Введение в GEANT4

Иванов Артем Викторович
E-mail: arivanov@jinr.ru

Содержание

- Создание материалов
- Описание магнитного поля

Geometry

```
G4NistManager* nist = G4NistManager::Instance();
G4Material* env_mat = nist->FindOrBuildMaterial("G4_Al");
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");
```

Создаем World, который у нас имеет форму куба

```
G4double world_sizeXY = 100;
G4double world_sizeZ = 100;
auto solidWorld =new G4Box("World",  0.5 * world_sizeXY, 0.5 * world_sizeXY, 0.5 * world_sizeZ);

auto logicWorld = new G4LogicalVolume(solidWorld,  world_mat, "World");

auto physWorld = new G4PVPlacement(0,   G4ThreeVector(0,0,0),  logicWorld,  "World",  0,  false,  0,  true);
```

Ноль, потому что это World

Создаем куб

```
G4double env_sizeXY = 10;
G4double env_sizeZ = 10;
auto solidEnv = new G4Box("Box",   0.5 * env_sizeXY, 0.5 * env_sizeXY, 0.5 * env_sizeZ);

auto logicEnv = new G4LogicalVolume(solidEnv,  env_mat, "Box");

auto physEnv = new G4PVPlacement(0,  G4ThreeVector(0,0,0),    logicEnv,  "Box", logicWorld,  false,  0, true);
```

Не ноль, потому что помещаем в World

Класса для описания вещества

Основной принцип: Geant4 повторяет реальное строение вещества — материалы состоят из элементов, а элементы — из изотопов.

Три основных класса для описания вещества:

G4Isotope (Изотоп)

Описывает атомные ядра с уникальными свойствами:

- Z — атомный номер (число протонов)
- N — число нуклонов (протоны + нейтроны)
- A — молярная масса
- Уровень изомера

G4Element (Элемент)

Описывает химические элементы:

- (Эффективный) атомный номер (Zeff)
- (Эффективное) число нуклонов
- (Эффективная) атомная масса (Aeff)
- Энергии связей на атомных оболочках

G4Material (Материал)

- Описывает макроскопические свойства вещества:
  - Плотность, агрегатное состояние, температура, давление
  - радиационная длина, длина поглощения и т.п.
- Состав определяется базовым материалом или списком компонентов. Компонентами могут быть элементы или другие материалы.

База материалов NIST

База материалов **NIST** представляет собой коллекцию различных баз данных и публикаций, созданных Национальным институтом стандартов и технологий США (NIST).

В данной базе:

- Доступно более 3000 изотопов
- Доступны часто используемые материалы: ткань, пластик, цитозин, тимин, кевлар и т.д.

Создание элементов с использованием базы NIST

Чтобы работать с материалами из базы NIST, необходимо получить указатель на менеджер:

G4NistManager* nist = G4NistManager::Instance();

Этот класс реализует паттерн *Singleton*, то есть в программе может существовать только один объект этого класса. Доступ к этому объекту возможен из любой части кода. Это гарантирует, что все материалы будут определяться однозначно и централизованно.

```
class G4NistManager {
   ..............
   static G4NistManager* instance;
    static G4NistManager* Instance() {
              if (!instance)  instance=new G4NistManager;
               return instance;  }
   ..............
}

G4NistManager* G4NistManager::instance = 0;
```

Создание элементов с использованием базы NIST

Для создания элементов из базы NIST нужно вызывать метод FindOrBuildElement() в качестве аргумента передав либо номер элемента либо его имя:

```
G4Element* H = nist->FindOrBuildElement(1);
```

или

```
G4Element* H = nist->FindOrBuildElement("H");
```

Важно понимать, что мы не создаём новые элементы, а лишь получаем указатели на них. Сами элементы уже были инициализированы в G4NistManager, поэтому их повторное создание избыточно.

Создание материалов с использованием базы NIST

Чтобы создать материал используя базу NIST необходимо воспользоваться методом **FindOrBuildMaterial()** в качестве аргумента передав имя материала:

```
G4Material* water = nist->FindOrBuildMaterial(“G4_WATER”);
```

Кроме того в базе материалов представлены «материалы – элементы».

```
G4Material* fe = nist->FindOrBuildMaterial(“G4_Fe”);
```

Эти материалы состоят из одного атома, и могут быть использованы для построения материалов, для которых не известна химическая формула, но доступно процентное содержание того или иного элемента:

Geant4 Material Database

Z	Name	ChFormula	density(g/cm^3)	I(eV)
1	G4_H		8.3748e-05	19.2
2	G4_He		0.000166322	41.8
3	G4_Li		0.534	40
4	G4_Be		1.848	63.7
5	G4_B		2.37	76
6	G4_C		2	81
7	G4_N		0.0011652	82
8	G4_O		0.00133151	95
9	G4_F		0.00158029	115
10	G4_Ne		0.000838505	137
11	G4_Na		0.971	149
12	G4_Mg		1.74	156
13	G4_Al		2.699	166
14	G4_Si		2.33	173
15	G4_P		2.2	173
16	G4_S		2	180
17	G4_Cl		0.00299473	174
18	G4_Ar		0.00166201	188
19	G4_K		0.862	190
20	G4_Ca		1.55	191
21	G4_Sc		2.989	216
22	G4_Ti		4.54	233
23	G4_V		6.11	245
24	G4_Cr		7.18	257
25	G4_Mn		7.44	272
26	G4_Fe		7.874	286
27	G4_Co		8.9	297
28	G4_Ni		8.902	311
29	G4_Cu		8.96	322
30	G4_Zn		7.133	330
31	G4_Ga		5.904	334
32	G4_Ge		5.323	350
33	G4_As		5.73	347
34	G4_Se		4.5	348
35	G4_Br		0.0070721	343
36	G4_Kr		0.00347832	352
37	G4_Rb		1.532	363
38	G4_Sr		2.54	366
39	G4_Y		4.469	379
40	G4_Zr		6.506	393
41	G4_Nb		8.57	417
42	G4_Mo		10.22	424
43	G4_Tc		11.5	428
44	G4_Ru		12.41	441
45	G4_Rh		12.41	449
46	G4_Pd		12.02	470
47	G4_Ag		10.5	470
48	G4_Cd		8.65	469
49	G4_In		7.31	488

14	G4_BLOOD_ICRP		1.06	75.2
	1	0.101866		
	6	0.10002		
	7	0.02964		
	8	0.759414		
	11	0.00185		
	12	4e-05		
	14	3e-05		
	15	0.00035		
	16	0.00185		
	17	0.00278		
	19	0.00163		
	20	6e-05		
	26	0.00046		
	30	1e-05		
8	G4_BONE_COMPACT_ICRU		1.85	91.9
	1	0.063984		
	6	0.278		
	7	0.027		
	8	0.410016		
	12	0.002		
	15	0.07		
	16	0.002		
	20	0.147		

Создание изотопов

У class-a G4Isotope следующий конструктор

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

два изотопа урана: U-235 и U-238.

```
G4Isotope* U235 = new G4Isotope("U235", 92, 235, 235.044*g/mole);
G4Isotope* U238 = new G4Isotope("U238", 92, 238, 238.051*g/mole);
```

Создание элементов

```
G4Element(const G4String& name,      //имя
          const G4String& symbol,    //символьное обозначение элемента
          G4int nbIsotopes)          //количество изотопов

G4Element(const G4String& name,      //имя
          const G4String& symbol,    //символьное обозначение элемента
          G4double  Z,               //атомный номер
          G4double  A)               //молярная масса
```

```
G4Element *enrichedU = new G4Element("enrichedU", "U", 2);
enrichedU->AddIsotope(U235, 5.0  * perCent);
enrichedU->AddIsotope(U238, 95.0 * perCent);
```

```
G4Element* naturalUranium = new G4Element("NaturalUranium", "U", 92.,
238.02891*g/mole);
```

Создание материалов

У class-a G4Material следующий конструктор

```
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
```

```
state может принимать следующие значения

 kStateSolid,      // Твердое состояние
 kStateLiquid,     // Жидкое состояние
 kStateGas,        // Газообразное состояние
 kStateUnefined    // Неизвестное состояние
```

```
Нормальное давление
1 атм

комнатная температура
293.15 K (20°C)
```

Создание материалов

Молекулы определяются из отдельных элементов (в данном случае мы используем число атомов)

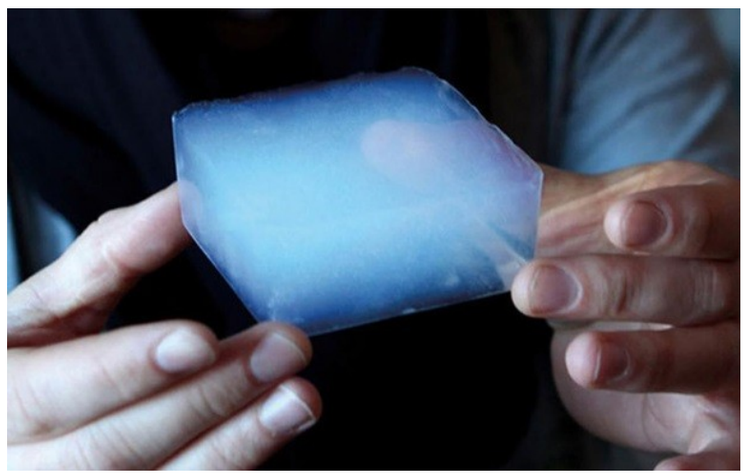
```
G4Element* elH  = new G4Element(“Hydrogen”,   symbol="H",  z=1.,  a = 1.01*g/mole   );
G4Element* elO  =  new G4Element(“Oxygen”,     symbol="O",  z=8.,   a = 16.00*g/mole );

G4Material* H2O = new G4Material(“Water”, density = 1.000*g/cm3, ncomp=2);

H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

Создание материалов

Мы также можем определять смеси, используя существующие материалы или элементы, используя массовую долю.



```
G4Element* elC  = ...;    // задали элемент углерод
G4Material* SiO2 = ...;  // задали материал стекло
G4Material* H2O = ...;   // задали материал вода
```

Силикатный аэрогеля

```
G4Material* Aerog =   new G4Material("Aerogel",
                                     density = 0.200*g/cm3,
                                     ncomponents=3);
```

```
Aerog->AddMaterial(SiO2, fractionmass=62.5*perCent);
Aerog->AddMaterial(H2O ,fractionmass=37.4*perCent);
Aerog->AddElement (elC ,  fractionmass= 0.1*perCent);
```

Создаем газ

```
G4Element* elC  = ...;    // задали элемент углерод
G4Element* elO  = ...;    // задали элемент кислород

density        = 27.*mg/cm3;    // Нормальное давление, комнатная температура



G4Material* CO2 = new G4Material(name="Carbonic gas", density, ncomponents=2);

CO2->AddElement(elC, natoms=1);
CO2->AddElement(elO, natoms=2);
```

Создаем газ

```
G4Element* elC  = ...;    // задали элемент углерод
G4Element* elO  = ...;    // задали элемент кислород

density     = 27.*mg/cm3;
pressure    = 50.*atmosphere;
temperature = 325.*kelvin;


G4Material* CO2 = new G4Material(name="Carbonic gas", density, ncomponents=2, kStateGas,
                                                      temperature,pressure);

CO2->AddElement(elC, natoms=1);
CO2->AddElement(elO, natoms=2);
```

Изменение плотности материала

Создавайте новый материал на основе существующего с требуемой плотностью.
Class **G4NistManager** имеет метод **BuildMaterialWithNewDensity** для изменения плотности

```
G4Material* CO2 = nist->FindOrBuildMaterial("G4_CARBON_DIOXIDE");


density     = 27.*mg/cm3;
pressure    = 50.*atmosphere;
temperature = 325.*kelvin;
G4Material* CO2_newDensity =
nist->BuildMaterialWithNewDensity("Atmosphere", "G4_CARBON_DIOXIDE”,
   density,temperature, pressure );
```

Class G4NistManager

```
// Build G4Material with user defined name and density on base
// of a material from Geant4 dataBase
//
G4Material* BuildMaterialWithNewDensity(const G4String& name, const G4String& basename,
  G4double density = 0.0, G4double temp = NTP_Temperature, G4double pres = CLHEP::STP_Pressure);

// Construct a G4Material from scratch by atome count
// temperature and pressure should be consistent with the density
//
inline G4Material* ConstructNewMaterial(const G4String& name, const std::vector<G4String>& elm,
  const std::vector<G4int>& nbAtoms, G4double dens, G4bool isotopes = true,
  G4State state = kStateSolid, G4double temp = NTP_Temperature,
  G4double pressure = CLHEP::STP_Pressure);

// Construct a G4Material from scratch by fraction mass
// temperature and pressure should be consistent with the density
//
inline G4Material* ConstructNewMaterial(const G4String& name, const std::vector<G4String>& elm,
  const std::vector<G4double>& weight, G4double dens, G4bool isotopes = true,
  G4State state = kStateSolid, G4double temp = NTP_Temperature,
  G4double pressure = CLHEP::STP_Pressure);

// Construct a gas G4Material from scratch by atome count
//
inline G4Material* ConstructNewGasMaterial(const G4String& name, const G4String& nameNist,
  G4double temp, G4double pres, G4bool isotopes = true);

// Construct an ideal gas G4Material from scratch by atom count
//
inline G4Material* ConstructNewIdealGasMaterial(const G4String& name,
  const std::vector<G4String>& elm, const std::vector<G4int>& nbAtoms, G4bool isotopes = true,
  G4double temp = NTP_Temperature, G4double pressure = CLHEP::STP_Pressure);
```

Class G4NistManager

```
// Get number of elements
//
inline std::size_t GetNumberOfElements() const;

// Get atomic number by element symbol
//
inline G4int GetZ(const G4String& symb) const;

// Get atomic weight by element symbol - mean mass in units of amu of
// an atom with electron shell for the natural isotope composition
//
inline G4double GetAtomicMassAmu(const G4String& symb) const;

// Get atomic weight in atomic units - mean mass in units of amu of an atom
// with electron shell for the natural isotope composition
//
inline G4double GetAtomicMassAmu(G4int Z) const;

// Get mass of isotope without electron shell in Geant4 energy units
//
inline G4double GetIsotopeMass(G4int Z, G4int N) const;

// Get mass in Geant4 energy units of an atom of a particular isotope
// with the electron shell
//
inline G4double GetAtomicMass(G4int Z, G4int N) const;
```

Оптические свойства

```
G4Material* Aerog =   new G4Material("Aerogel",
                                     density = 0.200*g/cm3,
                                     ncomponents=3);


std::vector<G4double> energy = { 7.0*eV, 7.07*eV, 7.14*eV };
std::vector<G4double> RIND    = { 1.00, 1.00, 1.00 };
std::vector<G4double> ABSL    = { 100*m, 100*m, 100*m};


Aerog->AddProperty("RINDEX",          energy,  RIND);
Aerog->AddProperty("ABSLENGTH",    energy , ABSL);
```

Вауум

Geant4 не позволяет использовать абсолютный вакуум (материал с нулевой плотностью). Поэтому создается материал с ультранизкой плотностью.

G4_Galactic (плотность = 1.0e-25 [г/см^3]) из предопределенной базы данных является примером такого материала.

```
G4Material* vacuum = nist->FindOrBuildMaterial("G4_Galactic");
```

Информация о материале

Как вывести информацию о материале и список материалов

```
G4cout << H2O;  // вывести информацию о заданном материале
G4cout << *(G4Material::GetMaterialTable()); // вывести список всех материалов
```

Как получить доступ к базе данных материалов через команды команды пользовательского интерфейса Geant4

```
/material/nist/printElement Fe          # вывести информацию об элементе по имени
/material/nist/printElementZ 13         # вывести информацию об элементе по атомному номеру(Z)
/material/nist/listMaterials type       # вывести материалы по типу = [hep, bio, ...]
/material/g4/printElement elmName       # вывести созданный элемент по имени
/material/g4/printMaterial matName      # вывести созданный материал по имени
```

Заключение по материалом

Используйте предопределенную базу данных материалов по возможности.
Это просто и точно!

Создание магнитного поля

Как создать (магнитное) поле?

Создать его в методе **ConstructSDandField()** вашего DetectorConstruction

DetectorConstruction.hh

```
#ifndef DETECTORCONSTRUCTION_HH
#define DETECTORCONSTRUCTION_HH
#include <G4VUserDetectorConstruction.hh>

class DetectorConstruction : public G4VUserDetectorConstruction{
public:
    G4VPhysicalVolume *Construct() override;
    void ConstructSDandField();


};
#endif
```

DetectorConstruction.cxx

```
#include "DetectorConstruction.hh"
#include "G4SystemOfUnits.hh"

void *DetectorConstruction::ConstructSDandField(){

        описания полей и чувстительных объемов

}
```

Создание магнитного поля

Однородное поле:

использовать класс **G4UniformMagField.**

конструктор G4UniformMagField (const G4ThreeVector &FieldVector)

```
G4MagneticField* magField =
  new G4UniformMagField(G4ThreeVector(1.*Tesla,0.,0.));
```

Создание магнитного поля

Не однородное поле:

- Использовать класс **G4QuadrupoleMagField**
  конструктор **G4QuadrupoleMagField (G4double pGradient)**

```
G4MagneticField* magField = new G4QuadrupoleMagField(
1.*tesla/(1.*meter));
```

- Создать свой собственный конкретный класс, унаследованный от **G4MagneticField**, и реализовать метод **GetFieldValue**.

```
void MyField::GetFieldValue(
    const double Point[4], double *field) const
```

Point[0..2] – координаты x,y,z в глобальной системе, **Point[3]** –время
field[0..2] – выходные компоненты x,y,z магнитного поля (T)

Создаем магнитное поле

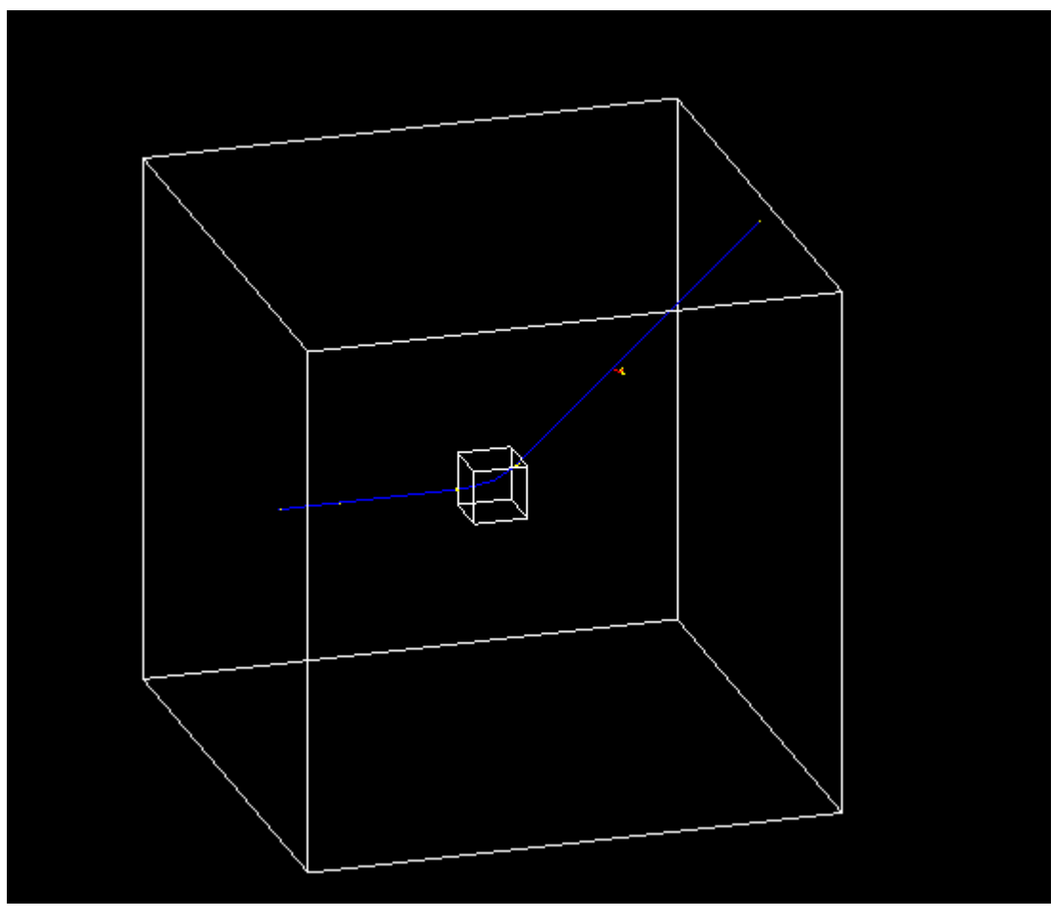
Создаем менеджер поля

```
G4FieldManager* fieldMgr = new G4FieldManager();
fieldMgr->SetDetectorField(magField);
fieldMgr->CreateChordFinder(magField);
```

Создаем магнитное поле

Ассоциируем с объемом

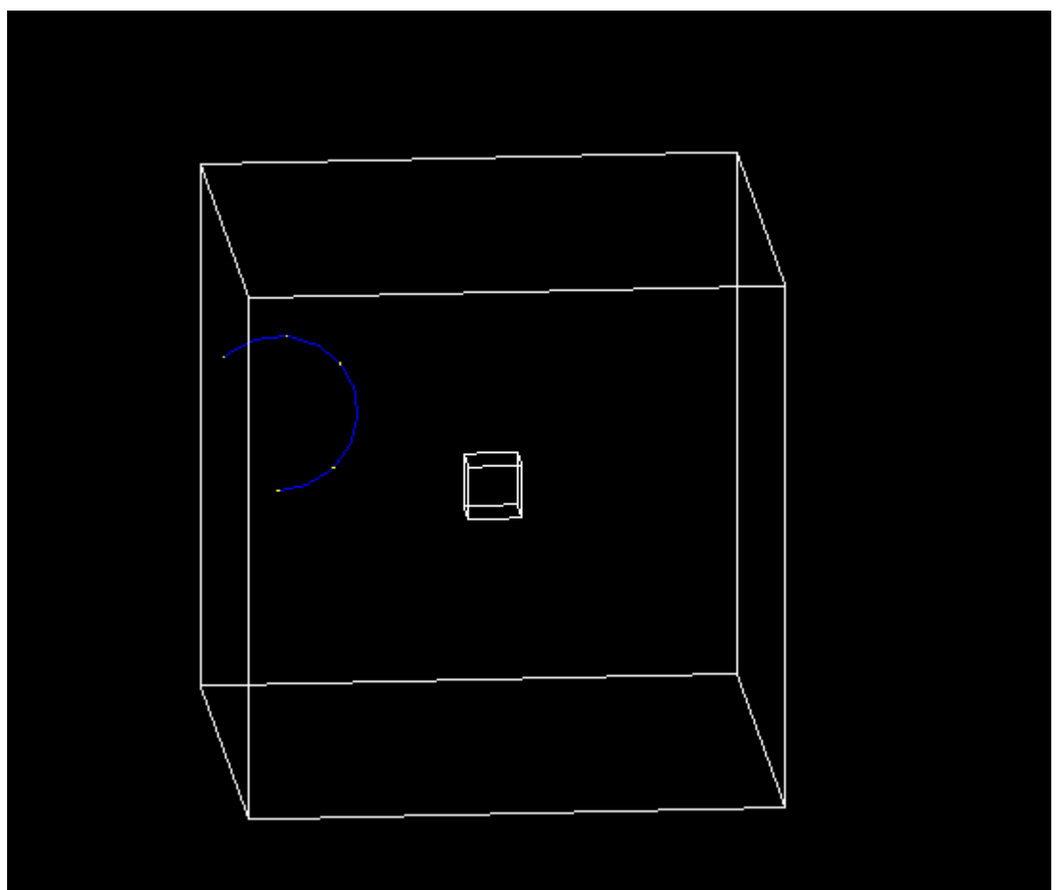
logicEnv->SetFieldManager(fieldMgr, false);



Создаем магнитное поле

Ассоциируем с world

G4TransportationManager::GetTransportationManager()->SetFieldManager(fieldMgr);



GDML

GDML (Geometry Description Markup Language) — это формат на основе **XML**, разработанный для описания геометрии детекторов в физике высоких энергий. Его основная цель — предоставить независимый от платформы и фреймворка способ хранения и обмена сложными трехмерными моделями детекторов.

В Geant4 GDML играет роль моста между декларативным описанием геометрии (файл) и программным кодом на C++.

```
<solids>
   <box name="WorldBox" x="700" y="700" z="700"/>
   <box name="XBox" x="100" y="005" z="005"/>
   <box name="YBox" x="005" y="200" z="005"/>
   <box name="ZBox" x="005" y="005" z="300"/>
   <box name="Origin" x="10" y="10" z="10"/>
   <tessellated name="Arrow">
      <quadrangular vertex1="v1" vertex2="v2" vertex3="v3" vertex4="v4"/>
      <triangular vertex1="v1" vertex2="v2" vertex3="v5"/>
      <triangular vertex1="v2" vertex2="v3" vertex3="v5"/>
      <triangular vertex1="v3" vertex2="v4" vertex3="v5"/>
      <triangular vertex1="v4" vertex2="v1" vertex3="v5"/>
   </tessellated>
</solids>
```

```
<materials>
   <material Z="1.0" name="AIR" state="gas">
      <D value="1e-24"/>
      <atom value="1.00794"/>
   </material>
   <material name="ALU" state="solid" Z="13.0">
      <D value="2.70"/>
      <atom value="26.98"/>
   </material>
</materials>
```