкторы: HadronPhysicsQGSP_BERT_HP, HadronPhysicsQGSP_BIC_HP, G4HadronElasticPhysicsHP
- Наборы: QGSP_BIC_HP, QGSP_BERT_HP

Достоверность

- Надежный расчет транспорт нейтронов (рассеяние, захват)
- Моделирование реакций (n, n'), (n, 2n), (n,3n) обычно достоверно, но в случае рождения нескольких нейтронов энергия и углы вылета могут не коррелировать
- Моделирование реакций с образованием заряженных частиц или гамма-квантов может быть неточным
- В целом, в многочастичных реакциях закон сохранения энергии-импульса может не выполняться.

Рассеяние тепловых нейтронов

Для корректного учета рассеяния тепловых нейтронов нужно либо создавать материалы на основе библиотеки NIST, либо использовать **названия** из следующего списка:

TS_Aluminium_Metal
TS_Beryllium_Metal
TS_Be_of_Beryllium_Oxide
TS_C_of_Graphite
TS_D_of_Heavy_Water
TS_H_of_Water
TS_H_of_Zirconium_Hydride
TS_H_of_Polyethylene
TS_Iron_Metal
TS_O_of_Uranium_Dioxide
TS_O_of_Beryllium_Oxide
TS_U_of_Uranium_Dioxide
TS_Zr_of_Zirconium_Hydride
TS_H_of_Para_Hydrogen
TS_H_of_Ortho_Hydrogen
TS_D_of_Para_Deuterium
TS_D_of_Ortho_Deuterium
TS_H_of_Liquid_Methane
TS_H_of_Solid_Methane
TS_Benzene

Альтернативные таблицы сечений

https://www-nds.iaea.org/geant4/

- BROND-2.2
- CENDL-31
- ENDF-B/VI.8
- ENDF-B/VII.0
- JEFF-3.0
- JEFF-3.1
- JENDL-3.3
- JENDL-4.0

Применение:
export G4NEUTRONHPDATA=/path/to/data/files/CENDL-31

Полезные переменные окружения

G4NEUTRONHP_SKIP_MISSING_ISOTOPES

- если объявлена, то в случае отсутствия сечений для данного изотопа в таблице используются сечения для природной смеси, и 0 если их тоже нет
- если не объявлена, используется ближайший по Z и A элемент с известными сечениями

G4NEUTRONHP_DO_NOT_ADJUST_FINAL_STATE

- если не объявлена, для выполнения закона сохранения энергии-импульса могут рождаться «лишние» гамма-кванты

G4NEUTRONHP_PRODUCE_FISSION_FRAGMENTS

- моделируется образование осколков деления ядра-мишени

LEND

- Альтернатива NeutronHP в Geant4
- Представляет собой интерфейс к GIDI - новому формату ядерных данных, разработанному в LLNL
- Реализация в соответствии с архитектурой адронных процессов в Geant4 (разделение полного сечения и моделирования конечного состояния)
- Содержит данные для нескольких температур:

  300, 1160 и 3590 K
- Файлы с оцененными данными распространяются независимо от Geant4

Трекинг

Как происходит один шаг в моделировании (1)

1. Считается скорость частицы в начале шага
2. Определяется длина свободного пробега независимо для каждого активного процесса. Выбирается наименьшая.
3. Рассчитывается расстояние до граничной поверхности ближайшего объема. Если это расстояние больше, чем наименьшее определенное для физических процессов, выбирается последнее, и дальнейшие геометрические расчеты не делаются

Как происходит один шаг в моделировании (2)

4. Производится расчет непрерывных взаимодействий согласно длине шага, и соответственно в конце изменяется значение кинетической энергии
5. Производится проверка, стоит ли продолжать трекинг данной частицы
6. Обновляются параметры частицы (энергия и положение)
7. Производится расчет дискретных процессов. Если требуется, создается список вторичных частиц.

Как происходит один шаг в моделировании (3)

8. Производится проверка, стоит ли продолжать трекинг данной частицы
9. Рассчитывается расстояние до граничной поверхности ближайшего объема.
10. Если шаг ограничен этой поверхностью, происходит переход в следующий объем
11. Производятся действия, определяемые пользователем (G4UserSteppingAction) и обрабатывается информация в детектирующих элементах
12. Добавляется точка в траектории движения

Пределы, устанавливаемые пользователем (UserLimits)

- Применяются в целях оптимизации
- Могут устанавливаться для отдельных частиц и отдельных логических объемов
- Позволяют устанавливать
  - **максимально допустимую длину шага**
  - **максимальную длину трека**
  - **максимальное время жизни**
  - **минимальную кинетическую энергию**
  - **минимальное расстояние пролета**

Примеры UserLimits (1)

Принудительное ограничение длины шага:

В **UserAppDetectorConstruction:**

```
G4double maxStep = 0.1*cm;
logicTracker->SetUserLimits(new G4UserLimits(maxStep));
```

В **UserAppPhysicsList**

```
pmanager->AddDiscreteProcess(new G4StepLimiter);
```

Примеры UserLimits (2)

Более общая конструкция:

В **UserAppDetectorConstruction**

```
G4double maxTime = 10*ms;
logicTracker->SetUserLimits(new
                    G4UserLimits(DBL_MAX,DBL_MAX,maxTime));
```

В **UserAppPhysicsList**

```
G4ProcessManager* pmanager =
                              G4Neutron::Neutron->GetProcessManager();
pmanager->AddProcess(new G4UserSpecialCuts(),-1,-1,1);
```

Повышение эффективности моделирования

Как повысить эффективность расчетов

$$\sigma \sim 1/\sqrt{N} \qquad T_{CPU} \sim N$$

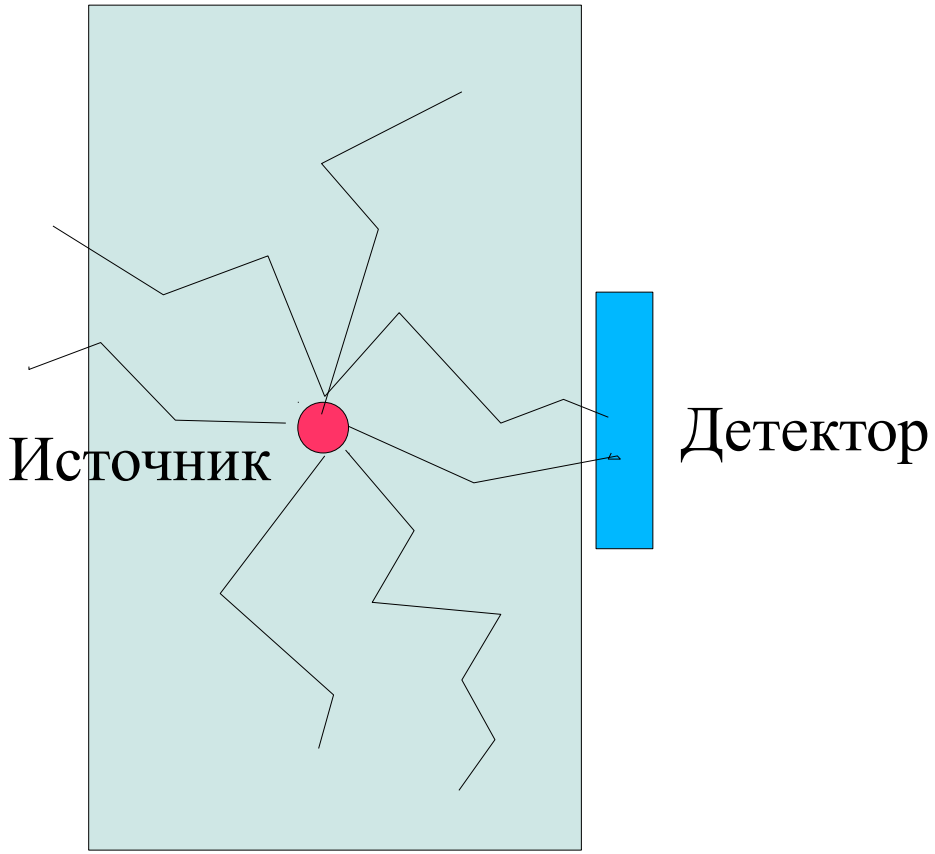
Повышение точности в два раза требует вчетверо больших затрат процессорного времени

Как достичь большей точности при меньших затратах CPU:

- Подробное моделирование более интересных областей установки за счет менее подробного моделирования неинтересных (выборка по значимости или importance sampling)
- Изменение вероятности интересных физических процессов для их более частого моделирования
- У каждого трека появляется дополнительный вес

  G4double w = aTrack->GetWeight();
  aTrack->SetWeight(G4double w);

Выборка по значимости объема



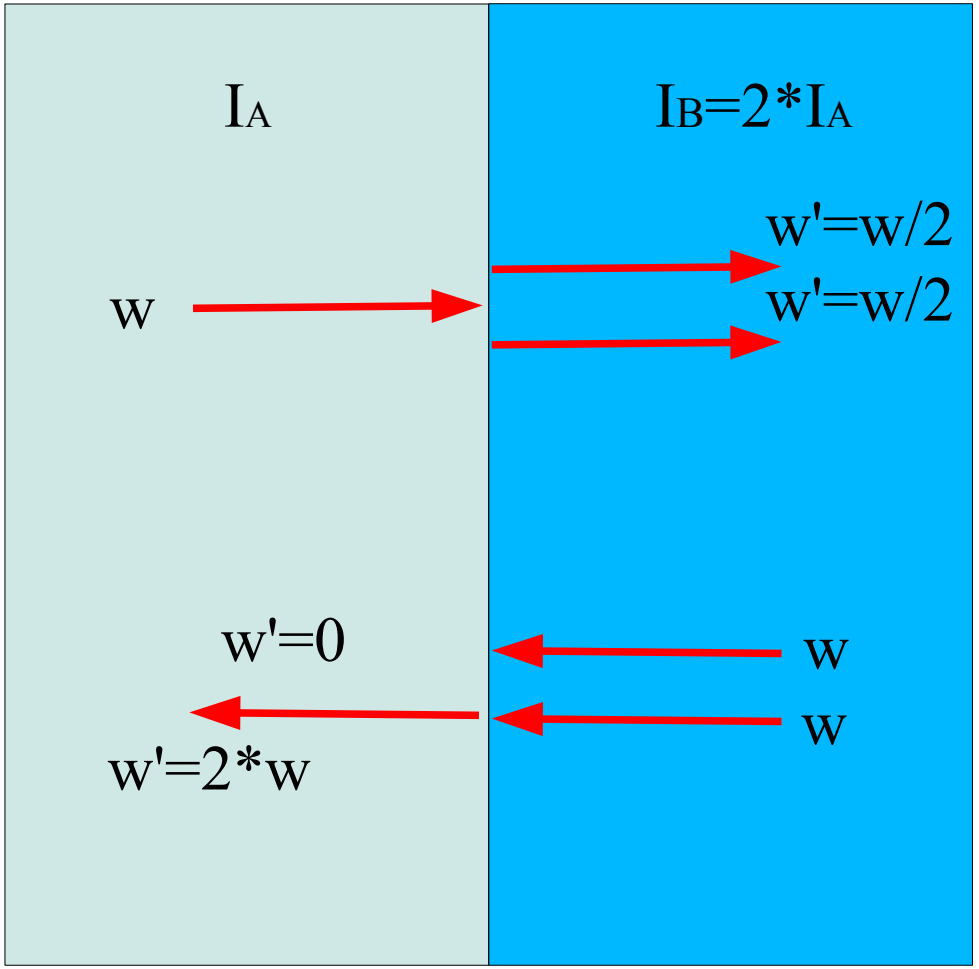
Выборка по значимости объема

- Детектор разбивается на области (cells) и каждой области в зависимости от ее важности для моделирования присваивается значимость
  - *Вариант 1: cell = реальный физический объем*
  - *Вариант 2: cell = объем в параллельной геометрии*

- Более значимые области моделируются более подробно = моделируется больше треков

- При пересечении границ объемов трек либо останавливается, либо дублируется, в зависимости от относительного изменения значимости

Выборка по значимости объема



- $r=I_B/I_A$
- $r=1$ обычный трекинг
- $r<1$ трек останавливается с вероятностью 1-r (Russian roulette)
- $r>1$
  - если r целое, создается r копий трека
  - если r не целое, то создается r+1 копий с вероятностью p=r-floor(r) или r копий с вероятностью 1-p
- Треку присваивается вес, сохраняющий число частиц

Реализация в программе

- DetectorConstruction.hh

```
class DetectorConstruction : public G4VUserDetectorConstruction {
public:
 ...
 G4GeometrySampler* sampler;
```

- DetectorConstruction.cc

```
sampler = new G4GeometrySampler(physWorld,"gamma");
G4VImportanceAlgorithm* ialg = new G4ImportanceAlgorithm();
G4IStore* istore = new G4IStore(*physWorld);
istore->AddImportanceGeometryCell(1,*physWorld);
istore->AddImportanceGeometryCell(1,*physSand);
istore->AddImportanceGeometryCell(2,*physWater);
istore->AddImportanceGeometryCell(4,*physIron);
sampler->PrepareImportanceSampling(istore, ialg);
```

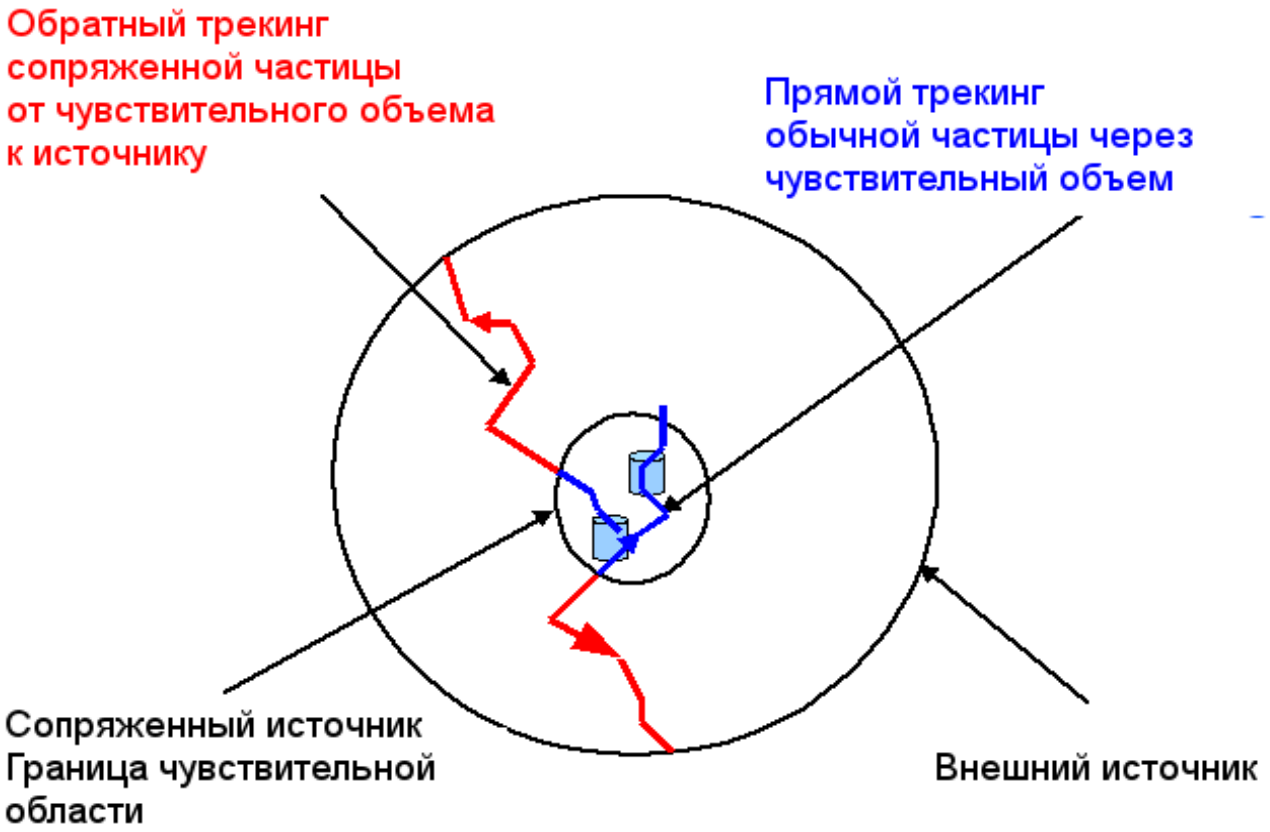
- main.cc

```
runManager->Initialize();
detectorConstruction->sampler->Configure();
```

Весовое окно

- Аналогичная техника может применяться не только для объемов, но и для «ячеек» в фазовом пространстве
- Применяется на границе объемов, при столкновении и при столкновении на границе
- Пользователь определяет
  - *нижнюю границу весового окна*
  - *отношение максимального и минимального весов*
  - *отношение веса выживания и минимального веса*
- Веса определяются для различных физических объемов и диапазонов энергии
- Наиболее эффективно в сочетании с выборкой по значимости (например, чтобы убрать треки со слишком маленьким или слишком большим весом)

Обратное моделирование Adjoint/Reverse MC



- Реализовано только для некоторых ЭМ процессов (ионизация, многократное рассеяние, комптоновское рассеяние, тормозное излучение, фотоэффект)
- Пример: *examples/extended/biasing/ReverseMC01*

Смещенная выборка для физических процессов

- Вместо функции плотности вероятности p(x) используется функция p'(x). Каждому треку присваивается вес w=p(x)/p'(x)
- Таким образом можно изменять на более выгодные угловые распределения, энергетический спектр вторичных частиц, сечение взаимодействия и т.д.
- В Geant4 такая возможность реализована через «псевдопроцесс» G4WrapperProcess

G4WrapperProcess

```
class G4WrapperProcess : public G4VProcess {
G4VProcess* pRegProcess;
…
inline void G4WrapperProcess::RegisterProcess(G4VProcess* process)
{
      pRegProcess=process;
      …
}

…

inline G4VParticleChange*
G4WrapperProcess::PostStepDoIt(const G4Track& track, const G4Step& stepData)
{
      return pRegProcess->PostStepDoIt(track, stepData);
}
```

аналогично для остальных методов DoIt

Пример реализации

```
class MyWrapperProcess : public G4WrapperProcess {
…
G4VParticleChange* PostStepDoIt(const G4Track& track, const G4Step& step)
 {
      // Your own implementation
  }
}


void MyPhysicsList::ConstructProcess()
{
      …
      G4LowEnergyBremsstrahlung* bremProcess =
                                                  new G4LowEnergyBremsstrahlung();
      MyWrapperProcess* wrapper = new MyWrapperProcess();
      wrapper->RegisterProcess(bremProcess);
      processManager->AddProcess(wrapper, -1, -1, 3);
}
```

Изменение сечения адронных процессов

```
void MyPhysicsList::ConstructProcess()
 {
   ...
   G4ElectroNuclearReaction * theElectroReaction =
     new G4ElectroNuclearReaction;

   G4ElectronNuclearProcess theElectronNuclearProcess;
   theElectronNuclearProcess.RegisterMe(theElectroReaction);
   theElectronNuclearProcess.BiasCrossSectionByFactor(100);

   pManager->AddDiscreteProcess(&theElectronNuclearProcess);
   ...
 }
```

Метод работает для любого наследника G4HadronicProcess

Радиоактивный распад

- В классе G4RadioactiveDecay реализованы следующие способы смещенной выборки:
  - Изменение вероятности распада (sampling rate biasing)
  - Дублирование распадающихся нуклидов (nuclear splitting)
  - Изменение вероятностей мод распада (branching ratio biasing)

- Пример использования:
  examples/extended/radioactivedecay/exrdm