Geant4 学习心得—灵敏探测器（SD）的使用

SD 使几何结构中的一部分变得“敏感”，敏感区域中的各种事件信息会被记录。使用 SD 有两种途径，即创建自己的 SD 类或使用内置的多功能 SD。下面以计算能谱为例，介绍如何创建自己的 SD 类。

这种方法主要通过继承 G4VSensitiveDetector 类来实现。

G4VSensitiveDetector 类

- Initialize()　//每个事件开始时被调用
- ProcessHits() //探测器区域内粒子的每一个 Step 完成时被调用
- EndOfEvent()　//事件结束时被调用

计算能谱，可以在 ProcessHits()函数中选择保存探测器里每一个 Step 的能量沉积或其他数据，在 EndOfEvent()函数中对每一个 Event 的结果进行处理。采用的数据保存结构为：G4VHit<G4THitsCollection<G4HCofThisEvent。G4VHit 保存一个 Step 的数据，G4THitsCollection 保存 Event 中所有 Setp 的 G4VHit 数据，G4HCofThisEvent 保存 Event 中可能存在的多个 G4THitsCollection 数据。G4HCofThisEvent 中保存的数据可以在其他类中被调用。各个部分实现的具体示例如下：

1. 注册探测器　cnscott 的博客

在 G4VUserDetectorConstruction 子类的 Contruct()函数末尾返回之前加入下面内容：

```
G4VSensitiveDetector* mySD= new MySensitiveDetector("MyDetector");
G4SDManager* SDManager= G4SDManager::GetSDMpointer();
SDManager->AddNewDetector(mySD);//向探测器的管理器注册
logicWorld->SetSensitiveDetector(mySD);//向探测器对应逻辑体注册
```

2. 获得我们关心的数据　cnscott 的博客

- 编写用于保存数据的 G4VHit 子类

```
class MyHit: public G4VHit
{
public:
MyHit();
virtual ~MyHit();
```

```
inline void SetEnergyDeposit(double energy)
{ energyDeposit= energy; }
inline double GetEnergyDeposit() { return energyDeposit;}

private:
G4double energyDeposit; //  用于记录能量
};
//typedef G4THitsCollection<MyHit> MyHitsCollection;
```

- 变量初始化。

hitsCollection 成员变量是一个 G4THitsCollection<MyHit>集合类型的指针，你可以把集合类型看成是一个可以保存不同数据类型的动态数组。

```
void MySensitiveDetector::Initialize(G4HCofThisEvent* HCE)
{
//为整形成员变量 collectionID 赋值，取得对应 ID 值。
if (collectionID< 0)collectionID= GetCollectionID(0);
hitsCollection= new MyHitsCollection(SensitiveDetectorName, collectionName[0]);
//根据 ID 值将 hitsCollection 保存，便于其他类调用。
HCE -> AddHitsCollection(collectionID, hitsCollection);
}
```

- 保存关心数据

在 G4VSensitiveDetector 子类的 ProcessHits()函数中处理。该函数与 G4UserSteppingAction 类的 UserSteppingAction()函数非常相似，不同的是前者只有在粒子进入到探测器区域是才会被调用，而后者总是被调用。

```
G4bool MySensitiveDetector::ProcessHits(G4Step* step, G4TouchableHistory* ROhist)
{
MyHit* hit= new MyHit();
G4double energyDeposit= step ->GetTotalEnergyDeposit();
hit->SetEnergyDeposit(energyDeposit); //记录沉积能量
hitsCollection-> insert(hit);//保存信息
```

```
return true;
}
```

- 当前事件结束，处理探测器记录数据

```
void MySensitiveDetector::EndOfEvent(G4HCofThisEvent* HCE)
{
if (hitsCollection)//判断是否有数据
{
G4double energy=0;
int numberHits= hitsCollection->entries();//取得数据个数
for (int j= 0; j < numberHits; j++)
{
MyHit* hit= (*hitsCollection)[j];
energy += hit ->GetEnergyDeposit();//累加各 Step 沉积能量
}
}
}
```