Создание материалов.

Структура материалов в Geant4 отражает то, что реально существует в природе: материалы состоят из отдельных элементов или их смесей, элементы в свою очередь состоят из отдельных изотопов или их смесей. Более подробно материалы в Geant4 рассмотрим на примере моделирования воды.

Как было сказано выше, создание материала начинается с моделирования изотопов, входящих в состав элементов, образующих целевой материал. За изотопы отвечает класс G4Isotope.

```
G4Isotope(const G4String& name,    //имя
          G4int z,                 //атомный номер
          G4int n,                 //количество нуклонов
          G4double a = 0.,         //молярная масса
          G4int m = 0);            //изомерный уровень
```

Как видно из конструктора класса, обязательными полями являются: имя, атомный номер и количество нуклонов. Молярная масса рассчитается автоматически на этапе конструирования объекта (если не указано значение по умолчанию).

Соответственно, создание изотопа водорода(протий) в простейшем случае будет выглядеть следующим образом:

```
auto isoH = new G4Isotope("1H",1,1);
```

Теперь перейдем к созданию элемента "Водород". Конструктор элемента в Geant4 имеет следующий вид:

```
G4Element(const G4String& name,          //имя
          const G4String& symbol,        //символ
          G4int nbIsotopes);             //количество изотопов
```

Необходимо отметить, что имя может быть любым, а символ должен соответствовать символу элемента в таблице Менделеева, к примеру:

```
auto protium = new G4Isotope("1H", 1, 1);
auto deuterium = new G4Isotope("2H", 1, 2);

auto elH_1 = new G4Element("my_hydrogen_1", "H", 1);
elH_1->AddIsotope(protium, 1.0);

auto elH_2 = new G4Element("my_hydrogen_2", "H", 1);
elH_2->AddIsotope(deuterium, 1.0);
```

Здесь создается 2 элемента, один из которых на 100% состоит из протия, а другой на 100% из дейтерия. В обоих случаях это водород, поэтому symbol у них "H". Кроме того, стоит обратить внимание на метод AddIsotope:

```
void AddIsotope(G4Isotope* isotope,            //указатель на изотоп
                G4double   RelativeAbundance); //массовая доля изотопа
```

Для каждого создаваемого элемента данный метод должен вызываться ровно столько раз, сколько изотопов было указано в конструкторе.

Существует альтернативный конструктор элементов.

При использовании этого конструктора не нужно создавать изотопы напрямую.

```
G4Element(const G4String& name,        //имя
          const G4String& symbol,      //символ
          G4double  Zeff,              //атомный номер
          G4double  Aeff);             //молярная масса
```

Они будут сконструированы автоматически, исходя из природного соотношения, хранимого в базе.

Теперь перейдем к созданию материалов. За материалы отвечает класс G4Material. Конструктор, позволяющий создавать материалы из созданных ранее элементов выглядит следующим образом:

```
G4Material(const G4String& name,                    //имя
           G4double density,                        //плотность
           G4int nComponents,                       //количество компонентов
           G4State state = kStateUndefined,         //состояние
           G4double temp = NTP_Temperature,         //температура
           G4double  pressure = CLHEP::STP_Pressure); //давление
```

Вначале рассмотрим поля *state*, *temp* и *pressure*. Поле *state* отвечает за "состояние" материала. Оно может быть: твердым, жидким, газообразным или "неопределенным".

Данное поле принимает значение в виде энумерации, где

0	*kStateUndefined*
1	*kStateSolid*
2	*kStateLiquid*
3	*kStateGas*

По умолчанию значения температуры(*temp*) и давления(*pressure*) соответствуют нормальным условиям.

Рассмотрим теперь поле *nComponents*. Оно отвечает за количество компонентов в материале, причем под компонентами могут пониматься как отдельные элементы, так и уже готовые материалы.

Теперь можно перейти непосредственно к моделированию воды. Код будет выглядеть следующим образом:

```
auto Water = new G4Material("Water", 1 * g / cm3, 2);
Water->AddElement(elH, 2);
Water->AddElement(elO, 1);
```

Метод AddElement имеет две перегрузки. В данном случае используется

```
void AddElement(G4Element* element, G4int nAtoms);
```

версия,

принимающая целочисленное количество атомов в молекуле.

Кроме того, у данного метода есть иная форма:

```
void AddElement (G4Element* element, G4double fraction);
```

В данном случае в качестве второго аргумента передается массовая доля элемента в конечном материале. Соответственно, при использовании данной перегрузки исходный код по созданию воды примет следующую форму:

```
auto Water2 = new G4Material("Water2", 1 * g / cm3, 2);
Water2->AddElement(elH, 11.19*perCent);
Water2->AddElement(elO, 88.81*perCent);
```

Аналогично элементам для материалов предусмотрен конструктор, который позволяет пропустить стадию создания изотопов и элементов. Это достаточно легкий способ генерации "моно-материалов" (материалов, состоящих из одного элемента).

```
auto mat_O = new G4Material("Oxygen", 8., 16. * g / mole, 1. * g / cm3);
```

```
auto mat_H = new G4Material("Hydro_", 1., 1. * g / mole, 1. * g / cm3);
```

В Geant4 материалы можно создавать не только из элементов, но и из уже созданных ранее других материалов. К примеру, из созданных выше моно-материалов водорода и кислорода можно получить воду следующим образом:

```
auto Water3 = new G4Material("Water3", 1 * g / cm3, 2);
Water3->AddMaterial(mat_H, 11.19 * perCent);
Water3->AddMaterial(mat_O, 88.81 * perCent);
```

Где метод AddMaterial аналогичен перегрузке метода AddElement, принимающей в качестве аргумента массовую долю.

Наконец, для получения необходимого количества компонентов материала можно комбинировать вызовы методов AddMaterial и AddElement:

```
auto Water4 = new G4Material("Water4", 1 * g / cm3, 2);
Water4->AddMaterial(mat_H, 11.19 * perCent);
Water4->AddElement(elO, 88.81 * perCent);
```

Однако стоит отметить, что использование перегрузки метода AddElement с количеством атомов, и любых методов с массовым содержанием, может привести к некорректному результату.

В завершении стоит сказать, что при создании материала из других материалов не учитываются промежуточные свойства его компонентов. В конечном итоге, материал состоит из элементов, а его плотность, температура и т.п. определяются только финальным объектом.

Рассмотрим еще один конструктор класса G4Material

```
G4Material(const G4String& name,                    //имя
           G4double  density,                       //плотность
           const G4Material* baseMaterial,          //базовый материал
           G4State   state    = kStateUndefined,    //состояние
           G4double  temp     = NTP_Temperature,    //температура
           G4double  pressure = CLHEP::STP_Pressure); //давление
```

Данный конструктор позволяет создавать материалы с новыми свойствами (плотность, состояние, температура, давление) из уже существующих.

```
auto new_Water = new G4Material("new_Water", 5 * g / cm3, Water, kStateLiquid, 1000 * kelvin);
```

База материалов NIST.

В Geant4 представлена база материалов, составленная NIST Physical Measurement Laboratory. В данной базе доступно более 3000 изотопов, а все представленные элементы составлены исходя из природного баланса изотопов. Элементы доступны от Водорода до Калифорния (98). Кроме того, в базе доступны различные готовые материалы, такие как:

- составные вещества и смеси (ткань, эквивалентная пластику; морской воздух и т.д.)
- биохимические материалы (жировая ткань, цитозин, тимин и т.д.)
- композитные материалы (например, кевлар)

Для того, чтобы получить доступ к базе NIST, следует воспользоваться статическим методом

```
static G4NistManager* Instance();
```

Данный метод возвращает указатель, с помощью которого можно обращаться к шаблонам уже готовых материалов, например, водород можно получить следующим образом

```
auto nist = G4NistManager::Instance();
auto nist_H_1 = nist->FindOrBuildElement("H");
```

В данном случае метод FindOrBuildElement позволяет найти элемент в базе по его символу, однако существует перегрузка для доступа по атомному номеру:

```
auto nist_H_2 = nist->FindOrBuildElement(1);
```

Для поиска готовых материалов в качестве ключа следует использовать имя материала. Полный список доступных материалов в Geant4 можно найти в разделе: Geant4 Book For Application Developers > Appendix > Geant4 Material Database

Например, получаемый таким образом моно-материал "кислород"

```
auto nist_O_mat = nist->FindOrBuildMaterial("G4_O");
```

создаваемую в предыдущей части воду можно получить как

```
auto nist_Water = nist->FindOrBuildMaterial("G4_WATER");
```

В завершение следует сказать, что создаваемые вручную изотопы, элементы и материалы должны иметь уникальные имена в рамках своей группы. Однако при повторных вызовах nist-методов FindOrBuild*(Material/Element) возвращается указатель на материал/элемент, который был создан при первом вызове, и ошибок не возникает.