



Генерация первичного излучения

ЛЕКЦИЯ 5



Содержание

- ❑ **G4VUserActionInitialization**
- ❑ События и генерация первичной частицы
- ❑ **G4VUserPrimaryGeneratorAction**
- ❑ G4ParticleGun
- ❑ Пример построения простейшего генератора первичных частиц
- ❑ Использование спектра при построении генератора первичных частиц



G4VUserActionInitialization

Данный абстрактный базовый класс используется для инициализации всех классов действий, одним из которых является класс, отвечающий за запуск первичных частиц - **G4VUserPrimaryGeneratorAction**.

Класс **G4VUserActionInitialization** содержит один чисто виртуальный метод **G4VUserActionInitialization::Build()**. Следовательно простейший класс, наследующий данный, будет выглядеть следующим образом:

```
10 class Action: public G4VUserActionInitialization{  
11     public:  
12         virtual void Build() const;  
13  
14     };
```



G4VUserActionInitialization::Build()

Метод **Build()** отвечает за инициализацию всех классов действий, и может быть реализован следующим образом:

```
19 void Action::Build() const {  
20     SetUserAction(new PrimaryPat);  
21 }
```

Примечание: В данном случае PrimaryPat – экземпляр потомка G4VUserPrimaryGeneratorAction

Инициализация экземпляров классов действий осуществляется за счет методов **G4VUserActionInitialization::SetUserAction()**

```
void SetUserAction(G4VUserPrimaryGeneratorAction*) const;  
void SetUserAction(G4UserRunAction*) const;  
void SetUserAction(G4UserEventAction*) const;  
void SetUserAction(G4UserStackingAction*) const;  
void SetUserAction(G4UserTrackingAction*) const;  
void SetUserAction(G4UserSteppingAction*) const;
```

Чтобы связать классы действий с ядром необходимо передать указатель на экземпляр данного класса экземпляру G4RunManager

```
runManager->SetUserInitialization(new Action());
```



Событие и генерация первичной частицы

Событие – представляет собой единичный цикл от зарождения первичной частицы, до окончания отслеживания последней вторичной частицы.

Первичная частица, это та частица с которой начинается событие:

- ❑ Т.е. гамма, альфа-частица, электрон, позитрон и т.п.
- ❑ Первичная частица описывается её направлением, id и ,к примеру, поляризацией.

Первичная вершина включает в себя позицию и время. Одна или более первичных частиц могут быть связаны с одной первичной вершиной. Каждое событие может включать одну или более первичных вершин.

G4PrimaryVertex objects
= {position, time}



G4PrimaryParticle objects
= {PDG, momentum,
polarization...}



G4VUserPrimaryGeneratorAction

G4VUserPrimaryGeneratorAction абстрактный базовый класс для управления генерацией первичных частиц. Сам по себе данный класс генерацию НЕ ОСУЩЕСТВЛЯЕТ.

- ❑ Потомок данного класса должен содержать один или более экземпляров потомка абстрактного класса **G4VPrimaryGenerator**.
- ❑ В экземпляре данного класса устанавливаются и изменяются свойства частиц
- ❑ Данный класс содержит один чисто виртуальный метод **G4VUserPrimaryGeneratorAction::GeneratePrimaries()** который вызывается из G4RunManager в течении цикла событий.

Генерация первичного излучения осуществляется классом **G4VPrimaryGenerator**.

У данного класса существует несколько потомков, для упрощения процесса генерации частиц. Один из этих потомков **G4ParticleGun**



G4ParticleGun

G4ParticleGun — потомок класса **G4VPrimaryGenerator**. Он запускает частицы определенного типа в указанном направлении и с заданной кинетической энергией. Данный класс не содержит специальных методов для рандомизации свойств вылетающих частиц, но пользователь может вручную менять свойства на каждом старте события (*или сформировать алгоритм, по которому эти свойства будут меняться*).

Конструкторы данного класса выглядят следующим образом:

❑ Без определения типа первичных частиц

```
G4ParticleGun(G4int numberofparticles)           //в качестве аргумента указывается
                                                    //количество первичных частиц на одно
                                                    //событие
```

❑ С заранее указанным типом частиц

```
G4ParticleGun(G4ParticleDefinition * particleDef, //указатель на тип первичных частиц
               G4int numberofparticles = 1)       //количество первичных частиц
```

Примечание: Если количество первичных частиц задано больше одного, то все они имеют одинаковые свойства



Пример простейшего генератора первичных частиц: Шаг 1

□ Унаследуем класс **G4VUserPrimaryGeneratorAction**

```
11 class PrimaryPat: public G4VUserPrimaryGeneratorAction{
12     private:
13         G4ParticleGun* gun;
14     public:
15         PrimaryPat();
16         ~PrimaryPat();
17         virtual void GeneratePrimaries(G4Event* anEvent);
18     };
```

Пусть все частицы буду иметь одинаковые свойства, тогда имеет смысл задать их в конструкторе а в методе **GeneratePrimaries()** лишь передавать их событию



Пример простейшего генератора первичных частиц: Шаг 2

□ Укажем свойства первичных частиц

```
10 PrimaryPat::PrimaryPat() {  
11     gun = new G4ParticleGun(1);  
12     gun->SetParticleDefinition(G4Gamma::GammaDefinition());  
13     gun->SetParticleEnergy(0.661*MeV);  
14     gun->SetParticlePosition(G4ThreeVector(0,0,-80));  
15     gun->SetParticleMomentumDirection(G4ThreeVector(0,0,1));  
16 }
```

В данном случае это гамма-кванты с энергией 661кэВ вылетающие из точки с координатами $X = 0$, $Y = 0$, $Z = -80$ в направлении оси Z



Пример простейшего генератора первичных частиц: Шаг 3

- В методе **GeneratePrimaries()** будем передавать свойства полученной первичной вершины на каждом цикле событий:

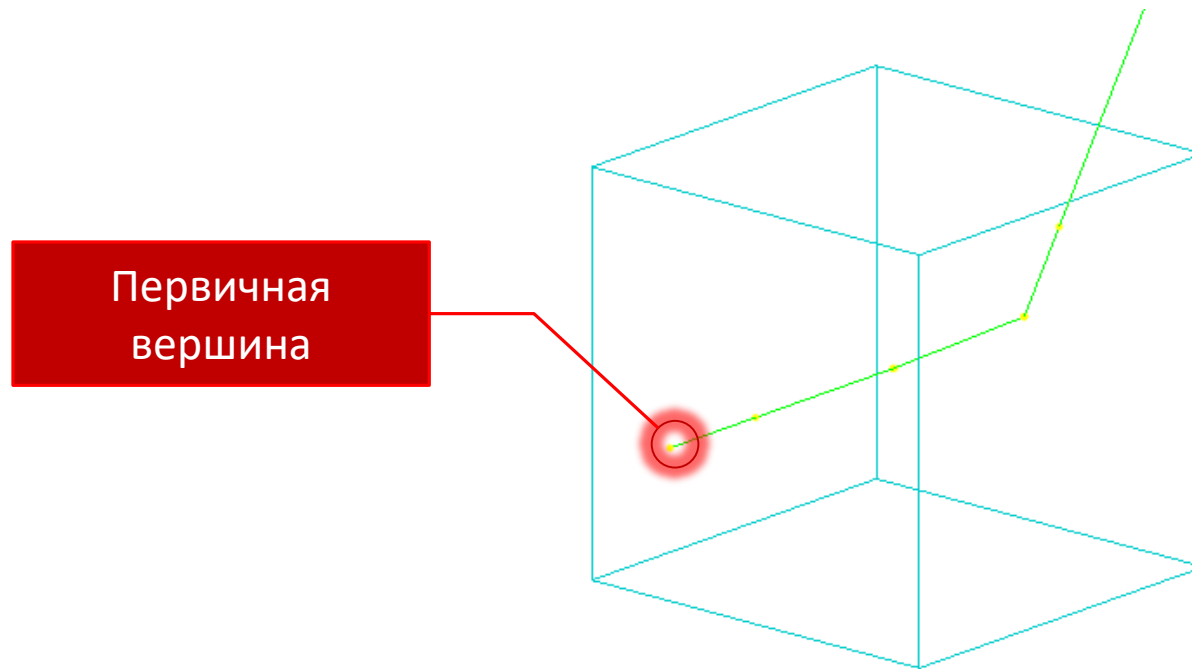
```
49 void PrimaryPat::GeneratePrimaries(G4Event* anEvent){  
50     gun->GeneratePrimaryVertex(anEvent);  
51 }
```



Пример простейшего генератора первичных частиц: Финал

Чтобы запустить первичные частицы в интерактивном режиме(и не только) следует ввести команду `:/run/beamOn <n>`, где `<n>` количество событий.

В данном примере геометрия была задана как: кубик с длиной грани 100 мм из материала BGO

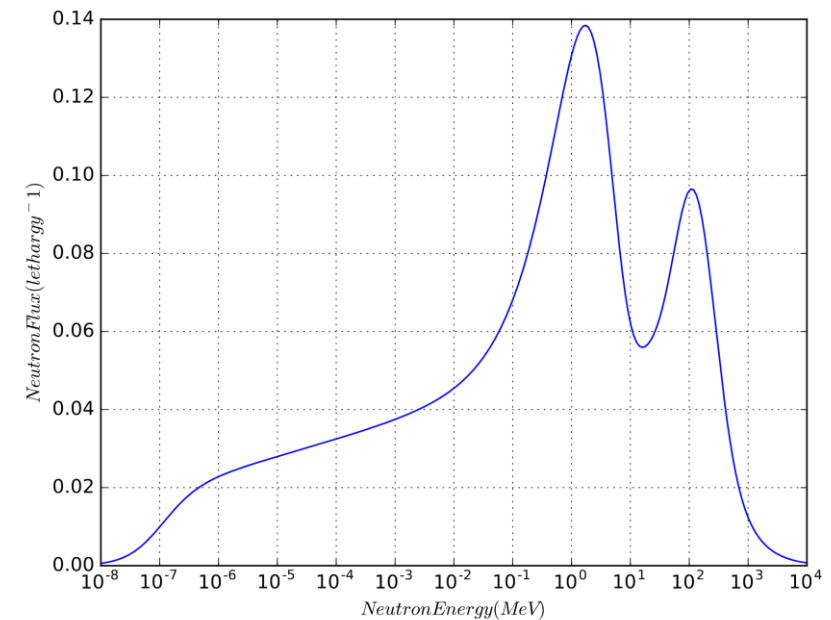




Построение источников первичного излучения с энергией описанной спектром

Достаточно часто возникает проблема по использованию для описания источника первичного излучения некоего спектра. Данную задачу в Geant4 можно решить следующим образом:

- Спектр нейтронов задан некой эмпирической формулой и выглядит следующим образом

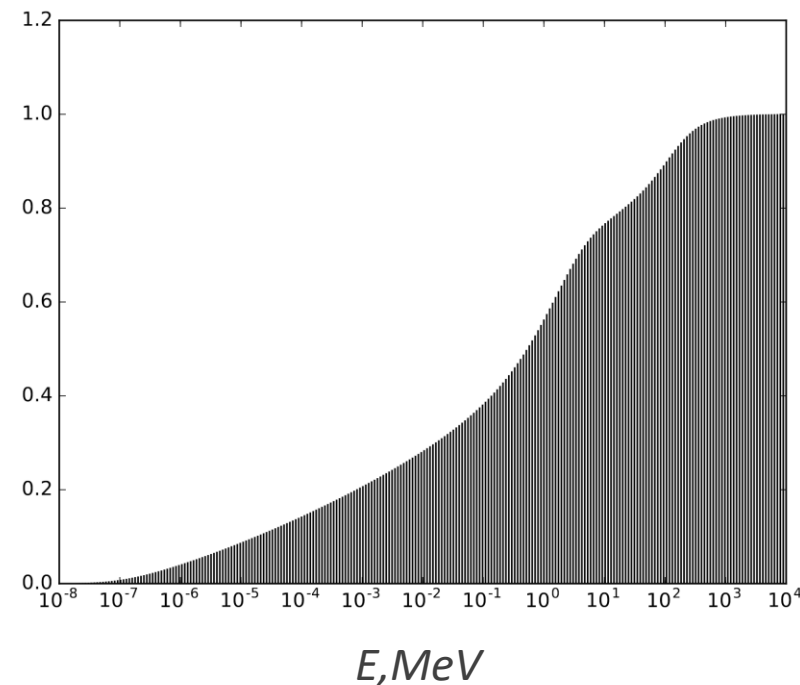
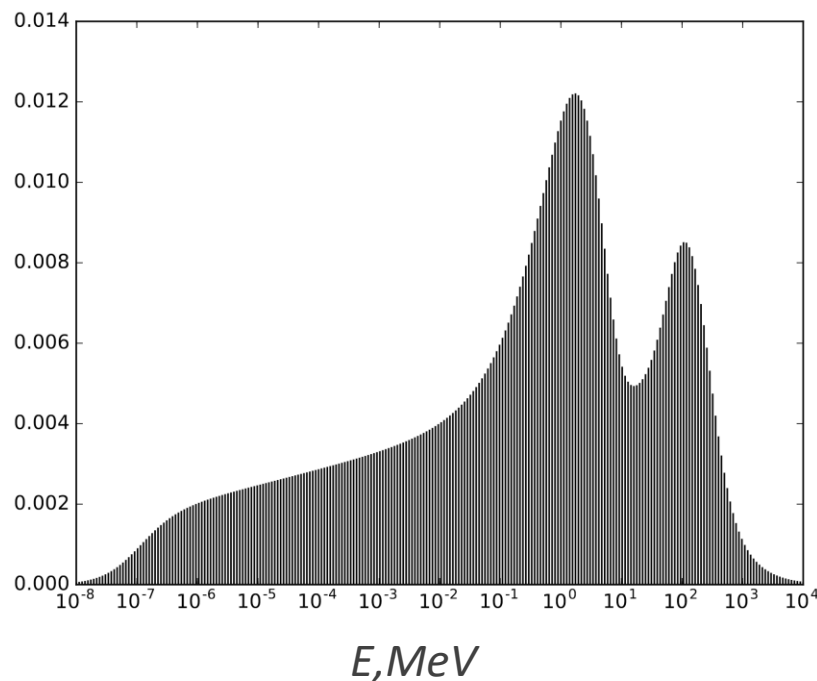




Построение источников первичного излучения с энергией описанной спектром

□ Приведем его к дифференциальному виду

□ Построим распределение случайной величины





Построение источников первичного излучения с энергией описанной спектром

*Примечание: данные процедуры можно провести как на Си++, так и в сторонней программе. К материалу данной лекции не относится получение функции распределения случайной величины поэтому перейдем к этапу когда у нас уже есть некая карта (**std::map**) в которой вероятность играет роль первого поля или ключа (**first**) а соответствующая энергия – значение (**second**).*

- ❑ Разыграем случайное значение от 0 до 1 через **G4UniformRand()**
- ❑ Получим указатель на соответствующий элемент через метод **std::map::lower_bound()** возвращающий итератор на первый элемент, чей ключ больше или равен передаваемому
- ❑ Соответствующее значение энергии передадим в **G4ParticleGun**

```
G4double rand = G4UniformRand();  
std::map<G4double, G4double>::iterator it = ew->lower_bound(rand);  
gun->SetParticleEnergy(it->second * MeV);
```

*Примечание: ew – это std::map<G4double,G4double>**