清華大學
Tsinghua University

粒 子 物 理 模 拟

第五讲

王喆
清华大学

本节内容

- **Geant4运行流程和用户控制**
- 如何得到感兴趣的物理量
- 添加新的物理过程

Geant4的运行流程
与用户的干预

Geant4运行流程

PreInit - 材料，几何，粒子，物理过程

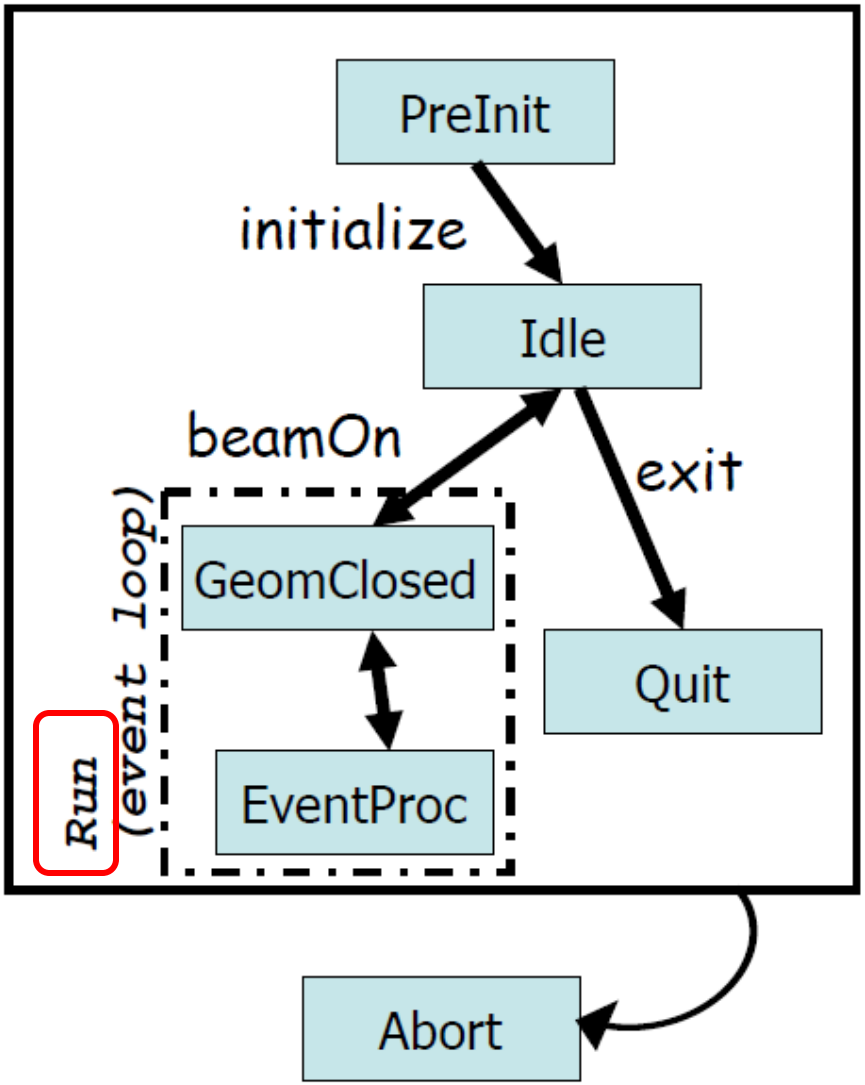
Idel - 可以开始

GeomClosed - 几何关闭

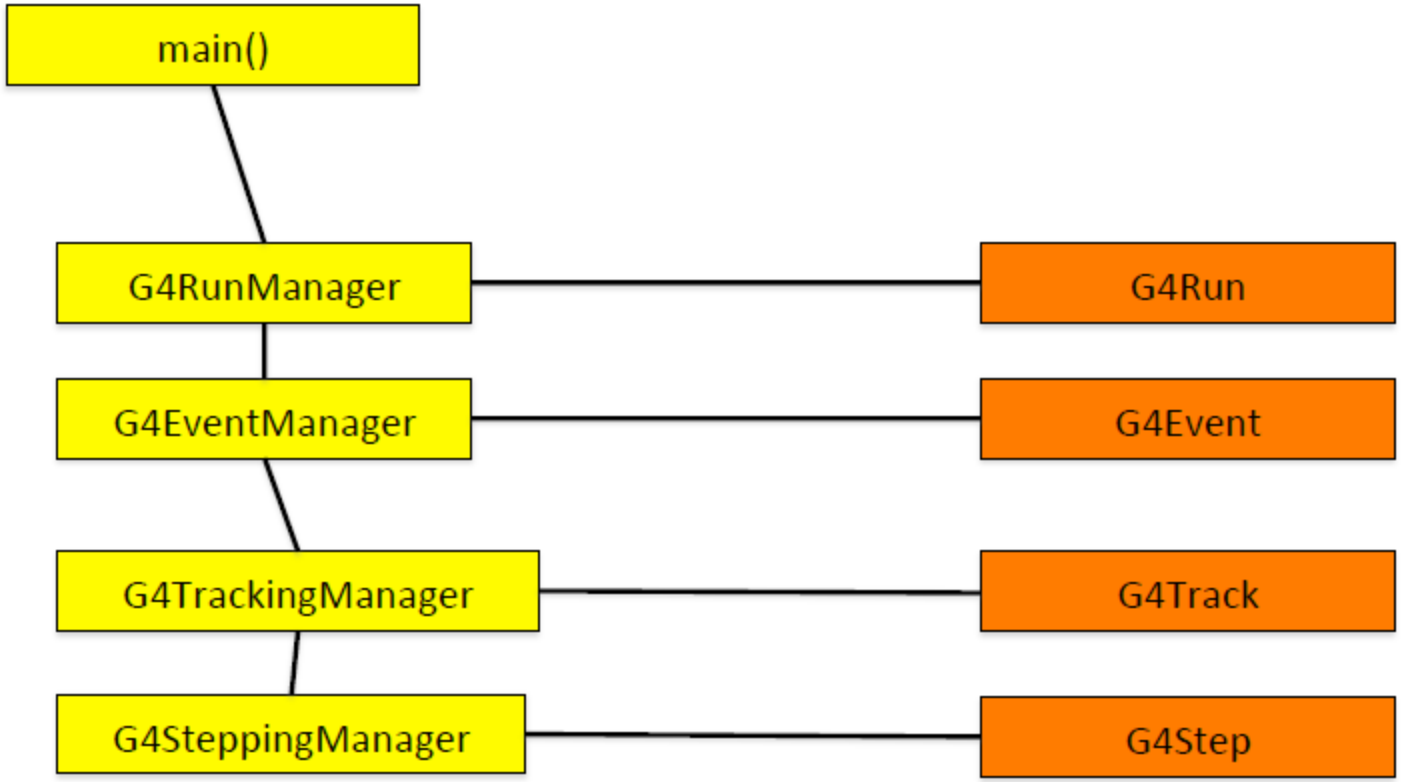
EventProc - 事例循环

Quit - 正常退出

Abort - 异常中断



Geant4运行的几个层次



G4Run

- Run是Geant4模拟的最大单元。一次Run中，探测器几何、敏感探测器、物理过程都不能改变。

  G4RunManager调用BeanOn()时开始一次Run。可以包含很多Event。
- G4UserRunAction类中有BeginOfRunAction()和EndOfRunAction()。

  前者主要用于进行run号设定、直方图或TTree,TFile定义等，后者主要进行存储直方图或者文件等。

在概念上，一个 run收集的是同一个探测器条件下的事例。

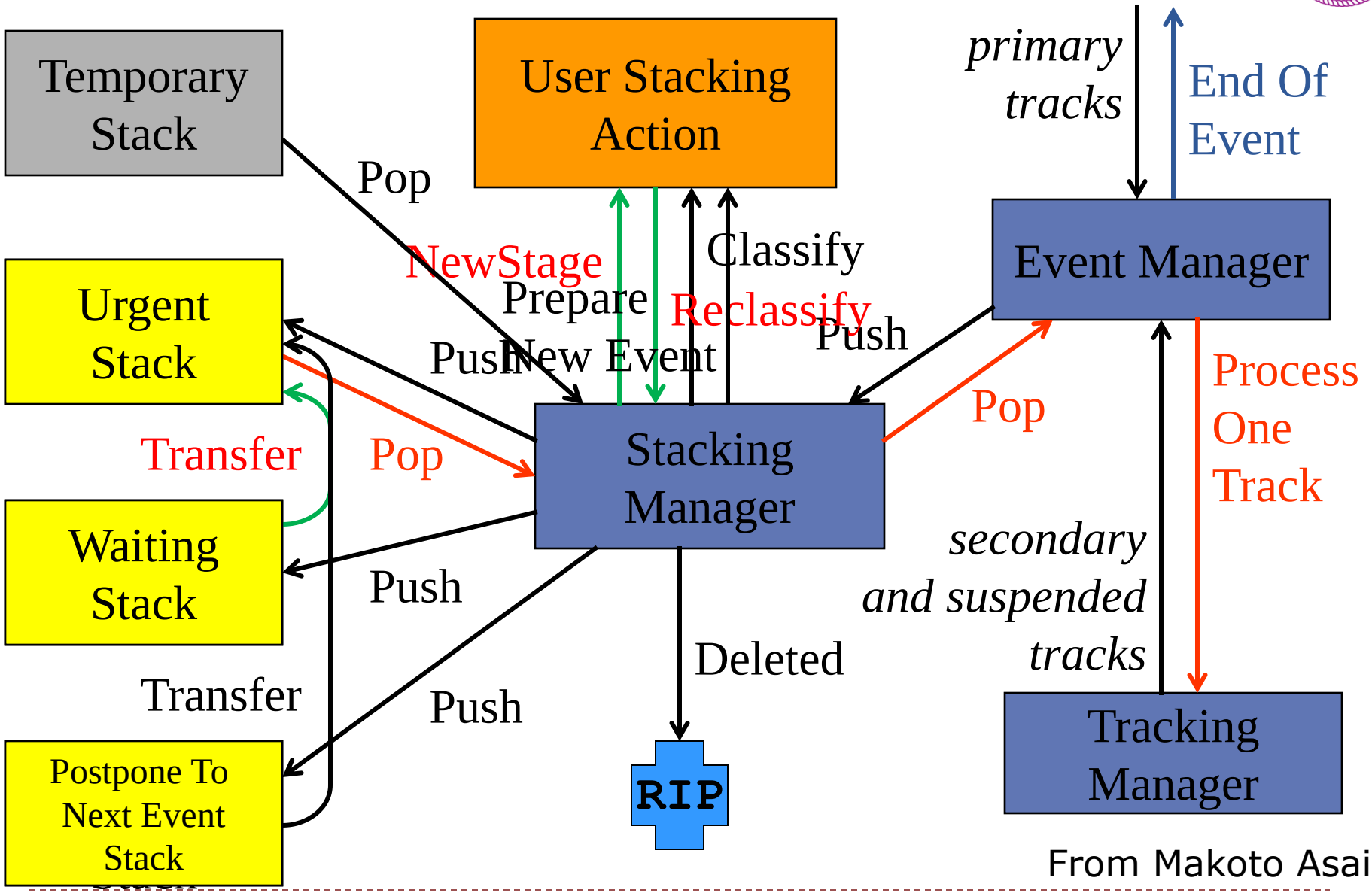
在调用BeanOn()的过程中，将调用5种(如存在)用户作用类：

G4UserRunAction, G4UserEventAction, G4UserStackingAction, G4UserTrackingAction, G4UserSteppingAction

G4Event

- G4Event表示一个事例。一个event对象包含有模拟事例的所有输入和输出信息，主要是4大类：主顶点和主粒子、径迹、击中以及数字化集合。
- G4UserEventAction类中有BeginOfEventAction()和EndOfEventAction()，前者可以作事例开始的预备工作，后者可以将事例的有用信息提取出来，填充到直方图或者TTree中。
  - 用作输入的原初顶点和粒子列表
  - 所收集的各种在探测器的击中或响应
  - 所收集的各种运动轨迹信息
  - 所收集的各种数字化信息

Stacking机制 -- Track管理

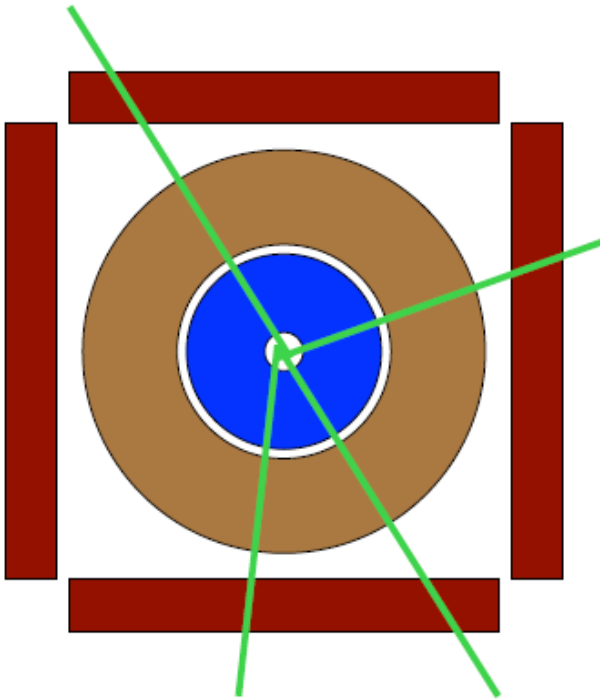


Stacking的三个用户可重载函数

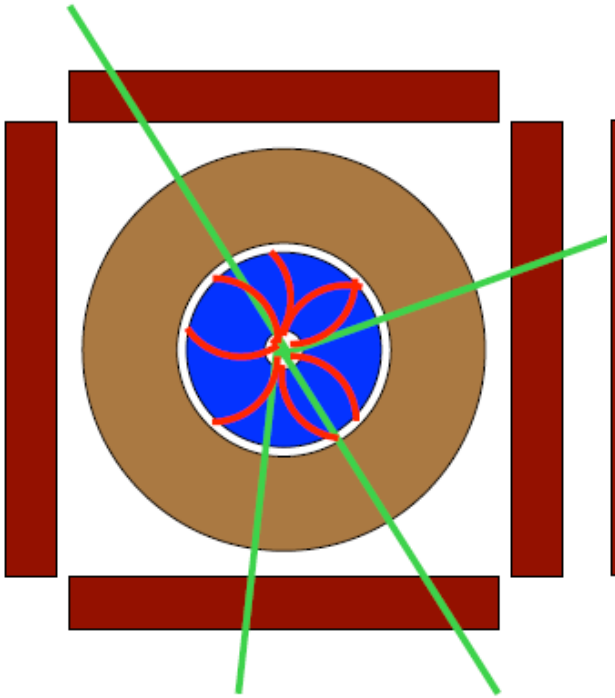
- Stacking的机制让用户有机会控制模拟的进程，快慢程度，细致程度
- ClassifyNewTrack(const G4Track*)
  每次得到一个新的track的时候将被调用
  可以分类为fUrgent，fWaiting，fPostpone，fKill
- 另外还有NewStage()和PrepareNewEvent()

可参见例子B3

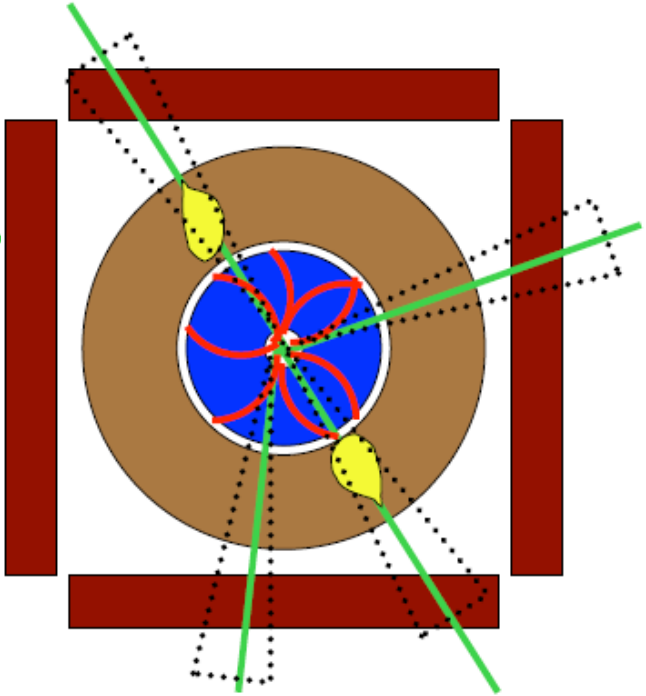
使用实例



Urgent：
初级的高能量缪子

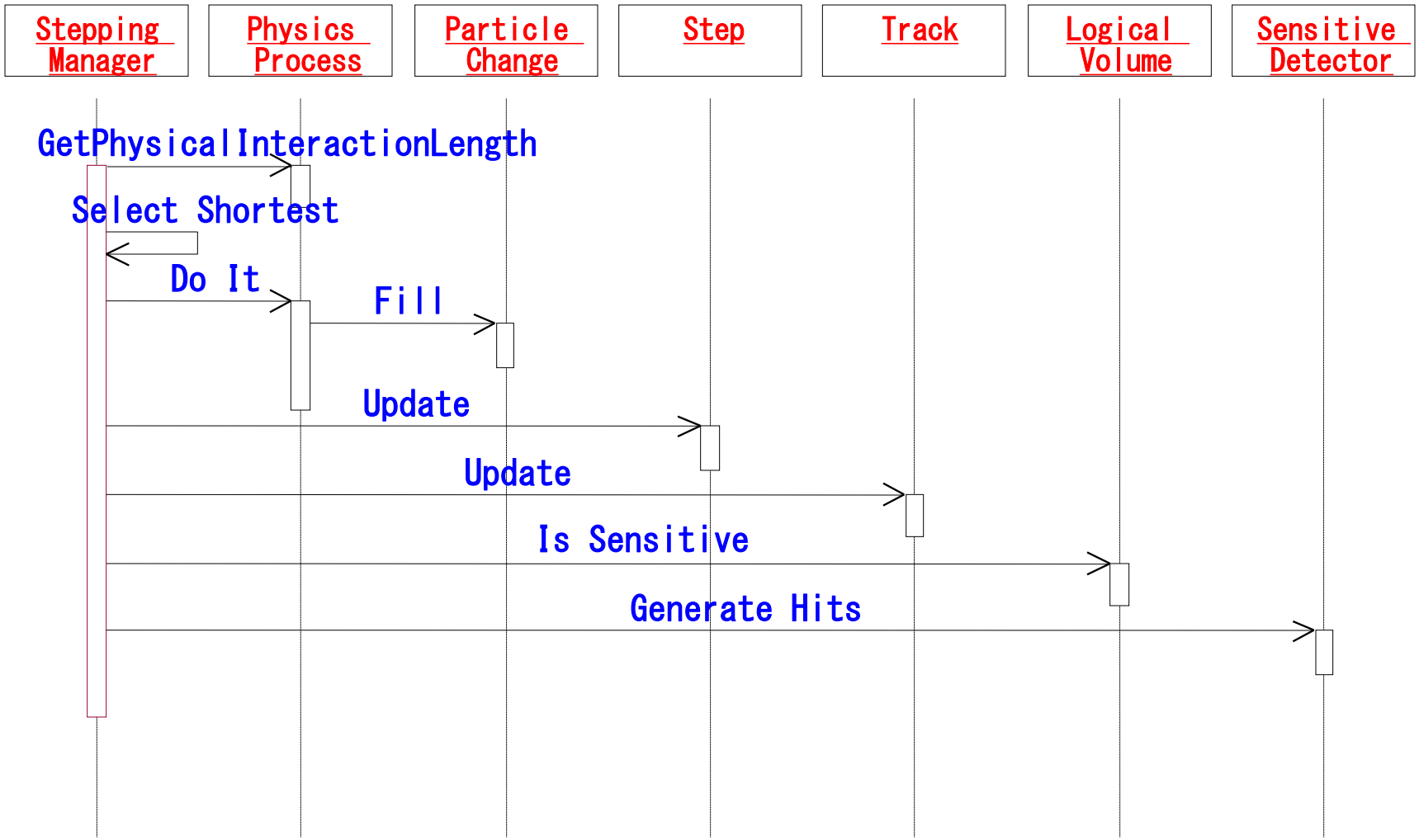


Urgent：
漂移室内的次级带电粒子



Killed：
光子可以去掉，反而替代为快速模拟，节约大量时间

Stepping流程

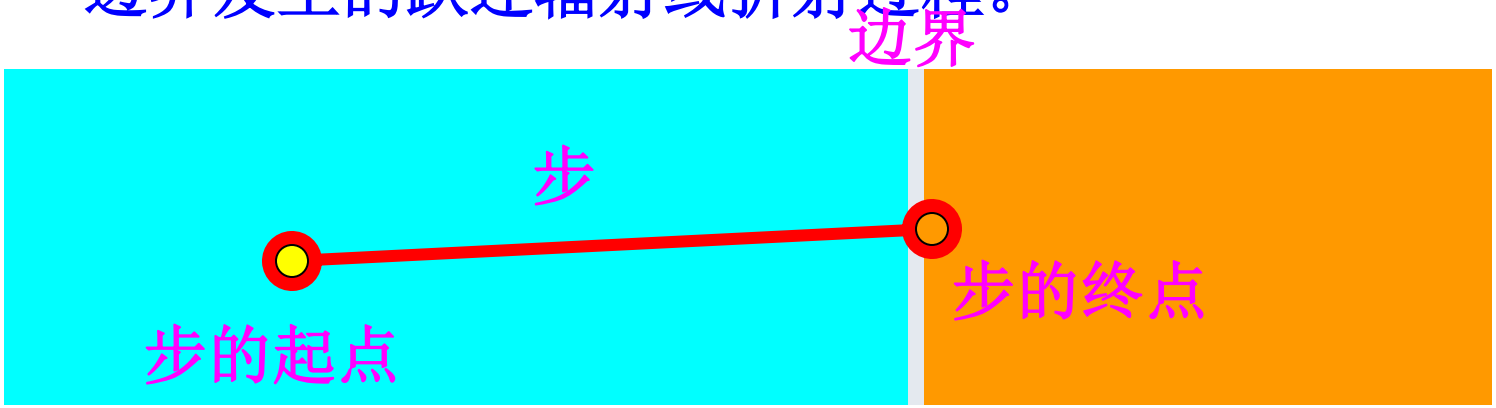

Stepping Manager
Physics Process
Particle Change
Step
Track
Logical Volume
Sensitive Detector
GetPhysicalInteractionLength
Select Shortest
Do It
Fill
Update
Update
Is Sensitive
Generate Hits

在 Geant4 中的迹

- 迹是粒子在探测器中留下的痕迹。
  - 只体现出当时粒子的位置和物理量。
- 步是粒子径迹的一小段“Δ”信息。
  - 在**Geant4**内部粒子径迹并不是步的集合。
- 在下列情况下，迹要被删除：
  - 迹离开广义中的大体积
  - 迹消失（例如发生了衰变）
  - 粒子动能为零，在(**AtRest**)时也无其它物理过程的要求。
  - 用户决定要将其删除。
- 在每个事例结尾，不保留迹的目标模块。
  - 用运功轨迹的类目标模块来记录粒子的径迹。
- *G4TrackingManager*负责管理处理迹的进程，迹由 *G4Track* 类表示。

在 Geant4 中的步

- 每一步都有两个点和粒子的“Δ”信息（在该步的能损，所需的飞行时间，等等）。
- 在每一点上，都应该知道其所处在的体积(与材料)内。如有一步跨越边界，该步的截止点物理上就设在该边界上，逻辑上该点属于下一个体积。
  - 由于一步能知道两个体积的物质材料，因此可以模拟在边界发生的跃迁辐射或折射过程。



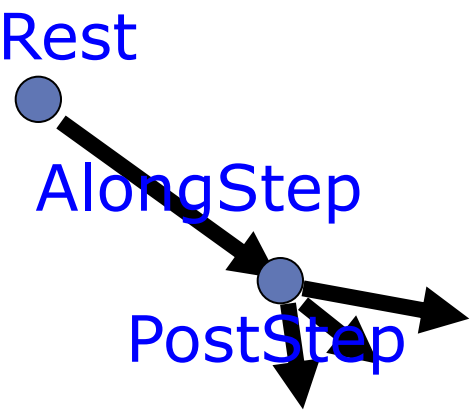
- G4SteppingManager 类负责管理步的处理，步由 G4Step 类表示。

跟踪与物理过程处理

- **Geant4 迹跟踪是很普遍的。**
  - 它无关于
    - 粒子的种类
    - 粒子所涉及的物理过程
  - 它给所有物理过程提供了
    - 帮助确定步的长度的机会
    - 帮助对任何可能对迹的物理量进行改变的机会
    - 对迹状态改变给出建议的机会
      - 例如，中止，搁置或删除。

物理过程与步

- 每个物理过程有着下列一个或几个结合的性质。
  - AtRest
    - 例如 muon 在静止时衰变
  - AlongStep
    - 例如 期仑科夫过程
  - PostStep
    - 例如 在飞行中衰变



物理过程需要指明发生在AtRest，AlongStep或PostStep上，且需要给出顺序。如下面函数调用后面的3个参数分别表示这三个状态下该物理过程是否发生以及顺序：

```
pmanager->AddProcess(new G4MultipleScattering,-1, 1,1);
```

详见ExN02PhysicsList.cc中ConstructEM()部分。

“-1”表示该物理过程不发生。

Geant4 的切割点

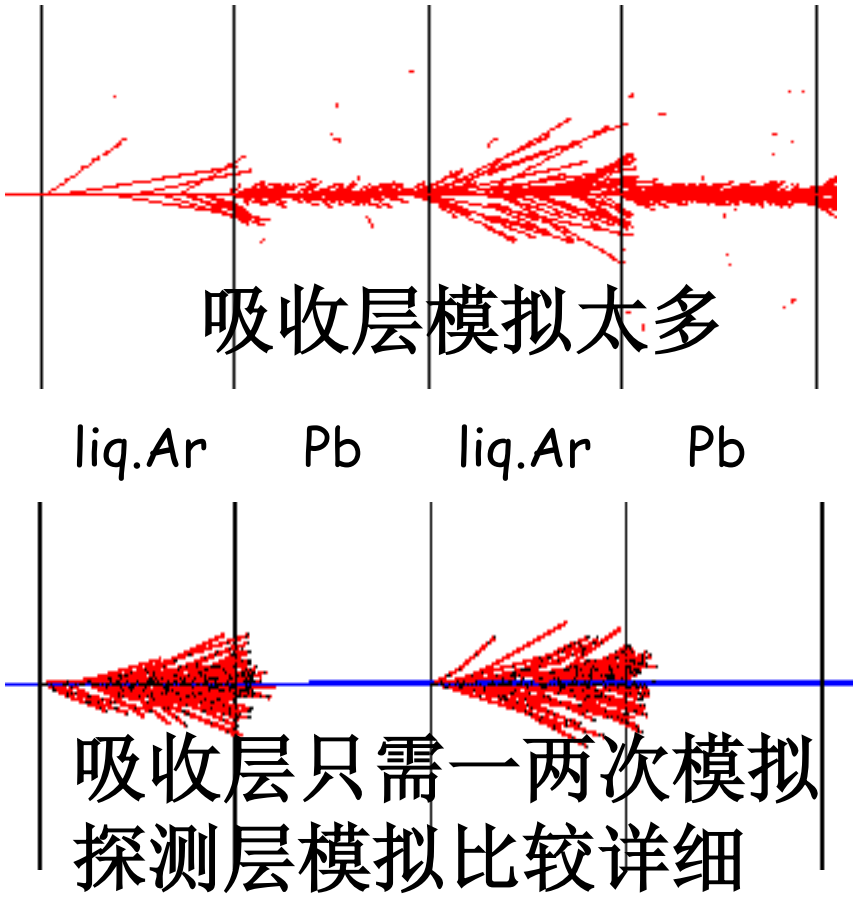
- 在Geant4切割点指的是产生阈。
  - 只对于有红外发散的物理过程。
  - 不存在粒子迹的切割
- 能量阈必须确定在能损的连续值上
  - Geant4 的处理方法：
    - 在连续能损开始那一点，指定射程（转换到每个材料对应的能量）跟踪原初粒子直到射程为零。
    - 只在给定的射程上产生次级粒子，能量低于所在物质要求射程的次级粒子，其能损叠加到原初粒子上。

能量切割与射程切割之比较

- **500 MeV/*c*** 质子液态 Ar (4mm) / Pb (4mm) 取样量能器中的输运过程

- **Geant3 (能量切割)**
  - **Ecut = 450 keV**

- **Geant4 (射程切割)**
  - **Rcut = 1.5 mm**
  - 对应于 **Ecut in liq.Ar = 450 keV, Ecut in Pb = 2 MeV**



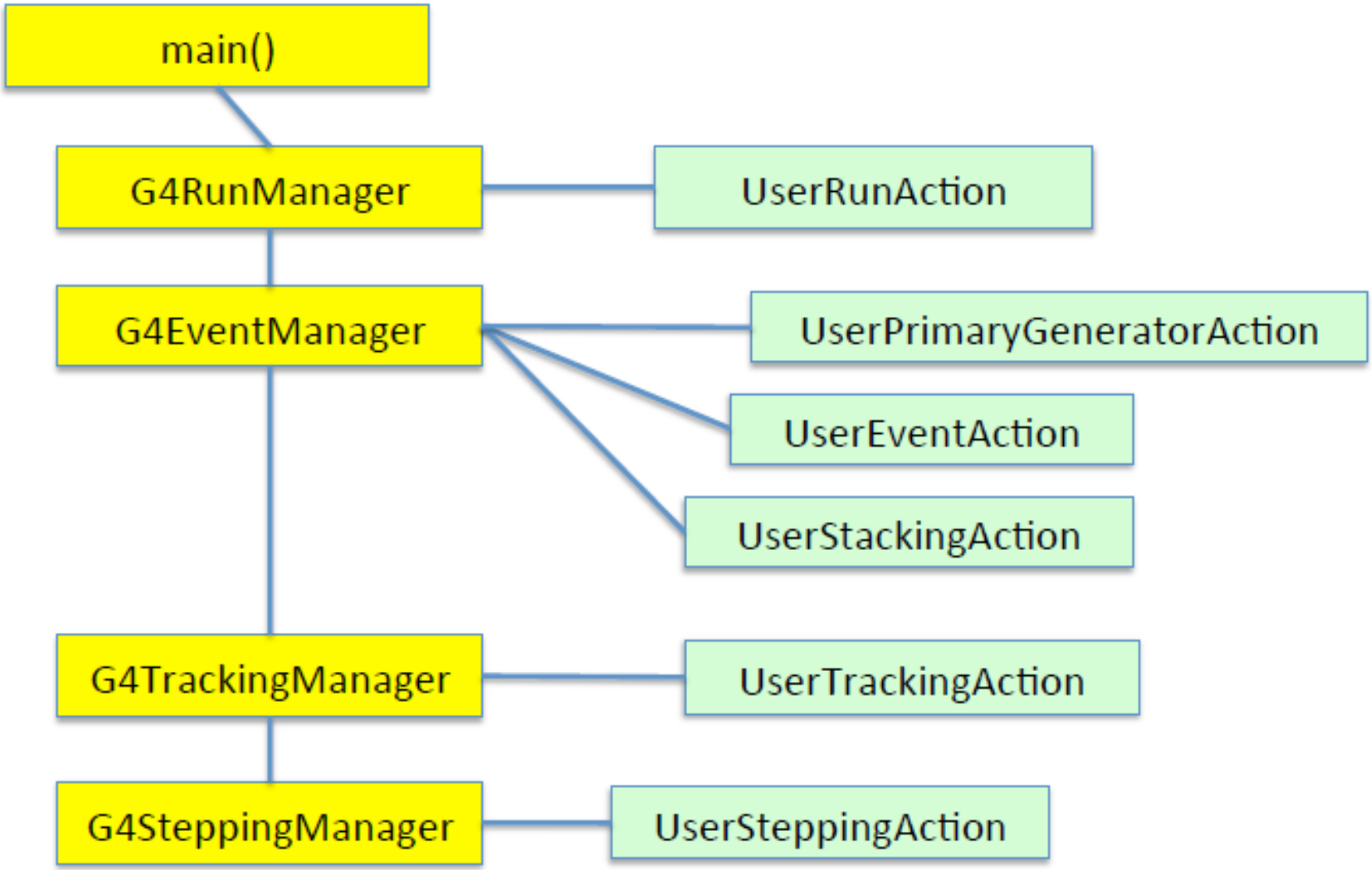
使用SteppingAction

- UserSteppingAction(const G4Step* step)

- G4Step包含了一步中的全部的状态信息，从preStep到postStep，可以查阅，并作出一些工作。

  例如，截取一些信息dE/dx，位移，路程，填充直方图等

Geant4运行机制及用户控制总结



如何得到感兴趣的物理量

如何得到感兴趣的物理量

Geant4并不明确用户需要哪些输出，需要用自行定义

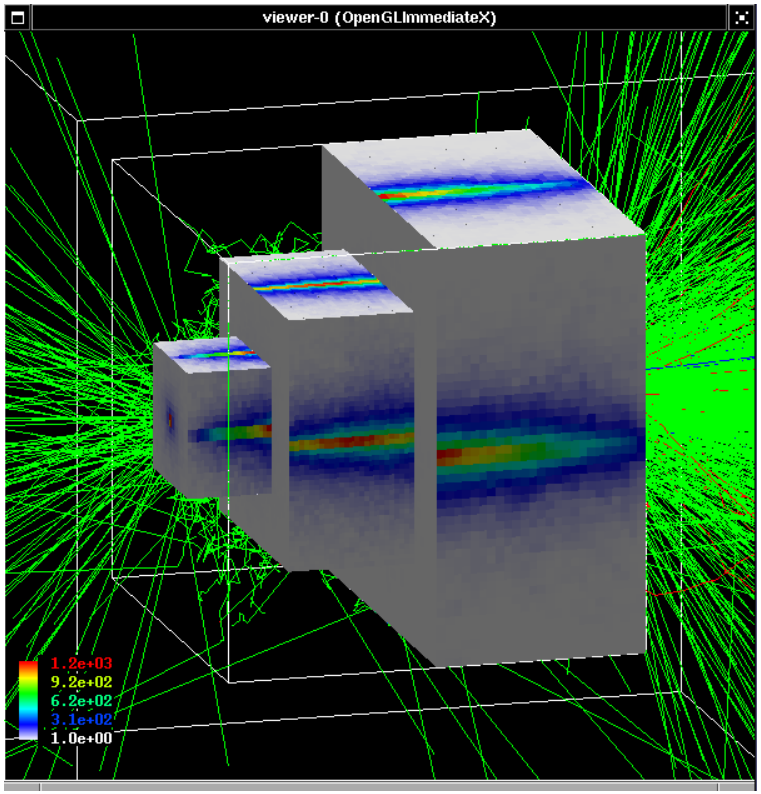
- **用户接口函数（hooks）：**

  G4UserTrackingAction

  G4UserSteppingAction

  非常直接，可以拿到全部信息。

- **定义敏感探测器：**
  - 类似
- **Scoring**
  - 略显功能有限



使用SteppingAction

- **SteppingAction的调用时间？**
- **The hook is:**

  `G4UserSteppingAction::UserSteppingAction(const G4Step*)`
- **In N02, this is inherited by** `ExN02SteppingAction`

  `void ExN02SteppingAction::UserSteppingAction(const G4Step*)`

  You need to add your code here, for example, to add up all the energy deposit.
- **Make use of G4Step to get your interested quantities:**
  1. **Where the step was?**
  2. **What particle?**
  3. **What the particle did?**

利用G4TouchableHandel取得能量沉积

How to get these information quickly?

Where the step is?

```
G4StepPoint* point1 = step->GetPreStepPoint();
G4StepPoint* point2 = step->GetPostStepPoint();

G4TouchableHandle touch1 = point1->GetTouchableHandle();
G4VPhysicalVolume* volume = touch1->GetVolume();
G4String name = volume->GetName();
G4int copyNumber = touch1->GetCopyNumber();
```

What particle?

```
G4Track* track = step->GetTrack();
G4ThreeVector momentum = track->GetMomentum();
```

What the particle did?

```
G4double eDeposit = step->GetTotalEnergyDeposit();
G4double sLength = step->GetStepLength();
G4ThreeVector displace = step->GetDeltaPosition();
G4double tof = step->GetDeltaTime();
```

G4Step其他方面

Take a look!

G4Step的位置：

……/source/track/include/G4Step.hh

G4Step的其他公共成员函数：

```
G4TrackVector* GetSecondary() const;
G4ThreeVector displace = step->GetDeltaPosition();
G4double tof = step->GetDeltaTime();
……
```

利用sensitive detector

定义方法：参见**N02**的**ExN02TrackerSD**

基本思想： **1.** 从**G4VSensitiveDetector**生成新的子类

2. 重载（请注意他们的接口参数）

void Initialize(G4HCofThisEvent*);

G4bool ProcessHits(G4Step*, G4TouchableHistory*);

可见其基本原理与利用SteppingAction是一致的，同样是利用截获每一步的G4Step信息。

```
G4double edep = aStep->GetTotalEnergyDeposit();
ExN02TrackerHit* newHit = new ExN02TrackerHit();
newHit->SetEdep (edep);
```

void EndOfEvent(G4HCofThisEvent*);

3. 在G4SDManager中注册该子类

4. 与对应的逻辑探测器相关联

在事例模拟结束时，有机会去整理全部的**Hit**信息

存成直方图，**Tree**等等

```
void G4VSensitiveDetector::EndOfEvent(G4HCofThisEvent*);
```

在G4SDManager中注册该子类

与对应的逻辑探测器相关联

```
G4LogicalVolume* myLogCalor = ……;
G4VSensetiveDetector* pSensetivePart =
  new MyDetector(“/mydet”);
G4SDManager* SDMan = G4SDManager::GetSDMpointer();
SDMan->AddNewDetector(pSensitivePart);
myLogCalor->SetSensitiveDetector(pSensetivePart);
```

添加新的物理过程

虚基过程G4VProcess

- Base class of all physics processes in Geant4

- Example: in Geant4 example area: extended/exoticphysics/monopole/include/G4MonopoleTransportation.hh

关键virtual functions of G4VProcess - DoIt

What we should add?

```
virtual G4VParticleChange* PostStepDoIt(
                           const G4Track& track,
                           const G4Step&  stepData
                          ) = 0;

virtual G4VParticleChange* AlongStepDoIt(
                           const G4Track& track,
                           const G4Step& stepData
                          ) = 0;
virtual G4VParticleChange* AtRestDoIt(
                           const G4Track& track,
                           const G4Step& stepData
                          ) = 0;
```

关键virtual functions of G4VProcess - Length

What else is needed?

```
virtual G4double AlongStepGetPhysicalInteractionLength(
                         const G4Track& track,
                         G4double  previousStepSize,
                         G4double  currentMinimumStep,
                         G4double& proposedSafety,
                         G4GPILSelection* selection) = 0;

virtual G4double AtRestGetPhysicalInteractionLength(
                         const G4Track& track,
                         G4ForceCondition* condition
                        ) = 0;

virtual G4double PostStepGetPhysicalInteractionLength(
                         const G4Track& track,
                         G4double   previousStepSize,
                         G4ForceCondition* condition
                        ) = 0;
```

生成次级粒子

- Create a few hundred secondary photons

```
aSecondaryTrack->SetParentID(aTrack.GetTrackID());

aParticleChange.AddSecondary(aSecondaryTrack);
```

加入定制的过程

当构造全部的物理过程的时候

```
void G4MonopolePhysics::ConstructProcess()
{
```

可以注册你定制的过程

```
    pmanager->AddProcess(new G4MonopoleTransportation(fMpl),-1, 0, 0);
```

本节课总结

- **Geant4的运行机制和用户干预**
- 如何得到感兴趣的物理量
- 添加新的物理过程