



清华大学
Tsinghua University

粒子物理模拟

第二讲

王喆
清华大学



主要内容:

1. 简介
2. 安装简要说明
3. 寻找帮助
4. 整体结构
5. **Run, Event, Track...**
6. 关键的C++语法, 虚函数, 类库
7. 理解、编译、执行第一个**Geant4**实例
8. 几何物质初步
9. 图形显示初步
10. 宏命令初步
11. 并行模拟
12. 练习



简介

- ▶ GEANT是GEometry ANd Tracking的缩写。
- ▶ Geant3的第一版可以追回到1974年，这时它以FORTRAN为编程语言，主要的开发机构为CERN。
- ▶ 与Geant3同期的有探测器模拟软件还有EGS，主要的开发机构为SLAC，LNL和KEK。
- ▶ Geant4计划始于1994年，C++为主要编程语言，主要开发机构为CERN，目前已经发布了10个稳定版本，目前每年一版。
- ▶ Geant的主要成功在于1) 它的易用性，简单清晰的结构，可视化；2) 得到了广泛的应用，在各种场合都有使用需求，有充分的更新、调试和维护。



Geant4 – A Simulation Toolkit

Geant 4



<http://www.geant4.org/>



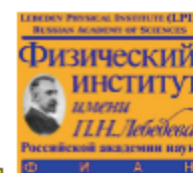
Laboratoire d'Annecy-le-Vieux
de Physique des Particules



S. Agostinelli et al.

Geant4: a simulation toolkit

NIM A, vol. 506, no. 3, pp. 250-303, 2003



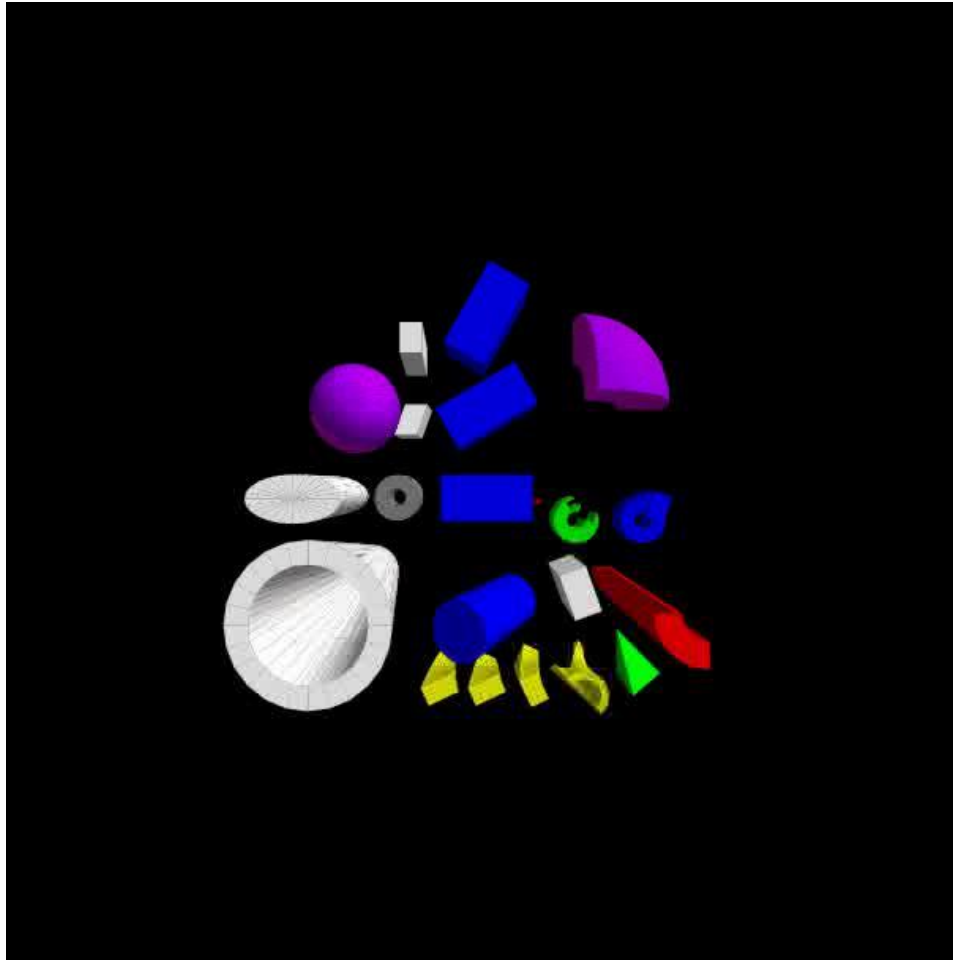
J. Allison et al.

Geant4 Developments and Applications

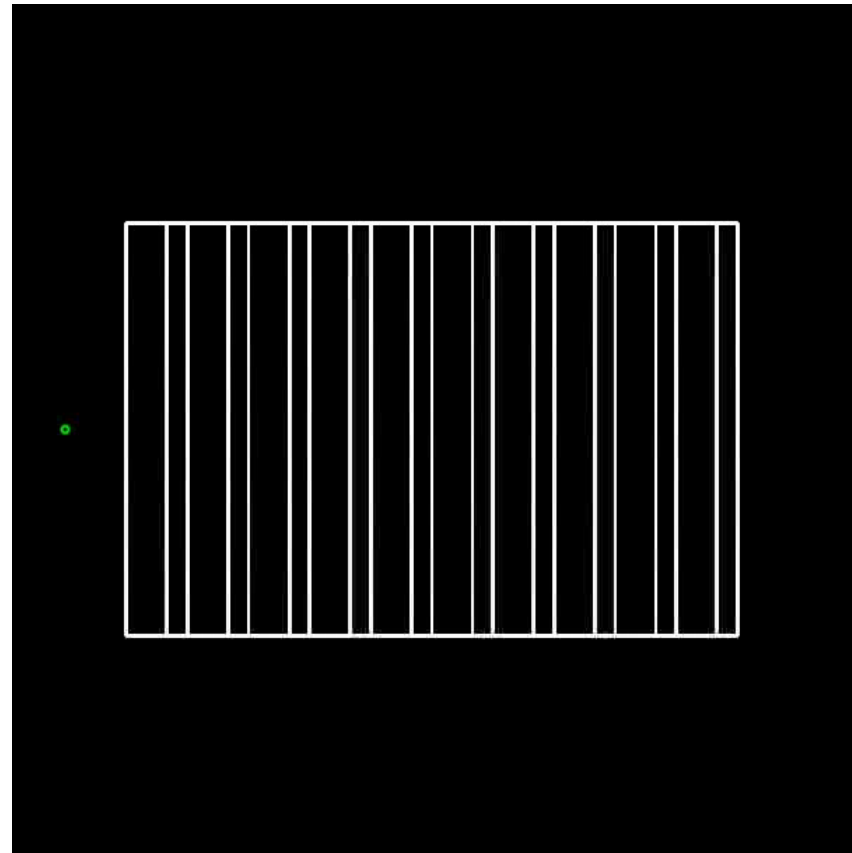
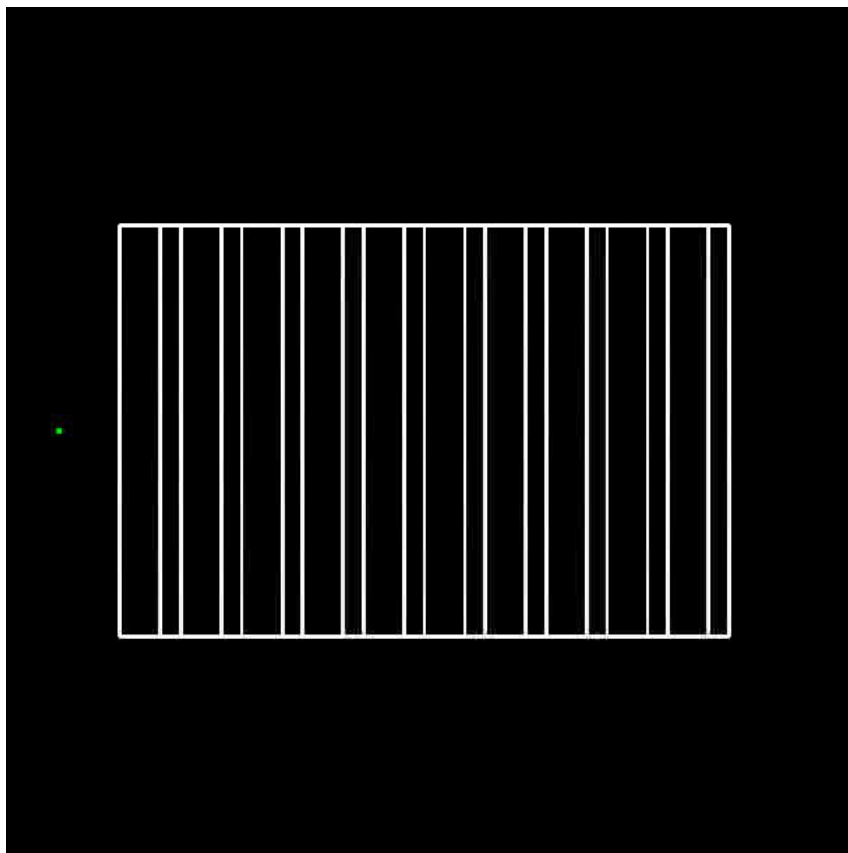
IEEE Trans. Nucl. Sci., vol. 53, no. 1, pp. 270-278, 2006



Geant4中的探测器



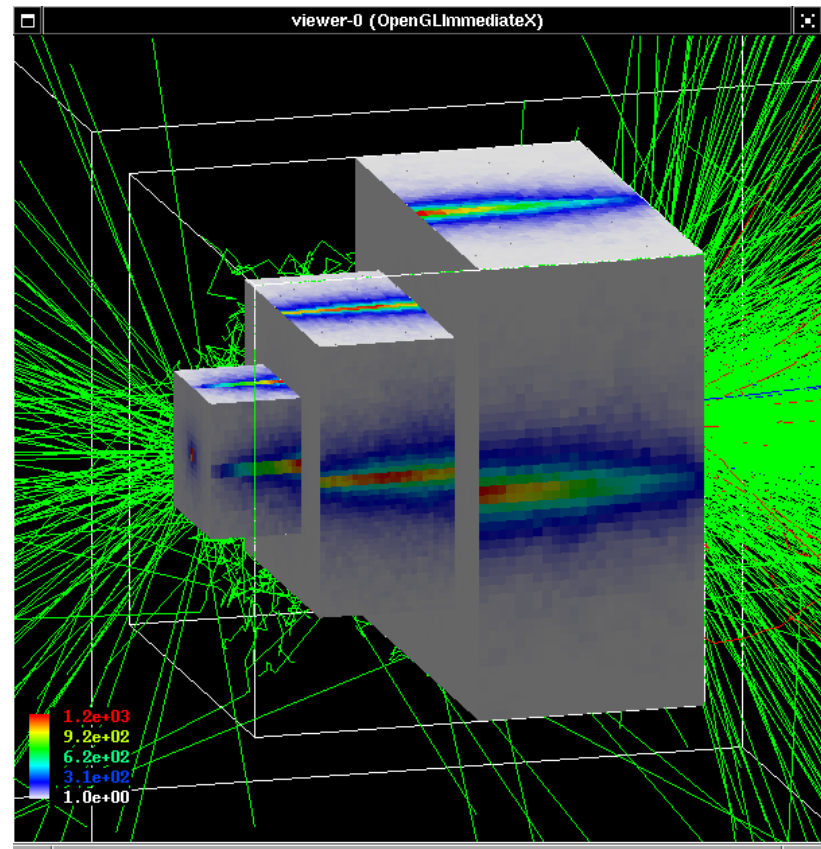
Geant4中的相互作用



哪一个电子？ 哪一个中子？

基本粒子模拟思路

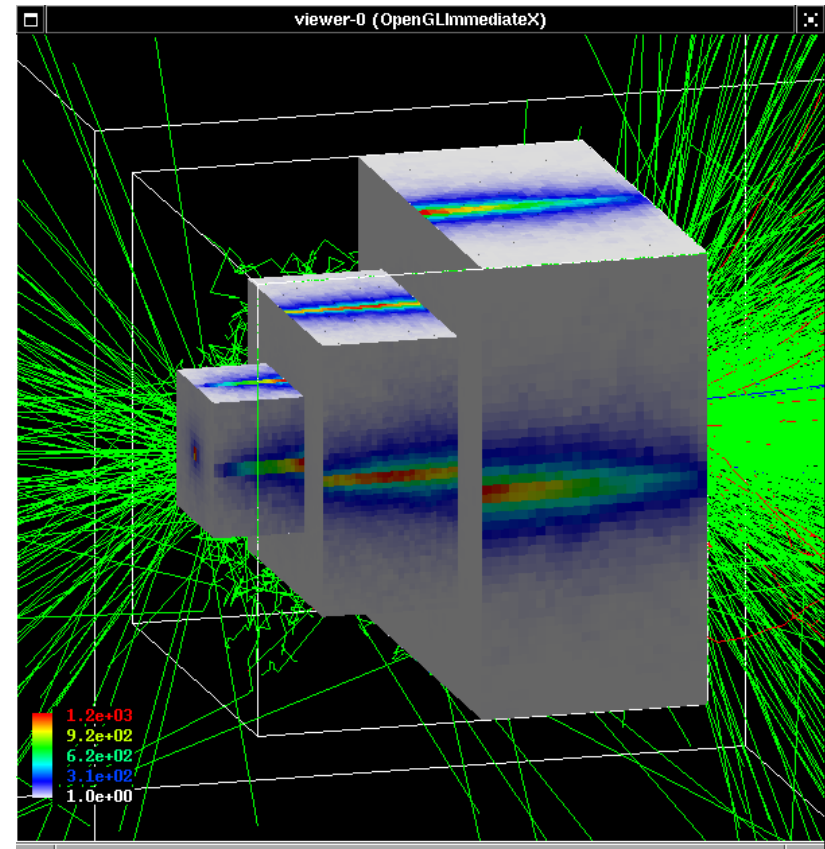
- ▶ 追溯每一个微观粒子，利用已知的粒子与物质的相互作用截面（粒子与粒子的相互作用截面），模拟粒子在物质里面的输运过程。
- ▶ 查看，计算粒子在某段物质内的能量沉积，时间特征，空间分布，来模拟探测器的实际响应。



Geant4在高能物理中的应用

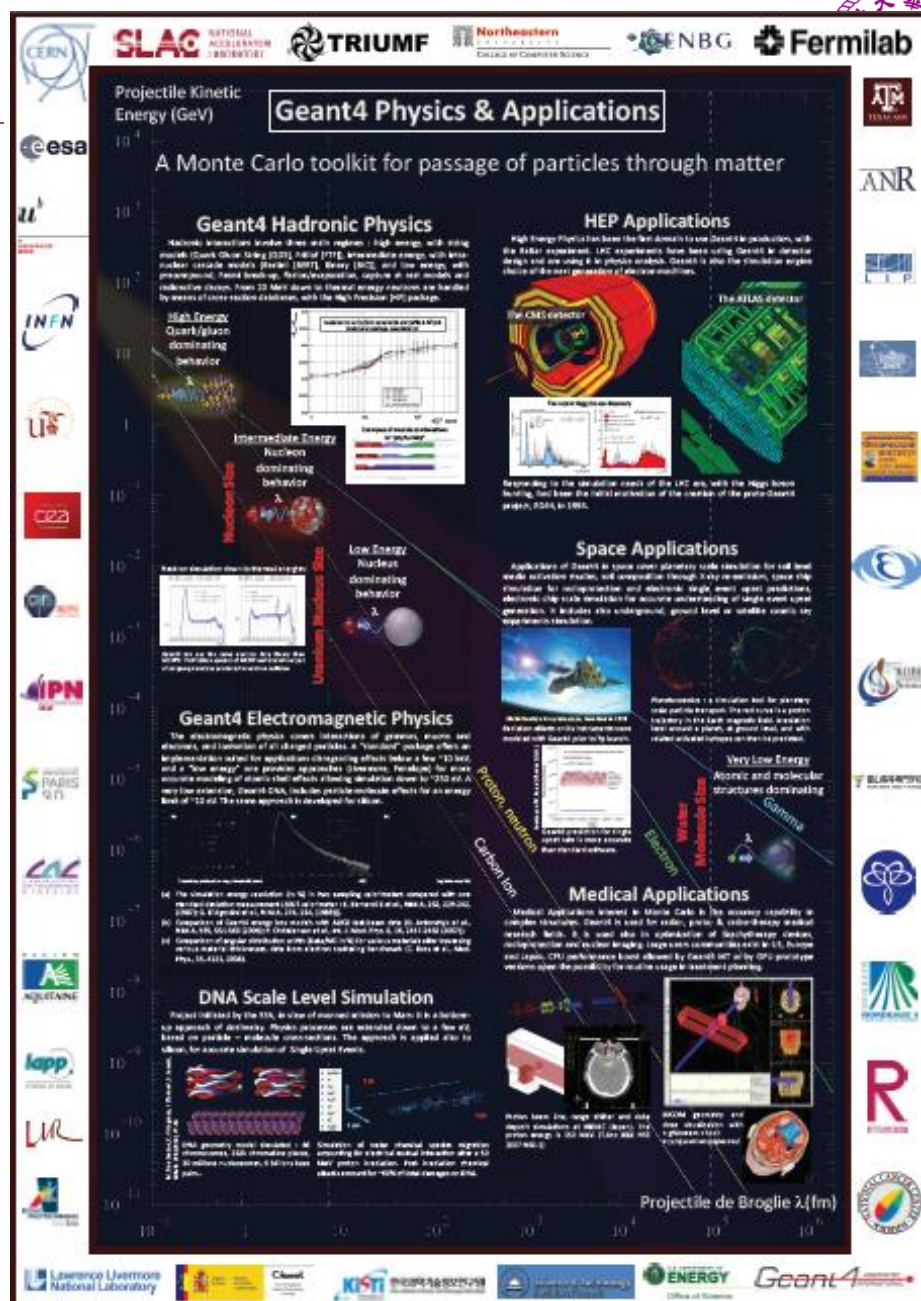
经常会发现，虽然了解第一原理，例如QED，但很多现象无法准确预期：

- ▶ 对电子及次级粒子的测量效率——能量分辨
- ▶ 对探测物质的选择，尺寸的选择——实验成本
- ▶ 加入新的相互作用，强子散射——理解新物理
- ▶ 等等.....



应用范围

- ▶ Geant4主要提供一套通用的模拟软件，模拟基本粒子穿过物质时发生的相互作用。
- ▶ 目前被应用于高能物理，核物理，空间工程，医学物理，材料科学，辐射防护和安全。
- ▶ 它所提供的帮助涵盖电磁，强，光强子，轻强子相互作用，几何光学等等。而且用户可以定制，扩展。





Geant4的版权声明

个人诠释，如需要细节，注意查看原文

- ▶ 著作权，所有权属于Geant4开发人员，不能拿Geant4的内容去发表文章，申请专利
- ▶ 可以免费使用，传播
- ▶ 不提供任何明确的或是隐含的性能、质量保证，不提供任何担保

如使用，请务必引用：

- ▶ S. Agostinelli *et al.*, Nucl. Instr. Meth. A **506**, 250 (2003)
- ▶ Allison, J. *et al.*, IEEE Trans. Nucl. Sci. **53**, 270 (2006)



安装简要说明

▶ 存在两个编译链接方式

1. ≤ 9.4 版, 使用GNU make
(大量的著名的实验还在使用)

2. ≥ 9.5 版, 使用Cmake
(最新版的, 以后的趋势, 我们的主要讲解部分)
Cmake: <http://www.cmake.org/>
(一个“十页”的文档)

详细的帮助在下页:

CMake版:

- 下载源代码包，例如 geant4.10.00.p02.tar.gz
<http://geant4.web.cern.ch/geant4/support/download.shtml>
 - 把源代码包拷贝到 /YourPath/
 - `cd /YourPath`
 - `tar xvfz geant4.10.00.p02`
- 你要替换该YourPath为真实的路径
- 与之平行的位置创建 `mkdir /YourPath/geant4.10.00.p02-build`
 - `cd /YourPath/geant4.10.00.p02-build`
 - `cmake -DGEANT4_USE_OPENGL_X11=ON \\ (OpenGL)`
`-DGEANT4_USE_GDML=ON \\ (使用GDML)`
`-DGEANT4_USE_QT=ON \\ (使用Qt)`
`-DGEANT4_INSTALL_DATA=ON \\ (安装数据)`
`-DCMAKE_INSTALL_PREFIX=/YourPath/geant4.10.00.p02-install \\`
`(安装目录)`
`/YourPath/geant4.10.00.p02 \\ (源代码目录)`
 - `make [-j 4]`
 - `make install`
- \\ 后为注释，应删去



编译做了些什么？为什么这样？

- 支持多样性，Linux（Ubuntu，Redhat，Scientific linux...），Mac，Windows
Cmake正是帮助我们鉴别各个系统，找出差别，解决各个系统的软件依赖性问题。
- 安装完成后，将有三个目录：
geant4.10.00.p02 （源代码）
geant4.10.00.p02-build （编译目录）
geant4.10.00.p02-install （安装目录）
- ▶ Geant4可方便的支持多编译选项，多安装目录，不冲突：
调试编译，优化编译，包含多线程选项的编译，不同C++版本的编译等等。

几个关键目录（请查看）



- 安装目录 **geant4.10.00.p02-install**

包含 bin, include, lib, share

- bin – binary 二进制文件，可执行的
- include – 全部的头文件 *.hh
- lib – 全部的库文件 *.so （后面一直用到该目录）
- share – 其它：数据，例子等

- ▶ 源代码目录 **geant4.10.00.p02**

1. source – 结构化的源代码目录，最终极的说明文档
2. examples – 例子
3.



C++的类库，开源与非开源的区别

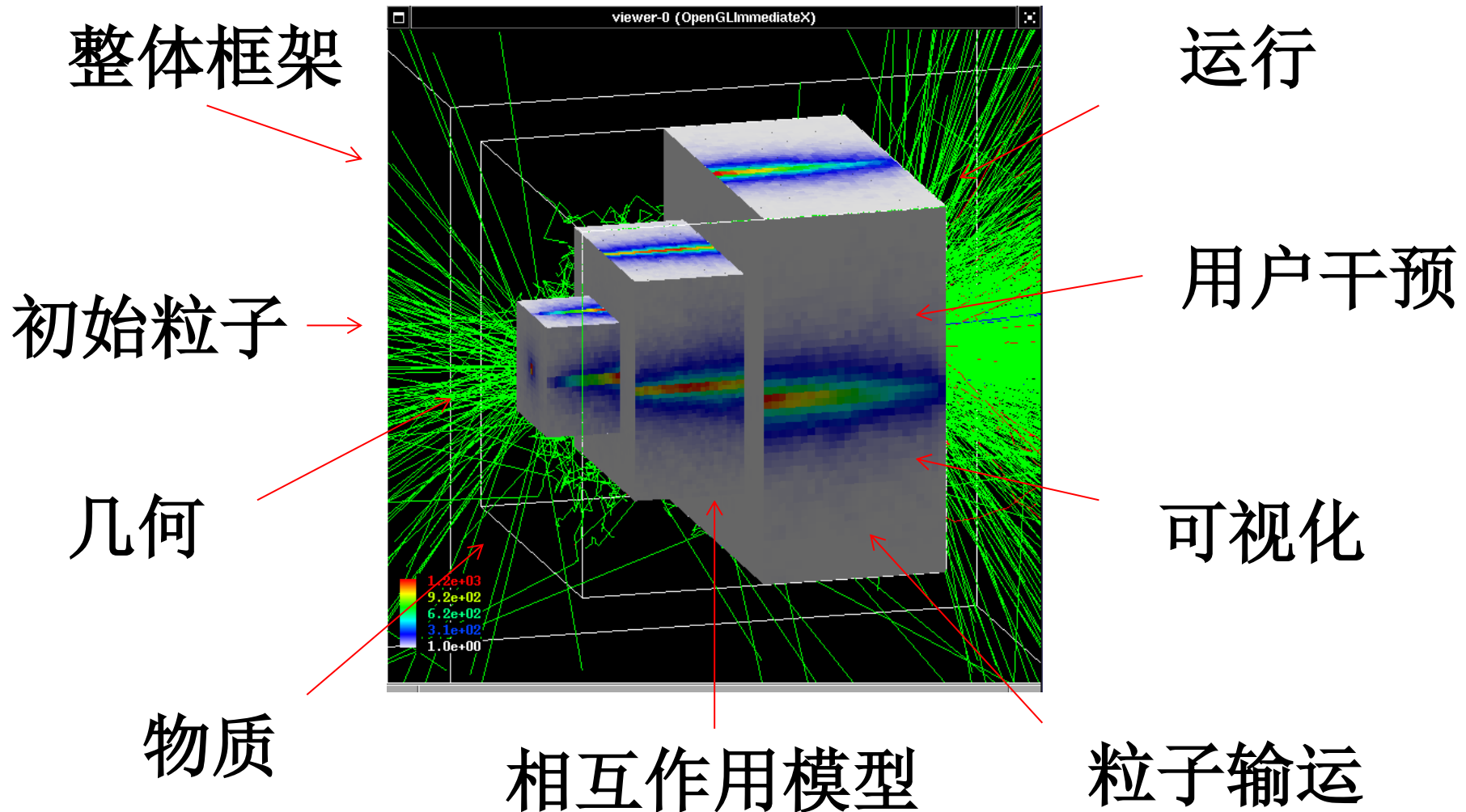
- Geant4提供了
 - 一系列头文件，例如
`G4TrackingManager.hh`
 - 它里面定义了一系列功能，或者说函数，例如
`void SetUserAction(G4UserTrackingAction* apAction);`
`void SetUserAction(G4UserSteppingAction* apAction);`
- 还有可执行的二进制代码（编译链接生成）
`libG4trcking.so`
- 用户需要做的是在自己的源代码中include头文件，编译的过程中link库文件
- ▶ 上述文件为完成了一个商业工具包（**toolkit**）
- 1. Geant4还公布了执行细节（开源与非开源的区别）
`G4TrackingManager.cc`



寻找帮助（还有很多）

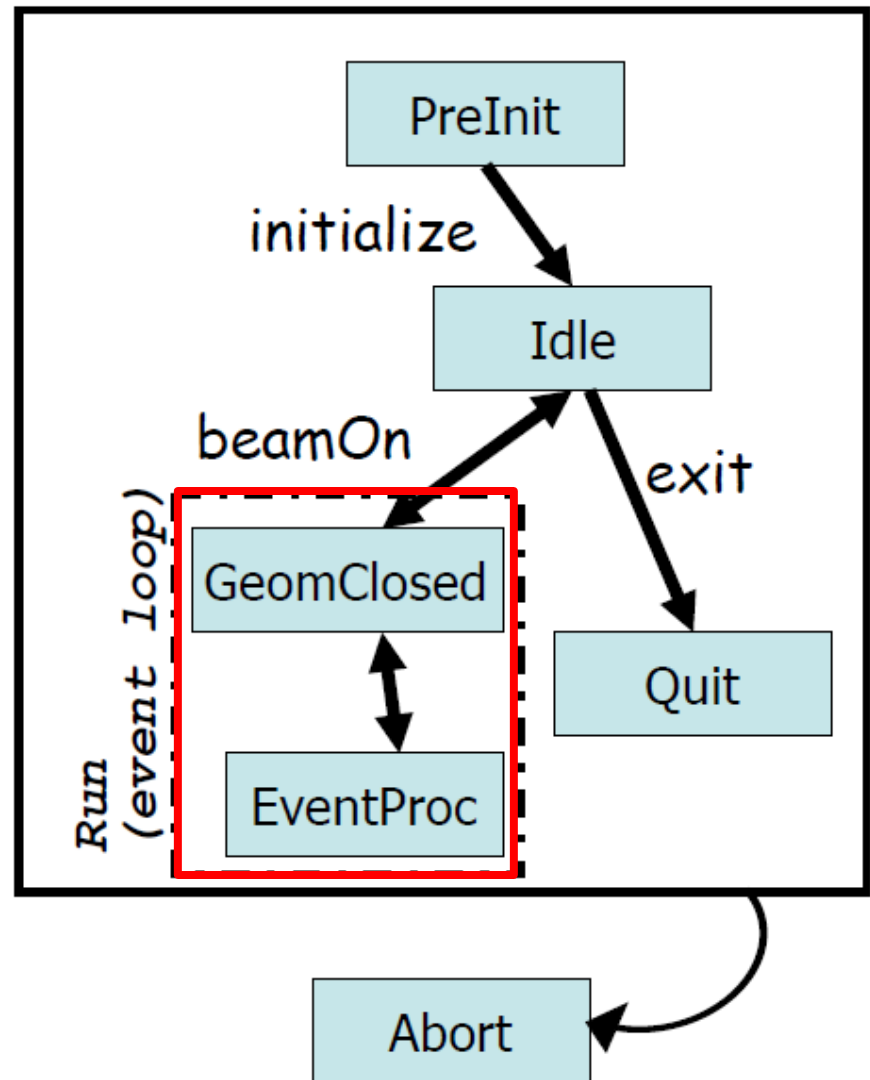
1. 主页 <http://geant4.cern.ch>
2. 用户支持<http://geant4.cern.ch/support/index.shtml>
3. 在上面的链接可以找到
[Application Developers Guide](#)
基于**Geant4**的应用开发指南 （用户）
[Toolkit Developers Guide](#)
Geant4工具包开发指南 （开发者）
[Physics Reference Manual](#)
物理参考手册
4. **Geant4**论坛
<http://hypernews.slac.stanford.edu/HyperNews/geant4/cindex>
5. **Geant4@SLAC**
<http://www-public.slac.stanford.edu/geant4/>

Geant4使用的几个重要方面



整体结构

- ▶ **PreInit** – 材料，几何，粒子，物理过程
- ▶ **Idle** – 可以开始
- ▶ **GeomClosed** – 几何关闭
- ▶ **EventProc** – 事例循环
- ▶ **Quit** – 正常退出
- ▶ **Abort** – 异常中断





Run, Event, Track, Step

Geant4继承了加速器实验概念

- ▶ Run来自于加速器实验的概念，指束流打开，采集一段时间的数据
- ▶ 对于对撞机物理，或打固定靶实验，截面足够小，或电子学足够快，基本保证每次电子学得到触发信号，只能够采集到一次对撞，称为Event
- ▶ 对撞后生成许多新粒子，每个粒子可以是一根径迹，称为Track
- ▶ Step为Track上的步，可以认为是一个粒子的两次相互作用之间一段距离

Run和Event的概念在现代有些模糊，例如一个暗物质实验，没有打开束流的概念；Event的概念也不清晰，中微子实验，核子对撞实验，经常有延迟信号，并和各种本底在时间上交叠。



温习：C++语法，虚函数

- ▶ 虚函数是C++中用于实现多态(polymorphism)的机制
- ▶ 翻译一：通过基类访问派生类定义的函数
- ▶ 翻译二：我用来规范派生类的行为，实际上就是所谓的“接口”。告诉使用者，我的派生类都会有这个函数。

```
class A
{
public:
    virtual void foo() { cout<<"A::foo() called."<<endl; }
};

class B: public A
{
public:
    virtual void foo() { cout<<"B::foo() called."<<endl; }
};

main()
{
    A* ptr = new B;
    ptr->foo();
}
```

会发现输出是B::foo() called.



最基本的主函数 (N01)

```
#include "G4RunManager.hh"  
#include "G4UImanager.hh"  
#include "ExN01DetectorConstruction.hh"  
#include "ExN01PhysicsList.hh"  
#include "ExN01PrimaryGeneratorAction.hh"
```

```
int main()  
{
```

```
// 1. 构造默认的运行管理器(强制)
```

```
G4RunManager* runManager = new G4RunManager;
```

```
// 2. 初始化探测器模型(强制)
```

```
G4VUserDetectorConstruction* detector = new ExN01DetectorConstruction;
```

```
runManager->SetUserInitialization(detector);
```

```
// 3. 初始化物理模型(强制)
```

```
G4VUserPhysicsList* physics = new ExN01PhysicsList;
```

```
runManager->SetUserInitialization(physics);
```

```
// 4. 设置起始粒子(强制)
```

```
G4VUserPrimaryGeneratorAction* gen_action = new ExN01PrimaryGeneratorAction;
```

```
runManager->SetUserAction(gen_action);
```

```
//.....接下页
```

例子N01课程附件
一起来读

问题. 这个过程是不是和你想象的是一致的? 是不是和你的直觉是一致的? 这个过程还缺少什么?



最基本的主函数(N01)

一起来读

```
// 接上页...  
// 5. 初始化G4 内核(强制)  
runManager->Initialize();
```

问题. 这个过程是不是和你想象的是一致的? 是不是和你的直觉是一致的?

```
// 获取UI管理器的指针, 并设置verbosities。  
G4UImanager* UI = G4UImanager::GetUIpointer();  
UI->ApplyCommand("/run/verbose 1");  
UI->ApplyCommand("/event/verbose 1");  
UI->ApplyCommand("/tracking/verbose 1");
```

```
// 6. 开始运行(强制)  
G4int numberOfEvent = 3;  
runManager->BeamOn(numberOfEvent);
```

```
// 7. 结束作业(强制)  
// 释放内存: 用户行为、物理过程以及探测器描述属于运行管理器,  
// 将被运行管理器自动删除, 所以不应该在主函数中删除之。  
// 只需要删除运行管理器和其它动态指针即可。  
delete runManager;
```

```
return 0;  
}
```



ExN01DetectorConstruction的关键部分

```
// 1. 头文件
#include "G4VUserDetectorConstruction.hh"

// 2. 虚继承
class ExN01DetectorConstruction : public
    G4VUserDetectorConstruction
{
public:

    ExN01DetectorConstruction();
    ~ExN01DetectorConstruction();
    // 3. 从虚基类继承的功能性函数
    G4VPhysicalVolume* Construct();

private:
    // .....

};
#endif
```

大家一起来找到G4VUserDetectorConstruction.hh,
并检查一下是不是该基类中有Construct()的定义?

另外我们还可以一起看下G4RunManager.cc,
在这里我们是不是可以找到
userDetector->Construct()?

是不是明白Geant4是如何运行的了?



ExN01DetectorConstruction的执行

```
#include "ExN01DetectorConstruction.hh"
```

```
ExN01DetectorConstruction::ExN01DetectorConstruction() {...}
```

```
ExN01DetectorConstruction::~~ExN01DetectorConstruction() {...}
```

// 从虚基类继承的功能性函数，实际执行了用户的意图

```
G4VPhysicalVolume* ExN01DetectorConstruction::Construct()
```

```
{
```

```
    G4double a; // atomic mass
```

```
    G4double z; // atomic number
```

```
    G4double density;
```

```
    G4Material* Ar =
```

```
    new G4Material("ArgonGas", z= 18., a= 39.95*g/mole, density=
    1.782*mg/cm3);
```

```
    ...
```

```
    return experimentalHall_phys;
```

```
}
```


几何，物质各类的概念层次

- Solid: 几何, Box, Tube...

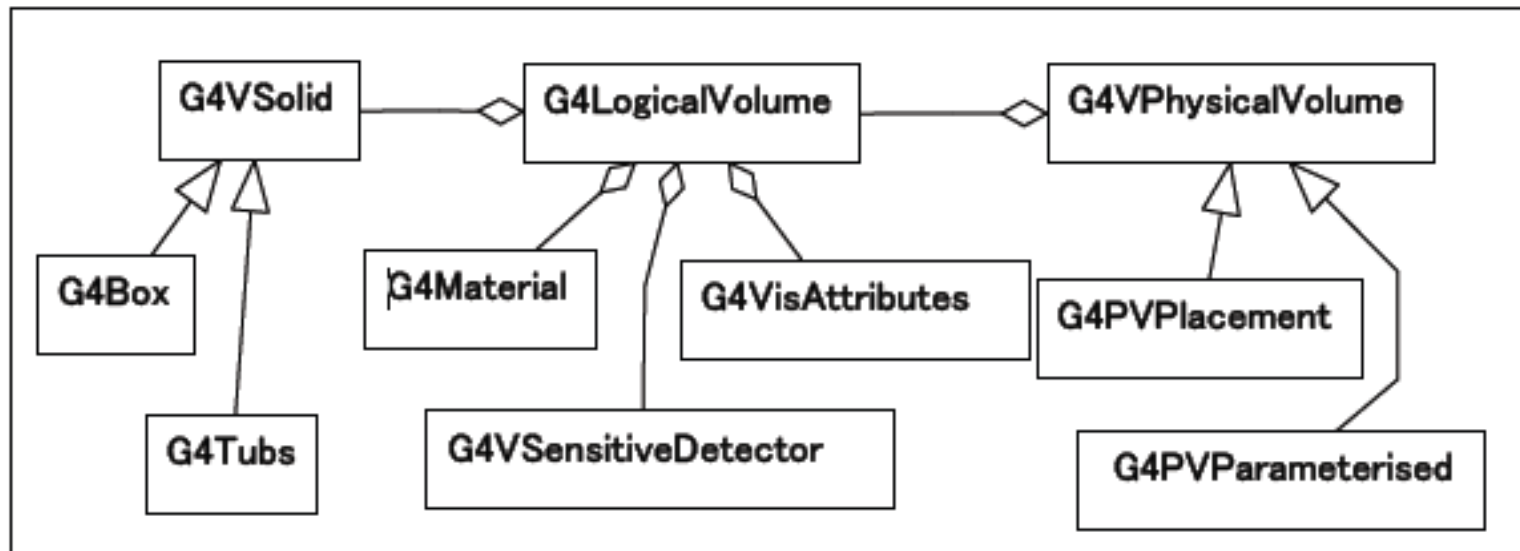
Solid描述了几何的抽象概念，无绝对坐标，无物质填充，与粒子无任何关系

- LogicVolume: 逻辑体

逻辑体和物质，可视化属性关联，可定义为敏感探测器

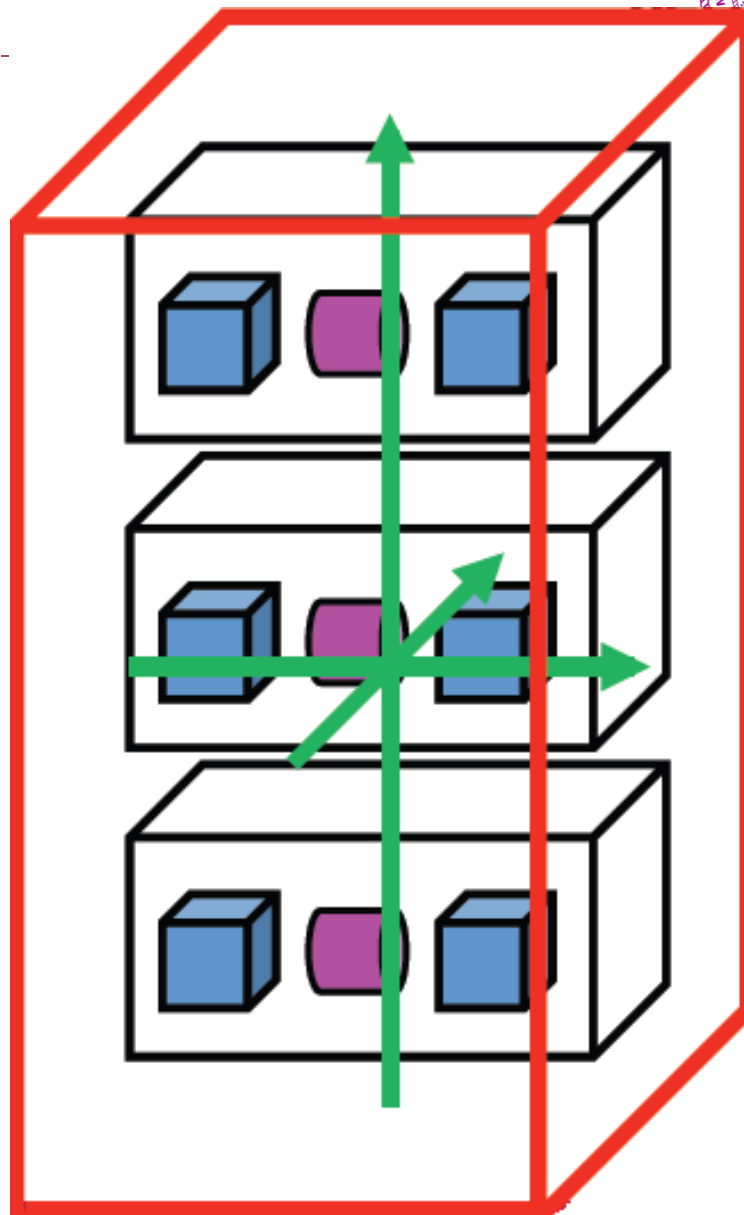
- PhysicalVolume: 实体

可复制，旋转，参数化，被放置到绝对坐标的逻辑体



几何的层次

- ▶ 第一个为World，包含全部子探测器
- ▶ 其他的子探测器可以放在World之中
- ▶ 子探测器中可以继续做母探测器，放入子探测器，形成一系列包含关系
- ▶ 放置时声明的相对位置为相对母探测器中心，比如Box的几何中心





物质的声明与定义

- ▶ 对微观粒子来讲，分子结构及以上的尺度的结构是不敏感的
- ▶ 对原子该尺度以下的结构，原则上已知质子数，中子数，电子数，微观粒子和物质的相互作用已经确定
- ▶ 对于一些特殊物质，如等离子体，应注意定制

```
double density = 1.390*g/cm3;//密度
double a = 39.95*g/mole;      //原子量
double z=18.;                 //原子序数
G4Material* lAr = new G4Material("liquidArgon",z,a,density);
```

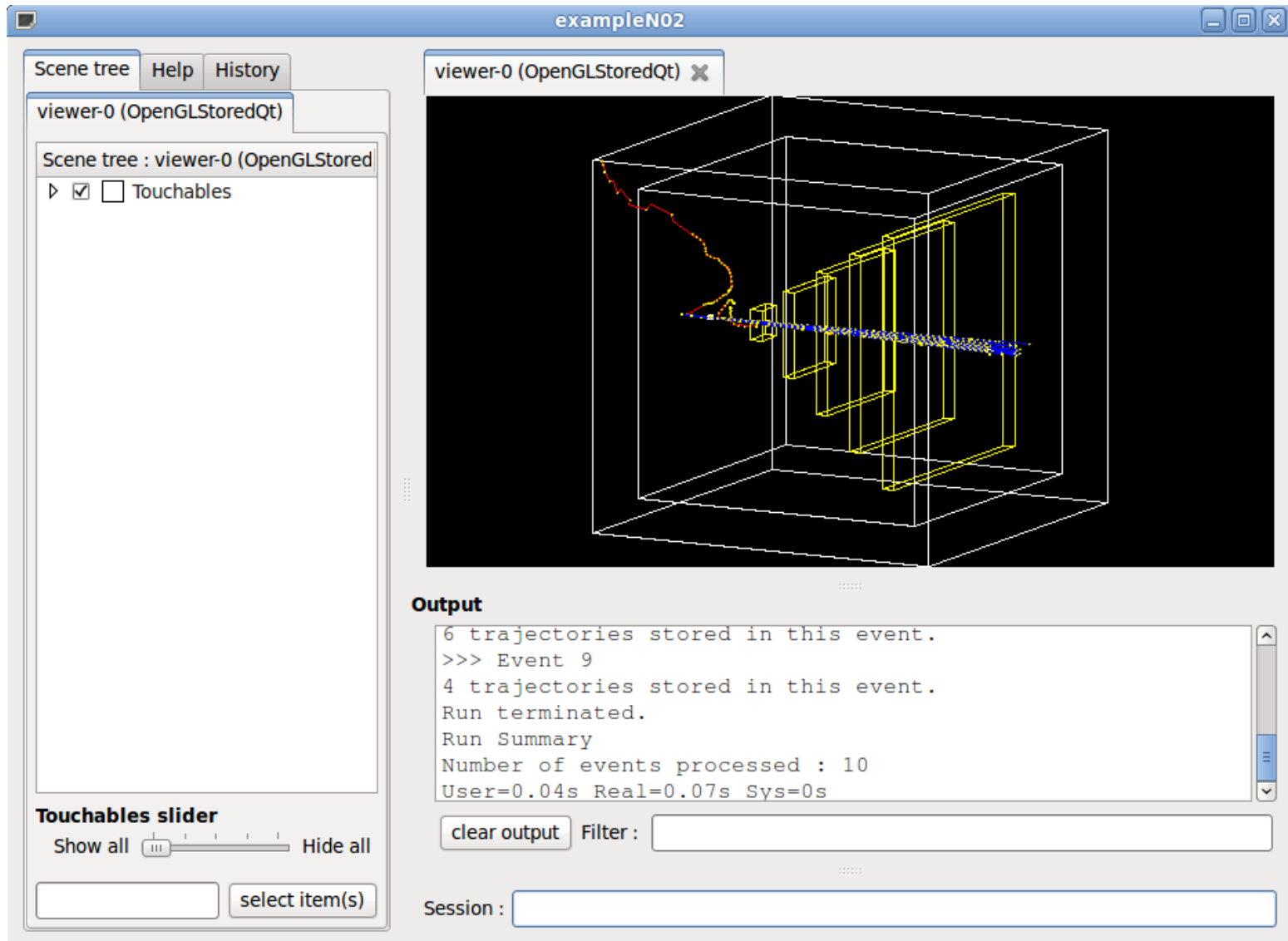
```
G4Element* H = new G4Element(name="Hydrogen",symbol="H" , z= 1.,
a);
G4Element* O = new G4Element(name="Oxygen" ,symbol="O" , z= 8., a);
density = 1.000*g/cm3;
G4Material* H2O = new G4Material(name="Water", density,
ncomponents=2);
H2O->AddElement(H, natoms=2);
H2O->AddElement(O, natoms=1); //定义水，给定密度、元素种类数目、添加元素
```



最简单的几何定义例子

- 定义一个方形的实验厅，里面充满了氩气

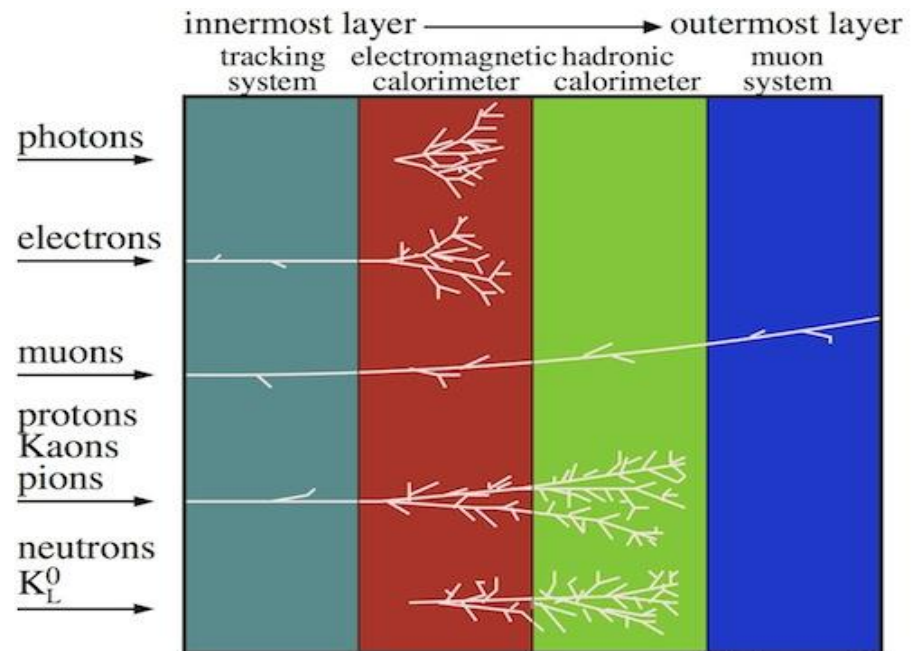
```
G4double expHall_x = 3.0*m;  
G4double expHall_y = 1.0*m;  
G4double expHall_z = 1.0*m;  
//Solid，指定几何形状和尺寸  
G4Box* experimentalHall_box  
    = new G4Box("expHall_box",expHall_x,expHall_y,expHall_z);  
  
//Logical，指定具体物理特性，如其中物质为Ar气  
experimentalHall_log = new G4LogicalVolume(experimentalHall_box,  
                                            Ar,"expHall_log",0,0,0);  
  
//Physical，指定放置位置以及旋转角度等  
experimentalHall_phys = new G4PVPlacement(0,G4ThreeVector(),  
                                            experimentalHall_log,"expHall",0,false,0);
```



几个用户宏命令

- ▶ Interactive模式 vs. Batch模式（交互式 vs. 批处理）
- ▶ 常用的几个命令：
/vis/open OGL
/vis/draw Volume
/gun/particle mu+
 （其他粒子，gamma, proton, e+, neutron,
 photon, pi）
/run/beamOn 10
/gun/energy 10 GeV

虚拟实验
什么样的厚度合适？

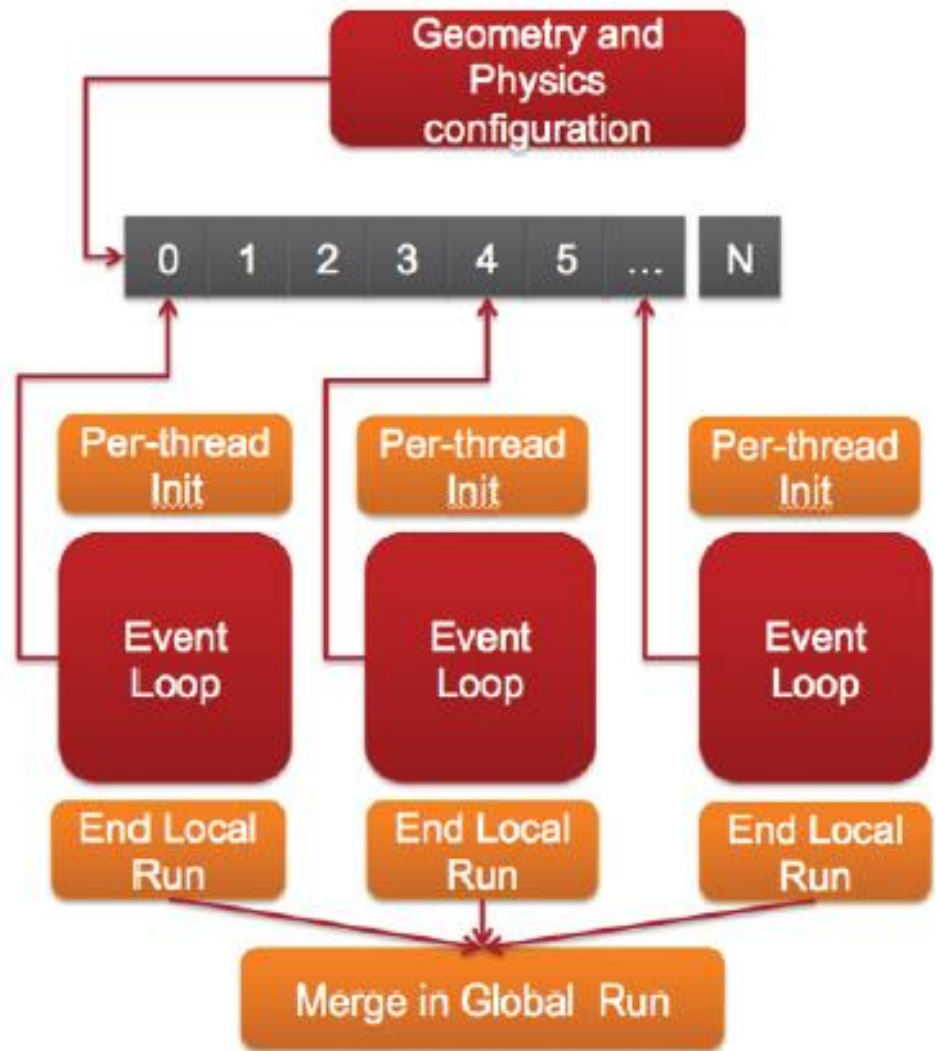


C. Lippmann – 2003

- ▶ 现代电脑，（个人笔记本）
 1. 经常包括多个子核，并支持多线程
 2. 多内存，32G
- ▶ 而如果一个Geant4作业只能使用一个线程，只能使用2G内存，显然浪费了很多的资源。
- ▶ 多线程是解决该问题的首选方案
- ▶ 在安装阶段，我们已经使用了`make -j 4`

多线程的基本思路

- ▶ 预先准备好随机种子的序列
- ▶ 每个线程竞争完成全部的模拟事例
- ▶ 最后将生成的结果组合在一起





使用多线程

- ▶ **cmake时加入**
-DGEANT4_BUILD_MULTITHREADED=ON
- ▶ **使用时:**
 1. **/run/numberOfThreads**
 2. 或者 **G4MTRunManager::SetNumberOfThreads()**
 3. 或者 环境变量
G4FORCENUMBEROFTHREADS=...
(可以用**max**代表系统允许的最大值)



多线程的问题

- ▶ 用户程序必须有清晰的Event概念，如需要事例混合，前后关联，则用户一定要把这些关系处理好
- ▶ 有一定的随机性，因为多线程的竞争能力受其他因素影响
- ▶ 不能保证两次运行的结果全同，所以错误也有随机性，不容易差错，由多线程引起的错误很难找到
- ▶ 在程序调试成熟后可以尝试使用



Geant4例子的运行方法

- > cd `path_to_Geant4_installation`/share/Geant4- \ 10.2.1/geant4make
- > source geant4make.csh

注意替换 `path_to_Geant4_installation`
参考例子 `g4.csh`

- > cd `path_to_exampleXYZ` \ 即例子的父目录，注意替换
- > mkdir `exampleXYZ_build` \ 与例子的目录平行
- > cd `exampleXYZ_build`
- > cmake ../exampleXYZ
- > make
- > ./exampleXYZ



作业：开始N01例子

- ▶ 安装GEANT4
- ▶ 尝试读懂N01
- ▶ 目标：找到在哪里程序设定了几何形状和物质



作业：练习运行N01

► 编译

► 运行

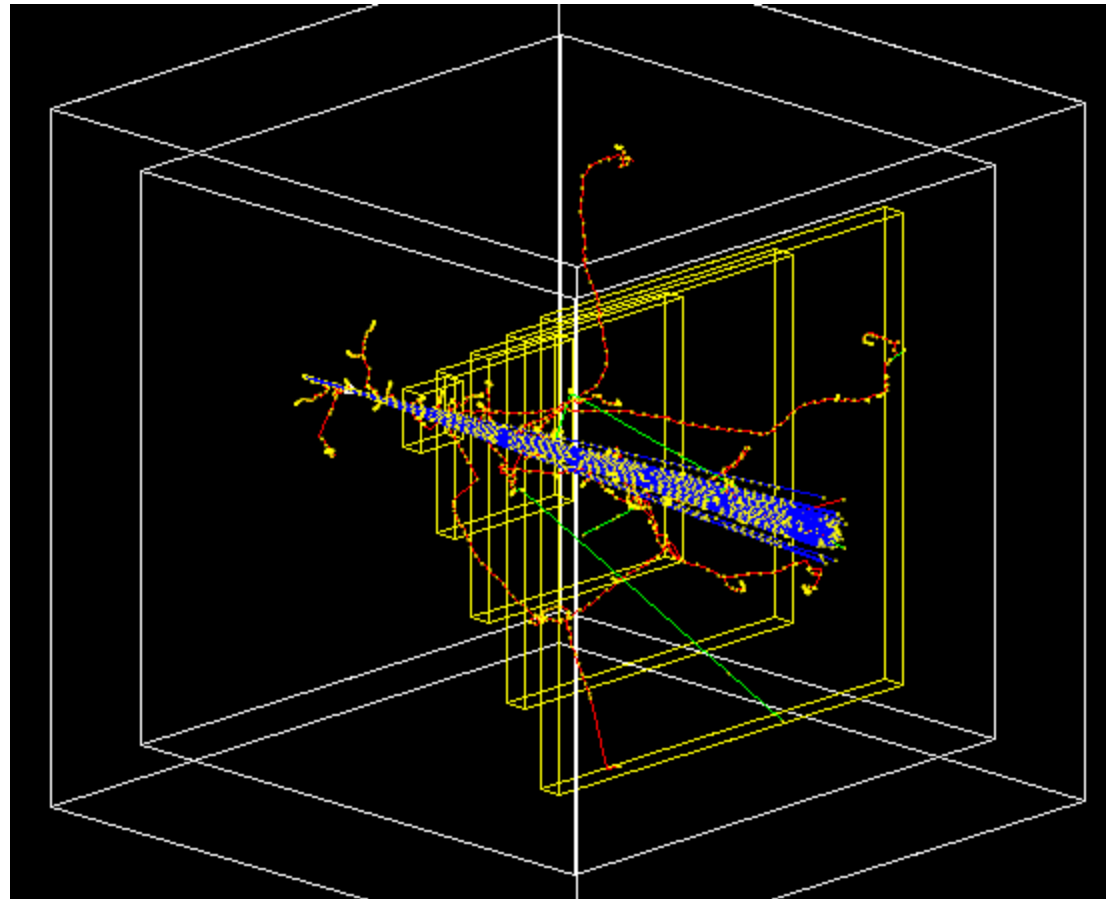
将看到类似如下信息：

```
*****
* G4Track Information: Particle = geantino, Track ID = 1, Parent ID = 0
*****
```

Step#	X(mm)	Y(mm)	Z(mm)	KinE(MeV)	dE(MeV)	StepLeng	TrackLeng	NextVolume	ProcName
0	-2e+03	0	0	1e+03	0	0	0	expHall	initStep
1	-1.6e+03	0	40	1e+03	0	402	402	tracker	Transportation
2	-400	0	160	1e+03	0	1.21e+03	1.61e+03	expHall	Transportation
3	-7.27e-14	0	200	1e+03	0	402	2.01e+03	caloBlock	Transportation
4	90	0	209	1e+03	0	90.4	2.1e+03	caloLayer	Transportation
5	110	0	211	1e+03	0	20.1	2.12e+03	caloBlock	Transportation
6	190	0	219	1e+03	0	80.4	2.2e+03	caloLayer	Transportation
7	210	0	221	1e+03	0	20.1	2.22e+03	caloBlock	Transportation
8	290	0	229	1e+03	0	80.4	2.3e+03	caloLayer	Transportation
9	310	0	231	1e+03	0	20.1	2.32e+03	caloBlock	Transportation
10	390	0	239	1e+03	0	80.4	2.4e+03	caloLayer	Transportation
11	410	0	241	1e+03	0	20.1	2.42e+03	caloBlock	Transportation
12	490	0	249	1e+03	0	80.4	2.5e+03	caloLayer	Transportation
13	510	0	251	1e+03	0	20.1	2.52e+03	caloBlock	Transportation

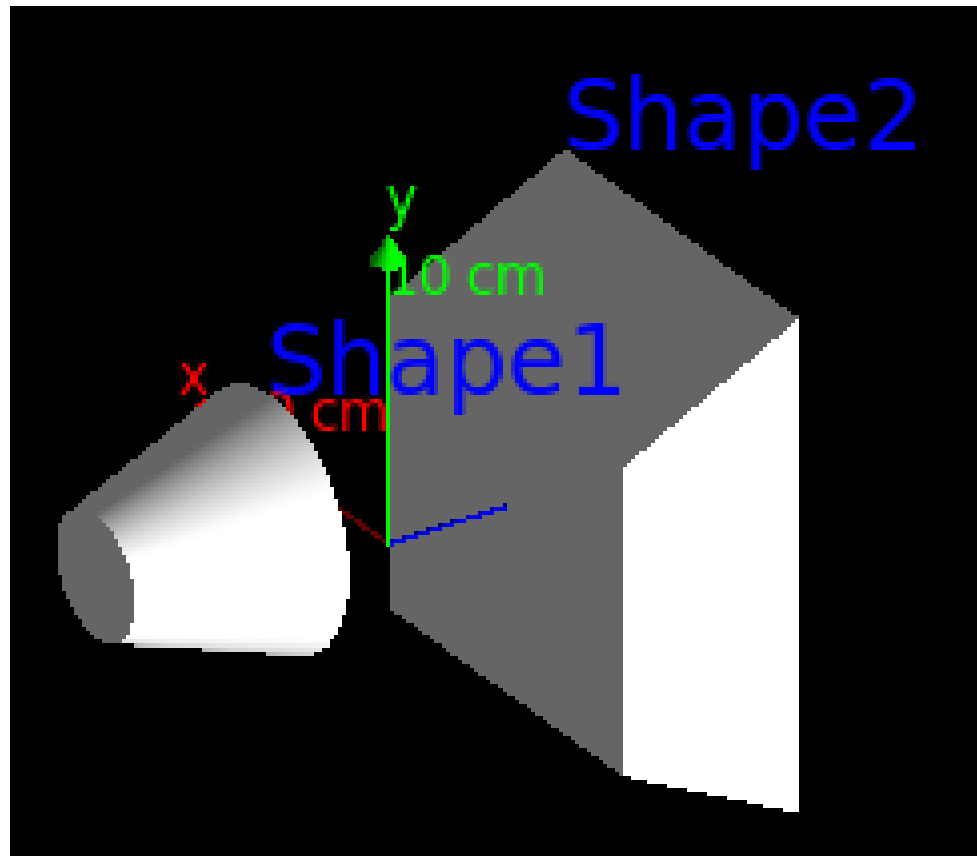
作业：练习运行N02

- ▶ 编译
- ▶ 运行，将看到这样的结果
- ▶ 观察打入不同粒子的结果：
mu+, gamma
proton, neutron



作业：练习运行B1

- ▶ 编译
- ▶ 运行，将看到这样的结果





作业：练习调整N02，B1的事例显示

- ▶ 角度旋转
- ▶ 生成postscript矢量图文件
矢量图：[矢量图](#)也称为[面向对象](#)的图像或绘图图像，是计算机图形学中用点、[直线](#)或者多边形等基于[数学](#)方程的几何图元表示图像。
- ▶ [矢量图](#)形最大的优点是无论放大、缩小或旋转等不会失真。一般[发表文章](#)时必须使用矢量图。

提示，这里要去看一下各种资源，尝试自己找到帮助。



继续研究之前的一个作业题

- ▶ 产生1000个有100个事例的高斯分布, $\mu=0$, $\sigma=1$
 1. 用高斯函数来拟合, 全部参数自由, 记录每次的拟合结果, μ , σ , $\text{chi2}/\text{ndf}$
 - 画上述三个变量的分布
 2. 仍用高斯函数拟合, 固定 $\mu=0.05$, σ 自由, 记录得到的 σ , $\text{chi2}/\text{ndf}$
 - 画上述两个变量的分布
 3. 固定 $\mu=0.2$, σ 自由, 记录得到的 σ , $\text{chi2}/\text{ndf}$
 4. 固定 $\mu=0.5$, σ 自由, 记录得到的 σ , $\text{chi2}/\text{ndf}$
- ▶ 对比上述四种拟合的 σ , $\text{chi2}/\text{ndf}$ 的分布, 画在一起