探测器模拟基础

曹国富

中国科学院高能物理研究所

第十四届南山清北粒子物理暑期学校

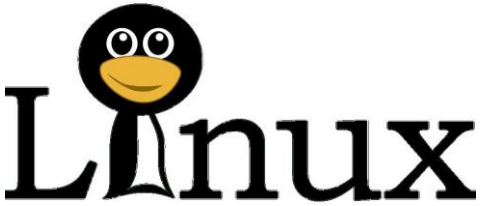
2026.07.25

课程安排

- 探测器模拟原理、方法和对象
- GEANT4模拟工具介绍和使用
- 上机演示

基本要求

- 计算机基础
- C++
- Linux
- 探测器基础
- ROOT基础
- Monte Carlo方法



❄ 设计并实现一个探测器模拟程序，实现以下功能：

- 构建一个内直径37m，壁厚20cm的有机玻璃球，球内装满液闪。整个球体置于边长40m的立方体中，球心与立方体中心重合，立方体内的物质为水。
  - 有机玻璃→ 密度: 1.18g/cm3; C: 59.984%; H: 8.055%; O: 31.961%
  - 液闪→ 密度： 0.859g/cm3; C: 87.924%; H: 12.01%; O: 0.034%; N: 0.027%; S: 0.005%
- 在上述球体内产生10万个IBD事例，事例顶点均匀分布在球体内。
- 请将模拟过程中的以下信息存储到ROOT文件中：
  - 事例编号
  - IBD产生顶点 (x, y, z)
  - 中微子初始能量 (nu_energy)
  - 正电子初始能量 (eplus_energy)
  - 中子俘获时间 (capture_time)
  - 快信号沉积能量 (prompt_depE)
  - 慢信号沉积能量 (delayed_depE)
  - 快信号沉积能量 (加入能量分辨3%/sqrt(E)，prompt_smearedE)
  - 慢信号沉积能量 (加入能量分辨3%/sqrt(E)，delayed_smearedE)

探测器模拟原理、方法和对象

什么是探测器模拟?

一句话总结什么是探测器模拟

OI **openai/gpt-5**

探测器模拟是指在计算机中基于物理模型与数值方法，仿真粒子与探测器材料及电子学的相互作用与响应，以预测可观测信号并用于装置设计优化与数据分析校正的过程。

OI **hepai/deepseek-v3.2**

探测器模拟是利用计算机软件对探测器物理过程（如粒子相互作用、信号生成与处理）进行数值仿真的技术，以预测或复现其实验响应。

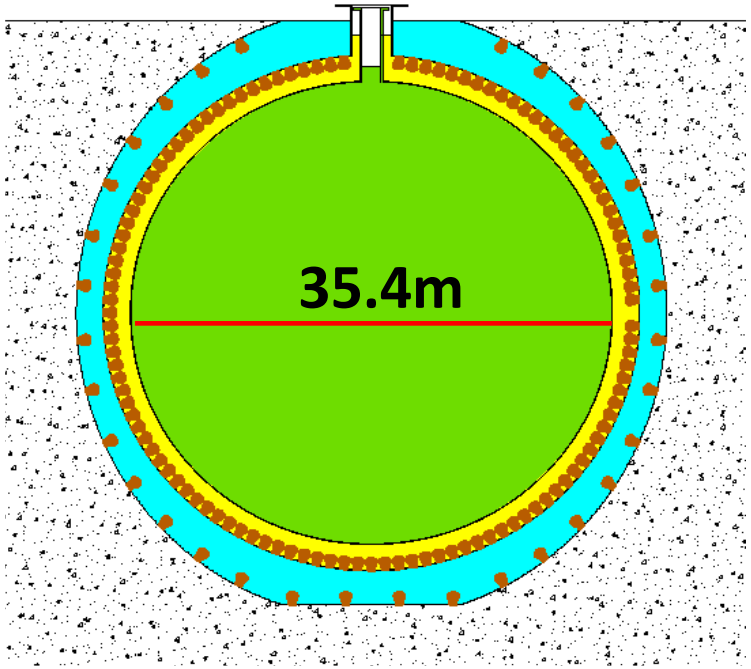
OI **aliyun/qwen3.6-plus**

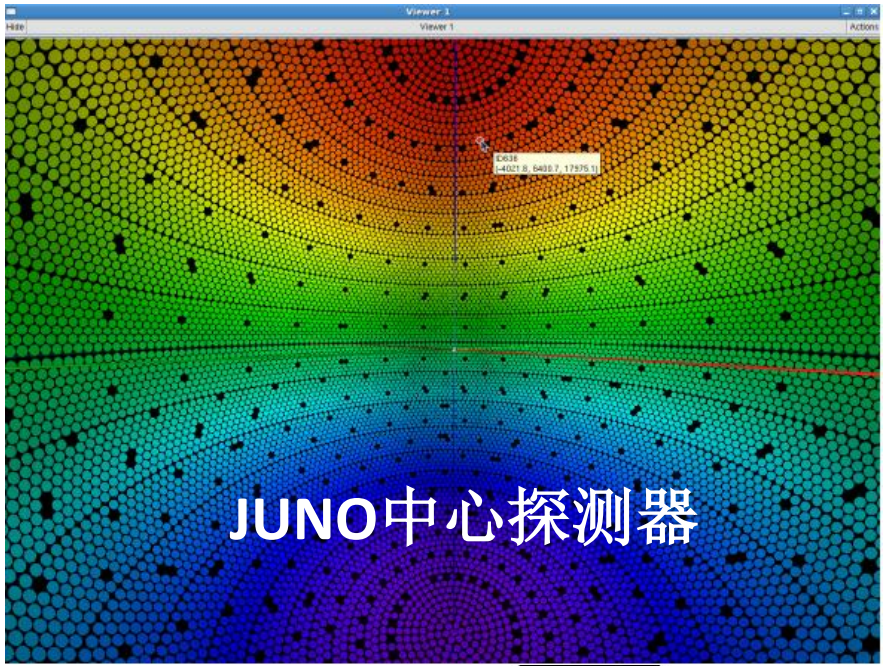
思考用时 16 秒

探测器模拟是利用计算模型与物理规律，虚拟复现探测器对入射信号的响应过程，以辅助设计优化、性能评估与数据分析。

什么是探测器模拟?

- 探测器模拟是把一个真实探测器或一个假想探测器在计算机中实现的过程。





- 探测器模拟是虚拟实验，已有理论、模型、经验和实验数据是探测器模拟的基础。
- 粒子与物质相互作用是探测器模拟的核心。
- 深刻理解探测器是做好探测器模拟的关键。

探测器模拟与实验间的关系

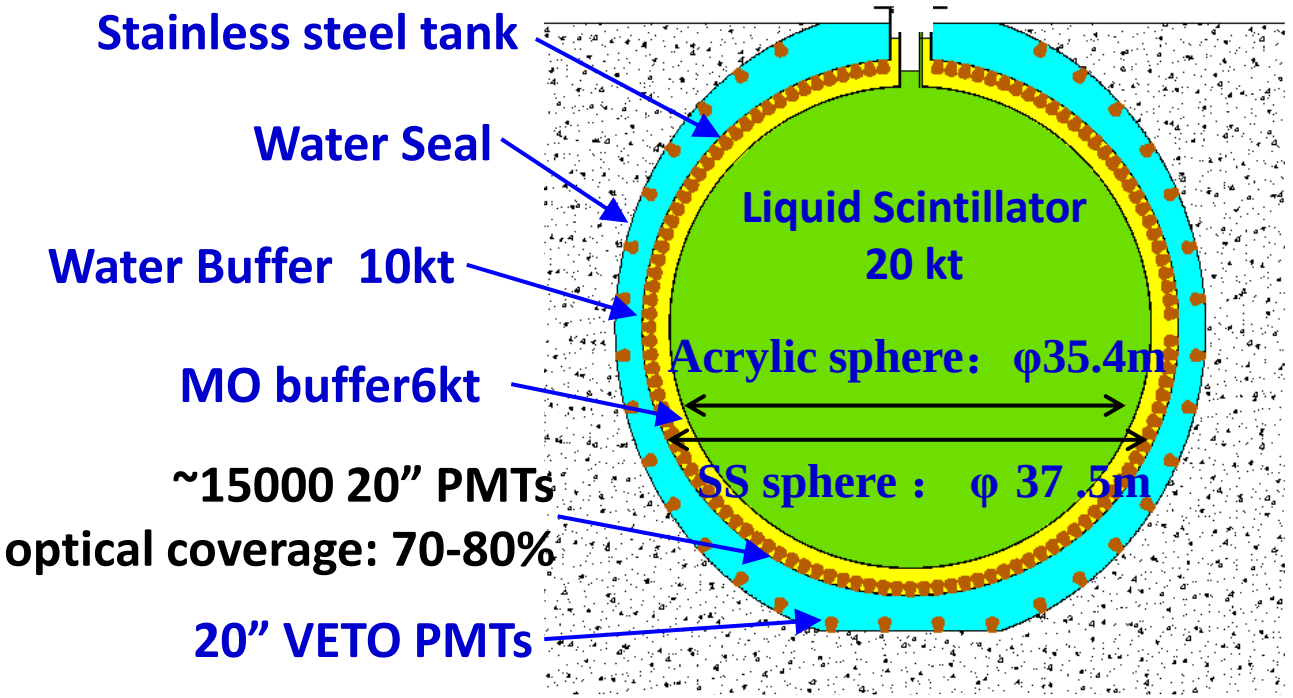
- 模拟是工具、手段、方法。
- 模拟无法替代实验，更无法替代理论。
- 模拟结果具有指导性，是高能物理等实验中不可或缺的一个环节。
- 高精度的模拟结果在某些领域是必需的。
- 实验是检验模拟正确性的唯一标准。
- 模拟解决的是：“如果世界按我们的理解运行，会发生什么”
- 实验回答的是：“世界到底会不会按我们的理解运行”

探测器模拟的角色

❄ 探测器R&D阶段

- ➢ 从物理上比较探测器不同设计方案优劣的重要手段
- ➢ 确定探测器的各项设计参数的重要途径
- ➢ 研究探测器的预期性能指标

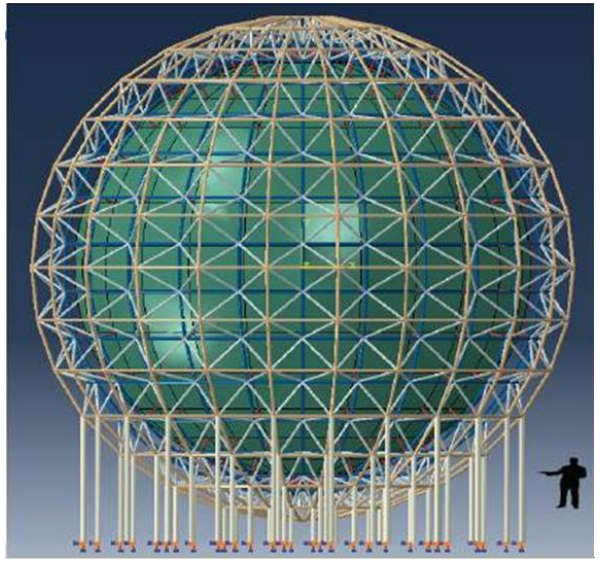
江门中微子实验(JUNO)探测器概念图



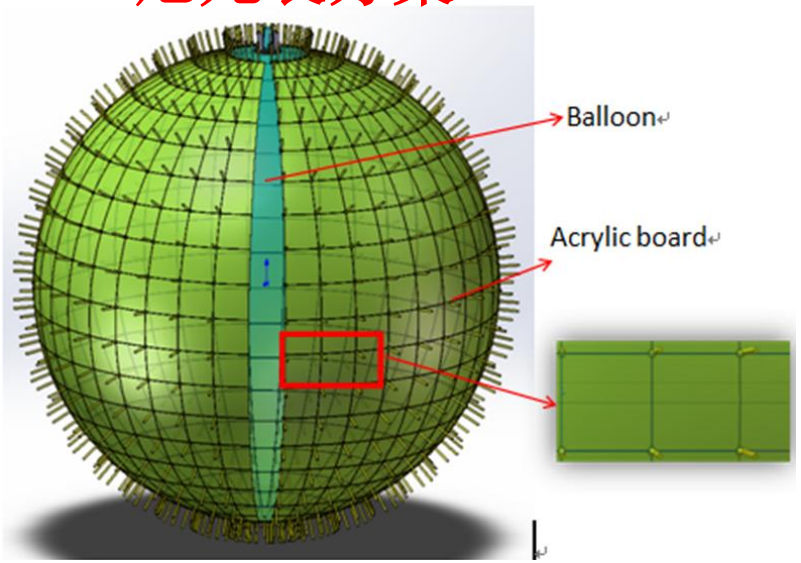
主要物理目标：
确定中微子质量等级

探测器主要性能指标：
能量分辨率 < 3%/sqrt(E)，
这对应于1MeV能量沉积
产生1200个光电子。

有机玻璃球方案



尼龙袋方案



选哪个好那？

探测器模拟结果显示：

1. 有机玻璃球方案的能量分辨率稍差。
2. 尼龙袋方案的本底略高。

最后，综合考虑模拟结果，机械设计和实验风险等因素，选择了有机玻璃球方案。

探测器设计参数

- 满足物理目标是探测器性能指标确定的基本要求。
- 达到探测器性能指标是设计探测器各项参数的前提。
- 以JUNO为例，通过探测器模拟确定了很多探测器设计参数：
  - 液闪衰减长度
  - 液闪光产额
  - PMT覆盖率
  - PMT探测效率
  - 水屏蔽层的厚度
  - ……
- 所有设计参数的确定必须综合考虑模拟结果，机械设计，成本，风险等因素。

探测器建造阶段

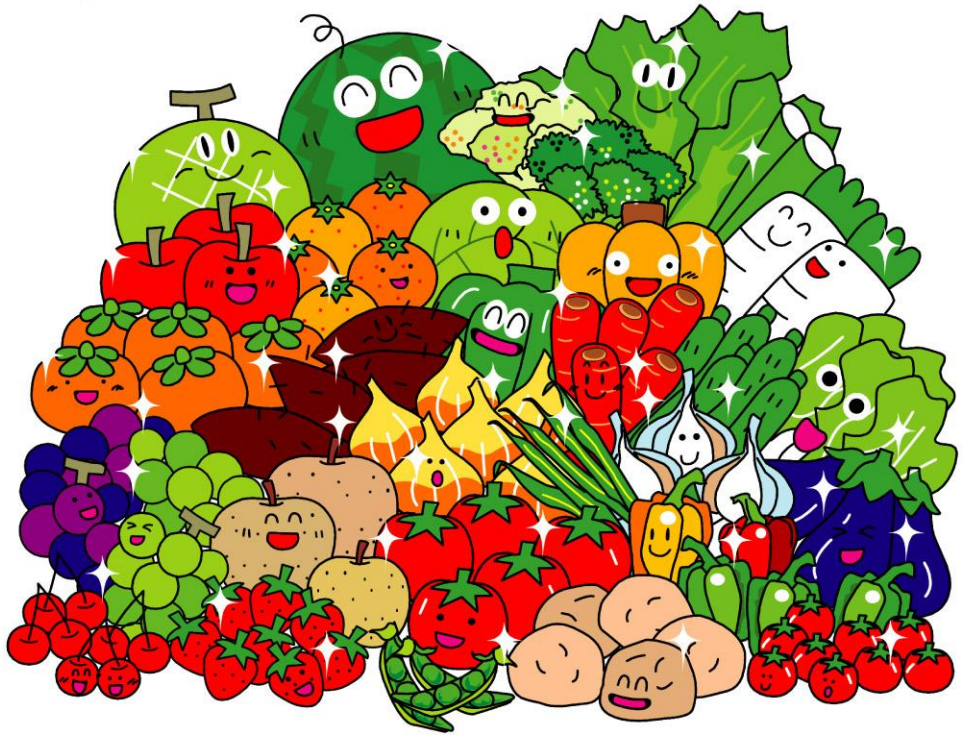
- **探测器模拟是离线软件数据处理系统的重要组成部分。**
  - 是重建、刻度等离线软件开发的前提和基础。
  - 检验、调试重建和刻度算法的重要手段。

- **在探测器建造的数年时间里，是完善和优化探测器模拟的重要契机，为实现精细探测器模拟做好准备。**
  - 完善和细化探测器几何和物质描述。
  - 利用一切可能的测量数据，确定模拟中的各种参数。
  - 考虑探测器性能随时间的变化，实现从静态、理想探测器模拟到探测器模拟真实化的转变。
  - 精细模拟电子学系统，触发判选系统和数据读出系统。
  - 检查模拟软件中的错误，优化模拟软件设计和运行速度。

探测器运行及数据分析阶段

- **模拟调节 (MC tuning)，目的是减少模拟与数据间的差别。**
  - **模拟调节←→模拟与数据对比**
- **借助探测器模拟理解探测器响应。**
- **在实验数据分析中，模拟在事例选择条件优化，本底分析，绝对探测效率确定和变量形状描述等方面起着关键性作用。**
- **探测器模拟在物理成果的发现中扮演者非常重要的角色。**

一个例子



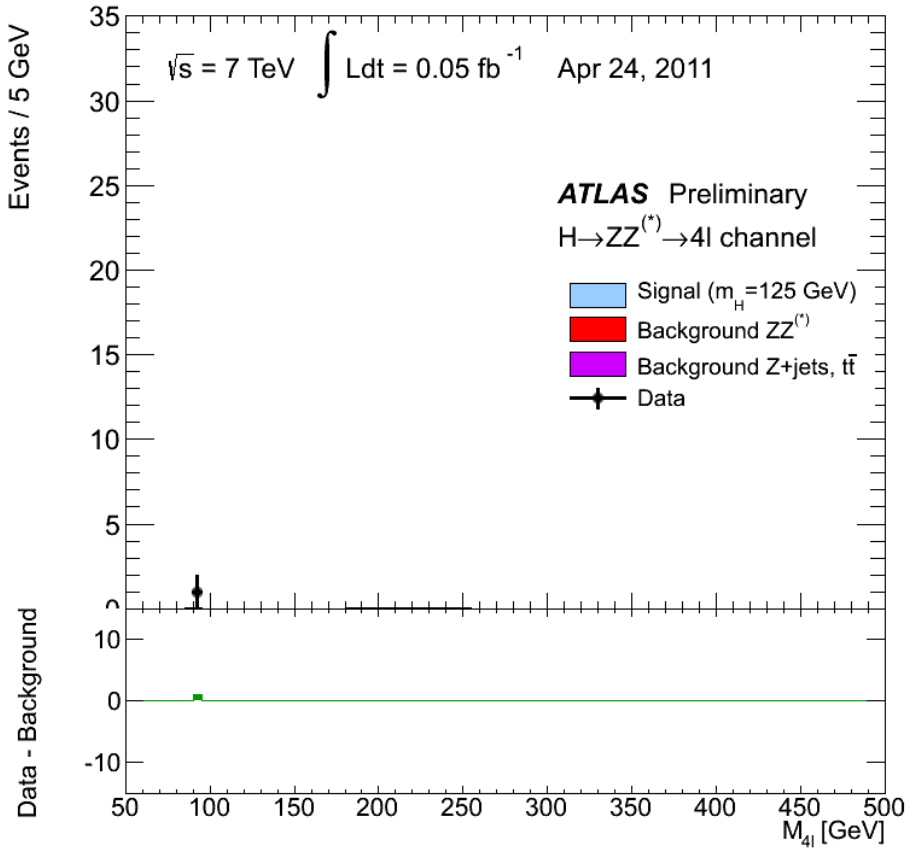
信号：黄瓜
本底：其它蔬菜、水果

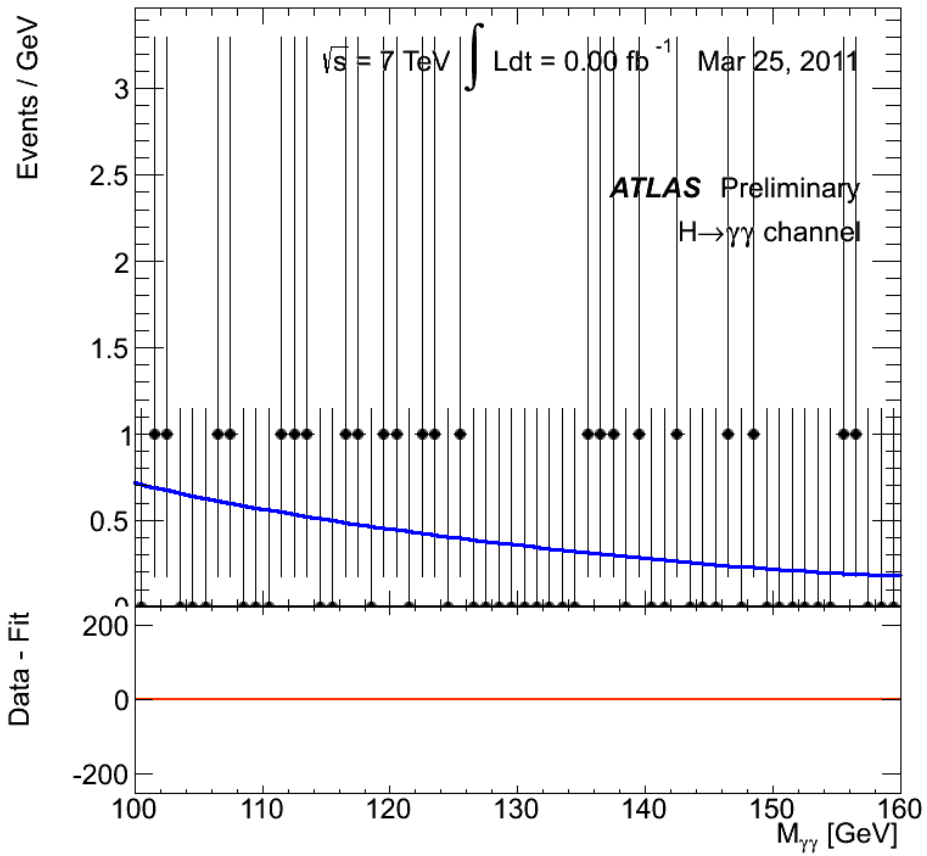
通过模拟研究信号事例和本底事例的特征。
a) 绿色
b) 细长
c) 瓜皮较厚
d) 表皮有刺
通过以上选择条件优化以后，黄瓜(信号)+少量丝瓜(本底)

通过模拟纯信号事例，得到选择效率。

通过模拟本底事例，分析通过选择条件的本底成分。

Higgs粒子(上帝粒子)寻找

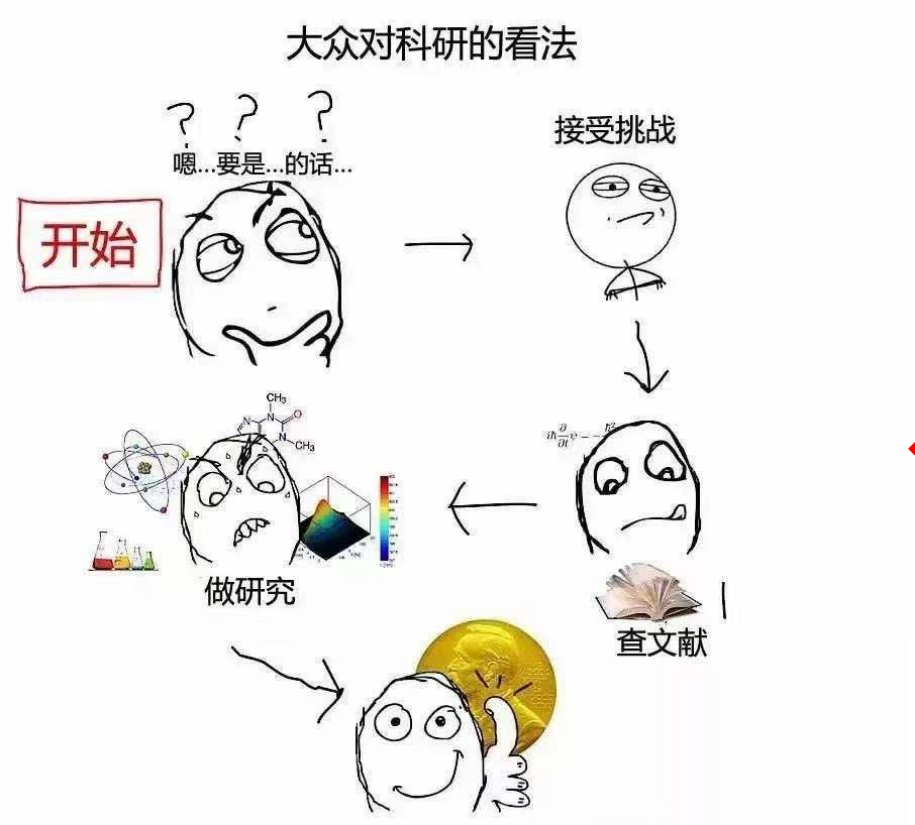




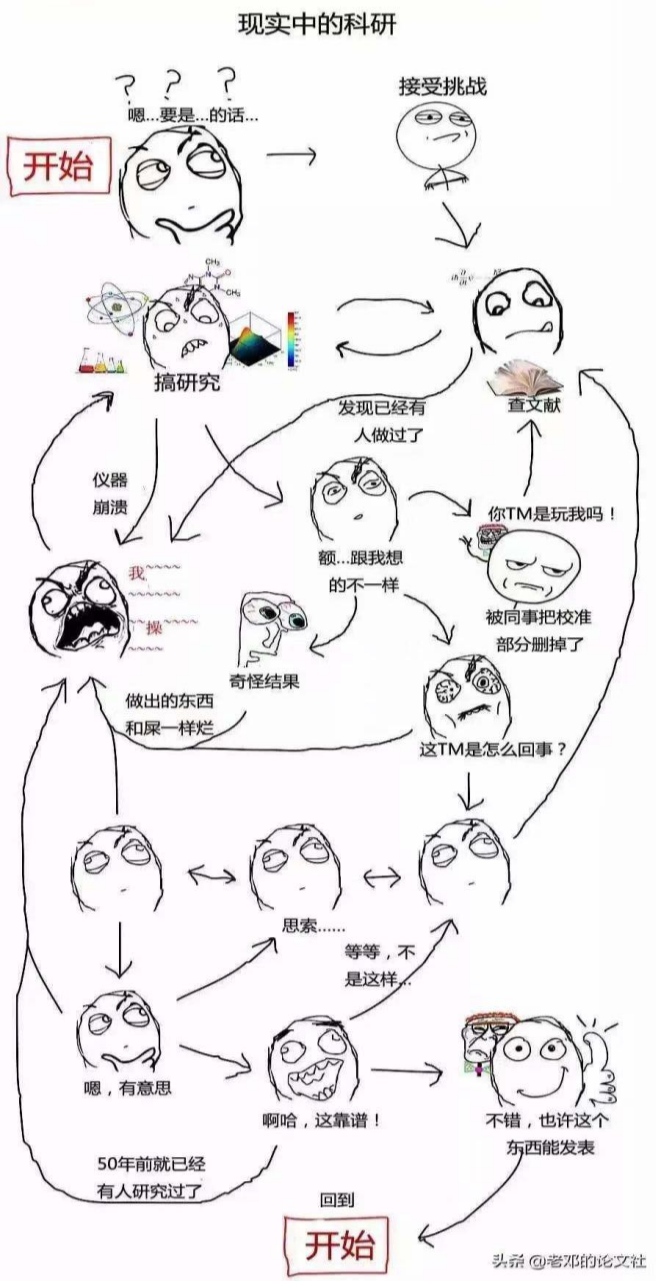
探测器模拟用于确定Higgs衰变到γγ末态的不变质量谱形状。

探测器模拟用于分析Higg寻找中的本底。

所以，模拟会助力你的科研工作



MC Simulation



探测器模拟的构成

❄ **一个完整的探测器模拟系统通常包括以下几部分：**

- **事例产生子**：产生进入探测器的粒子。
- **物质与几何**：构建探测
- **粒子与物质相互作用**：粒子在探测器中的传输和反应过程。
- **电子学系统模拟**：实现种物理量到电信号的转换，模拟电子学系统的响应。
- **触发判选系统模拟**：以电子学系统的输出为输入，模拟触发判选系统的触发逻辑，触发算法等。
- **数据读出系统模拟**：实现数据从电子学系统输出后的事例组装，数据存盘等过程。

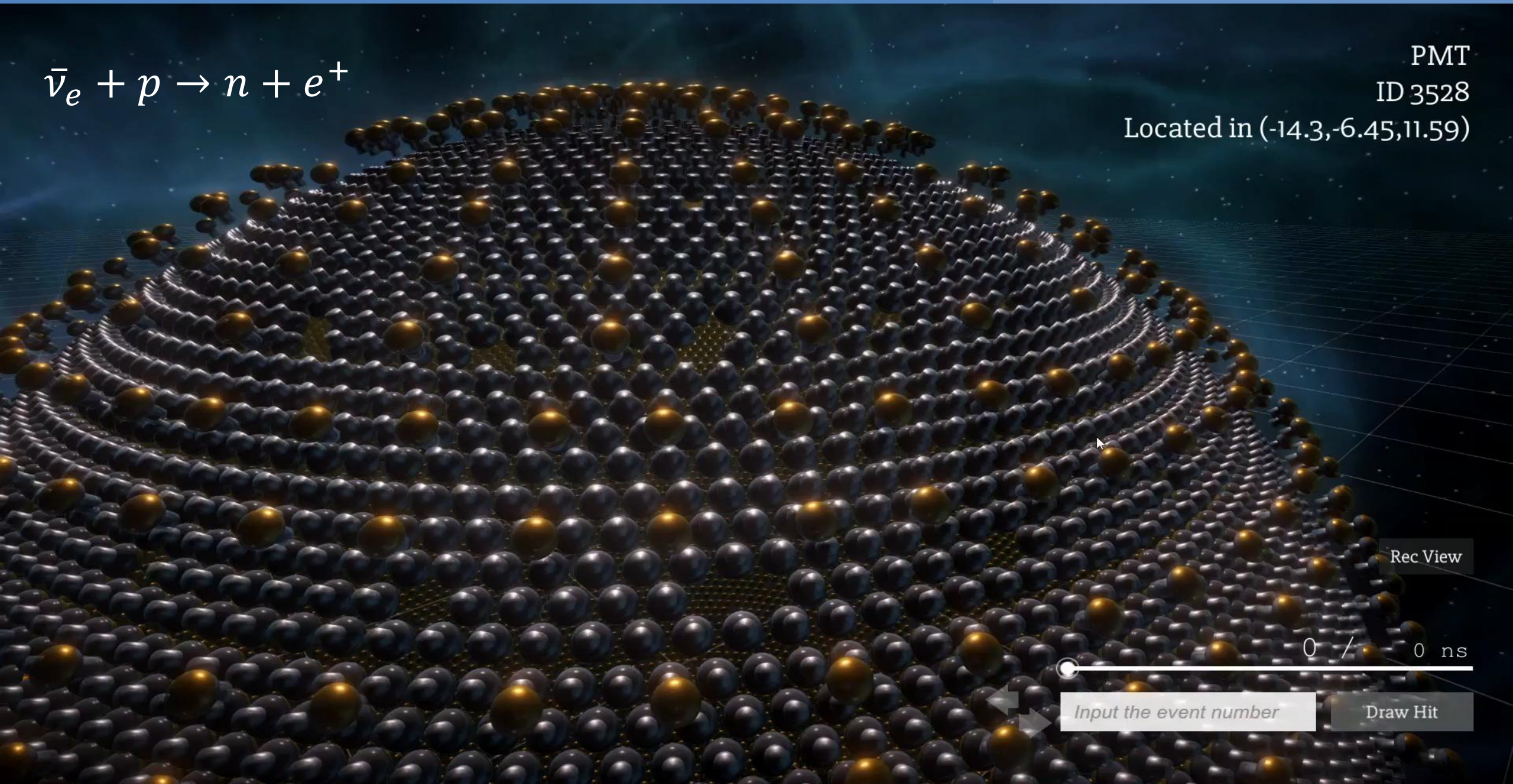
基于模拟工具完成，如Pythia, GEANT4, FLUKA等

一般由用户自己开发

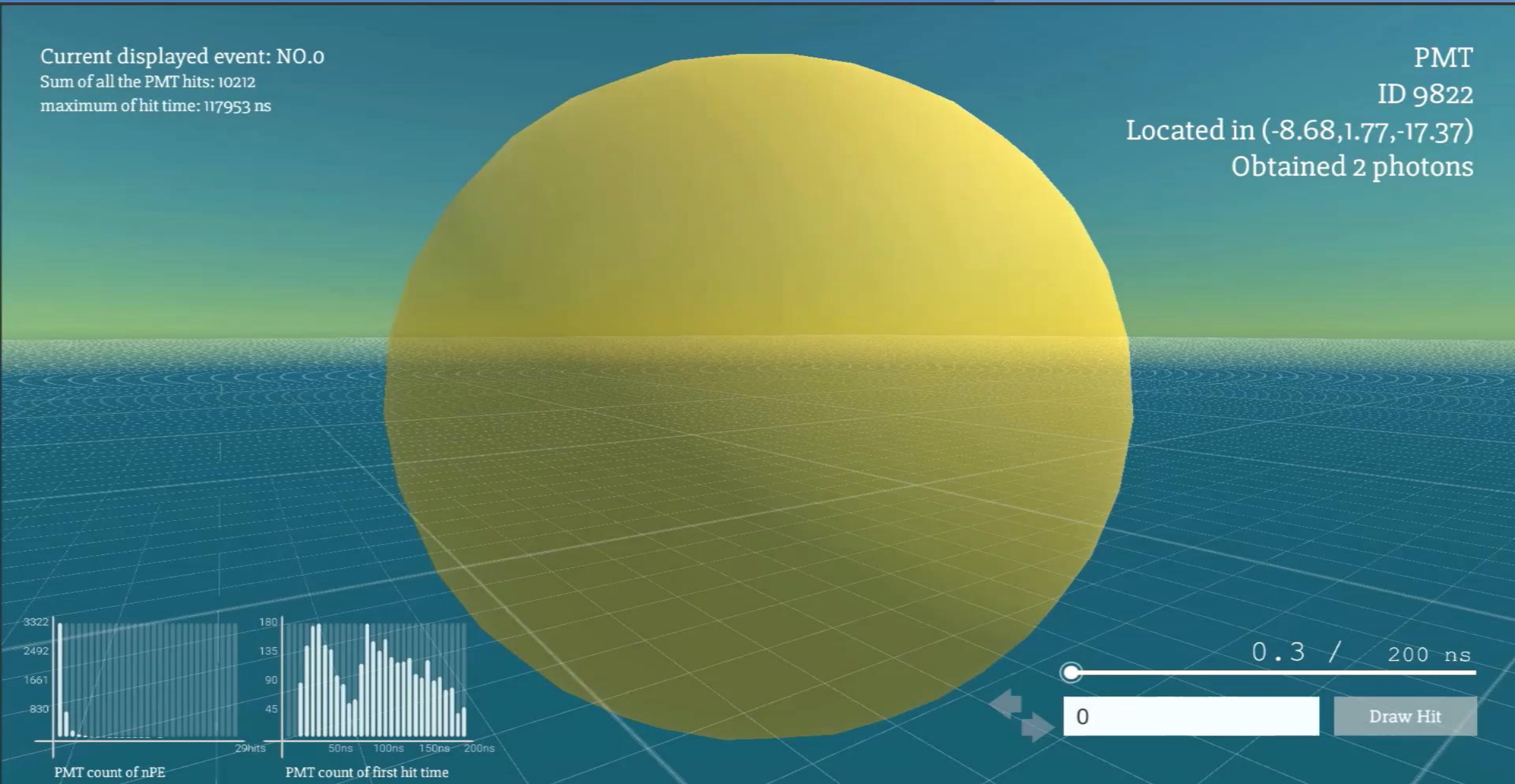
❄ **除以上部分，通常探测器模拟系统还具备以下功能：**

- 图形显示
- 信息获取
- 模拟控制与干预
- …… …

JUNO模拟事例显示


$\bar{\nu}_e + p \rightarrow n + e^+$
PMT
ID 3528
Located in (-14.3,-6.45,11.59)
Rec View
0 / 0 ns
Input the event number
Draw Hit

光学光子追踪


Current displayed event: NO.0
Sum of all the PMT hits: 10212
maximum of hit time: 117953 ns
PMT
ID 9822
Located in (-8.68,1.77,-17.37)
Obtained 2 photons
3322
2492
1661
830
29hits
PMT count of nPE
180
135
90
45
50ns
100ns
150ns
200ns
PMT count of first hit time
0.3 / 200 ns
0
Draw Hit

事例产生子

- 事例产生子用于产生每个物理事例的原初粒子，并给出这些原初粒子的衰变模式和末态。

- 事例产生子的输出是探测器模拟的输入，通常
  - 探测器模拟工具，如GEANT4等，都已定义好和事例产生子间的接口。
  - 模拟工具本身会提供一些简单的事例产生子，如GEANT4中的粒子枪等。

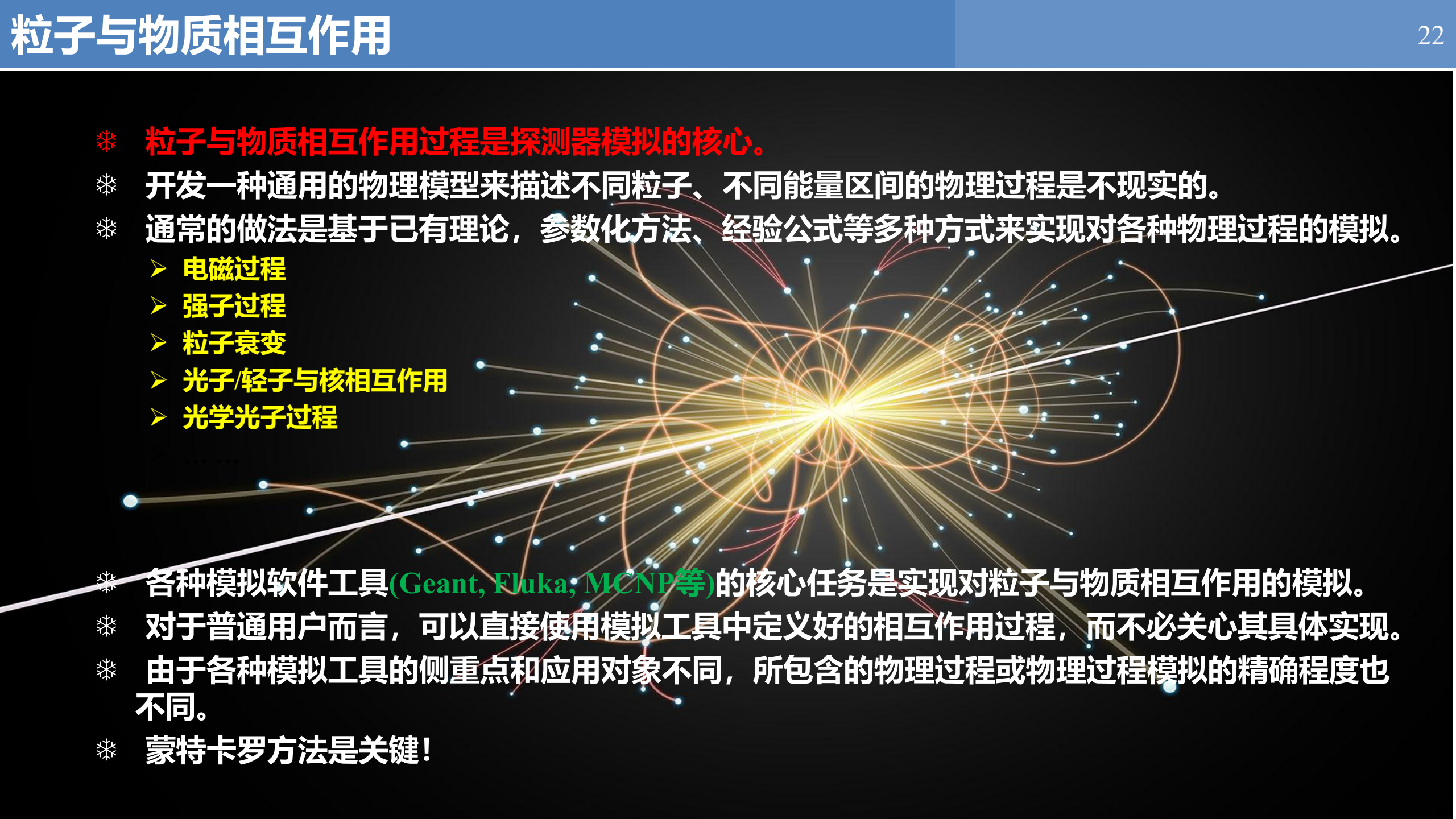
探测器物质和几何

- **原则上，模拟中所构建的探测器几何和物质与真实探测器一致性越高，模拟结果越精确。**
  - 在保证模拟精度的同时，也要保证模拟具备良好的计算性能。
  - 仔细构建灵敏探测器的几何和物质，对模拟结果影响小的复杂探测器区域通常采用等效的处理方法，如前端电子学，电缆，死区等。

- **探测器模拟工具(GEANT4，FLUKA等)提供基本的几何、物质及其属性、光学界面的定义方法。**
  - **几何：**形状，尺寸，位置，转动，几何间逻辑关系，物质设定，电磁场设置和灵敏探测器定义等。
  - **物质**(分子、化合物和混合物)由**元素**组成，元素由其各种**同位素**组成。
  - **物质属性：**密度，状态(气体/液体/固体)，光学属性(折射率，吸收长度，散射长度，光产额，发光时间等)，温度和压力等。
  - **光学界面：**定义光学光子在该界面上的反射，吸收等过程。

粒子与物质相互作用

- **粒子与物质相互作用过程是探测器模拟的核心。**
- 开发一种通用的物理模型来描述不同粒子、不同能量区间的物理过程是不现实的。
- 通常的做法是基于已有理论，参数化方法、经验公式等多种方式来实现对各种物理过程的模拟。
  - 电磁过程
  - 强子过程
  - 粒子衰变
  - 光子/轻子与核相互作用
  - 光学光子过程
  - ……



- 各种模拟软件工具(Geant, Fluka; MCNP等)的核心任务是实现对粒子与物质相互作用的模拟。
- 对于普通用户而言，可以直接使用模拟工具中定义好的相互作用过程，而不必关心其具体实现。
- 由于各种模拟工具的侧重点和应用对象不同，所包含的物理过程或物理过程模拟的精确程度也不同。
- 蒙特卡罗方法是关键！

❄ 几种常用的物理相互作用

- **不稳定粒子**
  - 飞行中衰变
- **光子**
  - 光电效应
  - 康普顿散射
  - 对产生
- **带电粒子**
  - 电离过程
  - 多次散射
  - 强相互作用

❄ 几种常用的物理相互作用

- **不稳定粒子**
  - 飞行中衰变
- **光子**
  - 光电效应
  - 康普顿散射
  - 对产生
- **带电粒子**
  - 电离过程
  - 多次散射
  - 强相互作用

Unstable particle

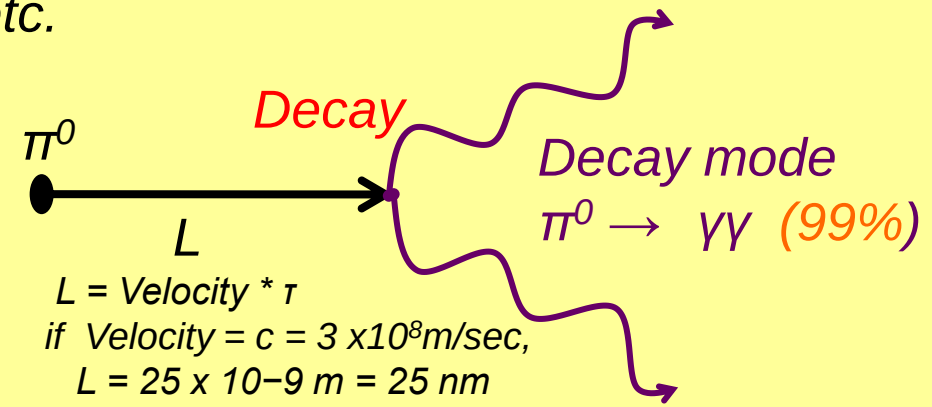
Decay in flight

Life time τ, for example,

π^0 $\tau = 8.4 \times 10^{-17}$ sec.

μ $\tau = 2.2 \times 10^{-6}$ sec.

etc.



*L = Velocity * τ*

if Velocity = c = 3 x10⁸m/sec,

L = 25 x 10−9 m = 25 nm

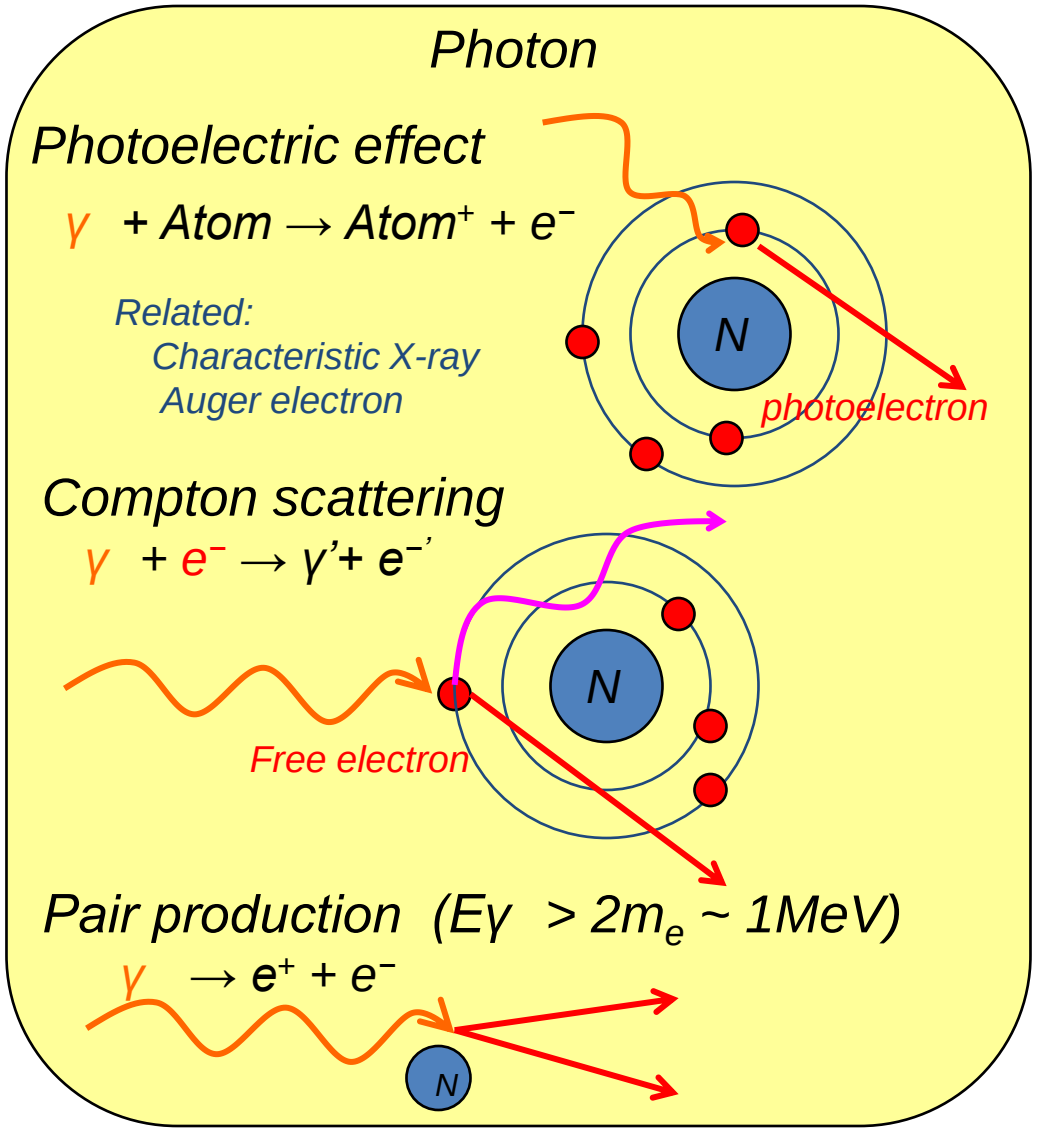
The other decay mode

$\pi^0 \rightarrow e^+e^-\gamma$ (1%)

Branching ratio

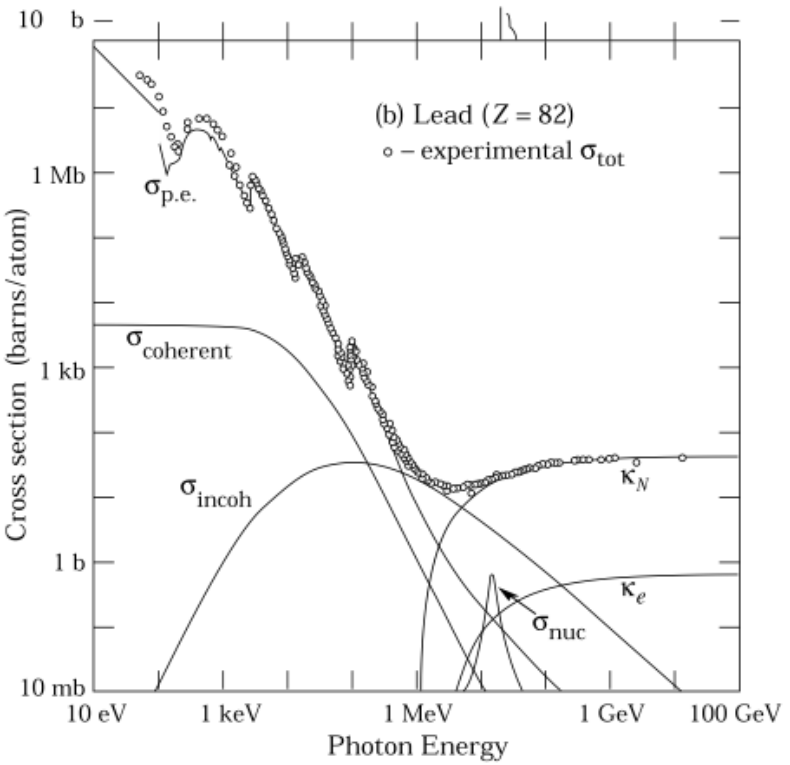
❄ 几种常用的物理相互作用

- ➢ 不稳定粒子
  - 飞行中衰变
- ➢ 光子
  - 光电效应
  - 康普顿散射
  - 对产生
- ➢ 带电粒子
  - 电离过程
  - 多次散射
  - 强相互作用

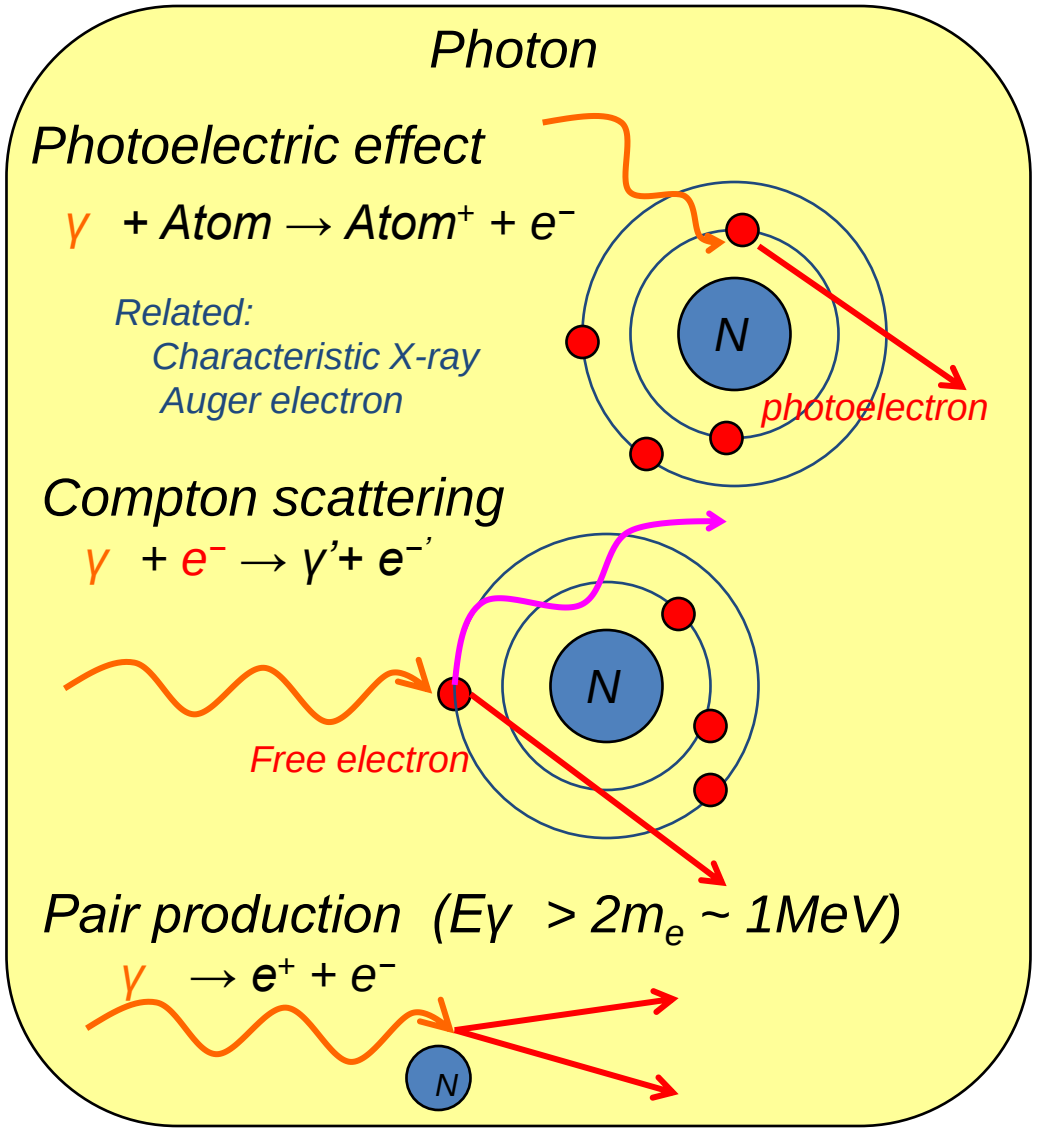


❄ 几种常用的物理相互作用

- ➢ 不稳定粒子
  - 飞行中衰变
- ➢ 光子
  - 光
  - 原
  - 双
- ➢ 带电
  - 电
  - 多
  - 弹



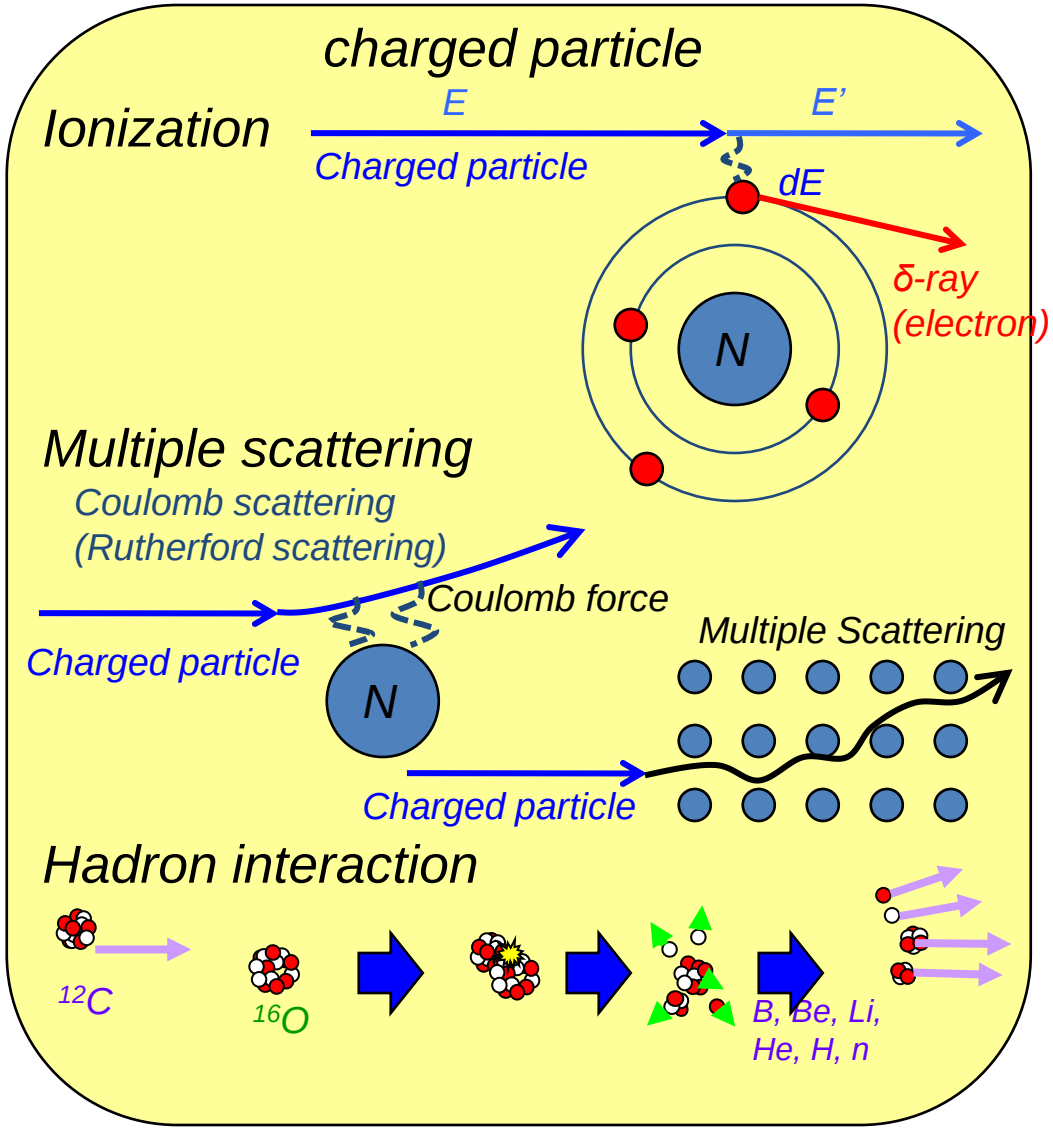
Photon Energy



几种基本物理过程简单回顾

❄ 几种常用的物理相互作用

- ➢ 不稳定粒子
  - 飞行中衰变
- ➢ 光子
  - 光电效应
  - 康普顿散射
  - 对产生
- ➢ 带电粒子
  - 电离过程
  - 多次散射
  - 强相互作用



❄ 指数定律 (The exponential law)

P(x): 在x距离内没有发生相互作用的几率。

ωdx: 在x和x+dx内发生相互作用的几率。

其中，$\omega = N \times \sigma$，*N:* 单位体积内靶粒子数；σ*:* 相互作用截面。

$$P(x + dx) = P(x)(1 - \omega dx)$$

dx内没有相互作用的几率

x距离内不发生相互作用的几率

两边积分后，得到：

$P(x) = exp(-\omega x)$, 概率分布函数是指数分布。

产生相互作用的一般考虑

- **基于上一页的讨论，在[x, x+dx]内发生相互作用的几率 $P_{int}(x)$ ，可以表示为：**

$$P_{int}(x)dx = P(x)\omega dx$$

$P_{int}(x)$: 概率密度函数(Probability Density Function, PDF)

- **累积分布函数(The cumulative distribution function, CDF)为：**

$$\int P_{int}(x)dx = \int P(x)\omega\, dx = \int \omega \exp(-\omega x)dx$$
$$= 1 - \exp(-\omega x)$$

- **连读并严格递增的累积分布函数是[0, 1]的均匀分布。**
- **采用逆变换的方法产生一个相互作用，**

$$\eta = 1 - \exp(-\omega x)$$

η是0到1之间的均匀分布。

$$x = -\ln(1-\eta)/\omega$$

不同物质中产生相互作用的处理方法

$$x = -\ln(1-\eta)/\omega$$

x是长度、距离，其依赖于物质。因此，对x的抽样过程同样依赖于物质。

通过如下变换，可使抽样过程与物质无关。

$$x\omega = -\ln(1-\eta)$$

引入平均自由程(mean free path) -- λ，并定义为：

$$\lambda = \int xP(x)dx / \int P(x)dx = 1/\omega$$

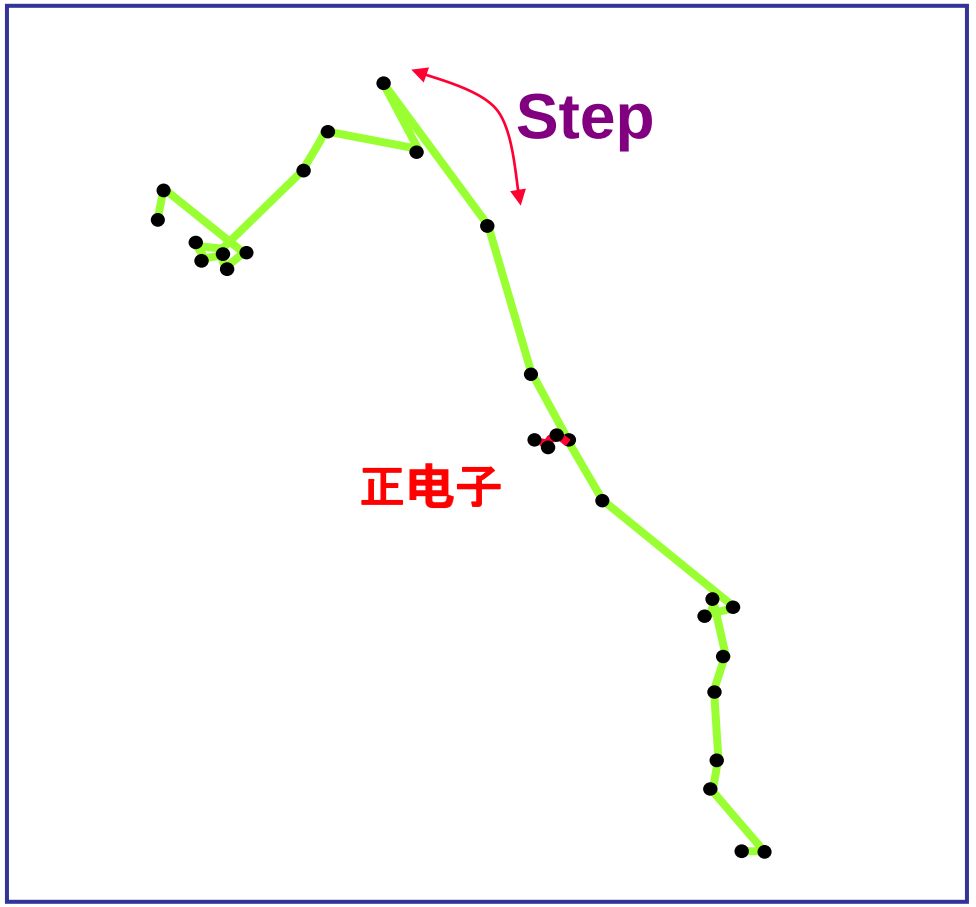
抽样过程以 λ为单位，并不依赖于物质。

$$x/\lambda = -\ln(1-\eta)$$

x/λ: 平均自由程数 (Number of Mean Free Path, NMFP)

如何考虑粒子在物质中的输运过程?

粒子在物质中的输运过程是一步一步进行模拟的，一步为一个step。



8MeV正电子在水中的湮灭过程

如何确定每一步的步长?

1. 在每个step的开始，对该粒子每一物理过程的NMFP进行抽样，这一抽样过程不依赖于物质。

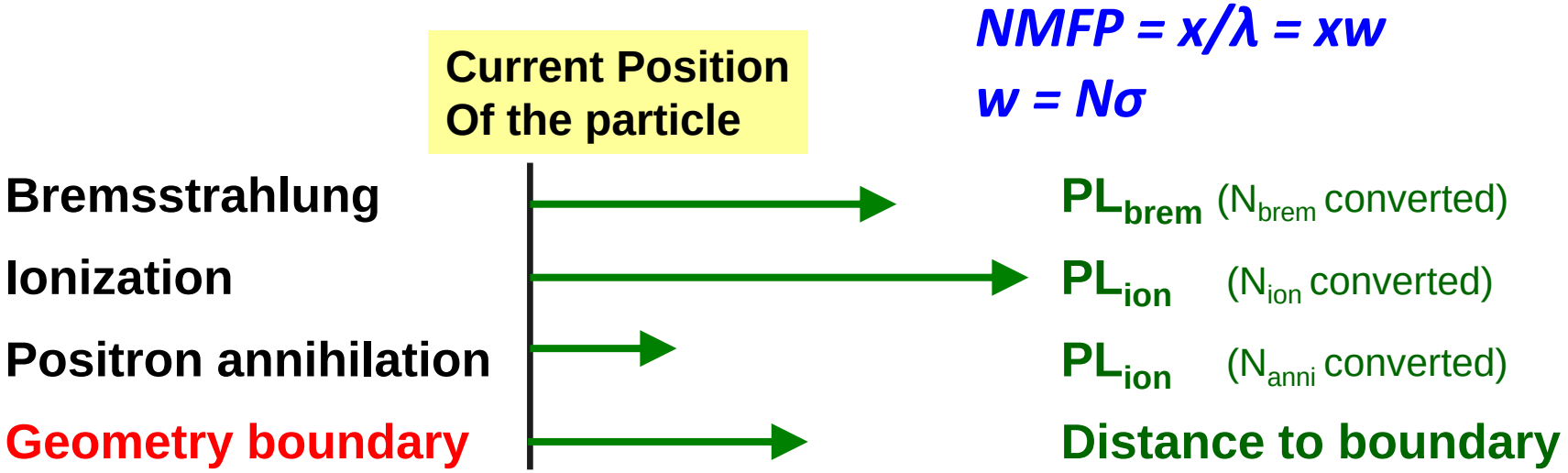
例如：正电子具有下面的物理过程，对每一过程都应用指数定律进行抽样，分别得到NMFP：

Bremsstrahlung $NMFP = N_{brem}$

Ionization $NMFP = N_{ion}$

Positron annihilation $NMFP = N_{anni}$

2. 利用粒子与物质的相互作用截面，将NMFP转换为相互作用长度(physical length, PL)。



3. 具有最小长度的物理过程或几何边界决定当前step的长度。

- 在上例中，step长度由正电子湮灭这一过程决定。

模拟粒子在物质中输运的流程

4. 完成该粒子在这一step的输运模拟。

5. 如果在完成相互作用以后，该粒子仍然处于存活状态，那么对所有的NMFPs重新做抽样，进行下一个step的模拟。

6. 如果在完成相互作用以后，该粒子消失，那么该粒子模拟结束

一个具体的例子

❄ 粒子衰变过程

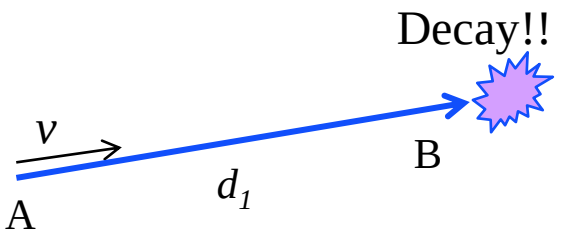
- 具有寿命τ的粒子衰变概率密度函数：

$$f(t)=\frac{1}{\tau}\exp\left(-\frac{t}{\tau}\right)\quad t\geq 0 \quad\Rightarrow\quad f(x)=\frac{1}{v\tau}\exp\left(-\frac{x}{v\tau}\right)$$

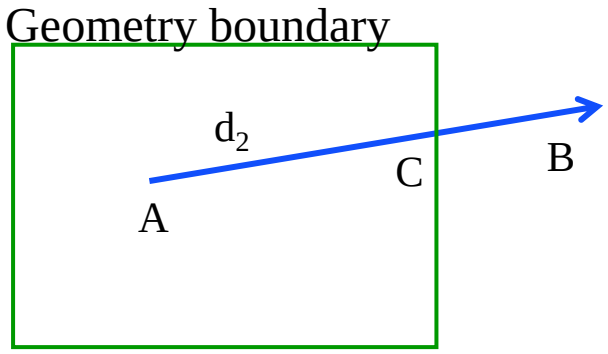
τ is the mean life of the particle

v is the velocity of the particle

$\omega = 1/(v\tau)$ *(non relativistic)*



- 抽样NMFP，$x = -\ln(1-\eta)\cdot v\tau$
- 衰变前运动距离，x = AB = d_1
- 粒子处于A位置，计算AC距离d_2。
- Step长度取决于min(d_1, d_2)。
- 如果B<C，粒子发生衰变。
  如果B>C，不做任何事情，但计算A到C的飞行时间。



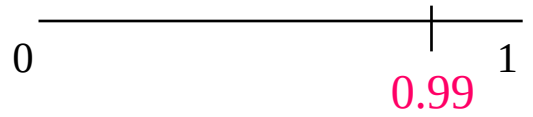
一个具体的例子

❄ 粒子衰变过程

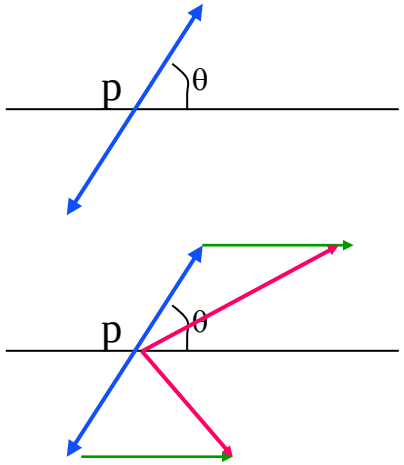
- 当粒子发生衰变。

  例如，$\pi^0 \rightarrow \gamma\gamma$ (~99%)

  $\rightarrow \gamma e^+ e^-$ (~1%)



- 根据分支比确定衰变末态，[0，1]之间随机选取r。
- 产生末态粒子
  - 在π0质心系中确定末态粒子4动量，$\delta\Omega = \sin\theta\, d\theta\, d\varphi$。
  - 通过Lorentz变换回到实验室系。



思考： 在上述过程中共使用了几次随机数?

虽然粒子衰变模拟是一个简单的过程，但它清楚地展示了analogue Monte Carlo (“honest” simulation)的基本思想。

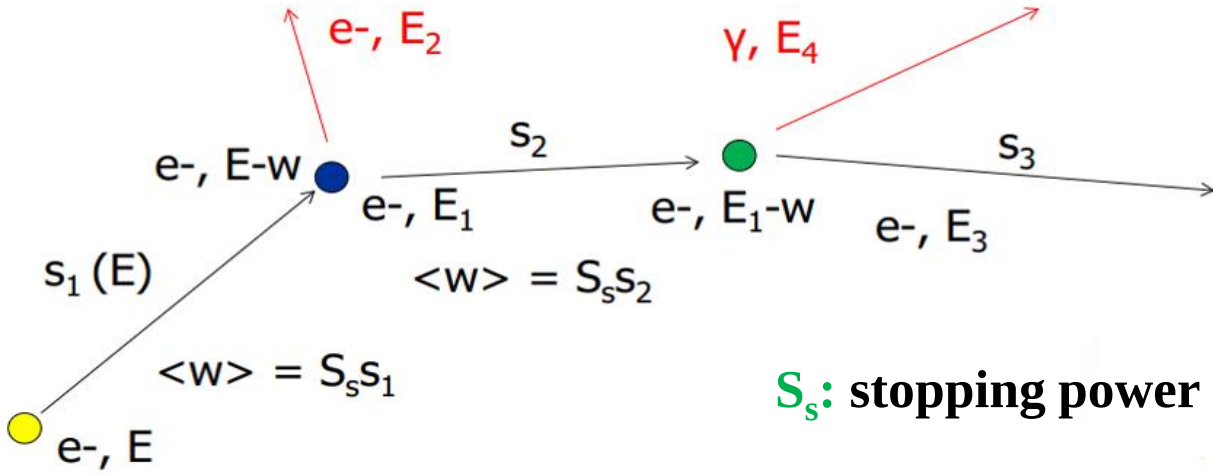
带电粒子在物质中的输运过程

- **Geant4没有tracking cut**
  - 追踪粒子直至其能量为零或离开探测器。

- **Step的长度取决于最短相互作用长度，但对带电粒子需要特殊处理**
  - 电离过程的截面很大，导致step长度很小，每个step的能量沉积很小 → 对计算性能是极大的挑战。

- **只模拟沉积能量(运动方向改变)大于特定阈值W_0的相互作用。**
  - W_0越小，模拟结果越精确，运行速度越慢。

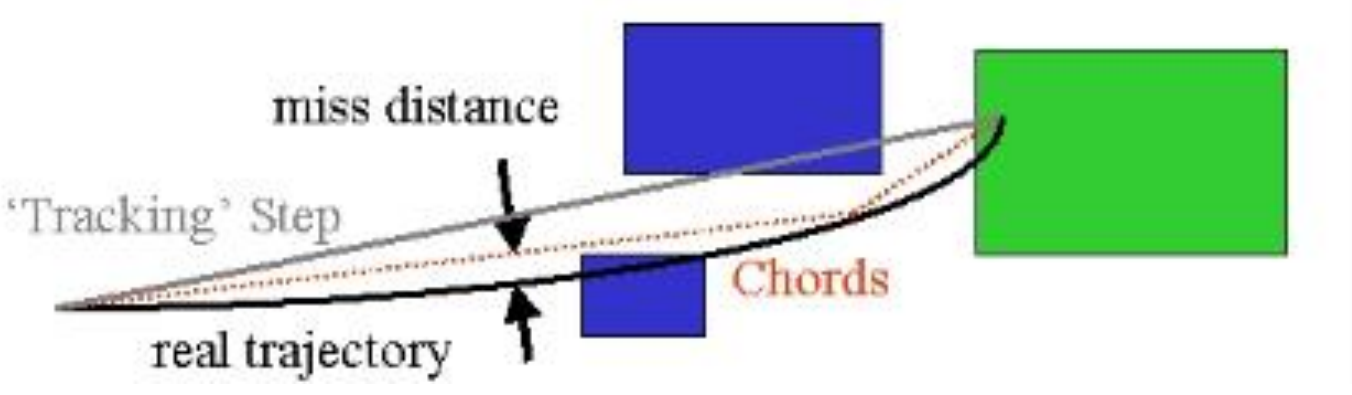
- **对能量沉积小于W_0的相互作用，采用统计的办法近似模拟。**



S_s: stopping power

粒子在电磁场中的输运

- 带电粒子在电磁场中的运动轨迹一般为曲线。
- 通过对粒子的运行方程进行数值积分完成对粒子输运过程的模拟。
  - 通常使用Runge-Kutta (RK)方法
- 真实径迹由多个弦(linear chord)来近似。
- 几个重要的参数用来控制模拟精度
  - miss distance (chord distance)：曲线被直线近似的几何误差
  - delta intersection：与边界求交的定位误差
  - delta one step：单步数值积分的末端误差



❄ 物理方法

- 计算机固有噪声
- 放射性物质的放射性
- 随机数序列无法重复，给验证结果带来很大困难。

❄ 数学方法 (伪随机数)

- 初值确定后，随机数序列唯一确定。
- 随机数序列会重复出现。
- 有利于验证结果。
- MC模拟中的主要应用

一般来说，模拟软件工具和ROOT等分析工具所提供的随机数产生器基本能够满足我们的需求，大家可以放心使用。

一些MC模拟工具列表

ETRAN (Berger, Seltzer; NIST 1978)

EGS4 (Nelson, Hirayama, Rogers; SLAC 1985) www.slac.stanford.edu/egs

EGS5 (Hirayama et al; KEK-SLAC 2005) rcwww.kek.jp/research/egs/egs5.html

EGSnrc (Kawrakow and Rogers; NRCC 2000) ww.irs.inms.nrc.ca/inms/irs/irs.html

Penelope (Salvat et al; U. Barcelona 1999) www.nea.fr/lists/penelope.html

Fluka (Ferrari et al; CERN-INFN 2005) www.fluka.org

Geant3 (Brun et al; CERN 1986) www.cern.ch

Geant4 (Apostolakis et al; CERN++ 1999) geant4.web.cern.ch/geant4

MARS (James and Mokhov; FNAL) www-ap.fnal.gov/MARS

MCNPX/MCNP5 (LANL 1990) mcnpx.lanl.gov

尝试安装 Geant4

http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/InstallationGuide/html/ 安装手册

Geant4安装前准备

- 操作系统
  - Linux（例如：CentOS 8，虚拟机），推荐使用
  - macOS
  - Windows 7+
- 编译器
  - C++11，例如：GCC 4.8.5+，clang 3.6+，Visual C++ 14.0
- CMake
  - 从www.cmake.org下载
  - unzip cmake-*.zip/tar.gz/Z
  - ./bootstrap, make, make install
- 外部库
  - Xerces-c，X11，OpenGL，…

- **虚拟机中已经安装了Geant4和ROOT等软件，可以直接使用。**
  - /usr/local

- **下载虚拟机 (10G左右)，VMware player和安装说明文件。**
  - 虚拟机文件(11GB)：
    https://ihepbox.ihep.ac.cn/ihepbox/index.php/s/lENNKTpzBAntlKM
  - VMware player：
    https://ihepbox.ihep.ac.cn/ihepbox/index.php/s/fd8RBVElPbV86Sr
  - Readme (说明文档):
    https://ihepbox.ihep.ac.cn/ihepbox/index.php/s/tzQyur1A4OQIw6i

- **推荐这种方式。**

❄ 修改CMakeLists.txt文件，添加下面内容：

➢ 在第20行附近，添加：

```
find_package(ROOT REQUIRED)
include(${ROOT_USE_FILE})
```

➢ 链接ROOT库文件

```
target_link_libraries(exampleB1 ${Geant4_LIBRARIES} ${ROOT_LIBRARIES})
```

GEANT4介绍和使用

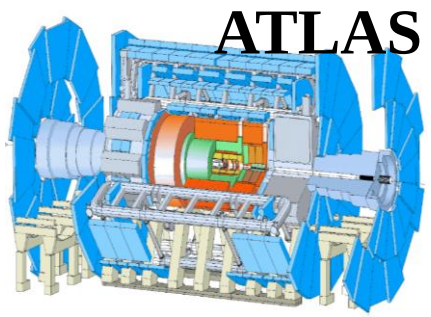
- Geant4 (GEometry ANd Tracking)是一个用来模拟粒子在物质中输运的工具包。
- Geant4提供了探测器模拟中所需要的一系列工具：几何/物质、粒子输运、相互作用、run/event/track管理、可视化、用户接口等。
- Geant4提供了非常丰富的物理过程，涵盖了不同粒子种类和较宽能量范围。通常对于一个特定的物理过程，具有多种物理模型可供选择。
- 用户需要基于Geant4所提供的工具包完成自己模拟程序的开发任务。
- Geant4基于C++语言开发，采用面向对象的软件技术。
- Geant4是开源的。

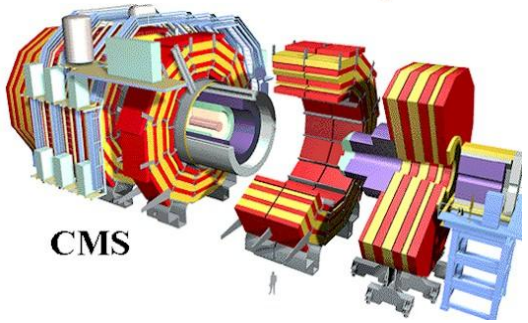
Geant4应用领域

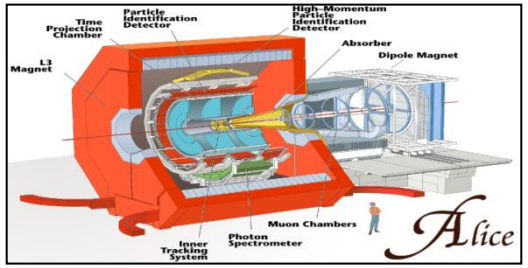
- **基于Geant4的探测器模拟广泛应用于，**
  - 高能物理 (探测器设计，刻度/重建，数据分析)
  - 核物理
  - 宇宙线物理

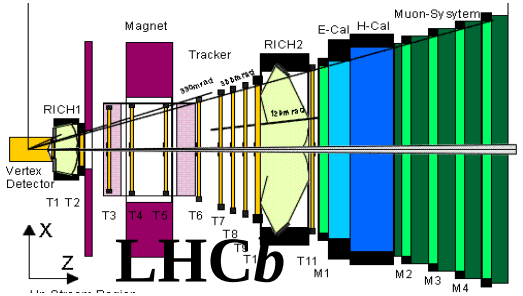
- **此外，还应用于很多其它领域，**
  - 空间科学
  - 放射科学
  - 医学 (束流治疗，放疗，成像，辐射研究)
  - 背景辐射计算
  - …… ……

Geant4诞生









Storage

raw recording rate 0.1–1 GByte/s
accumulating at 5-8 PBytes/year

Processing

200,000 of “today’s” fastest PCs

1000 person-years

“0ffline” software effort per experiment

~5000 physicists

around the world, around the clock

20 years software life-span

基于LHC上大型HEP实验的迫切需求，当时的Geant3 (Fortran)已经不能满足要求，在此基础上，Geant4正式被设计和开发。

Geant4发展历史简介

- **Pre-R&D阶段**
  - 1993 -- 在CERN和KEK，开始对GEANT3采用面向对象的技术进行重新设计。
- **R&D阶段**
  - 1994年12月 -- R&D计划书递交到CERN - *Geant4诞生！*
  - 1995年12月 -- 完成原型的基本设计和实现。R&D人员发展到~100多位科学家。
  - 1997年4月 -- 第一个α版本发布。
  - 1998年7月 -- 第一个β版本发布。
  - 1998年12月 -- 版本1发布，R&D阶段结束。
- **Geant4合作组成立**
  - 1998年12月 -- Geant4合作组成立，基于“MoU”(Memorandum of Understanding)。
  - ...
  - 2012年11月30日 -- Geant4 9.6版本发布。
  - 2013年12月06日 -- Geant4 10.0版本发布。
  - 2014年12月05日 -- Geant4 10.1版本发布。
  - 2015年12月04日 -- Geant4 10.2版本发布。
  - 2016年12月09日 -- Geant4 10.3版本发布。
  - 2021年12月10日 -- Geant4 11.0版本发布。
  - 2025年12月05日 - Geant4 11.4版本发布 ← *当前版本*

目前，Geant4合作组每年会发布一个版本。

Geant4合作组


Geant4 – A Simulation Toolkit
Geant 4
CERN
SLAC
NATIONAL ACCELERATOR LABORATORY
cnrs
IN2P3
Les deux infinis
TRIUMF
esa
cea
INFN
http://www.geant4.org/
KOBE UNIVERSITY Science
Science & Technology Facilities Council
富山高等専門学校
Toyama National College of Technology
S. Agostinelli et al.
Geant4: a simulation toolkit
NIM A, vol. 506, no. 3, pp. 250-303, 2003
JINR
Физический институт имени П.Н. Лебедева
RITSUMEIKAN
lapp
Laboratoire d'Annecy-le-Vieux de Physique des Particules
ANR
J. Allison et al.
Geant4 Developments and Applications
IEEE Trans. Nucl. Sci., vol. 53, no. 1, pp. 270-278, 2006
TEXAS A&M UNIVERSITY
LIP
Fermilab
KiSTi
한국과학기술정보연구원
Korea Institute of Science and Technology Information
ÉCOLE POLYTECHNIQUE ParisTech
UNIVERSITÉ BORDEAUX 1
NATIONAL CANCER CENTER
CENBG
Ciemat
UNIVERSITÉ PARIS SUD
LAL LABORATOIRE DE L'ACCÉLÉRATEUR LINÉAIRE
IPN ORSAY
REGION AQUITAINE
Northeastern UNIVERSITY COLLEGE OF COMPUTER SCIENCE
Lawrence Livermore National Laboratory
U.S. DEPARTMENT OF ENERGY Office of Science
Geant4 ASSOCIATES INTERNATIONAL
Experts in Radiation Simulation

http://geant4.cern.ch/support/userdocuments.shtml

❄ Geant4合作组提供

- Geant4介绍 (Introduction to Geant4)
- 安装指南 (Installation guide)
- 用户手册 (User's guide: For application developers)
- 高级用户手册 (User's guide: For Toolkit developers)
- 物理参考手册 (Physics reference manual)
- 源代码浏览器 (LXR source code browser)
- Doxygen -- classes and members reference guide

❄ 用户反馈

- 用户论坛 (HyperNews -- bug reports and fixes)
- FAQ

Geant4中的单位

- **在Geant4中，用户可以为任何的变量选择和使用合适的单位。**

- **在基本单位的基础上，Geant4使用一套内在的、自洽的单位集：**
  - millimeter (mm)
  - nanosecond (ns)
  - Mega electron Volt (MeV)
  - Positron charge (eplus)
  - Degree Kelvin (kelvin)
  - … …

- **所有的单位都可以通过基本单位进行定义。**
  - millimeter=mm=1;
  - meter = m = 1000*mm;
  - …….
  - m3 = m*m*m;
  - ……..

- **单位的定义包含在下面的文件中：**
  - Geant4/global/management/include/G4SystemOfUnits.hh

- **每一个数字都必须乘以其合适的单位。**
  - double Radius = 10.0*cm;
  - double kineticE = 1.0*GeV;
- **为了得到一个变量的值，必须要除以相应的单位。**
  - G4cout << eDep/MeV << " [MeV]" << G4endl;
  - G4cout <<G4BestUnit(eDep, "Energy") <<G4endl;
- **如果需要，你可以定义一个新的单位。**
- **注意单位的使用是否正确：**
  - G4double radius=10.*m; // internally converted: radius=10000
  - ….
  - G4double xposition = (radius*cos(alpha*rad))*cm; // is this really what you want to do??
  - G4double yposition = (radius*sin(alpha*rad))*cm;

运行Geant4提供的例子B1

- 开启虚拟机，并打开Terminal。
- Geant4, ROOT和cmake的环境变量都已经自动设置完成。

```
bash$ vim ~/.cshrc
```

- 从geant4目录下拷贝B1例子到自己的工作目录。

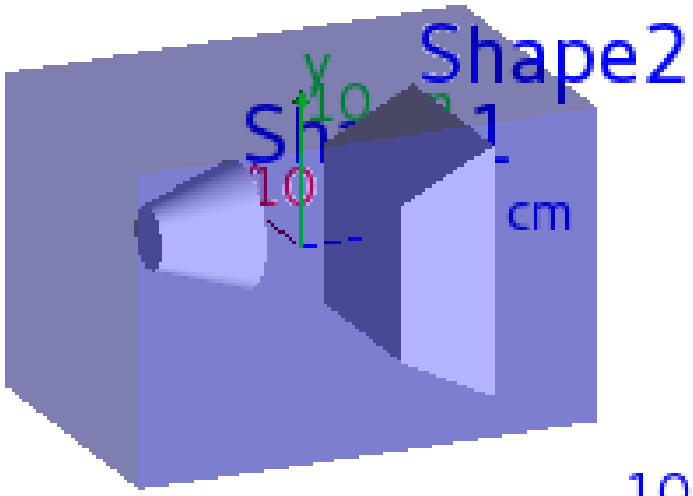
```
bash$ cd ~/
bash$ mkdir g4work
bash$ cp /usr/local/geant4.10.07/share/Geant4-10.7.0/examples/basic/B1 . -r
```

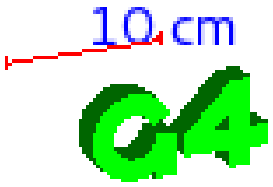
- 编译并运行B1。

```
bash$ cd ~/g4work/B1
bash$ mkdir build <建立一个编译目录>
bash$ cd build
bash$ cmake .. <生成B1的makefile>
bash$ make  <编译B1>
bash$ ./exampleB1  <运行B1>
geant4$ /run/beamOn 10
```

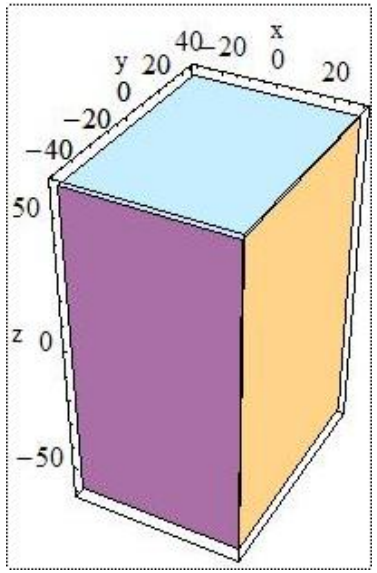
B1例子中的几何与物质

- B1例子中有3种几何形状：
  - *长方体(G4Box)*
  - *圆锥(G4Cons)*
  - *梯形(G4Trd)*
- B1例子中使用了4种物质：
  - *空气、水、组织、骨骼*

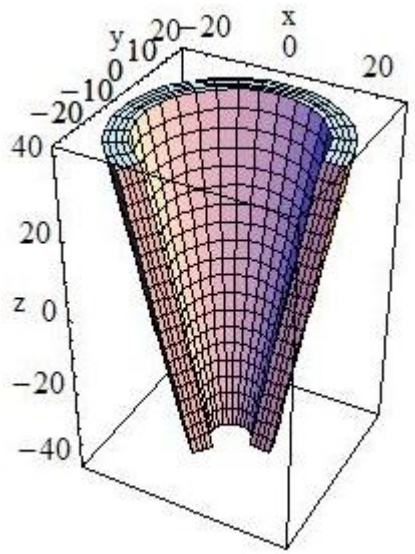




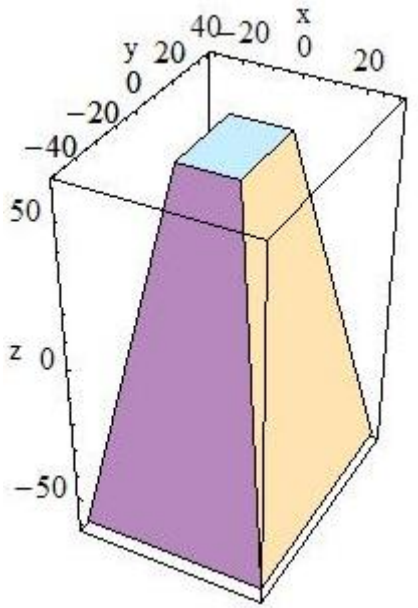
G4Box



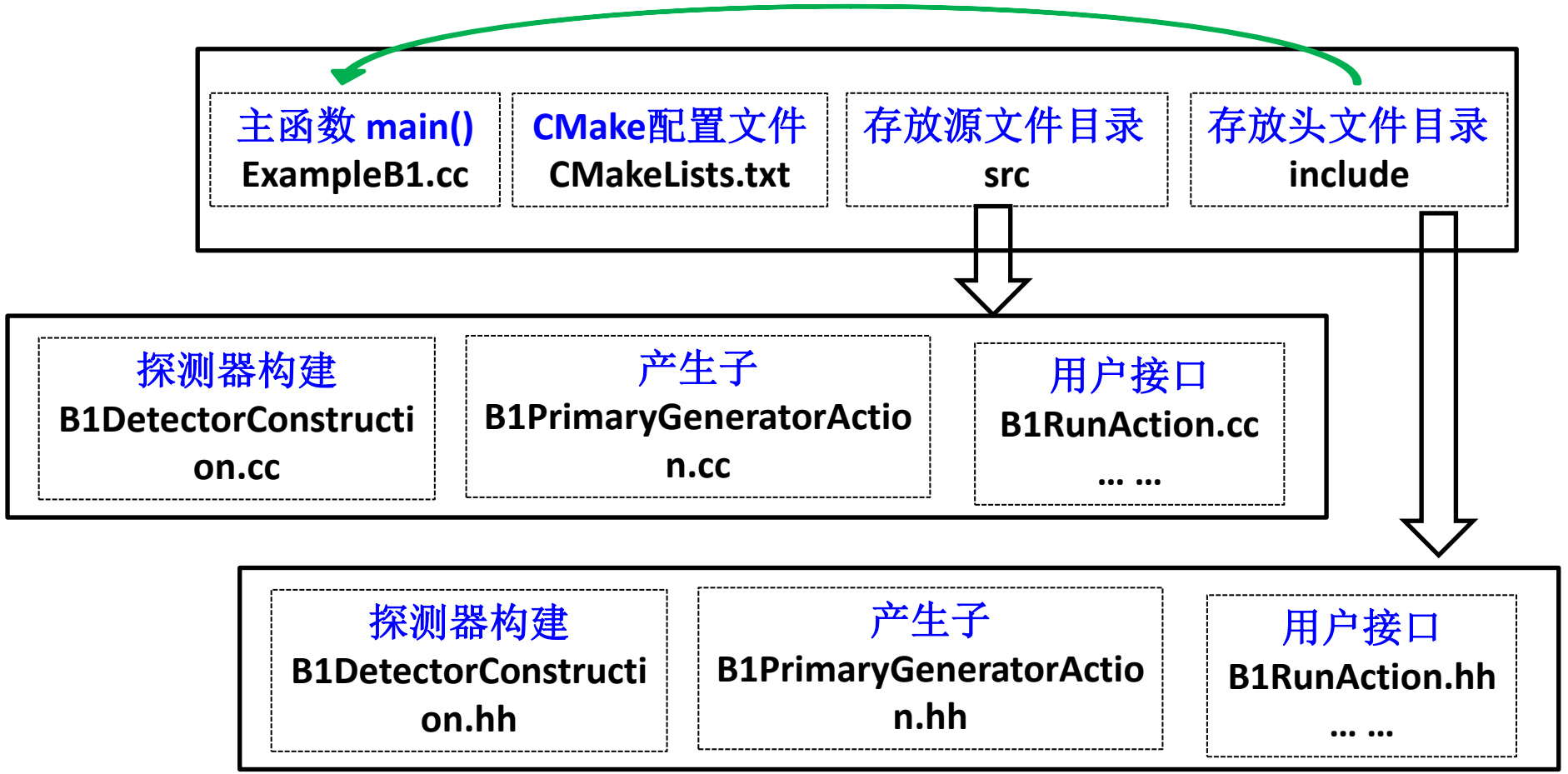
G4Cons



G4Trd



- Geant4只提供用户工具（库文件），没有主函数，也就没有可执行文件
- 用户在使用时，需要自己写主函数，例如B1例子



练习任务 (1)

- 安装虚拟机
- 熟悉Linux各种命令
- 熟悉B1例子中各个文件/类的功能，读代码
- 编译、运行B1例子

利用Geant4构建探测器（物质+几何）

- 如B1例子所示，用于构建探测器的类必须要继承其基类(G4VUserDetectorConstruction)

```
49 class G4VUserDetectorConstruction
50 {
51   public:
52     G4VUserDetectorConstruction();
53     virtual ~G4VUserDetectorConstruction();
54
55   public:
56     virtual G4VPhysicalVolume* Construct() = 0;
57
58     virtual void ConstructSDandField();
59     //This method is used in multi-threaded applications to build
60     //per-worker non-shared objects: SensitiveDetectors and Field managers
61
62     virtual void CloneSD();
63     virtual void CloneF();
64
65   public:
66     void RegisterParallelWorld(G4VUserParallelWorld*);
67
68   public:
69     G4int ConstructParallelGeometries();
70     void ConstructParallelSD();
```

- 用户需要具体实现探测器构建类中的Construct()函数，即，探测器的几何与物质定义需要在此函数中完成，并在最后返回”world physical volume”。

构建你自己的探测器

- 准备一个你自己的探测器构建类，需要从基类G4VUserDetectorConstruction继承。
- 具体实现Construct()函数：
  1) 构建所需的物质（G4Material）
  2) 定义探测器几何形状（Solid volume）
  3) 定义探测器几何属性（Logical volume）
  4) 完成探测器摆放（Physical volume）
  5) 定义探测器的显示属性*(可选)*
  6) 定义regions *(可选)*
- 具体实现ConstructSDandField()函数：
  1) 定义探测器内的电场或磁场(可选)
  2) 定义灵敏探测器(可选)
- 把你构建好的探测器交给G4RunManager。
- 当探测器结构较为复杂时，充分利用C++语言的特点，实现对探测器的模块化定义。

Geant4中物质的构建

- 物质(Material)由元素(Element)组成
  - *化合物 (molecules compounds), 混合物 (mixture)*
- 每种元素都包含若干个同位素(Isotopes)

Geant4给用户提供了3个类：

- Isotope ←→ G4Isotope
- Elements ←→ G4Element
- molecules, compounds and mixtures ←→ G4Material

G4Isotope and G4Element用来定义原子的属性。

- atomic number (Z), number of nucleons(N), molecular mass(A), shell structure, etc.
- 用在物质的定义中

G4Material

- 定义元素的组成
- 定义物质的属性：温度(temperature), 压力(pressure), 状态(state), 密度(density)
- 用在探测器的几何定义中

单一元素的物质定义

□ 由单一元素组成的物质 (液氩)

- 通过“名字，密度，Z，A”定义物质

```
G4double density = 1.390 * g/cm3;

G4double a = 39.95 * g/mole;

G4Material* lAr =

New G4Material("liquidArgon",z=18., a, density);
```

口 化合物：由多个原子组成 (H_2O)

```
G4double a = 1.01 * g/mole;
G4Element* elH  = new G4Element("Hydrogen”, symbol="H", z=1., a);
a = 16.00 * g/mole;
G4Element* elO  = new G4Element("Oxygen", symbol="O", z=8., a);

G4int ncomp;
G4double density = 1.000 * g/cm3;
G4Material* H2O = new G4Material("Water", density, ncomp=2);
G4int natoms;
H2O-> AddElement(elH, natoms=2);
H2O-> AddElement(elO, natoms=1);
```

混合物的定义 --- 元素质量百分比

□ 混合物 (Air)

```
G4double a = 14.01 * g/mole;
G4Element* elN  =
new G4Element(name="Nitrogen", symbol="N", z= 7., a);
a = 16.00 * g/mole;
G4Element* elO  =
new G4Element(name="Oxygen", symbol="O", z= 8., a);

G4int ncomp;
G4ouble density = 1.290 * mg/cm3;
G4Material* Air =
new G4Material(name="Air", density, ncomp=2);
G4double fracMass;
Air-> AddElement(elN, fracMass=70.0*perCent);
Air-> AddElement(elO, fracMass=30.0*perCent);
```

□ 混合物：气凝胶 (Aerogel)

```
G4Element* elC  = …;   // define “carbon” element
G4Material* SiO2 = …;  // define “quartz” material
G4Material* H2O = …;   // define “water” material

G4int ncomp;
G4double fractionmass;
G4double density = 0.200 * g/cm3;
G4Material* Aerog =
 new G4Material("Aerogel", density, ncomp=3);
Aerog-> AddMaterial(SiO2, fractionmass=62.5*perCent);
Aerog-> AddMaterial(H2O , fractionmass=37.4*perCent);
Aerog-> AddElement (elC , fractionmass= 0.1*perCent);
```

使用自己的同位素丰度来定义元素

例如：为核反应定义富集的铀元素。

```
G4Isotope* isoU235 = new G4Isotope("U235", iz=92, ia=235, a=235.0439242*g/mole);
G4Isotope* isoU238 = new G4Isotope("U238", iz=92, ia=238, a=238.0507847*g/mole);

G4Element* elenrichedU = new G4Element("enriched U", symbol="U", ncomponents=2);
elenrichedU-> AddIsotope(isoU235, abundance=90.*perCent);
elenrichedU-> AddIsotope(isoU238, abundance=10.*perCent);

G4Material* matenrichedU=
new G4Material("U for nuclear power generation", density=19.050*g/cm3,
ncomponents=1, kStateSolid);
matenrichedU-> AddElement(elenrichedU, fractionmass=1.);
```

物质定义的注意事项

- **真空(vacuum)可以定义为低密度物质**
  - 使用 “*universe_mean_density* (=1.e-25*g/cm^3)”

```
G4double z = 1.;
G4double a = 1.008*g/mole;
G4double density = 1.e-25*g/cm3;
G4double temperature = 2.73*kelvin;
G4double pressure = 3.e-18*pascal;
   G4Material* vacuum =
    new G4Material(“interGalactic", z, a, density, kStateGas, temperature, pressure);
```

- **对于强相互作用过程，正确定义探测器的物质对正确计算其反应截面是至关重要的。**
  - 避免使用"*averaged material*”.

NIST物质数据库

- **在Geant4中，可以直接使用NIST物质数据库。**
  - https://www.nist.gov/pml/atomic-weights-and-isotopic-compositions-relative-atomic-masses
- **对一些主要参数的描述都是非常准确的。**
  - 密度
  - 平均激发势
  - 化学键
  - 元素组成
  - 同位素组成
- **用户可以使用UI命令来打印物质和元素列表，如**
  - /material/nist/printElement (打印定义好的元素)
  - /material/nist/listMaterials (打印定义好的物质)
- **更加详细的信息，请参考Geant4用户手册附录6。**

NIST数据库中的元素与同位素

Z		A	m	error	(%)	Aeff
14	Si	22	22.03453	(22)		28.0855(3)
		23	23.02552	(21)		
		24	24.011546	(21)		
		25	25.004107	(11)		
		26	25.992330	(3)		
		27	26.98670476	(17)		
		28	27.9769265327	(20)	92.2297(7)	
		29	28.97649472	(3)	4.6832 (5)	
		30	29.97377022	(5)	3.0872 (5)	
		31	30.97536327	(7)		
		32	31.9741481	(23)		
		33	32.978001	(17)		
		34	33.978576	(15)		
		35	34.984580	(40)		
		36	35.98669	(11)		
		37	36.99300	(13)		
		38	37.99598	(29)		
		39	39.00230	(43)		
		40	40.00580	(54)		
		41	41.01270	(64)		
		42	42.01610	(75)		

同位素使用自然丰度，一共有3000多种同位素可供使用。

NIST中的物质


```
=========================================
###  Elementary Materials from the NIST Data Base
=========================================
 Z Name  ChFormula      density(g/cm^3)  I(eV)
=========================================
1   G4_H    H_2          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    N_2          0.0011652       82
8   G4_O    O_2          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                2.6989          166
14  G4_Si                2.33            173
```

- ❄ **NIST中的基本材料:**
  - ➢ H to Cf (Z=1 to 98)
- ❄ **NIST中的混合物:**
  - ➢ G4_ADIPOSE_TISSUE
- ❄ **高能物理中常用的物质:**
  - ➢ liquid Ar, $PbWO_4$, etc

```
=========================================
###  Compound Materials from the NIST Data Base
=========================================
 N Name      ChFormula      density(g/cm^3)  I(eV)
=========================================
13  G4_Adipose_Tissue                0.92          63.2
                1    0.119477
                6    0.63724
                7    0.00797
                8    0.232333
               11    0.0005
               12    2e-05
               15    0.00016
               16    0.00073
               17    0.00119
               19    0.00032
               20    2e-05
               26    2e-05
               30    2e-05
4  G4_Air                           0.00120479    85.7
                6    0.000124
                7    0.755268
                8    0.231781
               18    0.012827
2  G4_CsI                           4.51          553.1
               53    0.47692
               55    0.52308
```

如何使用NIST中定义好的物质?

不需要提前定义元素和物质。

```
G4NistManager* manager = G4NistManager::GetPointer();

G4Element* elm = manager->FindOrBuildElement("symb", G4bool iso);

G4Element* elm = manager->FindOrBuildElement(G4int Z, G4bool iso);

G4Material* mat = manager->FindOrBuildMaterial("name", G4bool iso);
// iso is omitable (true by default)

G4Material* mat = manager->ConstructNewMaterial("name",
                            const std::vector<G4int>& Z,
                            const std::vector<G4double>& weight,
                            G4double density, G4bool iso);

G4double isotopeMass = manager->GetMass(G4int Z, G4int N);
```

物质属性定义


```
G4Material* Acrylic = new G4Material("Acrylic", 1.18*g/cm3, 3);
Acrylic->AddElement(C, 0.59984);
Acrylic->AddElement(H, 0.08055);
Acrylic->AddElement(O, 0.31961);

Float energy[2] = {1*eV, 10*eV};
Float ABSLength[2] = {20*m, 1*m};
Float RayLength[2] = {27*m, 10*m};
Float index[2] = {1.6, 1.7};

G4MaterialPropertiesTable* AcrylicMPT = new G4MaterialPropertiesTable();
AcrylicMPT->AddProperty("ABSLENGTH", energy, ABSLength, 2);
AcrylicMPT->AddProperty("RAYLEIGH", energy, RayLength, 2);
AcrylicMPT->AddProperty("RINDEX", energy, index, 2);

Acrylic->SetMaterialPropertiesTable(AcrylicMPT);
```

练习任务 (2)

- 基于B1例子，完成模拟程序所需材料的定义，B1例子中的物质可以保留，将B1例子中圆锥几何体的材料替换为有机玻璃，梯形几何体材料替换为液闪

- 完成编译，重新显示几何

利用Geant4定义探测器几何

- 参考B1例子，与物质的定义一样，探测器几何的定义仍在探测器构建类中的Construct()函数中实现，并在最后返回”world physical volume”。

探测器几何的构建方法

- Introduction
- Solid and shape
- Logical volume
- Various ways of placement
  - Simple placement volume
  - Parameterized volume
  - Replicated volume
  - Divided volume

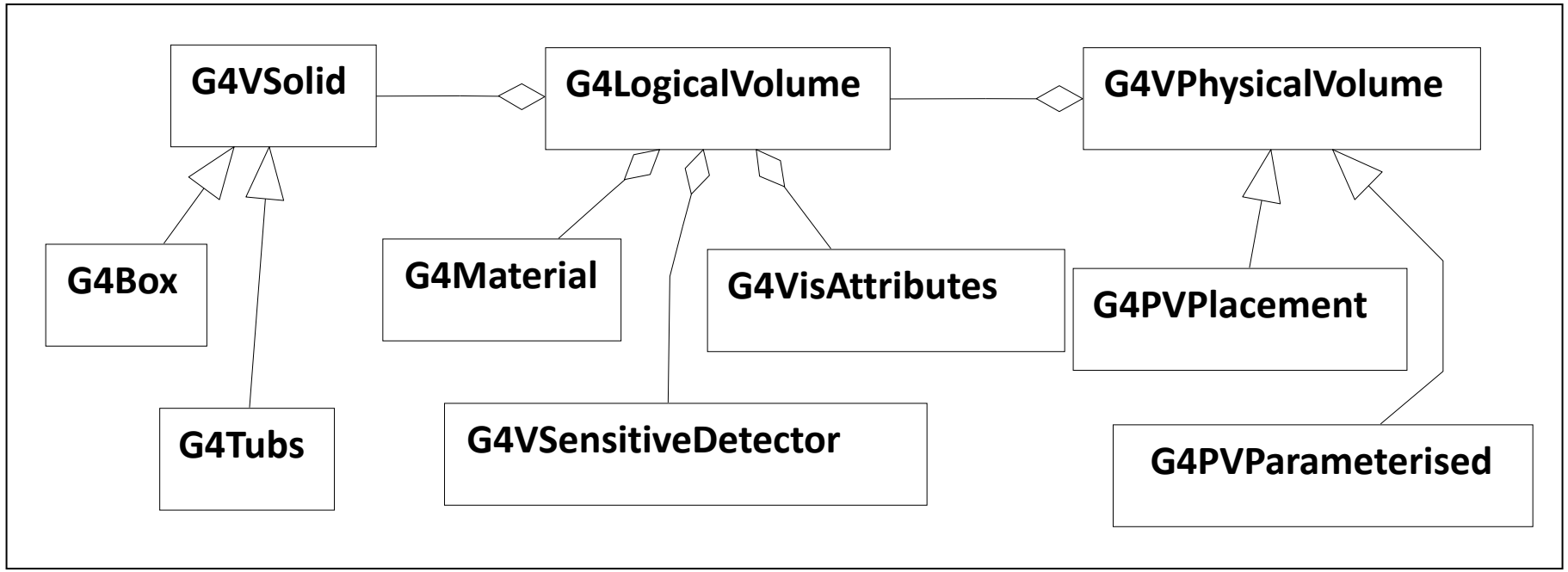
几何形状<长方形，球形，柱形，... ...>

几何属性<物质，显示属性，灵敏探测器，... ...>

几何的摆放<位置，旋转，... ...>

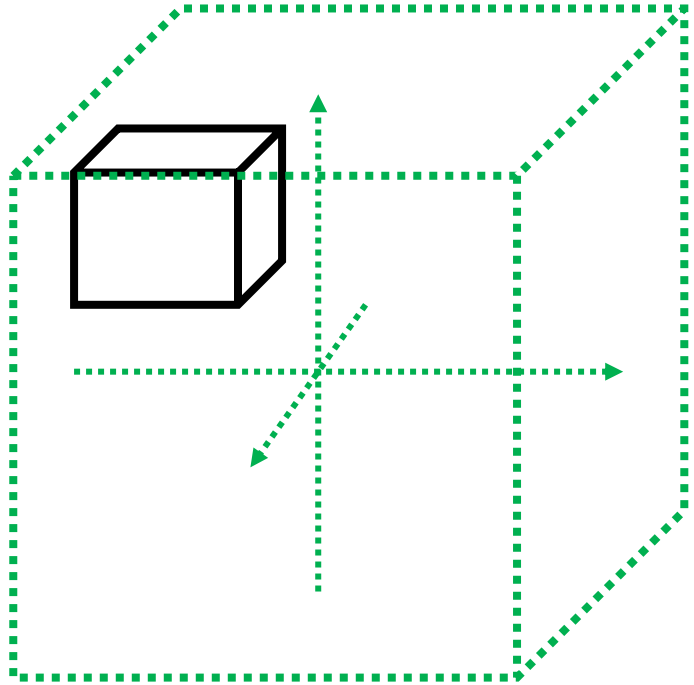
❄ 三步构建几何

- **G4VSolid** -- *shape, size*
- **G4LogicalVolume** -- *daughter physical volumes,*
  material, sensitivity, user limits, etc.
- **G4VPhysicalVolume** -- *position, rotation*



例子：定义一个长方体


```
G4VSolid* pBoxSolid =
new G4Box("aBoxSolid", 1.*m, 2.*m, 3.*m);

G4LogicalVolume* pBoxLog =
new G4LogicalVolume( pBoxSolid, pBoxMaterial,  "aBoxLog", 0, 0, 0);

G4RotationMatrix* rm = new G4RotationMatrix();
rm->RotateY(90*deg);
G4VPhysicalVolume* aBoxPhys =
new G4PVPlacement( rm, G4ThreeVector(posX, posY, posZ),  pBoxLog,
"aBoxPhys", pMotherLog,  0, copyNo);
```



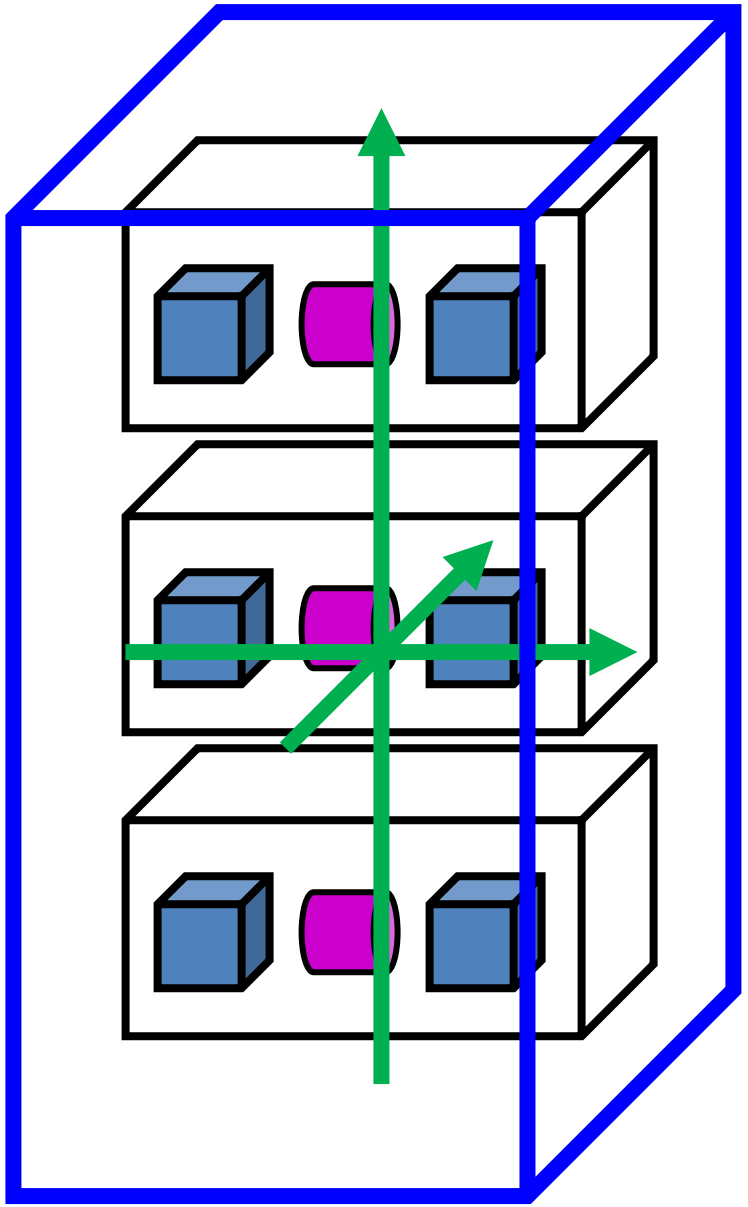
- **当把一个体积被放入其母体积时，其位置与旋转都是相对于其母体积的局域坐标来说的。母体积局域坐标系的原点在几何中心。**
  - **子体积的大小不能超出其母体积。**

几何体积间的层次关系

- 一个逻辑体积可以被多次摆放，一个母体积内可以放置一个或多个子体积

- 体积之间的母女关系是通过G4LogicalVolume来建立的
  - 如果一个母体积被多次摆放，那么这个母体积内的子体积会跟随母体积一起被多次摆放。

- 世界只能有一个！(world volume必须是唯一的，它包含所有其它体积)
  - World volume定义了全局坐标系，坐标原点在world的中心
  - 在模拟过程中，径迹的位置都是相对于全局坐标。

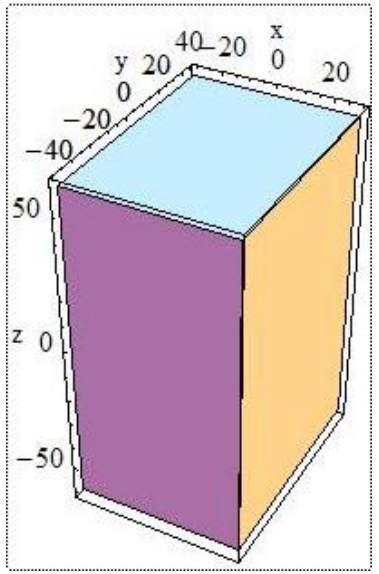


❄ Geant4中已经定义好的几何形状

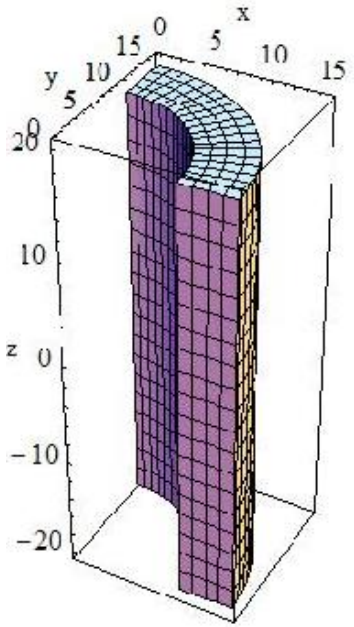
- CSG (Constructed Solid Geometry) solids
  - G4Box, G4Tubs, G4Cons, G4Trd, …
  - Analogous to simple GEANT3 CSG solids
- Specific solids (CSG like)
  - G4Polycone, G4Polyhedra, G4Hype, …
- Tessellated Solids
  - 定义几何体表面(G4VFacet)
  - 适用从CAD转换的复杂几何
- Boolean solids
  - G4UnionSolid, G4SubtractionSolid, …

CSG: 长方体、圆柱体

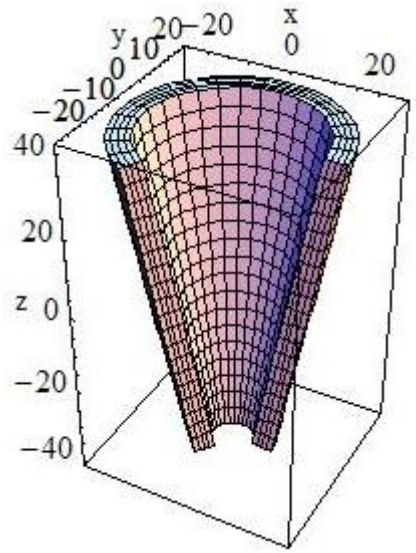
```
G4Box(const G4String &pname,    // name
            G4double half_x,    // X half size
            G4double half_y,    // Y half size
            G4double half_z);   // Z half size
```



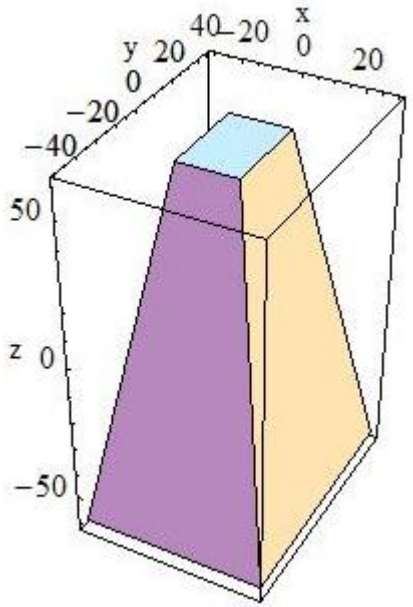

```
G4Tubs(const G4String &pname,   // name
             G4double  pRmin,   // inner radius
             G4double  pRmax,   // outer radius
             G4double  pDz,     // Z half length
             G4double  pSphi,   // starting Phi
             G4double  pDphi); // segment angle
```



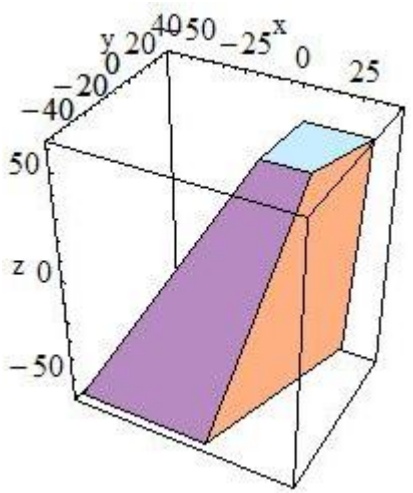
更多的CSG几何体



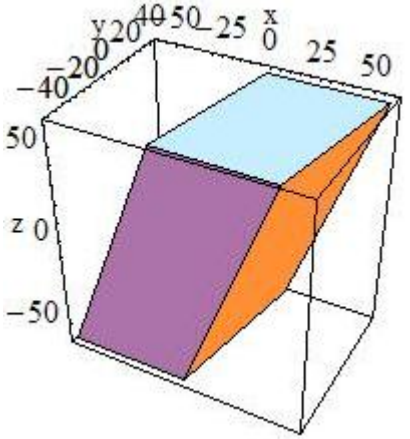
G4Cons



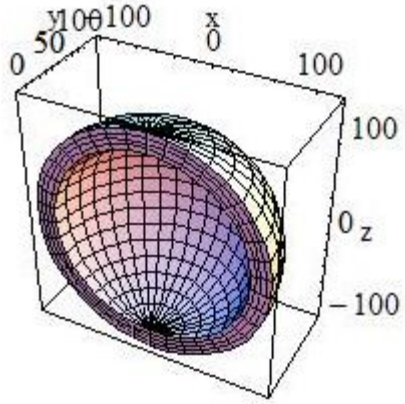
G4Trd



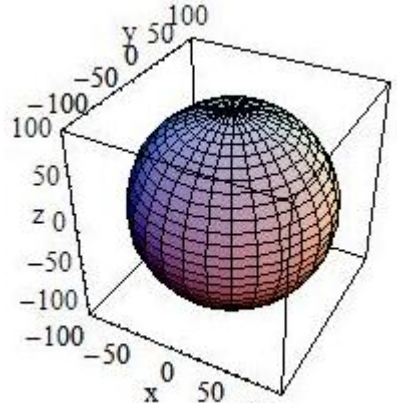
G4Trap



G4Para
(parallelepiped)

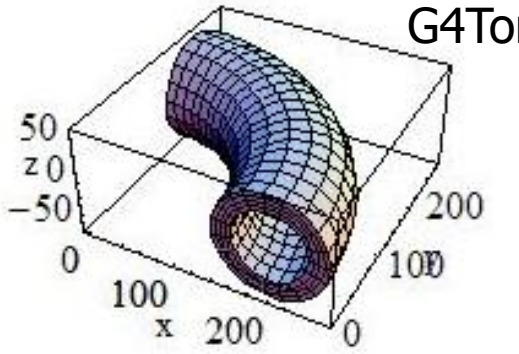


G4Sphere

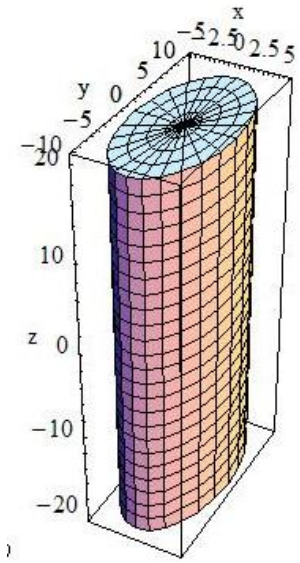


G4Orb
(full solid sphere)

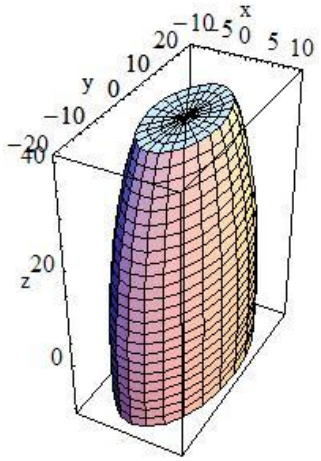
G4Torus



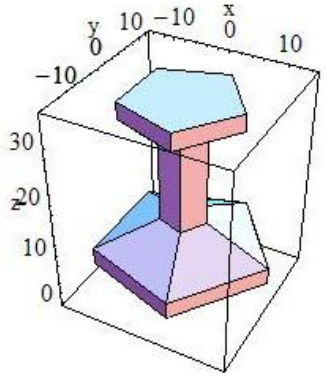
更多的CSG-like型几何体



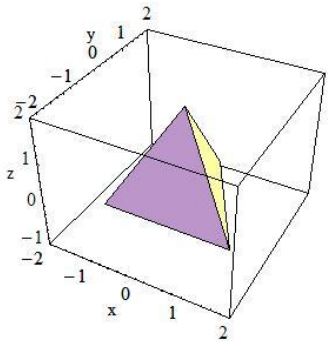
G4EllipticalTube



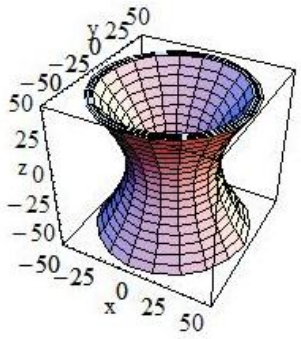
G4Ellipsoid



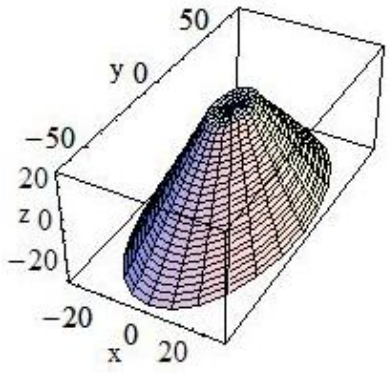
G4Polyhedra



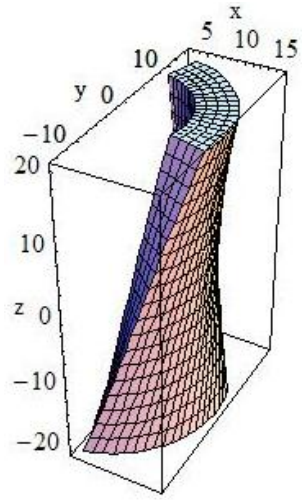
G4Tet
(tetrahedra)



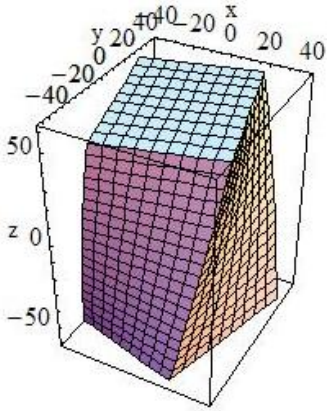
G4Hype



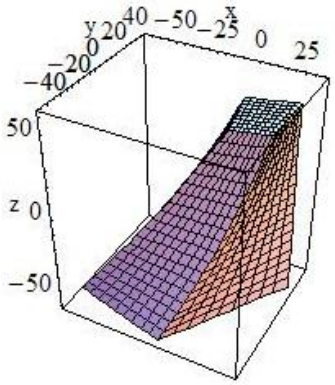
G4EllipticalCone



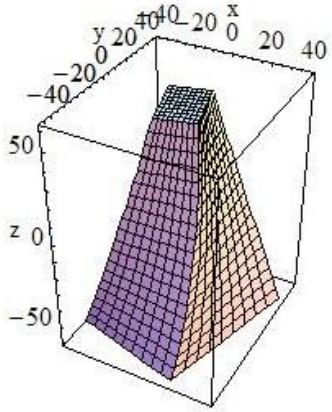
G4TwistedTubs



G4TwistedBox



G4TwistedTrap



G4TwistedTrd

更多的形状，请参考 **Section 4.1.2 of Geant4 Application Developers Guide**

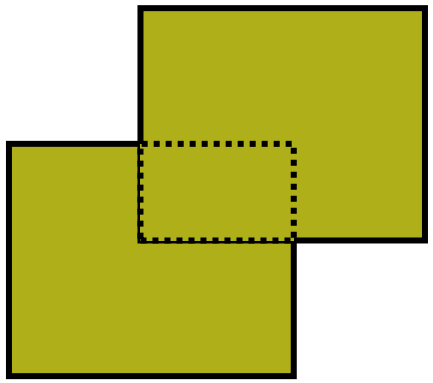
布尔型几何

- **几何体间可以通过布尔操作进行合并：**
  - G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid.
  - 要求：2个几何体, 1个布尔运算, 第2个几何的平移与转动（可选）。
  - 第2个几何的位置是相对于第一个几何的坐标系而言的。
  - 2个几何通过布尔操作后，变成一个几何，因此，它还可以和第3个几何通过布尔操作进行合并。

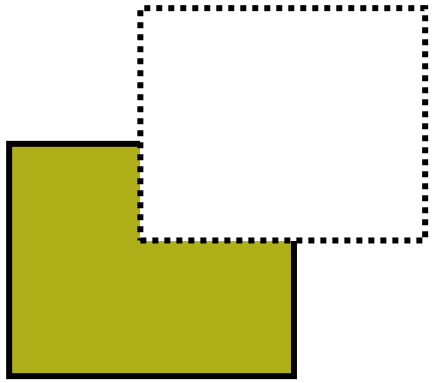
- **CSG型几何或者布尔型几何都可以通过此种方式进行合并。**

- **注意：在构建探测器几何时，应避免使用过多的布尔型几何，否则，运行速度会降低。**

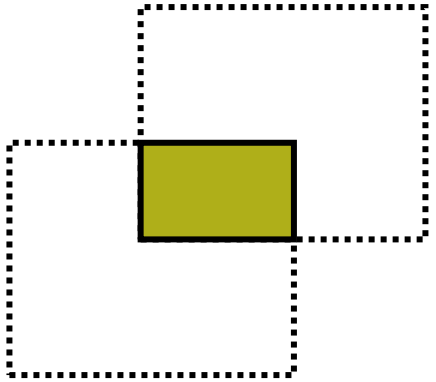
G4UnionSolid



G4SubtractionSolid



G4IntersectionSolid



如何构建布尔型几何(Boolean Solid)?


```
G4VSolid* box = new G4Box("Box",50*cm,60*cm,40*cm);
G4VSolid* cylinder
 = new G4Tubs("Cylinder",0.,50.*cm,50.*cm,0.,2*M_PI*rad);
G4VSolid* union
 = new G4UnionSolid("Box+Cylinder", box, cylinder);
G4VSolid* subtract
 = new G4SubtractionSolid("Box-Cylinder", box, cylinder,
       0, G4ThreeVector(30.*cm,0.,0.));
G4RotationMatrix* rm = new G4RotationMatrix();
rm->RotateX(30.*deg);
G4VSolid* intersect
   = new G4IntersectionSolid("Box&&Cylinder",
       box, cylinder, rm, G4ThreeVector(0.,0.,0.));
```

组合以后几何的坐标系及其原点与用于构建布尔型几何的第一个几何体相同

如何构建布尔型几何(Boolean Solid)?


```
G4VSoli
G4VSoli
 = new
G4VSoli
 = new
G4VSoli
 = new
G4Rotat
rm->Rot
G4VSoli
   = ne
```

组合

体相同

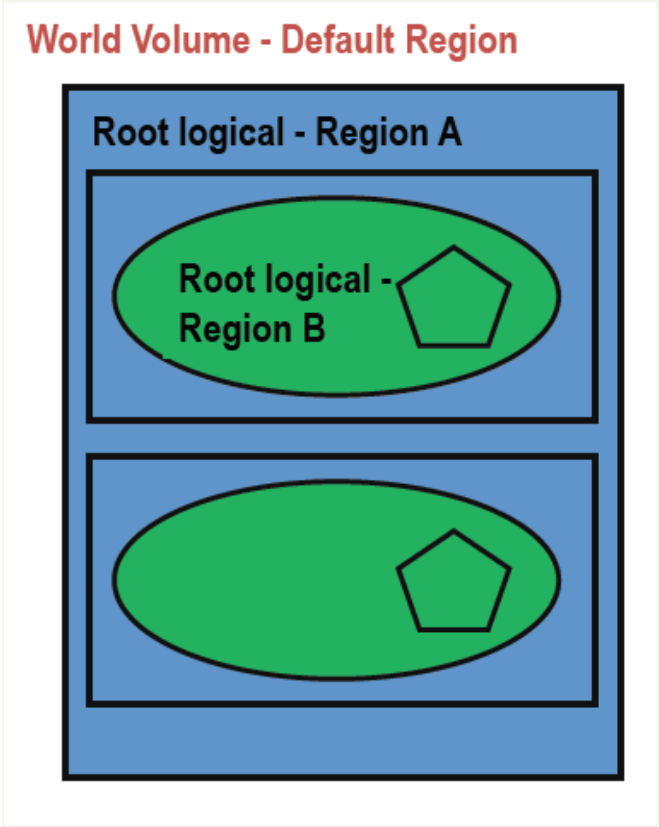
逻辑几何 (G4LogicalVolume)


```
G4LogicalVolume(G4VSolid* pSolid,
                G4Material* pMaterial,
                const G4String &name,
                G4FieldManager* pFieldMgr=0,
                G4VSensitiveDetector* pSDetector=0,
                G4UserLimits* pULimits=0);
```

- **包含几何体除位置和转动外的所有信息**
  - 形状和尺寸 (G4VSolid)
  - 物质，灵敏探测器，可视化属性
  - 子体积的位置
  - 电磁场，用户限制（User limits），区域（Region）
- **相同类型的物理几何体(Physical volumes)可以共用同一个逻辑几何。**
- **几何形状和尺寸(solid)的指针不能为空。**
- **物质的指针不能为空。**

几何区域(Region)

- 一个逻辑几何体可以被定义为一个区域，多个逻辑几何可以属于同一区域。
- 区域(region)是几何层次关系中的一个组成部分。例如，一系列的几何体组成一个集合，可以定义为一个区域，形成一个子系统。
- 一个逻辑几何一旦被指定为一个区域，那么它就成为一个根逻辑几何(root logical volume)。
  - 所有根逻辑几何的子几何体共享同一个区域，除非某一个子几何成为另一个根逻辑几何。
- 一个逻辑几何只能属于一个区域，不能被共享。
- World logical volume已经被默认指定为一个区域，因此，用户不能再为world指定一个区域。



区域的定义与使用(G4Region)

- 可以用如下方法定义一个区域:

```
G4Region* aRegion = new G4Region("region_name");
aRegion->AddRootLogicalVolume(aLogicalVolume);
```

- 给一个区域指定粒子产生阈(Production cuts)

```
G4Region* aRegion = G4RegionStore::GetInstance()-
>GetRegion("region_name);
G4ProductionCuts* cuts = new G4ProductionCuts;
cuts->SetProductionCut();
aRegion->SetProductionCuts(cuts);
```

G4Region类

G4Region class may take following quantities.

- void SetProductionCuts(G4ProductionCuts* cut);
- void SetUserInformation(G4VUserRegionInformation* uri);
- void SetUserLimits(G4UserLimits* ul);
- void SetFastSimulationManager(G4FastSimulationManager* fsm);
- void SetRegionalSteppingAction(G4UserSteppingAction* rusa);
- void SetFieldManager(G4FieldManager* fm);

Please note:

- If any of the above properties are not set for a region, properties of the world volume (i.e. default region) are used. Properties of mother region do not propagate to daughter region.

几何可视化属性(Visualization Attributes)

- 几何可视化属性由类G4VisAttributes进行管理。
- 用户可以通过该类对某一个几何体的可视化属性进行设置，如：

```
myLogical->SetVisAttributes(G4VisAttributes::Invisible);

G4VisAttributes* yourLogVis = new  G4VisAttributes(G4Colour(0.,1.,1.));
yourLogical->SetVisAttributes(yourLogVis);
```

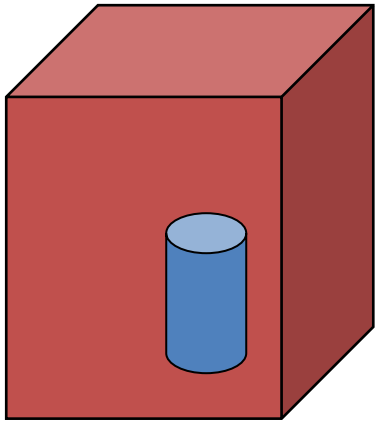
- 尝试对B1例子中的几何设置不同的可视化属性。

颜色属性 (G4Colour)

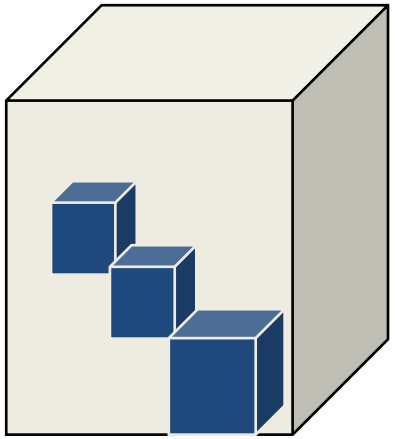

```
G4Colour white     ()              ;     // white
G4Colour white     (1.,1.,1.)      ;     // white
G4Colour gray      (.5,.5,.5)      ;     // gray
G4Colour black     (0.,0.,0.)      ;     // black
G4Colour red       (1.,0.,0.)      ;     // red
G4Colour green     (0.,1.,0.)      ;     // green
G4Colour blue      (0.,0.,1.)      ;     // blue
G4Colour cyan      (0.,1.,1.)      ;     // cyan
G4Colour magenta(1.,0.,1.)         ;     // magenta
G4Colour yellow    (1.,1.,0.)      ;     // yellow
```

物理几何体 (Physical volume)

- **Placement volume：摆放一次**
  - 一个物理几何体代表着一个真实的探测器单元。
- **Repeated volume：重复摆放**
  - 一个物理几何体代表着任意多个真实的探测器单元。
  - 可以有效降低内存的使用。
  - Parameterised (参数化)
    - 根据几何体编号(copy number)进行参数化，并重复摆放。
  - Replica and Division
    - 沿着一个坐标轴重复摆放。
- **一个母体积内可以包含：**
  - 多个一次摆放的几何
  - **或者是**, 一个重复摆放的几何
- **两个几何体之间发生重叠(overlap)是绝对不允许的。**



placement



repeated

物理几何体 (Physical volume) (1)

❄ **G4PVPlacement** **1 Placement = One Placement Volume**

- 几何体在其母体积内摆放一次。

❄ **G4PVParameterised** **1 Parameterized = Many Repeated Volumes**

- 根据几何体编号(copy number)进行参数化摆放。
  - 根据编号，可以对几何体的形状，大小，材料，灵敏探测器，可视化属性，位置和旋转实现参数化。
  - 用户必须实现一个具体的类，继承于G4VPVParameterisation。
- 降低内存消耗。
- 目前，参数化摆放几何的方法仅适用于下面的几何体中

  a) 几何体内没有子体积，或者

  b) 几何体有相同的尺寸和形状，以确保没有overlap的情况。
- 如果使用G4PVNestedParameterisation作为基类进行参数化，那么，物质、灵敏探测器和可视化属性可以通过原初几何体的编号进行参数化。

Physical volume (2)

❄ G4PVReplica　　1 Replica = Many Repeated Volumes

- 具有相同形状的子体积沿着一个坐标轴整齐摆放。
- 子体积完全填充在母体积内，子母体积间没有缝隙。

❄ G4PVDivision　　1 Division = Many Repeated Volumes

- 具有相同形状的子体积沿着一个坐标轴整齐摆放。
- 母体积与最外侧的子体积间有空隙。
- 子体积之间没有空隙。

❄ G4ReflectionFactory　　1 Placement = a pair of Placement volumes

- 按照镜像对称的方式摆放几何。
- 常见于端盖量能器中。

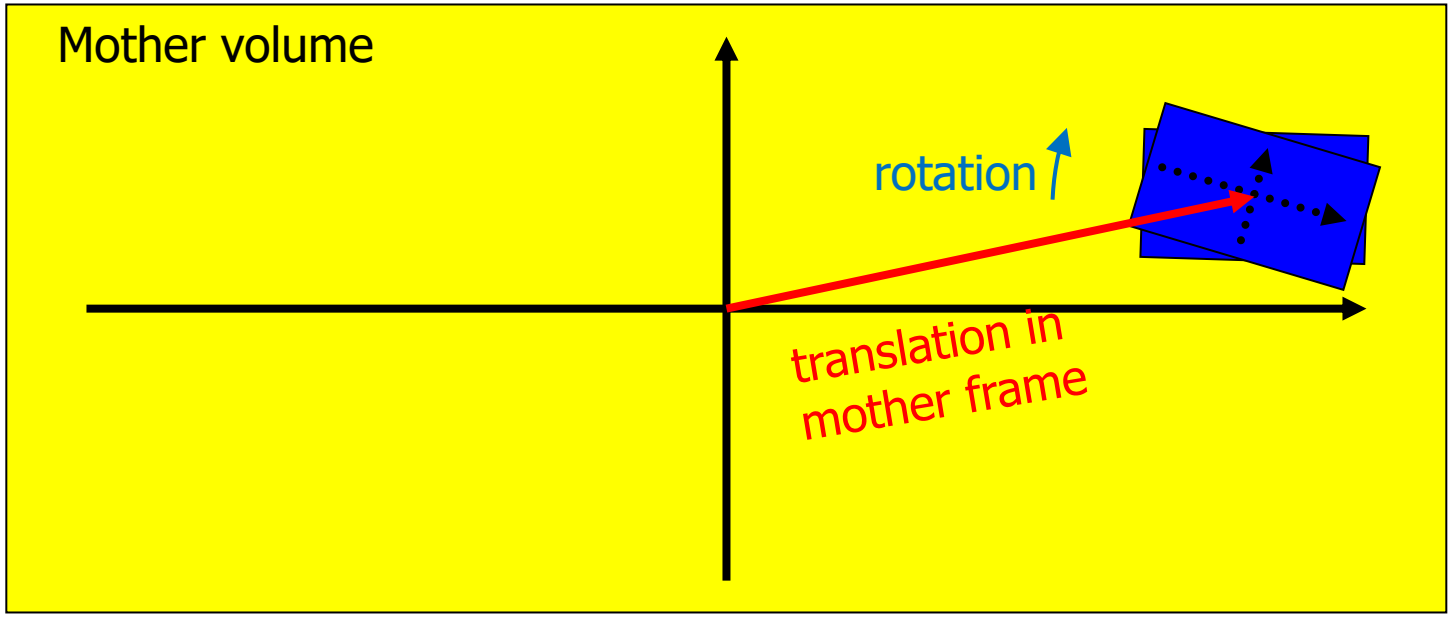
❄ G4AssemblyVolume　　1 Placement = a set of Placement volumes

- 单次摆放的一个合集。

G4PVPlacement

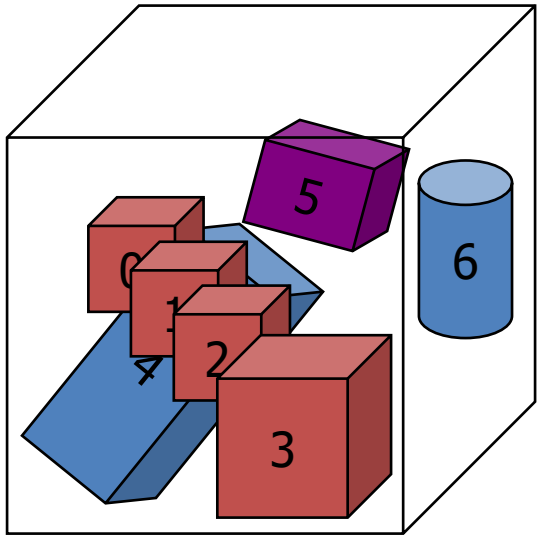

```
G4PVPlacement(G4RotationMatrix* pRot,   // rotation of mother frame
              const G4ThreeVector &tlate, // position in mother frame
              G4LogicalVolume *pDaughterLogical,
              const G4String &pName,
              G4LogicalVolume *pMotherLogical,
              G4bool pMany, // 'true' is not supported yet…
              G4int pCopyNo, // unique arbitrary integer
              G4bool pSurfChk=false); // optional boundary check
```

- **Single volume positioned relatively to the mother volume.**



G4PVParameterised

```
G4PVParameterised(const G4String& pName,
                  G4LogicalVolume* pLogical,
                  G4LogicalVolume* pMother,
                  const EAxis pAxis,
                  const G4int nReplicas,
                  G4VPVParameterisation* pPar,
                  G4bool pSurfChk=false);
```



- 在母体积pMother内，使用参数化方法pPar，重复摆放几何体nReplicas次。
- 用户需要实现一个自己的参数化几何类，以G4VPVParameterisation为基类，并实现下面的函数 (依赖于几何体编号):
  - where it is positioned (transformation, rotation)
  - the size of the solid (dimensions)
  - The type of the solid, material, sensitivity, vis attributes
- 对简单的CSG几何适用。

G4VPVParameterisation


```
class G4VPVParameterisation
{
  public:
   … …
   virtual void ComputeTransformation // position, rotation
     (const G4int copyNo, G4VPhysicalVolume* physVol) const;
   virtual void ComputeDimensions // size
     (G4Box& trackerLayer, const G4int copyNo,
         const G4VPhysicalVolume* physVol) const;
   … …
   virtual G4VSolid* ComputeSolid // shape
     (const G4int copyNo, G4VPhysicalVolume* physVol);
   virtual G4Material* ComputeMaterial // material, sensitivity, visAtt
     (const G4int copyNo, G4VPhysicalVolume* physVol,
         const G4VTouchable *parentTouch=0);
       // G4VTouchable should not be used for ordinary parameterization
};
```

G4PVParameterised例子 (B2b)

B2bDetectorConstruction.cc

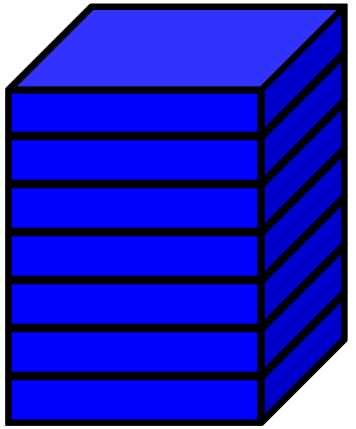
```
209   G4Tubs* chamberS
210     = new G4Tubs("tracker", 0,  100*cm,  100*cm,  0.*deg,  360.*deg);
211   fLogicChamber
212     = new G4LogicalVolume(chamberS,fChamberMaterial,"Chamber",0,0,0);
213
214   G4double firstPosition = -trackerSize + chamberSpacing;
215   G4double firstLength   = trackerLength/10;
216   G4double lastLength    = trackerLength;
217
218   G4VPVParameterisation* chamberParam =
219     new B2bChamberParameterisation(
220                                NbOfChambers,    // NoChambers
221                                firstPosition,   // Z of center of first
222                                chamberSpacing,  // Z spacing of centers
223                                chamberWidth,   // chamber width
224                                firstLength,     // initial length
225                                lastLength);     // final length
226
227   // dummy value : kZAxis -- modified by parameterised volume
228
229   new G4PVParameterised("Chamber",       // their name
230                         fLogicChamber,   // their logical volume
231                         trackerLV,       // Mother logical volume
232                         kZAxis,          // Are placed along this axis
233                         NbOfChambers,    // Number of chambers
234                         chamberParam,    // The parametrisation
235                         fCheckOverlaps); // checking overlaps
```

Replicated Volumes

- 子几何体完全填满母体积，子体积具有相同的尺寸和形状。

a daughter logical volume to be replicated

- 重复摆放几何时可以沿着下面的坐标轴进行：
  - **笛卡尔坐标轴(X, Y, Z)**
  - **径向坐标轴(Rho) – 共心的圆柱/圆锥形状**
  - **方位角(Phi) – 扇形**



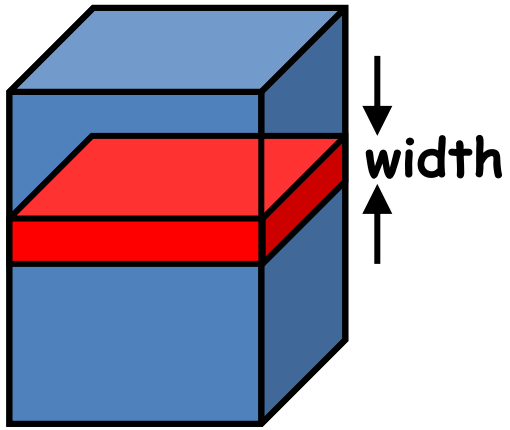
mother volume

❄ 笛卡尔坐标轴 - kXaxis, kYaxis, kZaxis

- 第n个子体积的中心在：

  `-width*(nReplicas-1)*0.5+n*width`

- Offset在这种情况不起作用。

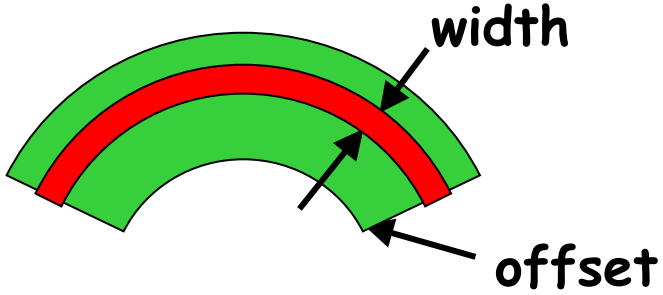


❄ 径向坐标轴 - kRaxis

- 第n个子体积的中心在：

  `width*(n+0.5)+offset`

- Offset的值必须是母体积的内半径。

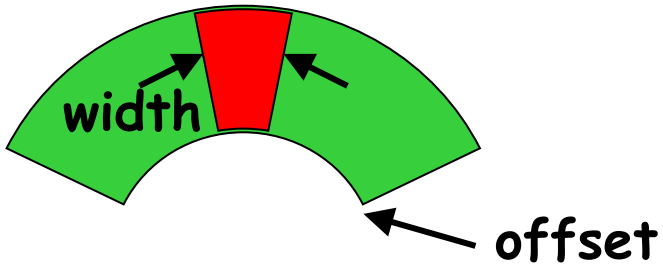


❄ Phi向坐标轴 - kPhi

- 第n个子体积的中心在：

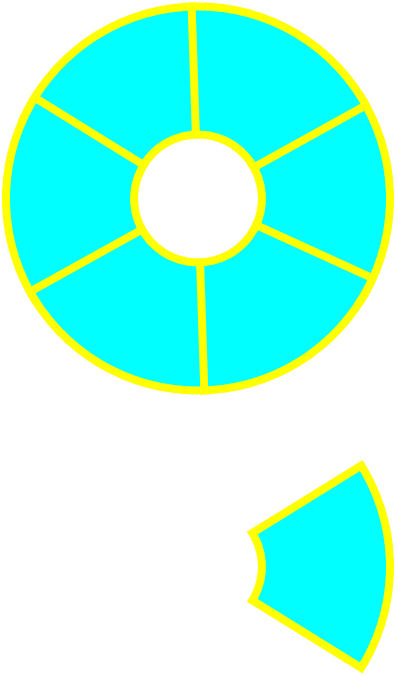
  `width*(n+0.5)+offset`

- Offset必须是母体积的初始角度。



G4PVReplica使用举例


```
G4double tube_dPhi = 2.* M_PI * rad;
G4VSolid* tube =  new G4Tubs("tube",20*cm,50*cm,30*cm,0.,tube_dPhi);
G4LogicalVolume * tube_log =  new G4LogicalVolume(tube, Air, "tubeL", 0, 0, 0);
G4VPhysicalVolume* tube_phys =  new G4PVPlacement(0,G4ThreeVector(-200.*cm,0.,0.),
      "tubeP", tube_log, world_phys, false, 0);

G4double divided_tube_dPhi = tube_dPhi/6.;

G4VSolid* div_tube = new G4Tubs("div_tube", 20*cm, 50*cm, 30*cm,
       -divided_tube_dPhi/2., divided_tube_dPhi/2.0);

G4LogicalVolume* div_tube_log = new G4LogicalVolume(div_tube,Pb,"div_tubeL",0,0,0);

G4VPhysicalVolume* div_tube_phys =   new G4PVReplica("div_tube_phys", div_tube_log,
  tube_log, kPhi, 6, divided_tube_dPhi);
```



如何构建详细的人体结构?

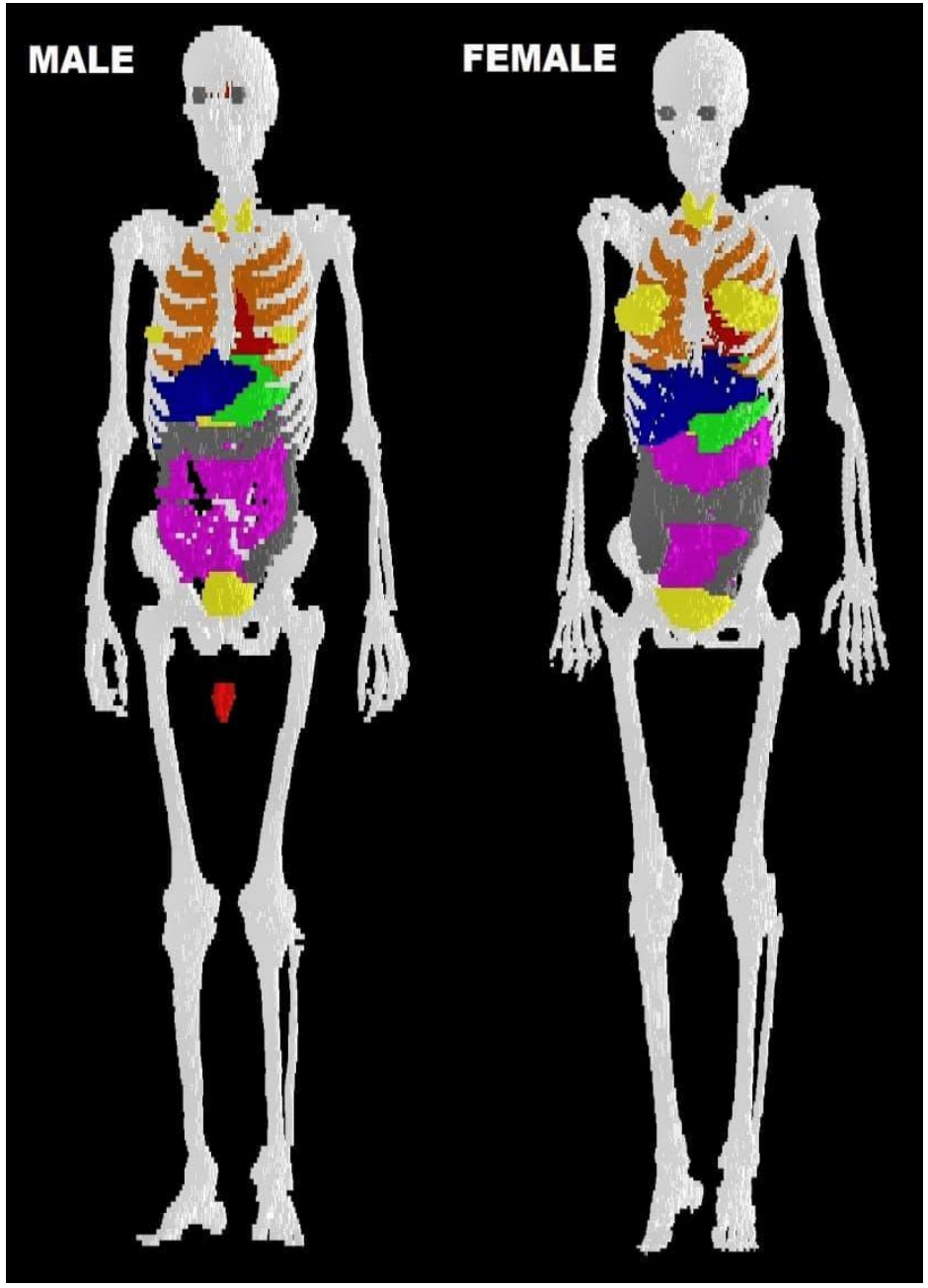
利用N个微小的格子(G4Box)构建人体所在空间
⇨ G4PVReplica

不同格子填充对应的物质
⇨ G4PVNestedParameterisation

需要实测数据

构建方法虽然简单，但占用内存空间较大

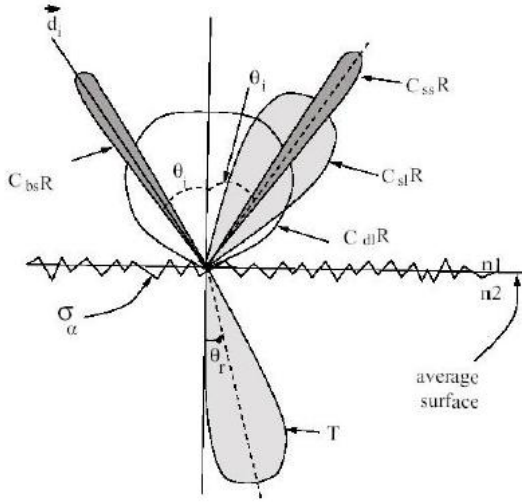
参考G4高级例子：ICRP110_HumanPhantoms



定义光学界面

需要在探测器构建类中实现!!!

```
G4VPhysicalVolume* volume1;
G4VPhysicalVolume* volume2;

G4OpticalSurface* OpSurface = new G4OpticalSurface("name");
G4LogicalBorderSurface* Surface = new G4LogicalBorderSurface("name", volume1, volume2, OpSurface);

G4double sigma_alpha = 0.1;
OpSurface -> SetType(dielectric_dielectric);
OpSurface -> SetModel(unified);
OpSurface -> SetFinish(groundbackpainted);
OpSurface -> SetSigmaAlpha(sigma_alpha);
const G4int NUM = 2;
G4double pp[NUM] = {2.038*eV, 4.144*eV};
G4double specularspike[NUM] = {1.0, 1.0};
G4double rindex[NUM] = {1.35, 1.40};
G4double reflectivity[NUM] = {0.3, 0.5};
G4double efficiency[NUM] = {0.8, 0.1};

G4MaterialPropertiesTable* SMPT = new G4MaterialPropertiesTable();
SMPT -> AddProperty("RINDEX", pp, rindex, NUM);
SMPT -> AddProperty("SPECULARSPIKECONSTANT", pp, specularlobe, NUM);
SMPT -> AddProperty("REFLECTIVITY", pp, reflectivity, NUM);
SMPT -> AddProperty("EFFICIENCY", pp, efficiency, NUM);

OpSurface -> SetMaterialPropertiesTable(SMPT);
```



Polar plot of the radiant intensity in the UNIFIED model

参考 “Section 5.2.5 of Geant4 Application Developers Guide ”

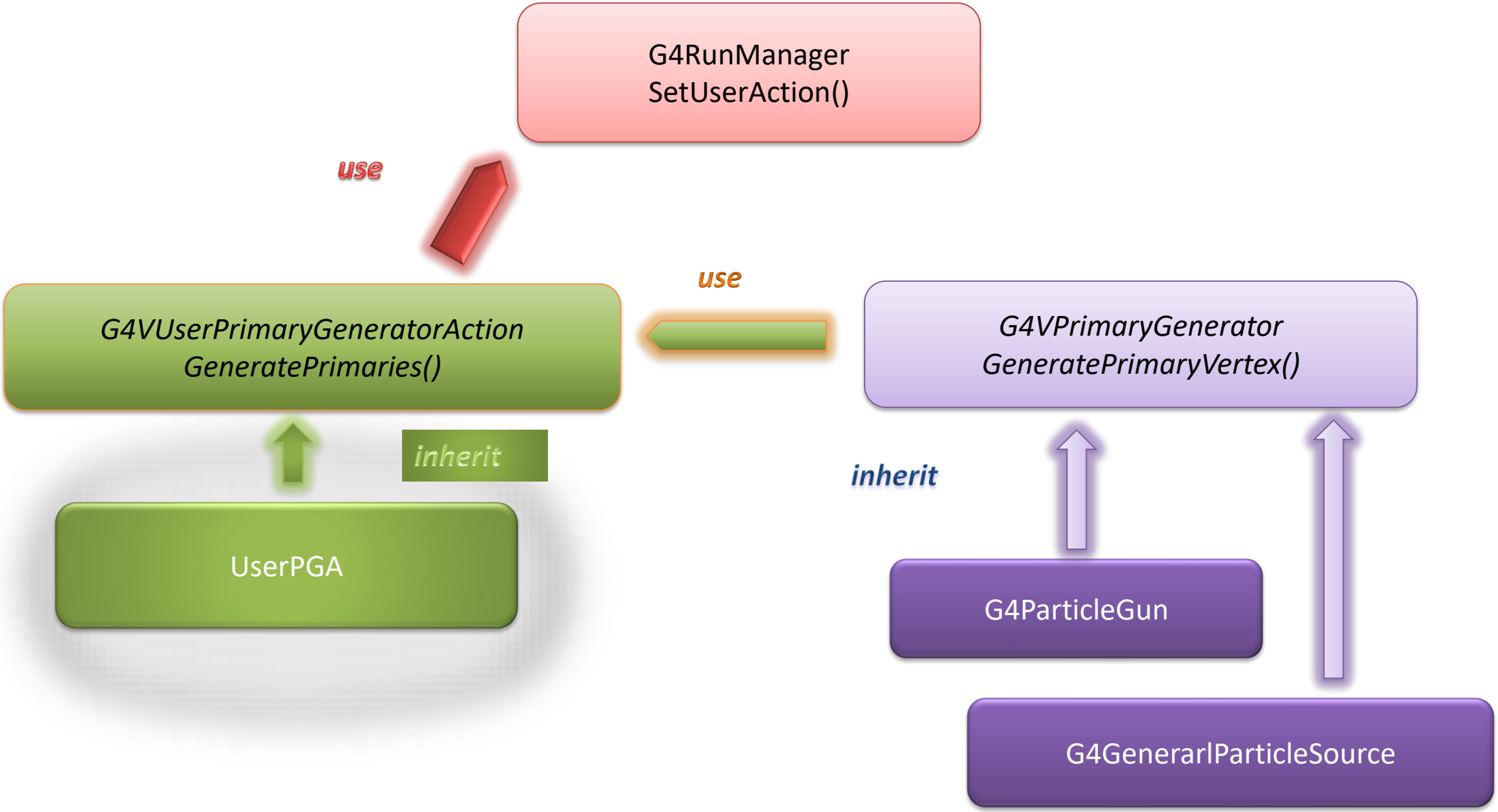
练习任务 (3)

- 基于B1例子，完成模拟程序所需几何形状的定义和摆放，移除B1中原有的几何

- 完成编译并显示新构建的几何形状

- **用户使用事例产生子时需要继承于其基类 G4VUserPrimaryGeneratorAction**
  - 是探测器模拟所必须的一个user action
- **Geant4内部提供的事例产生子**
  - 粒子枪 (G4ParticleGun)
  - 更高级一些的粒子产生器 (GPS) (G4GeneralParticleSource)
- **外部事例产生子与Geant4接口 (HEPEvt)**

产生子及用户操作的调用关系



用户产生子操作类的实现方法

- 用户的产生子操作类 (PGA class)需要继承其基类G4VPrimaryGeneratorAction
- 在PGA类成员函数GeneratePrimaries()中，调用产生子类的GeneratePrimaryVertex()函数
- Geant4中内建的产生子
  - G4ParticleGun
  - G4GeneralParticleSource (GPS)

```
// constructor
void T01PrimaryGeneratorAction::T01PrimaryGeneratorAction
{
  // particleGun is a member of this class.
  particeGun = new G4ParticleGun();
}

void T01PrimaryGeneratorAction::GeneratePrimaries
(G4Event* anEvent)
{
  particleGun-> GeneratePrimaryVertex(anEvent);
}
```

粒子枪产生子 (G4ParticleGun)

- *G4VPrimaryGenerator*的具体实现
- 粒子枪用来在特定点、特定时间以一个特定方向，向探测器内打入具有特定能量的粒子
  - 具备一系列基本功能
- 可以用UI命令来设置初值：

/gun/list	List available particles
/gun/particle	Set particle type to be generated
/gun/direction	Set momentum direction
/gun/energy	Set kinetic energy
/gun/momentum	Set momentum
/gun/momentumAmp	Set absolute value of momentum
/gun/position	Set starting position of the particle
/gun/time	Set initial time of the particle
/gun/polarization	Set polarization
/gun/number	Set number of particles to be generated (per event)
/gun/ion	Set properties of ion to be generated [usage] /gun/ion Z A Q

如何使用G4ParticleGun?


```
void T01PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent) {
  G4ParticleDefinition* particle;
  G4int i = (int)(5. * G4UniformRand());
  switch(i) {
    case 0: particle = positron;  break;
    case 1: ...
  }
  particleGun-> SetParticleDefinition(particle);

  G4double momentum = 100.*MeV;
  G4double pp = momentum + (G4UniformRand()-0.5) * sigmaMomentum;
  G4double mass = particle-> GetPDGMass();
  G4double Ekin = sqrt(pp*pp+mass*mass) - mass;
  particleGun-> SetParticleEnergy(Ekin);

  G4double sigmaAngle = 10.*deg;
  G4double angle = (G4UniformRand()-0.5) * sigmaAngle;
  particleGun-> SetParticleMomentumDirection(G4ThreeVector(sin(angle),0.,cos(angle)));

  particleGun-> GeneratePrimaryVertex(anEvent);
}
```

- **粒子产生的位置可以通过几种不同的方式进行抽样**
  - 从一个点、面、长方体、… 发射粒子。
- **粒子角分布**
  - 提供了几种不同的角分布：isotropic, cosine-law, focused, …
  - 同时也提供了一些控制参数 (min/max-theta, min/max-phi,…)
- **粒子的动能分布**
  - 提供了一些基本分布 (e.g. mono-energetic, power-law)
  - 此外，还有一些额外的分布(e.g. black-body)，或用户自定义的分布。
- **多个粒子源**
  - 需要用户定义不同源之间的相对亮度。
- **具备增加事例抽样权重的功能 (event biasing)**
  - 提高粒子类型、顶点分布、能量和方向分布的权重。

使用GPS的例子


```
// constructor
MyPrimaryGeneratorAction::MyPrimaryGeneratorAction()
{
  // gps is a member of this class
  gps= new G4GeneralParticleSource();
}

void MyPrimaryGeneratorAction::GeneratePrimaries(G4Event*
anEvent)
{
      gps-> GeneratePrimaryVertex(anEvent);
}
```

- **提供了很多的UI命令供用户选择使用**

- **下面的链接提供了很多使用GPS的例子，可供参考。**
  - http://hurel.hanyang.ac.kr/Geant4/Geant4_GPS/reat.space.qinetiq.com/gps/examples/examples.html

```
/gps/particle proton

/gps/ene/type Mono
/gps/ene/mono 500 MeV

/gps/pos/type Plane
/gps/pos/shape Rectangle
/gps/pos/rot1 0 0 1
/gps/pos/rot2 1 0 0
/gps/pos/halfx 46.2 cm
/gps/pos/halfy 57.2 cm
/gps/pos/centre 0. 57.2 0. cm

/gps/direction 0 -1 0
```

protons

mono energetic beam of 500 MeV

planar emission from a zx plane along -y axis

参考:
G4GeneralParticleSourceMessenger

❄ G4HEPEvtInterface

- 使用/HEPEVT/格式，对很多基于FORTRAN的产生子都适用
- ASCII文件输入

```
     411
2    25    2    3  0.19538240e+02  0.24846369e+02 -0.60465937e+01  0.30378159e+03
2    23    4    5  0.11302123e+01 -0.23156443e+02  0.11416953e+03  0.89038048e+02
2    23    6    7  0.18408028e+02  0.48002811e+02 -0.12021612e+03  0.90217377e+02
1    13
1   -13
1    11    0    0  0.56072354e+00 -0.13617630e+02 -0.84170624e+02  0.50999998e-03
1   -11    0    0  0.17847303e+02  0.61620441e+02 -0.36045494e+02  0.50999998e-03
2    -2   51   51 -0.24825256e+01 -0.14315886e+01  0.28472370e+00  0.55999998e-02
```

Extended/runAndEvent/RE05: pythia_event.data

NHEP
ISTHEP IDHEP JDAHEP1 JDAHEP2 PHEP1 PHEP2 PHEP3 PHEP5
ISTHEP IDHEP JDAHEP1 JDAHEP2 PHEP1 PHEP2 PHEP3 PHEP5
... [NHEP times]
NHEP
ISTHEP IDHEP JDAHEP1 JDAHEP2 PHEP1 PHEP2 PHEP3 PHEP5
ISTHEP IDHEP JDAHEP1 JDAHEP2 PHEP1 PHEP2 PHEP3 PHEP5
... [NHEP times]

```
ISTHEP   = status code
IDHEP    = HEP PDG code
JDAHEP   = first daughter
JDAHEP   = last daughter
PHEP1    = px in GeV
PHEP2    = py in GeV
PHEP3    = pz in GeV
PHEP5    = mass in GeV
```

使用HEPEvt (N04)的例子


```
ExN04PrimaryGeneratorAction::ExN04PrimaryGeneratorAction()
{
  const char* filename = "pythia_event.data";
  HEPEvt = new G4HEPEvtInterface(filename);
}

ExN04PrimaryGeneratorAction::~ExN04PrimaryGeneratorAction()
{
  delete HEPEvt;
}

void ExN04PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  HEPEvt->GeneratePrimaryVertex(anEvent); }
}
```

练习任务 (4)

- 基于B1例子，修改产生子部分，读入HEPEvt格式的IBD产生子文件，在探测器内均匀产生IBD事例
  https://ihepbox.ihep.ac.cn/ihepbox/index.php/s/Zq9BYXhjSnvVHyH

- 完成编译并尝试运行10个事例

❄ 什么是physics list?

- 首先它是一个类，该类包含了用户探测器模拟程序里面所有的粒子定义，物理过程和产生阈
- 需要告诉G4RunManager来调用物理相互作用
- 构建你自己物理列表的方式是很灵活的
  - 你可以选择你需要的粒子种类
  - 你可以给每种粒子选择想要的物理过程
- 但是, 你必须对物理过程有一个较好的了解
  - 忽略一些粒子或物理过程的定义可能导致错误或较差的模拟结果

我们为什么需要物理列表?

❄ **Geant4无法提供一个完整的物理过程列表给每个用户使用**

- Geant4中具有太多的物理模型和近似处理
  - 这对强相互作用过程尤为常见
  - 同时也存在于电磁相互作用中
- 运行速度也是一个问题
- 没有用户的需求是这样的：要Geant4中提供的所有物理过程/模型和粒子

❄ **基于这一原因，**

- Geant4提供的许多物理过程之间都是相互独立的
- 用户选择这些物理过程的方式和方法都是一致的

Geant4提供的物理过程

- **电磁相互作用**
  - 标准电磁过程，从 ~ 1 keV 到 ~PeV
  - 低能电磁过程，从 250 eV 到 ~PeV
  - 光学光子过程
- **弱相互作用**
  - 基本粒子的衰变
  - 原子核的放射性衰变
- **强相互作用**
  - 纯粹的强过程，从 0 到 ~TeV
  - 电子和伽玛与核的相互作用，从 10 MeV 到 ~TeV
- **参数化或“快速模拟”物理过程**

- **用户自己的物理列表类需要继承于该类**
  - 之后需要将该物理列表告诉 G4RunManager

- **例子:**

```
Class MyPhysicsList: public G4VUserPhysicsList {
    public:
    MyPhysicsList();
    ~MyPhysicsList();
    void ConstructParticle();
    void ConstructProcess();
    void SetCuts();
 }
```

- **用户需要具体实现ConstructParticle(), ConstructProcess()和SetCuts()**

ConstructParticle()


```
Void MyPhysicsList::ConstructParticle() {
   //construct boson
   G4Gamma::GammaDefinition();
   G4OpticalPhoton::OpticalPhotonDefinition();
   //construct lepton
   G4Electron::ElectronDefinition();
   … …
   //construct meson
   G4PionPlus::PionPlusDefinition();
   … …
   //construct baryon
   G4Proton::ProtonDefinition();
   … …
   //construct ion
   G4Alpha::AlphaDefinition();
   G4AntiAlpha::AntiAlphaDefinition();
   … …
   G4GenericIon::GenericIonDefinition();
   //construct short lived particle
   G4ParticleDefinition* particle = new G4Gluons(***)
   particle->SetAntiPDGEncoding(21);
   … …
}
```

另外一种方式

```
Void MyPhysicsList::ConstructParticle() {
   G4BosonConstructor::ConstructParticle();
   G4LeptonConstructor::ConstructParticle();
   G4MesonConstructor::ConstructParticle();
   G4BaryonConstructor::ConstructParticle();
   G4IonConstructor::ConstructParticle();
   G4ShortLivedConstructor::ConstructParticle();
}
```

轻子和玻色子

Particle name	Class name	Name (in GPS...)	PDG
electron	G4Electron	e-	11
positron	G4Positron	e+	-11
muon +/-	G4MuonPlus G4MuonMinus	mu+ mu-	-13 13
tauon +/-	G4TauPlus G4TauMinus	tau+ tau-	-15 15
electron (anti)neutrino	G4NeutrinoE G4AntiNeutrinoE	nu_e anti_nu_e	12 -12
muon (anti)neutrino	G4NeutrinoMu G4AntiNeutrinoMu	nu_mu anti_nu_mu	14 -14
tau (anti)neutrino	G4NeutrinoTau G4AntiNeutrinoTau	nu_tau anti_nu_tau	16 -16
photon (γ, X)	G4Gamma	gamma	22
photon (optical)	G4OpticalPhoton	opticalphoton	(0)
geantino	G4Geantino	geantino	(0)
charged geantino	G4ChargedGeantino	chargedgeantino	(0)

强子和离子等

Particle name	Class name	Name (in GPS...)	PDG
(anti)proton	G4Proton G4AnitProton	proton anti_proton	2212 -2212
(anti)neutron	G4Neutron G4AntiNeutron	neutron anti_neutron	2112 -2112
(anti)lambda	G4Lambda G4AntiLambda	lambda anti_lambda	3122 -3122
pion	G4PionMinus G4PionPlus G4PionZero	pi- pi+ pi0	-211 211 111
kaon	G4KaonMinus G4KaonPlus G4KaonZero G4KaonZeroLong G4KaonZeroShort	kaon- kaon+ kaon0 kaon0L kaon0S	-321 321 311 130 310
(anti)alpha	G4Alpha G4AntiAlpha	alpha anti_alpha	1000020040 -1000020040
(anti)deuteron	G4Deteuron G4AntiDeuteron	deuteron anti_deuteron	1000010020 -1000010020
Heavier ions	G4Ions	ion	100ZZZAAAI*

*ZZZ=proton number, AAA=nucleon number, I=excitation level

ConstructProcess()


```
void MyPhysicsList::ConstructProcess() {
    //method provided by G4VUserPhysicsList assigns
    //transportation process to all particles defined in ConstructParticle()
    AddTransportation();

    //put EM physics here, defined by user
   ConstructEM();

    //put hadronic physics here, defined by user
    ConstructHad();

    //put optical processes here, defined by user
    ConstructOP();

    //put all other processes here, defined by user
    ConstructGeneral();
}
```

ConstructEM()


```
void MyPhysicsList::ConstructEM() {
   G4PhyscisListHelper* ph = G4PhysicsListHelper::GetPhysicsListHelper();
    theParticleIterator->reset();
    while( (*theParticleIterator)() ) {
      G4ParticleDefinition* particle = theParticleIterator->value();
       if(particle == G4Gamma::Gamma()) {
         ph->RegisterProcess(new G4GammaConversion(), particle);
         … … // add more processes
       }
       if(particle == G4Electron::Electron()) {
         G4eMultipleScattering* msc = new G4eMultipleScattering;
         G4UrbanMscModel95* msc1 = new G4UrbanMscModel95();
         G4WentzelVIModel* msc2 = new G4WentzelVIModel();
          msc1->SetHighEnergyLimit(highEnergyLimit);
          msc2->SetLowEnergyLimit(highEnergyLimit);
          msc->AddEmModel(0, msc1);
          msc->AddEmModel(1, msc2);
         ph->RegisterProcess(msc, particle);
         … … // add more processes
       }
       … … //define processes for other particles
    }
}
```

ConstructHad()


```
void MyPhysicsList::ConstructHad() {
   G4PhyscisListHelper* ph = G4PhysicsListHelper::GetPhysicsListHelper();
    theParticleIterator->reset();
    while( (*theParticleIterator)() ) {
      G4ParticleDefinition* particle = theParticleIterator->value();
       if(particle == G4Neutron::NeutronDefinition()) {
         // define elastic scattering
         G4HadronElasticProcess* elasticproc = new G4HadronElasticProcess;
         G4HadronElastic* elasticModel = new G4HadronElastic;
         elasticModel->SetMinEnergy(19*MeV);
         G4NeutronHPElastic* elasticHP = new G4NeutronHPElastic;
         elasticHP->SetMaxEnergy(19*MeV);
         elasticproc->RegisterMe(elasticModel);
         elasticproc->RegisterMe(elasticHP);
         elasticproc->AddDataSet(new G4NeutronHPElasticData());
         ph->RegisterProcess(elasticproc, particle);
          … … // inelastic scattering
          … … // capture
          … … // fission
       }
       … … //define processes for other particles
    }
}
```

ConstructOP()


```
void MyPhysicsList::ConstructOP() {
  G4Cerenkov* theCerenkovProcess = new G4Cerenkov("Cerenkov");
  G4Scintillation* theScintillationProcess = new G4Scintillation("Scintillation");
  G4OpAbsorption* theAbsorptionProcess = new G4OpAbsorption();
  G4OpRayleigh* theRayleighScatteringProcess = new G4OpRayleigh();
  G4OpBoundaryProcess* theBoundaryProcess = new G4OpBoundaryProcess();
  theScintillationProcess->AddSaturation(G4LossTableManager::Instance()->EmSaturation());

  theParticleIterator->reset();
   while( (*theParticleIterator)() ) {
     G4ParticleDefinition* particle = theParticleIterator->value();
      G4ProcessManager* pmanager = particle->GetProcessManager();
      G4String particleName = particle->GetParticleName();
      if(theCerenkovProcess->IsApplicable(*particle)) {
        pmanager->AddProcess(theCerenkovProcess);
        pmanager->SetProcessOrdering(theCerenkovProcess, idxPostStep);
      }
      if(theScintillationProcess->IsApplicable(*particle)) { …}
      if(particleName == “opticalphoton”) {
        pmanager->AddDiscreteProcess(theAbsorptionProcess);
        pmanager->AddDiscreteProcess(theRayleighScatteringProcess);
        pmanager->AddDiscreteProcess(theBoundaryProcess);
      }
   }
}
```

ConstructGeneral()


```
void MyPhysicsList::ConstructGeneral() {
    G4PhyscisListHelper* ph = G4PhysicsListHelper::GetPhysicsListHelper();
    G4Decay* theDecayProcess = new G4Decay();
    theParticleIterator->reset();
    while( (*theParticleIterator)() ) {
       G4ParticleDefinition* particle = theParticleIterator->value();
       if(theDecayProcess->IsApplicable(*particle)) {
         ph->RegisterProcess(theDecayProcess, particle);
       }
       … … // Add other processes
    }
}
```

```
void MyPhysicsList::SetCuts() {
    defaultCutValue = 0.7*mm;
    SetCutValue(defaultCutValue, “gamma”);
    SetCutValue(defaultCutValue, “e-”);
    SetCutValue(defaultCutValue, “e+”);
    SetCutValue(defaultCutValue, “proton”);

    // These are all the production cuts you need to set, not
   // required for any other particle.
}
```

G4VModularPhysicsList – 第二种方式

❄ G4VModularPhysicsList的特征

- 从G4VUserPhysicsList继承
- AddTransportation()自动被添加
- 直接使用已经构建好的physics constructors
- 允许用户定义自己的物理模块，如：电磁过程，强过程，光学过程等等

```
class MyPhysicsList : public G4VModularPhysicsList {
public:
    MyPhysicsList();              // define physics constructors
    void ConstructParticle();     // optional
    void ConstructProcess();      // optional
    void SetCuts();               // optional
}
```

如何使用G4VModularPhysicsList?


```
MyPhysicsList::MyPhysicsList() {
   defaultCutValue = 0.7*mm;
   //define hadronic processes here
   RegisterPhysics(new G4HadronElasticPhysics());
   //define EM processes here
   RegisterPhysics(new G4EmStandardPhysics());
}

void MyPhysicsList::SelectAlternativePhysics() {
   //add a physics in the created list
   AddPhysics(new G4OpticalPhysics);
   //remove a physics from the list
   RemovePhysics(fDecayPhysics);
   //replace a physics with another one
   ReplacePhysics(new G4EmLivermorePhysics);
}

void MyPhysicsList::SetCuts() {
    SetCutsWithDefault();
}
```

```
MyPhysicsList::MyPhysicsList() {
   defaultCutValue = 0.7*mm;
   emList = new G4EmLivermorePhysics();
   emExtraList = new G4EmExtraPhysics();
   decayList = new G4DecayPhysics();
   raddecayList = new G4RadioactiveDecayPhysics();
   hadronESList = new G4HadronElasticPhysicsHP();
   hadronList = new G4HadronPhysicsQGSP_BERT_HP();
   stoppingList = new G4StoppingPhysics();
   ionList = new G4IonPhysicsPHP();
}
void MyPhysicsList::ConstructParticle() {
   //Construct particles
   … …
}
Void MyPhysicsList::ConstructProcess() {
   emList->ConstructProcess();
   emExtraList->ConstructProcess();
   … …
}
void MyPhysicsList::SetCuts() {
    SetCutsWithDefault();
}
```

Geant4提前定义好的Physics constructors

EM physics lists:

G4EmStandardPhysics – default (ATLAS, BESIII, …)
G4EmStandardPhysics_option1 – HEP fast but not precise (CMS)
G4EmStandardPhysics_option2 – Experimental (LHCb)
G4EmStandardPhysics_option3 – medical (most accurate standard)
G4EmStandardPhysics_option4 – standard+low (most accurate EM)

G4EmLivermorePhysics
G4EmLivermorePolarizedPhysics
G4EmPenelopePhysics
G4EmDNAPhysics

Combined Physics
Standard > 1 GeV
LowEnergy < 1 GeV

Hadronic elastic:
G4HadronElasticPhysics, …

https://geant4.kek.jp/lxr/source/physics_lists/constructors/

Hadronic inelastic:
G4HadronPhysicsQGSP_BIC, G4HadronPhysicsFTFP_Bert, …

Decay, optical physics, EM extras, …

- **包含电磁过程和强过程等**
  - 每一个打包好的物理列表都包含不同电磁过程和强过程的组合
  - 打包好的物理列表可以在这里找到
    geant4/source/physics_lists/lists/include

- **在这些打包好的物理列表中，有一些是Geant4推荐使用的 (“reference” physics lists)**
  - 这些物理列表都经过很好地维护和测试
  - 这些物理列表在多个高能物理实验中被使用(BESIII, ATLAS, CMS, ...)

- **如何使用?**
  - 参考B1例子，exampleB1.cc
    runManager->SetUserInitialization(new QBBC);

https://geant4.kek.jp/lxr/source/physics_lists/lists/

选择合适的物理列表

❄ **Geant4一共有多个打包好的物理列表**

- FTFP_BERT
- QBBC
- QGSP_BERT
- QGSP_BIC
- Shielding

https://geant4-userdoc.web.cern.ch/UsersGuides/PhysicsListGuide/html/index.html

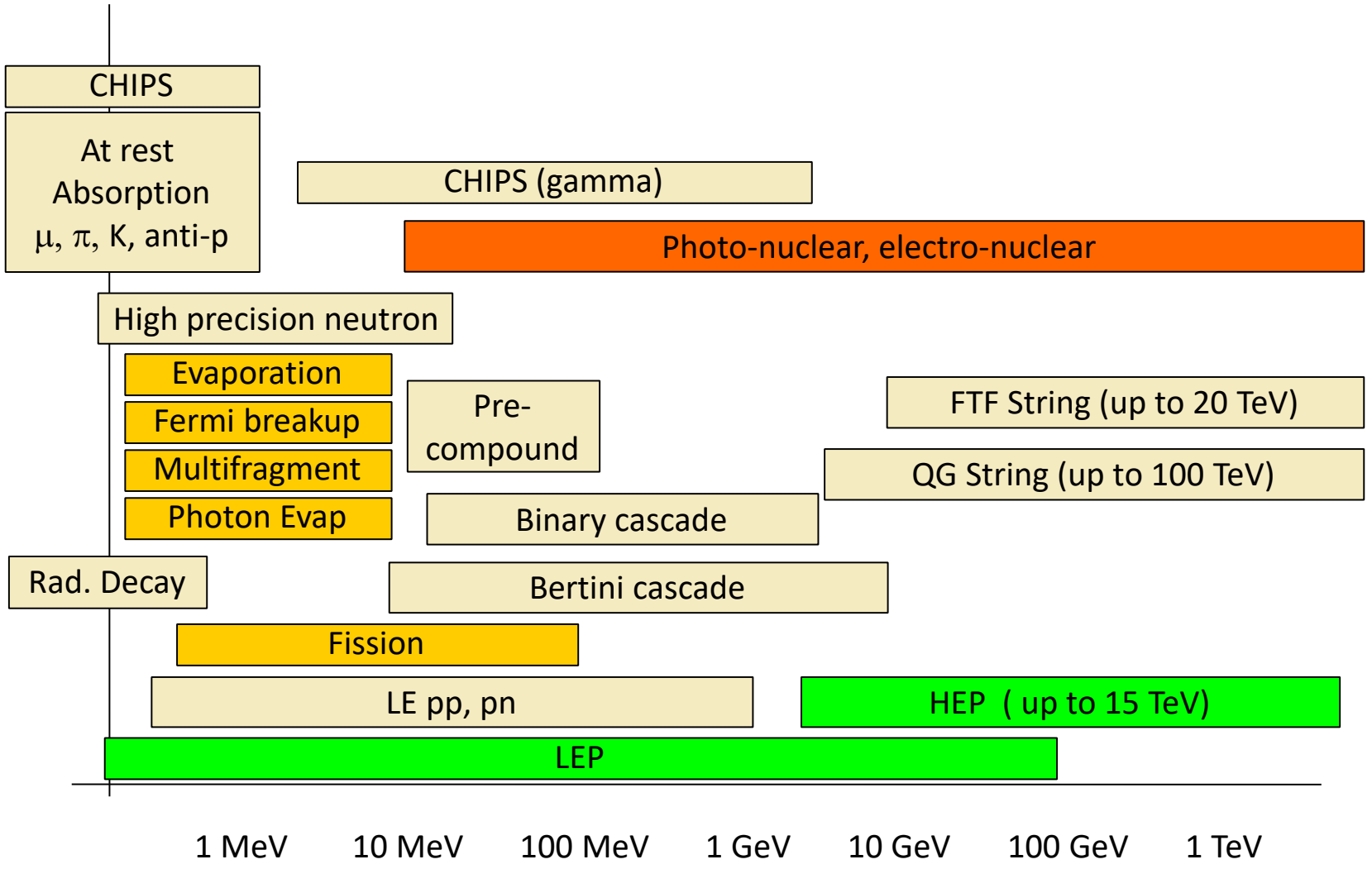
❄ **强子过程命名规则**

- QGS → Quark Gluon String model (>~20 GeV)
- FTF → Fritiof string model (>~5 GeV)
- BIC → Binary Cascade (< ~10 GeV)
- BERT → Bertini-style cascade (<~10 GeV)
- HP → High Precision neutron model (< 20 MeV)
- P → G4Precompund model used for de-excitation

❄ **EM过程通过后缀**

- No suffix: standard physics
- EMV suffix: older but faster EM processes
- Other suffixes for other EM options

强子过程所用的模型


CHIPS
At rest
Absorption
μ, π, K, anti-p
CHIPS (gamma)
Photo-nuclear, electro-nuclear
High precision neutron
Evaporation
Fermi breakup
Multifragment
Photon Evap
Pre-
compound
FTF String (up to 20 TeV)
QG String (up to 100 TeV)
Binary cascade
Rad. Decay
Bertini cascade
Fission
LE pp, pn
HEP (up to 15 TeV)
LEP
1 MeV
10 MeV
100 MeV
1 GeV
10 GeV
100 GeV
1 TeV

- **选择物理列表时需考虑：**
  - 物理性能
  - 计算性能

- **对大多数物理过程，Geant4都提供了validation的结果**
  - 这是一个持续的过程，并随时会保持更新。

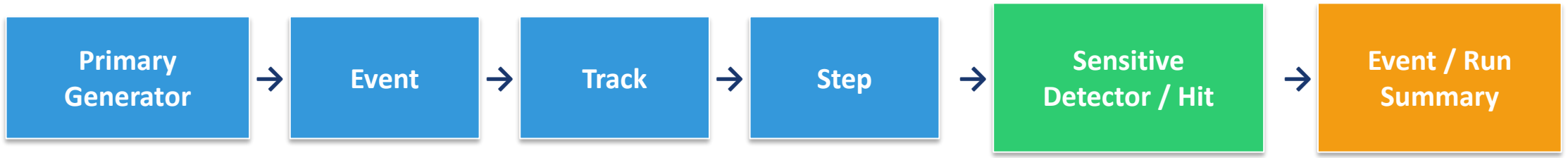
- **模拟与数据的对比结果可以在下面的链接找到**

  https://geant-val.cern.ch/

练习任务 (5)

- 基于B1例子，构建模拟程序所需的物理列表，要求使用低能电磁过程，中子过程使用HP
- 使用新物理列表类替换B1原有的物理列表
- 完成编译并运行10个事例

Geant4 中信息获取的总体框架



原始动态信息主要产生在 Step

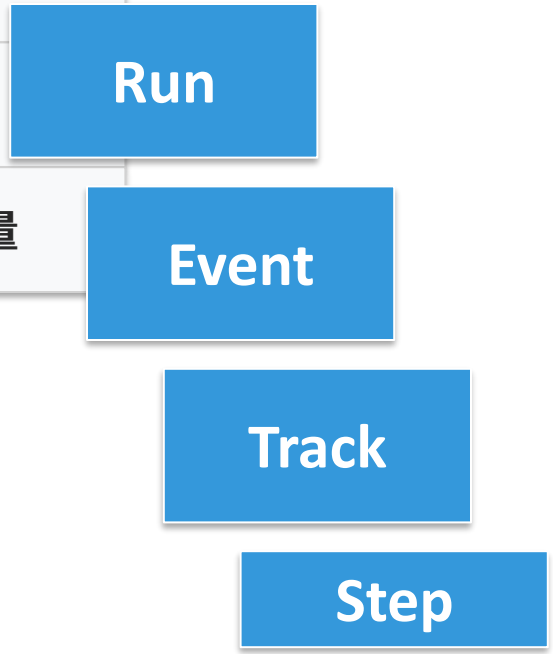
用户在不同层级进行汇总：

- **Step → Event**
- **Event → Run**
- **Hit → Detector Response**

最终输出到 Histogram / Ntuple / ROOT 文件

Geant4 的信息层级结构

层级	含义	典型类	关注信息
Run	一次运行	G4Run	总统计、均值、误差
Event	一个物理事例	G4Event	总沉积、触发、Hit集合
Track	一个粒子的径迹	G4Track	粒子种类、能量、父子关系
Step	一次传播步	G4Step	edep、位置、步长、过程
Hit	探测器命中	User Hit	时间、位置、单元编号、能量



- **Given geometry, physics and primary track generation, Geant4 does proper physics simulation "silently"**
  - **You have to add a bit of code to extract information useful to you**

- **There are two ways:**
  - **Use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)**
    - **You have an access to almost all information.**
    - **Straight-forward, but do-it-yourself**
  - **Use Geant4 scoring functionality**
    - **Assign G4VSensitiveDetector to a volume**
    - **Hits collection is automatically stored in G4Event object, and automatically accumulated if user-defined Run object is used.**
    - **Use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary**

Use your own sensitive detector (SD)

- Very common way for a large project
  - Highly customizable, save whatever you want
- Create a class of SD and register it to the SD Manager
- Assign the pointer of your SD to the logical volume of your detector geometry
- Must be done in ConstructSDandField() of the user geometry

```
G4VSensitiveDetector* mySensitive  = new MySensitiveDetector(SDname="/MyDetector");

G4SDManager* sdMan =G4SDManager::GetSDMpointer();

sdMan->AddNewDetector(mySensitive);

SetSensitiveDetector("LVname",mySensitive); // or

//logicVol->SetSensitiveDetector(mySensitive);
```

HitCollection in your SD

- Create a class of Hit, and define the variables that you want to save
- Should be derived from G4VHit
- The HitCollection is automatically managed by Geant4, and can be accessed in user hooks
- Add hits to HitCollection in your SD
- Refer to examples/basic/B2/B2a

```
class myHit : public G4VHit {
  public:
  …
  const myHit& operator=(const myHit&);
  G4int operator==(const myHit&) const;

  void SetTrackID  (G4int track)  { fTrackID = track; };
  G4int GetTrackID() const   { return fTrackID; };

  private:
 G4int  fTrackID;
};

typedef G4THitsCollection<myHit> myHitsCollection;
```

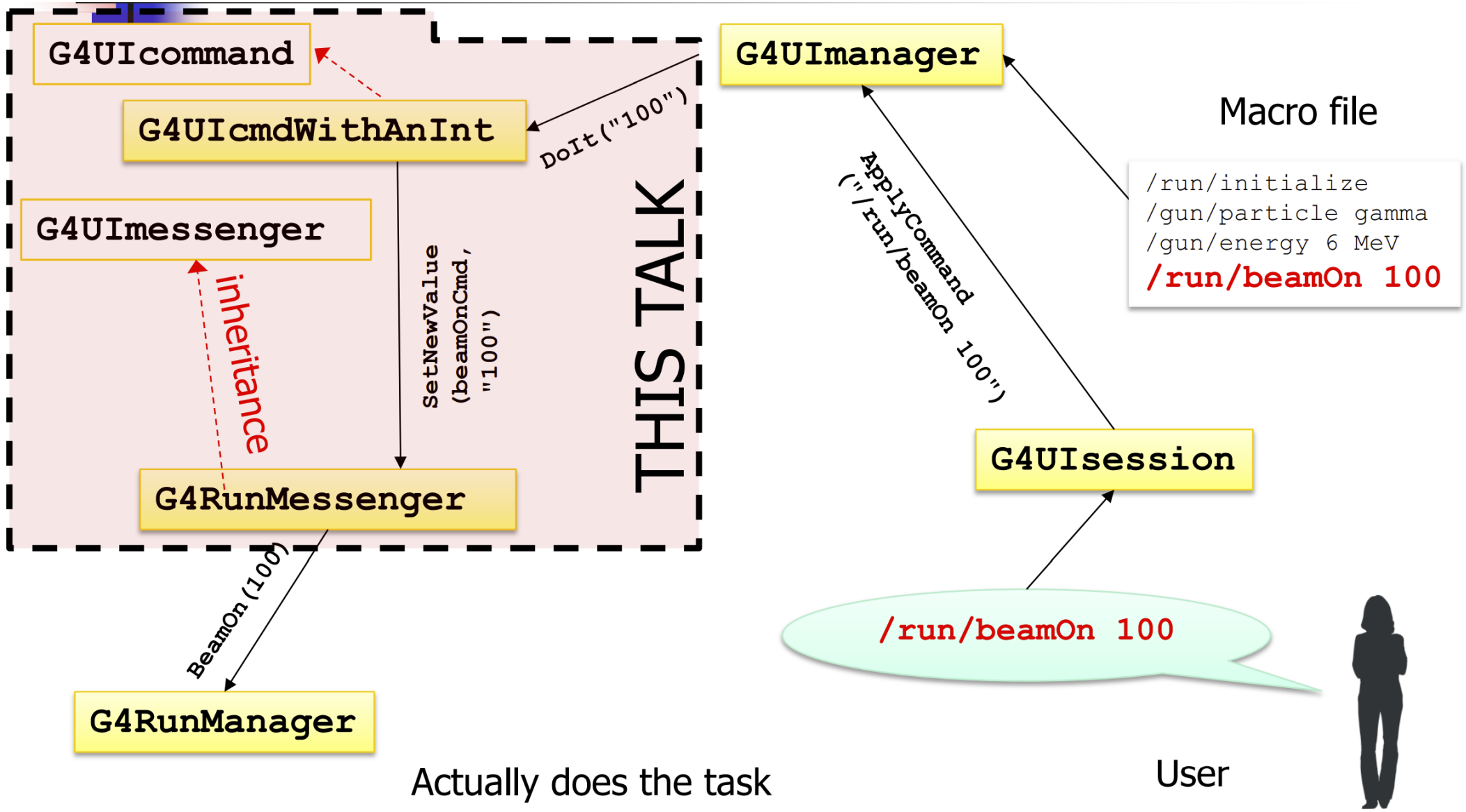
```
G4bool mySD::ProcessHit(G4Step* astep) {
  myHit*  newHit = new myHit();
  newHit->SetTrackID  (aStep->GetTrack()->GetTrackID());
  fHitsCollection->insert( newHit );

  return true;
};

void mySD::EndOfEvent(G4HCofThisEvent*)
{
   G4int nofHits = fHitsCollection->entries();
   for ( G4int i=0; i<nofHits; i++ )
   (*fHitsCollection)[i]->Print();
}
```

练习任务 (6)

- 基于B1例子，完成模拟程序所需信息的存储
- 完成编译，运行少量事例，检查结果是否正确
- 批量运行所需的IBD事例样本

- **UI session**
  - G4UIExective
  - G4UIterminal
  - Batch mode
- **UI commands (/run/beamOn n), can be used**
  - in a UI session
  - in a macro file
  - by hard-coded implementation
- **Messenger**
  - If built-in commands are not enough, you define your own command.

UI Flow Chart



❄ There are built-in commands roughly organized according to Geant4 categories

```
Idle> ls
Command directory path : /
Sub-directories :
/control/   UI control commands.
/units/     Available units.
/geometry/  Geometry control commands.
/tracking/  TrackingManager and SteppingManager control commands.
/event/     EventManager control commands.
/run/       Run control commands.
/random/    Random number status control commands.
/particle/  Particle control commands.
/process/   Process Table control commands.
/material/  Commands for Materials
/vis/       Visualization commands.
/gun/       Particle Gun control commands.
```

A complete list of built-in commands is available in the Geant4 Application Developers Guide, Chapter 7.1

Built-in command types

G4UIcmdWithoutParameter: /run/initialize
G4UIcmdWithAnInteger: /run/beamOn 10
G4UIcmdWithABool: /process/em/auger true*
G4UIcmdWithADouble: /particle/property/decay/br 0.5
G4UIcmdWithADoubleAndUnit: /gps/time 1.5 us
G4UIcmdWithAString: /gps/particle e-
G4UIcmdWith3Vector: /gps/direction 0.707 0.707 0.0
G4UIcmdWith3VectorAndUnit:
/gps/position 0.0 0.0 0.0 m

***Note:** The **true** values are **t**, **true**, **y**, **yes** and **1** (case insensitive); everything else is **false**.

MyClassMessenger.hh

```
#include <G4UImessenger.hh>
#include <G4UIdirectory.hh>
#include <G4UIcmdWithADouble.hh>
#include <G4UIcmdWithoutParameters.hh>

#include "MyClass.hh"

class MyClassMessenger : public G4UImessenger
{
public:
    MyClassMessenger(MyClass* myObject);

    ~MyClassMessenger();

    void SetNewValue(G4UIcommand* command, G4String newValue) override;

private:

    MyClass* fObject;
    G4UIdirectory* fDirectory;
    G4UIcmdWithADouble* fSomeParameterCommand;
    G4UIcmdWithoutParameters* fDoSomethingCommand;
};
```

Geant4 headers

Header to the class to manage

Inherit from **G4UImessenger**

Commands are defined in constructor

Method reacting to **all command calls** managed by this messenger

MyClassMessenger.cc

```
#include "MyClassMessenger.hh"

MyClassMessenger::~MyClassMessenger() {
    delete fSomeParameterCommand;
    delete fDoSomethingCommand;
    delete fDirectory;
}

MyClassMessenger::MyClassMessenger(MyClass* myObject)
    : fObject(myObject) {
    fDirectory = new G4UIdirectory("/myClass/");
    fDirectory->SetGuidance("Commands for MyClass");

    fSomeParameterCommand =
    // ...
```

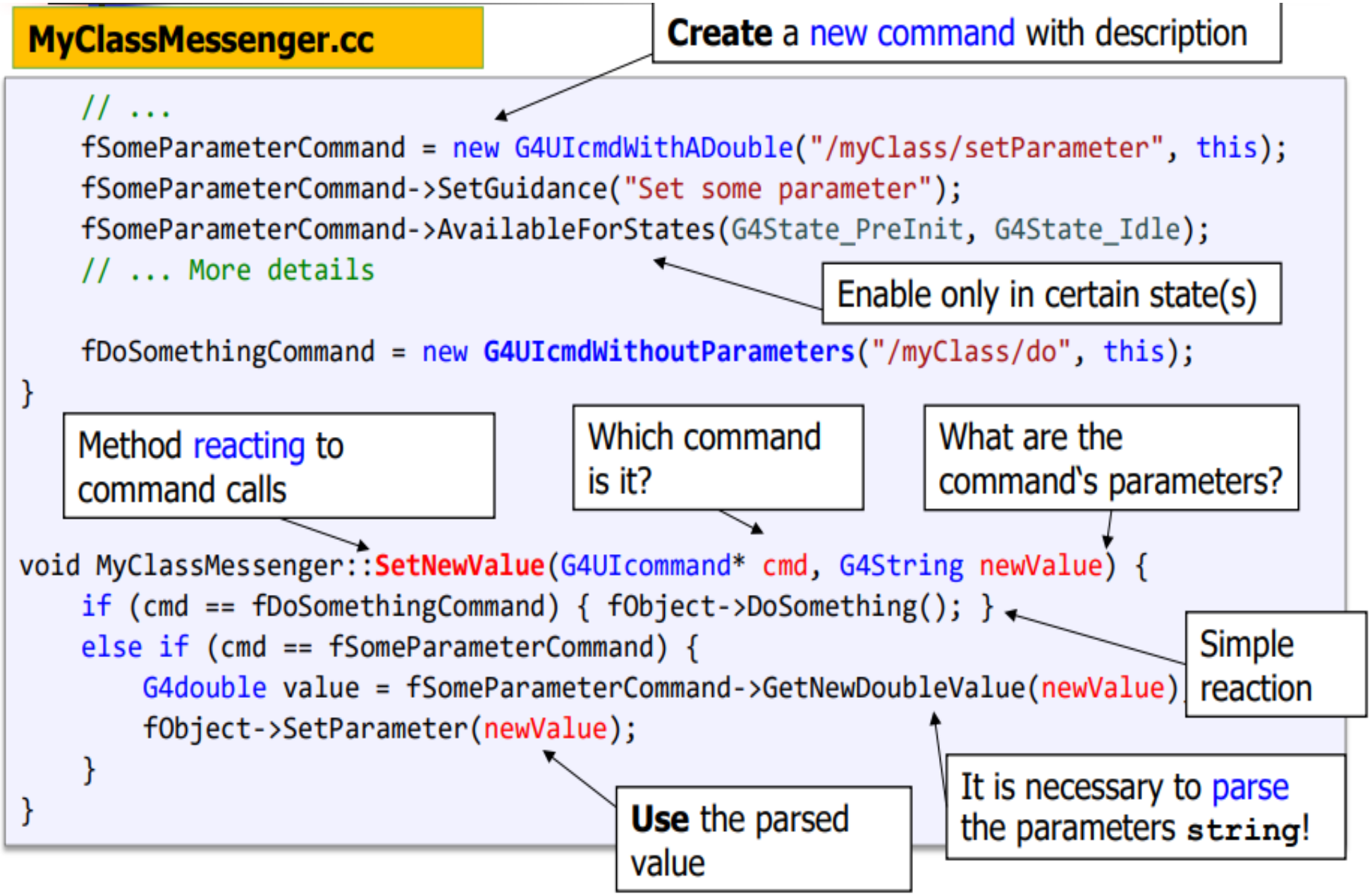
Include the header

UI command and directory pointers should be deleted!

Assign the pointer to the **associated** object

Create the command directory (with description)

To be continued in the next slide...


MyClassMessenger.cc
Create a new command with description
// ...
fSomeParameterCommand = new G4UIcmdWithADouble("/myClass/setParameter", this);
fSomeParameterCommand->SetGuidance("Set some parameter");
fSomeParameterCommand->AvailableForStates(G4State_PreInit, G4State_Idle);
// ... More details
Enable only in certain state(s)
fDoSomethingCommand = new G4UIcmdWithoutParameters("/myClass/do", this);
}
Method reacting to command calls
Which command is it?
What are the command's parameters?
void MyClassMessenger::SetNewValue(G4UIcommand* cmd, G4String newValue) {
if (cmd == fDoSomethingCommand) { fObject->DoSomething(); }
else if (cmd == fSomeParameterCommand) {
G4double value = fSomeParameterCommand->GetNewDoubleValue(newValue);
fObject->SetParameter(newValue);
}
}
Simple reaction
Use the parsed value
It is necessary to parse the parameters string!

Backup

Tessellated Solids


```
// First declare a tessellated solid
G4TessellatedSolid* solidTarget = new G4TessellatedSolid("Solid_name");
G4double targetSize = 10*cm ;
G4TriangularFacet *facet1 = new   G4TriangularFacet (G4ThreeVector(-targetSize, -targetSize,       0.0),
                                                     G4ThreeVector(+targetSize, -targetSize,       0.0),
                                                     G4ThreeVector(       0.0,       0.0,     +targetSize),
                                                     ABSOLUTE);
G4TriangularFacet *facet2 = new G4TriangularFacet (G4ThreeVector(+targetSize,  -targetSize,       0.0),
                                                    G4ThreeVector(+targetSize, +targetSize,       0.0),
                                                    G4ThreeVector(       0.0,       0.0, +targetSize),
                                                    ABSOLUTE);
```

```
G4TriangularFacet ( const G4ThreeVector     Pt0,
                    const G4ThreeVector     vt1,
                    const G4ThreeVector     vt2,
                          G4FacetVertexType fType )
```

i.e., it takes 4 parameters to define the three vertices:

`G4FacetVertexType`	`ABSOLUTE` in which case `Pt0`, `vt1` and `vt2` are the three vertices in anti-clockwise order looking from the outside.
`G4FacetVertexType`	`RELATIVE` in which case the first vertex is `Pt0`, the second vertex is `Pt0+vt1` and the third vertex is `Pt0+vt2`, all in anti-clockwise order when looking from the outside.

Multi-Union Structures


```
#include "G4MultiUnion.hh"

// Define two -G4Box- shapes
G4Box* box1 = new G4Box("Box1", 5.*mm, 5.*mm, 10.*mm);
G4Box* box2 = new G4Box("Box2", 5.*mm, 5.*mm, 10.*mm);

// Define displacements for the shapes
G4RotationMatrix rotm  = G4RotationMatrix();
G4ThreeVector position1 = G4ThreeVector(0.,0.,1.);
G4ThreeVector position2 = G4ThreeVector(0.,0.,2.);
G4Transform3D tr1 = G4Transform3D(rotm,position1);
G4Transform3D tr2 = G4Transform3D(rotm,position2);

// Initialise a MultiUnion structure
G4MultiUnion* munion_solid = new G4MultiUnion("Boxes_Union");

// Add the shapes to the structure
munion_solid->AddNode(*box1,tr1);
munion_solid->AddNode(*box2,tr2);

// Finally close the structure
munion_solid->Voxelize();

// Associate it to a logical volume as a normal solid
G4LogicalVolume* lvol = new G4LogicalVolume(munion_solid,    munion_mat,    "Boxes_Union_LV");
```

Since release 10.4, the possibility to define multi-union structures is part of the standard set of constructs in Geant4. A G4MultiUnion structure allows for the description of a Boolean union of many displaced solids at once, therefore representing volumes with the same associated material.

探测器中电磁场定义

❄ 可以在ConstructSDandField()函数中实现

➢ World内均匀磁场

方式一：

```
G4ThreeVector field(0,1.*tesla,0);
G4GlobalMagFieldMessenger* fMagFieldMessenger =
 new G4GlobalMagFieldMessenger(field)
```

方式二：

```
G4UniformMagField* magField = new G4UniformMagField(field);
G4FieldManager* fieldMgr =
G4TransportationManager::GetTransportationManager()
->GetFieldManager();
fieldMgr->SetDetectorField(magField);
fieldMgr->CreateChordFinder(magField);
```

❄ 方式二中，用户可以设置Stepper的精度和数值积分所使用的方法

探测器中电磁场定义

❄ 可以在ConstructSDandField()函数中实现

- ➢ 在World内定义非均匀磁场：创建用户的磁场类，从G4MagneticField类继承，实现GetFieldValue()函数

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

Geant4手册：http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/ForApplicationDeveloper/html/Detector/electroMagneticField.html#an-overview-of-propagation-in-a-field

Geant4例子：examples/extended/field

- ➢ 在某特定几何体内定义非均匀磁场：

```
MyField* myField = new MyField();
G4FieldManager* localFieldMgr = new G4FieldManager(myField);
G4bool allLocal = true;
logicVolWithField ->SetFieldManager(localFieldMgr, allLocal);
```

- **Run, event, track, step, step point**
- **Track ←→ trajectory, step ←→ trajectory point**
- **Process**
  - **At rest, along step, post step**
- **Cut = production threshold**

Run in Geant4

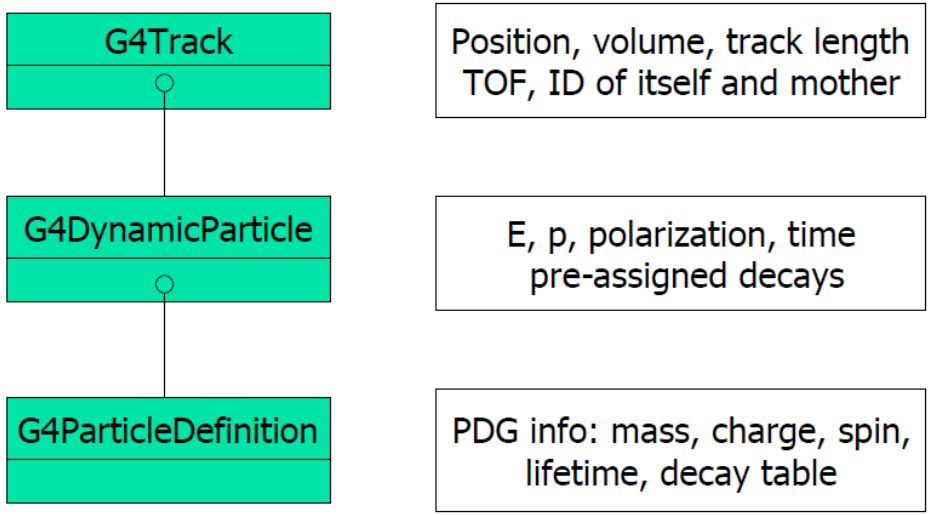
- As an analogy of the real experiment, a run of Geant4 starts with "Beam On"
- Within a run, the user cannot change
  - detector setup
  - settings of physics processes
- Conceptually, a run is a collection of events which share the same detector and physics conditions
  - A run consists of one event loop.
- At the beginning of a run, geometry is optimized for navigation and cross-section tables are calculated according to materials appear in the geometry and the cut-off values defined
- G4RunManager class manages processing a run, a run is represented by G4Run class or a user-defined class derived from G4Run
  - A run class may have a summary results of the run
- G4UserRunAction is the optional user hook

Event in Geant4

- An event is the basic unit of simulation in Geant4
- At beginning of processing, primary tracks are generated. These primary tracks are pushed into a stack
- A track is popped up from the stack one by one and "tracked". Resulting secondary tracks are pushed into the stack
  - This "tracking" lasts as long as the stack has a track
- When the stack becomes empty, processing of one event is over
- G4Event class represents an event. It has following objects at the end of its (successful) processing
  - List of primary vertices and particles (as input)
  - Hits and Trajectory collections (as output)
- G4EventManager class manages processing an event
- G4UserEventAction is the optional user hook

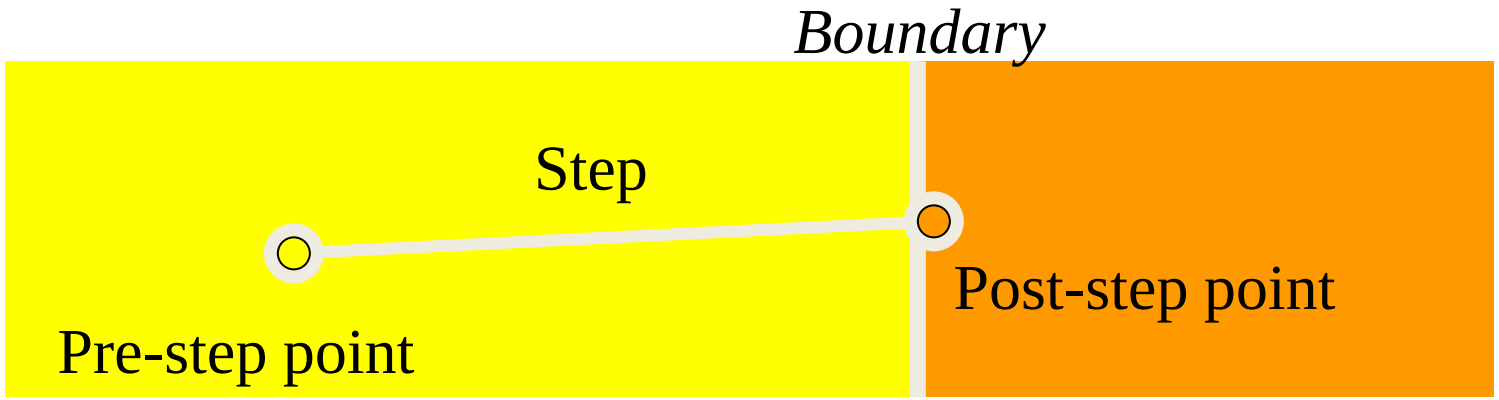
- **Track is a snapshot of a particle**
  - It has physical quantities of current instance only. It does not record previous quantities
  - Step is a "delta" information to a track. Track is not a collection of steps. Instead, a track is being updated by steps
- **Track object is deleted when**
  - it goes out of the world volume,
  - it disappears (by e.g. decay, inelastic scattering),
  - it goes down to zero kinetic energy and no "AtRest" additional process is required, or
  - the user decides to kill it artificially
- **No track object persists at the end of event**
  - For the record of tracks, use trajectory class objects
- **G4TrackingManager manages processing a track, a track is represented by G4Track class**
- **G4UserTrackingAction is the optional user hook**

- A particle in Geant4 is represented by three layers of classes
- G4Track
  - This is a class representing a particle to be tracked
- G4DynamicParticle
  - Each G4Track object has its own and unique G4DynamicParticle object
  - This is a class representing an individual particle
- G4ParticleDefinition
  - G4ProcessManager which describes processes involving to the particle
  - All G4DynamicParticle objects of same kind of particle share the same G4ParticleDefinition



Step in Geant4

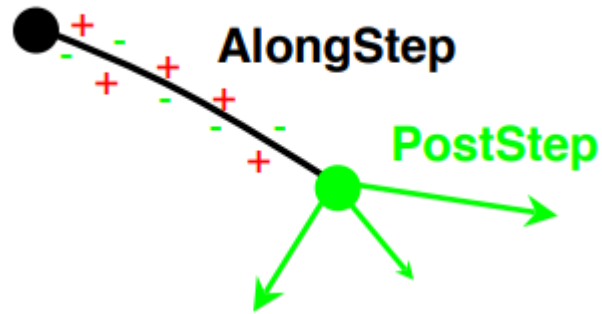
- **Step has two points and also "delta" information of a particle (energy loss on the step, time-of-flight spent by the step, etc.)**
- **Each point knows the volume (and material). In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume**
  - **Because one step knows materials of two volumes, boundary processes such as transition radiation or refraction could be simulated**
- **G4SteppingManager class manages processing a step, a step is represented by G4Step class**
- **G4UserSteppingAction is the optional user hook**



Trajectory and trajectory point

- **Track does not keep its trace. No track object persists at the end of event**
- **G4Trajectory is the class which copies some of G4Track information. G4TrajectoryPoint is the class which copies some of G4Step information**
  - **G4Trajectory has a vector of G4TrajectoryPoint.**
  - **At the end of event processing, G4Event has a collection of G4Trajectory objects.**
    - **/tracking/storeTrajectory must be set to 1.**
- **Keep in mind the distinction**
  - **G4Track ←→ G4Trajectory, G4Step ←→ G4TrajectoryPoint**
- **Given G4Trajectory and G4TrajectoryPoint objects persist till the end of an event, you should be careful not to store too many trajectories**
  - **E.g. avoid for high energy EM shower tracks.**
- **G4Trajectory and G4TrajectoryPoint store only the minimum information**
  - **You can create your own trajectory / trajectory point classes to store information you need. G4VTrajectory and G4VTrajectoryPoint are base classes.**

- **All the work of particle decays and interactions is done by processes**
  - Transportation is also handled by a process
- **A process does two things:**
  - Decides when and where an interaction will occur
    - Method: GetPhysicalInteractionLength()
  - Generates the final state (changes momentum, generates secondaries, etc)
    - Method: DoIt()
- **The process which requires the shortest interaction length limits the step**
- **Each process has one or combination of the following natures:**
  - Well-located in space → PostStep → Discrete process
    - e.g. decay on the fly, hadronic interaction, …
  - Not well-located in space → AlongStep
    → Continuous process
    - e.g. Cerenkov process, ionisation, …
  - Well-located in time → AtRest
    → AtRest process
    - e.g. muon decay at rest

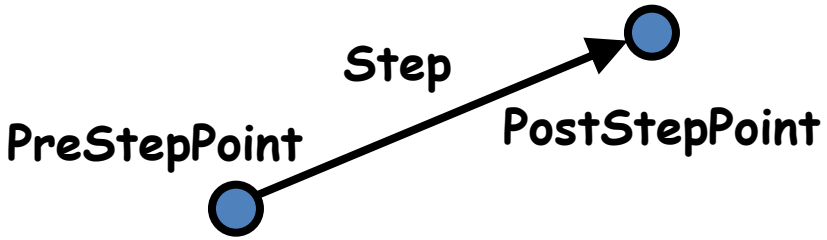


Track status

- **At the end of each step, according to the processes involved, the state of a track may be changed**
  - The user can also change the status in UserSteppingAction
  - Statuses shown in green are artificial, i.e. Geant4 kernel won't set them, but the user can set
- **fAlive**
  - Continue the tracking
- **fStopButAlive**
  - The track has come to zero kinetic energy, but still AtRest process to occur
- **fStopAndKill**
  - The track has lost its identity because it has decayed, interacted or gone beyond the world boundary
  - Secondaries will be pushed to the stack
- **fKillTrackAndSecondaries**
  - Kill the current track and also associated secondaries
- **fSuspend**
  - Suspend processing of the current track and push it and its secondaries to the stack
- **fPostponeToNextEvent**
  - Postpone processing of the current track to the next event
  - Secondaries are still being processed within the current event

Step status

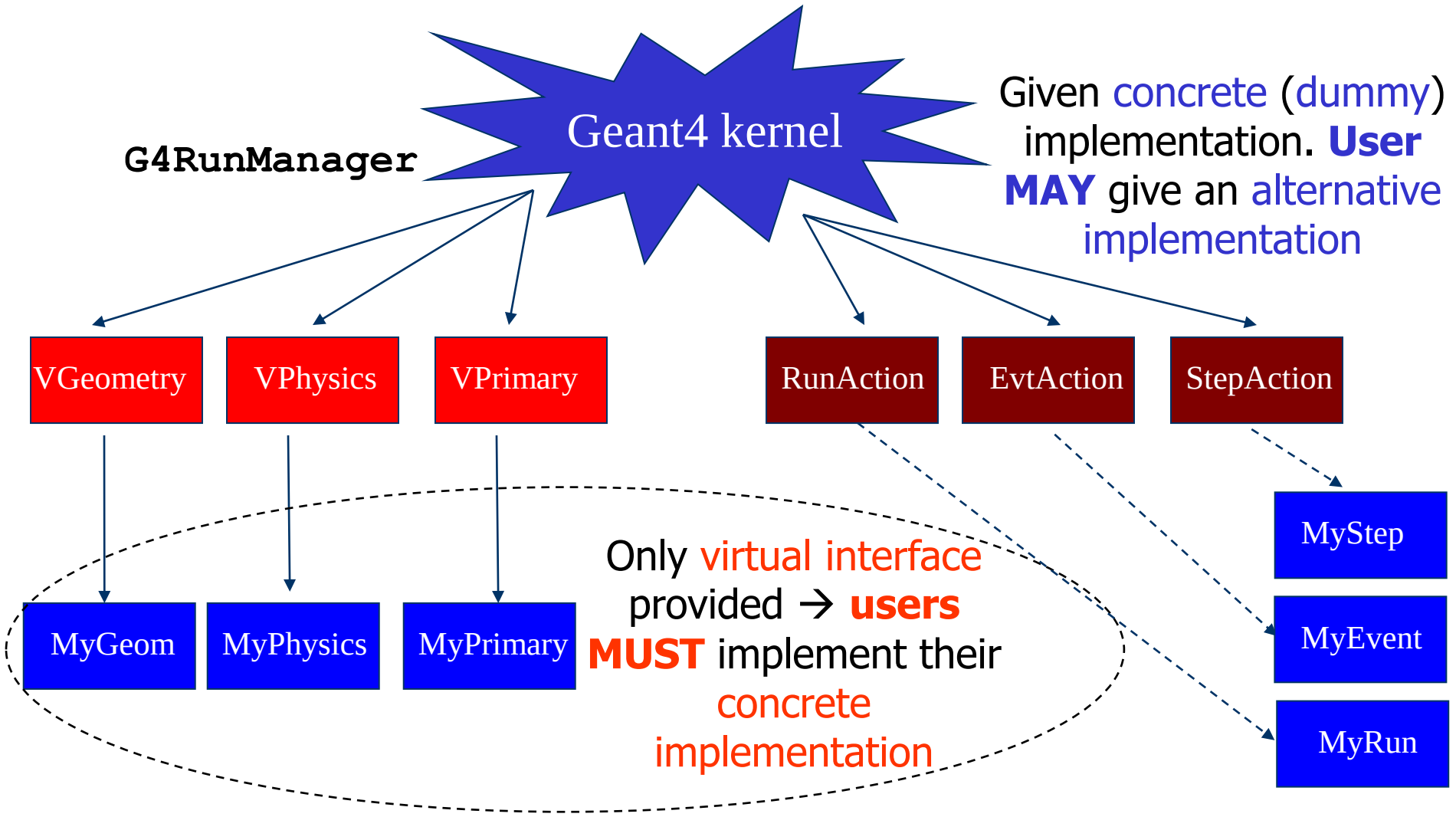
- Step status is attached to G4StepPoint to indicate why that particular step was determined.
  - Use "PostStepPoint" to get the status of this step.
  - "PreStepPoint" has the status of the previous step.



- fWorldBoundary
  - Step reached the world boundary
- fGeomBoundary
  - Step is limited by a volume boundary except the world
- fAtRestDoItProc, fAlongStepDoItProc, fPostStepDoItProc
  - Step is limited by a AtRest, AlongStep or PostStep process
- fUserDefinedLimit
  - Step is limited by the user Step limit
- fExclusivelyForcedProc
  - Step is limited by an exclusively forced (e.g. shower parameterization) process
- fUndefined
  - Step not defined yet

- If you want to identify the first step in a volume, pick fGeomBoudary status in PreStepPoint.
- If you want to identify a step getting out of a volume, pick fGeomBoundary status in PostStepPoint.

Interaction with Geant4 kernel



Initialisation classes

Invoked at the initialization

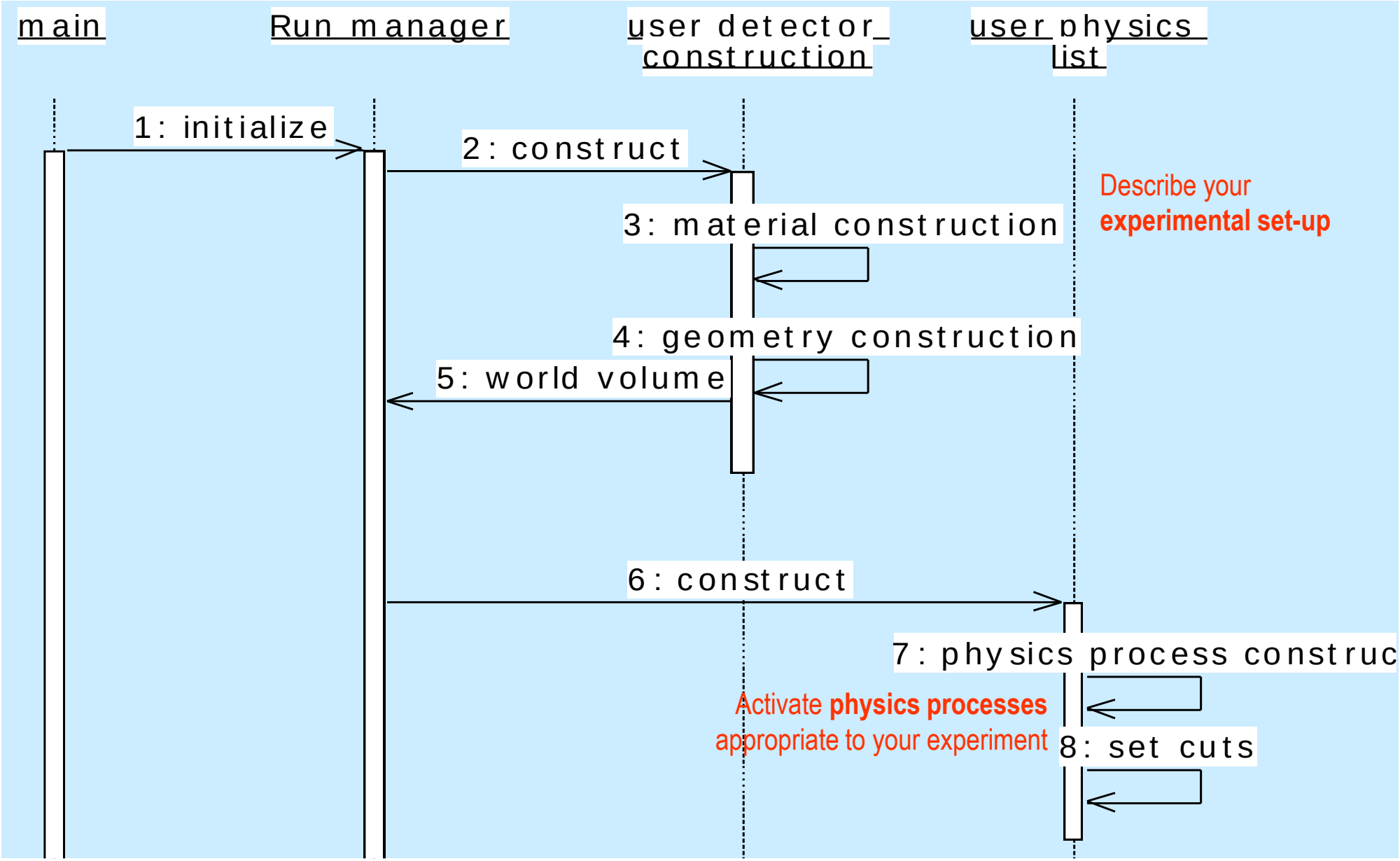
- **G4VUserDetectorConstruction**
- **G4VUserPhysicsList**

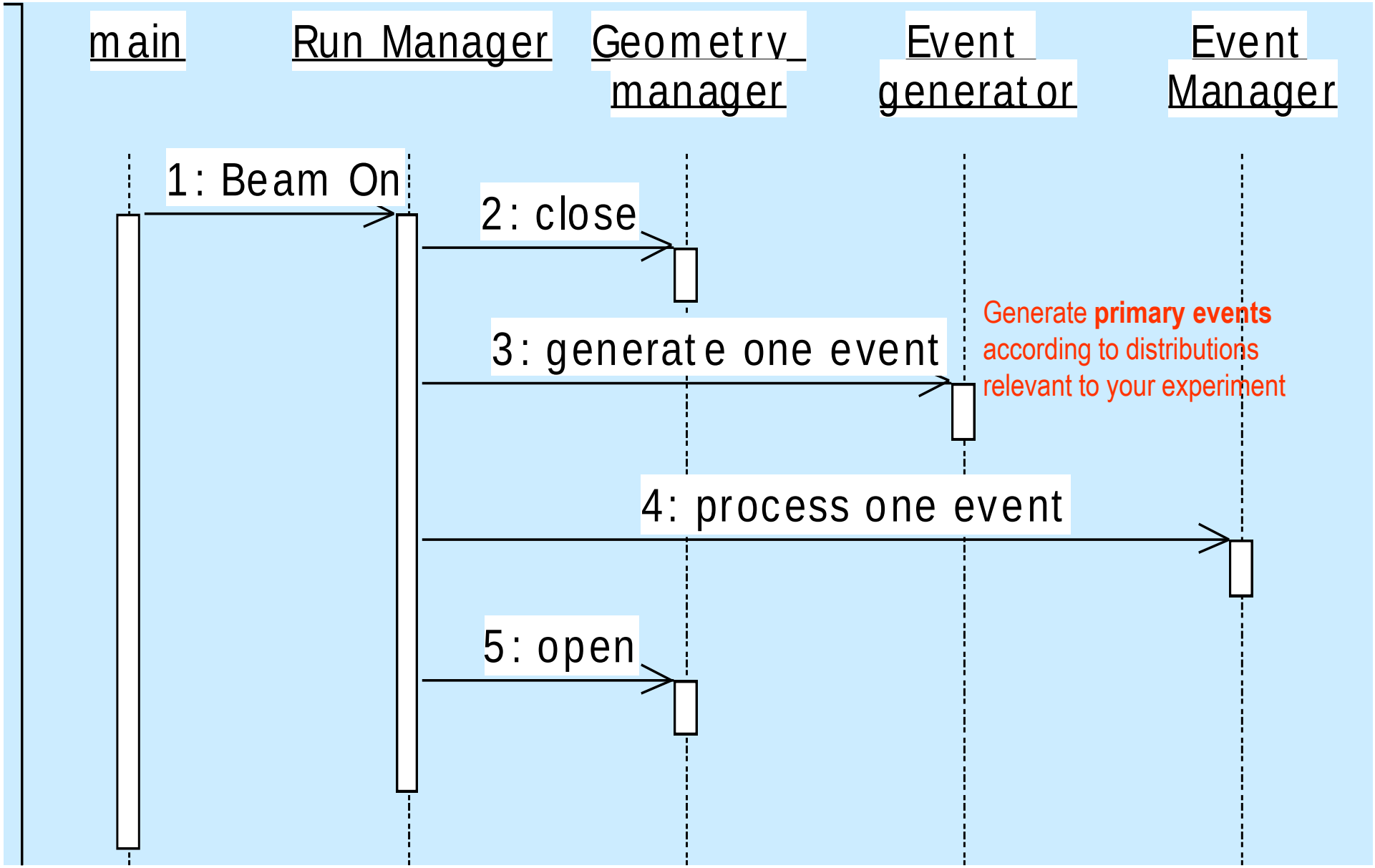
Classes having name starting with G4V are abstract classes (containing purely virtual methods)

Action classes

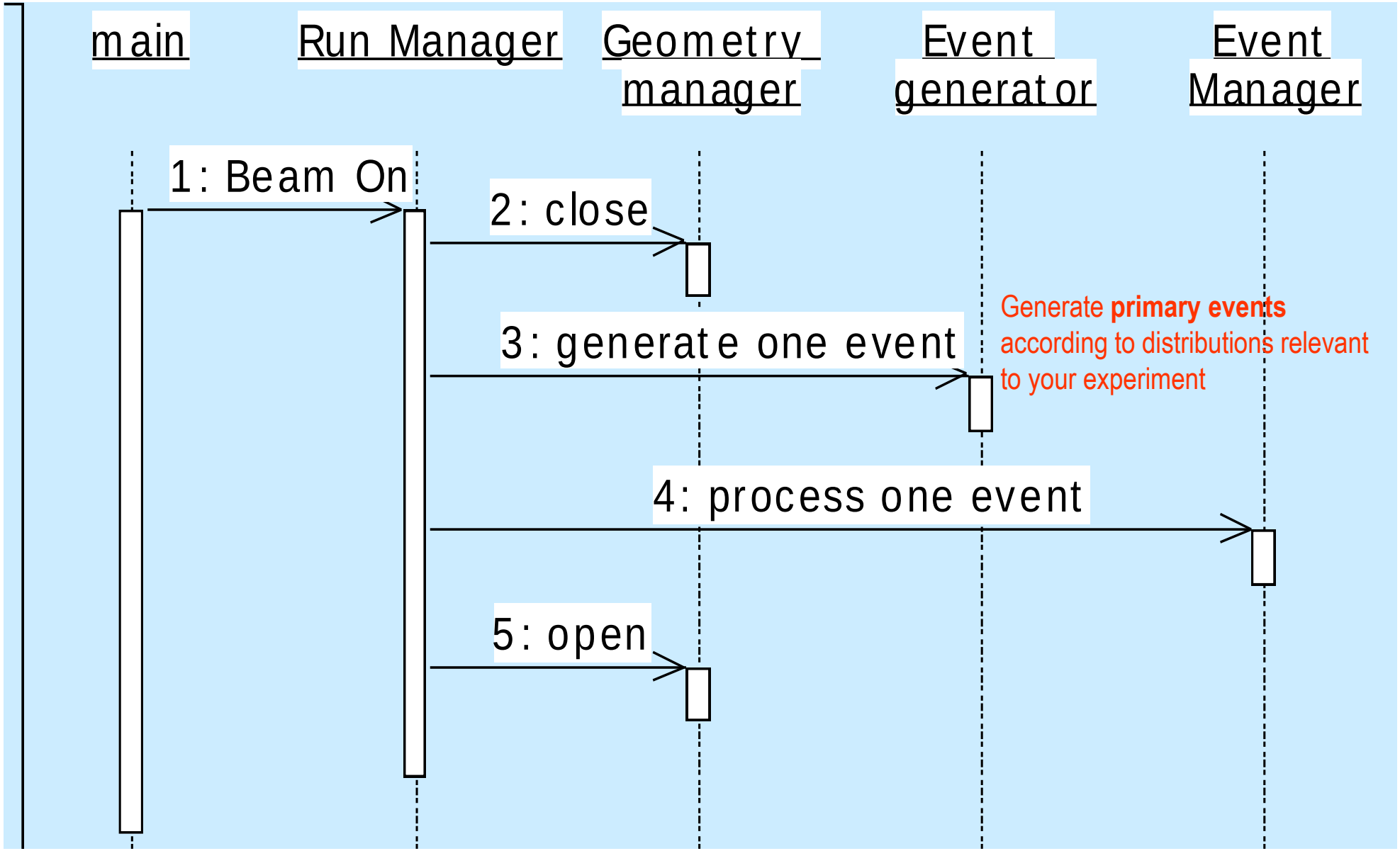
Invoked during the execution loop

- G4VUserPrimaryGeneratorAction
- G4UserRunAction
- G4UserEventAction
- G4UserTrackingAction
- G4UserStackingAction
- G4UserSteppingAction


main
Run manager
user detector construction
user physics list
1: initialize
2: construct
3: material construction
Describe your
experimental set-up
4: geometry construction
5: world volume
6: construct
7: physics process construc
Activate physics processes
appropriate to your experiment
8: set cuts

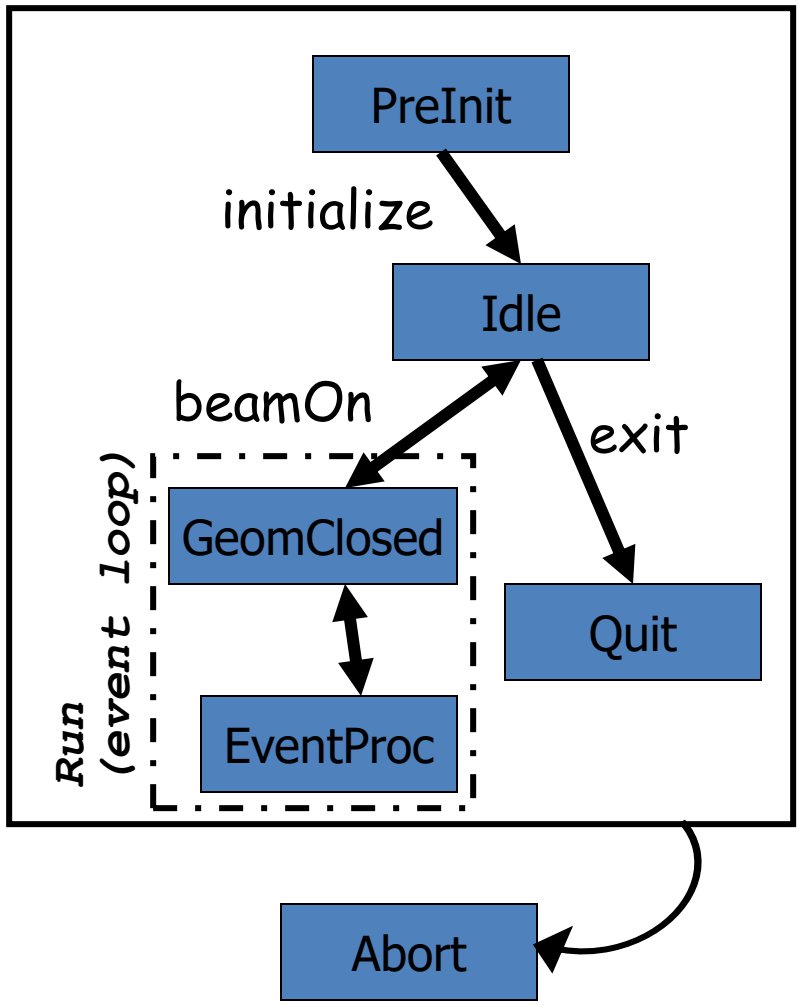

main
Run Manager
Geometry_ manager
Event generator
Event Manager
1: Beam On
2: close
3: generate one event
Generate primary events according to distributions relevant to your experiment
4: process one event
5: open

Event loop


main
Run Manager
Geometry_ manager
Event generator
Event Manager
1: Beam On
2: close
3: generate one event
Generate primary events according to distributions relevant to your experiment
4: process one event
5: open

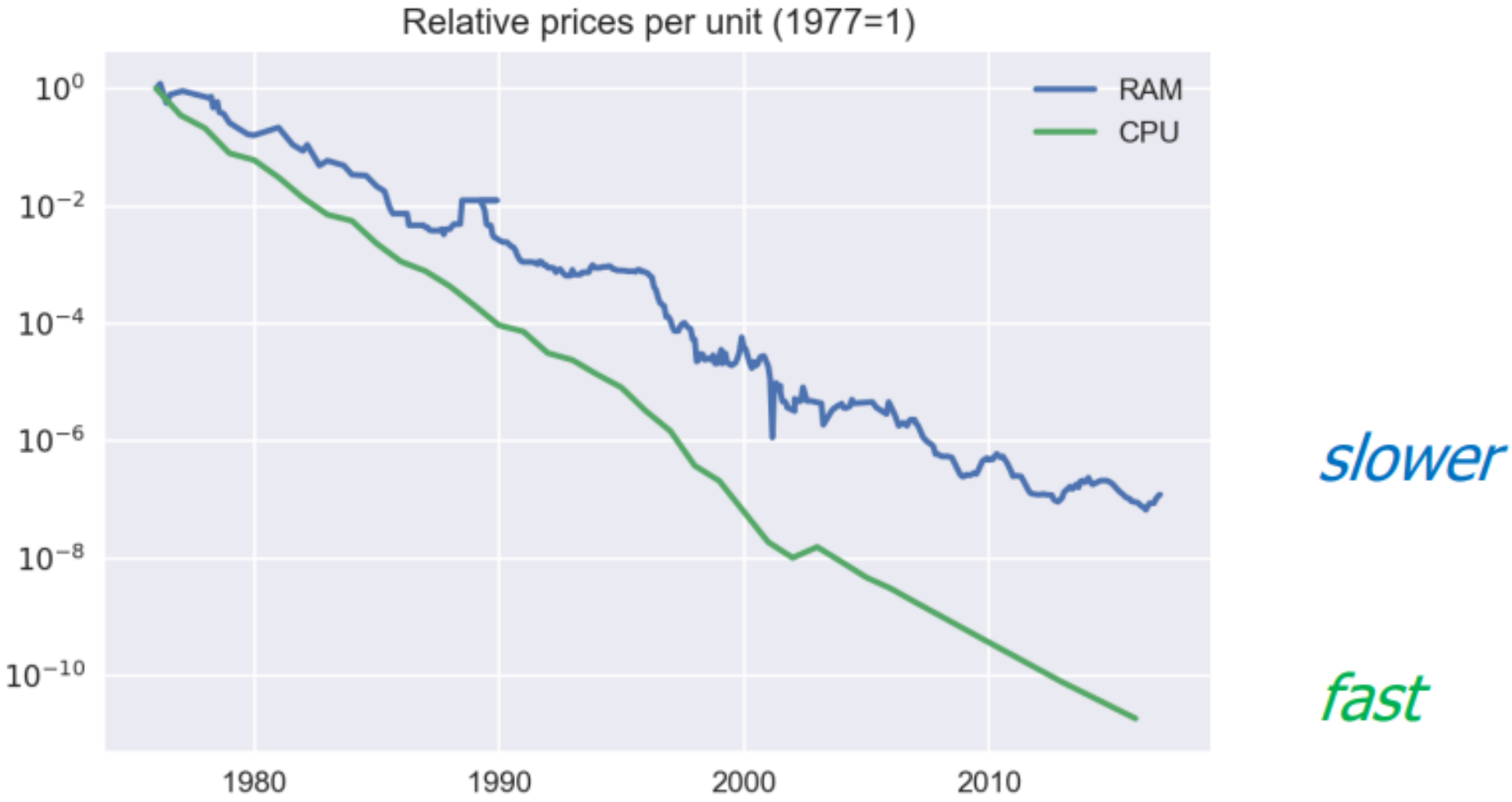
Geant4 as a state machine

- **Geant4 has six application states.**
  - **G4State_PreInit**
    - Material, Geometry, Particle and/or Physics Process need to be initialized/defined
  - **G4State_Idle**
    - Ready to start a run
  - **G4State_GeomClosed**
    - Geometry is optimized and ready to process an event
  - **G4State_EventProc**
    - An event is processing
  - **G4State_Quit**
    - (Normal) termination
  - **G4State_Abort**
    - A fatal exception occurred and program is aborting



Multithreading

- Multi-core CPUs
- Expensive memory

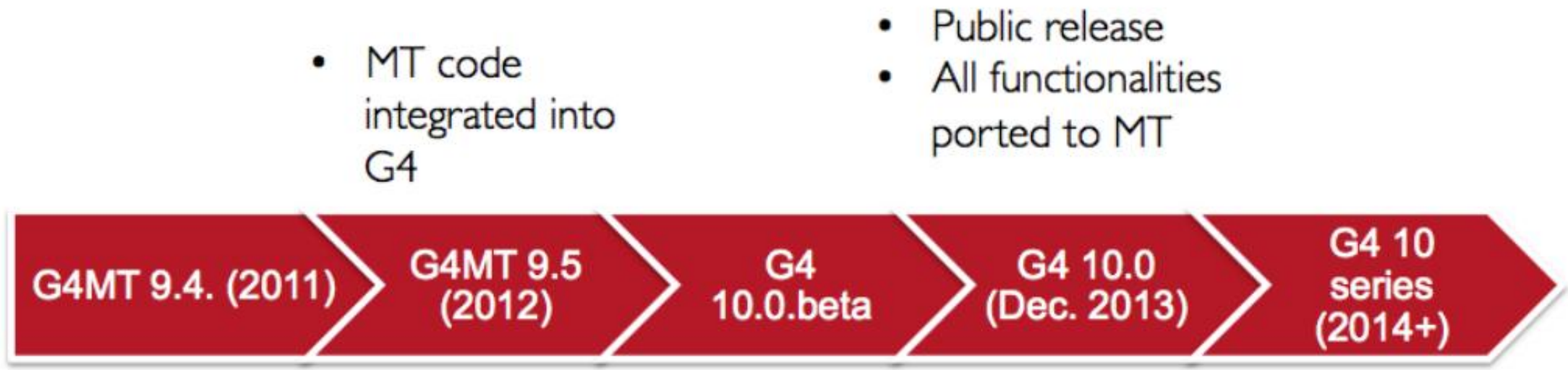


slower

fast

⇒ Memory optimization is more and more important!

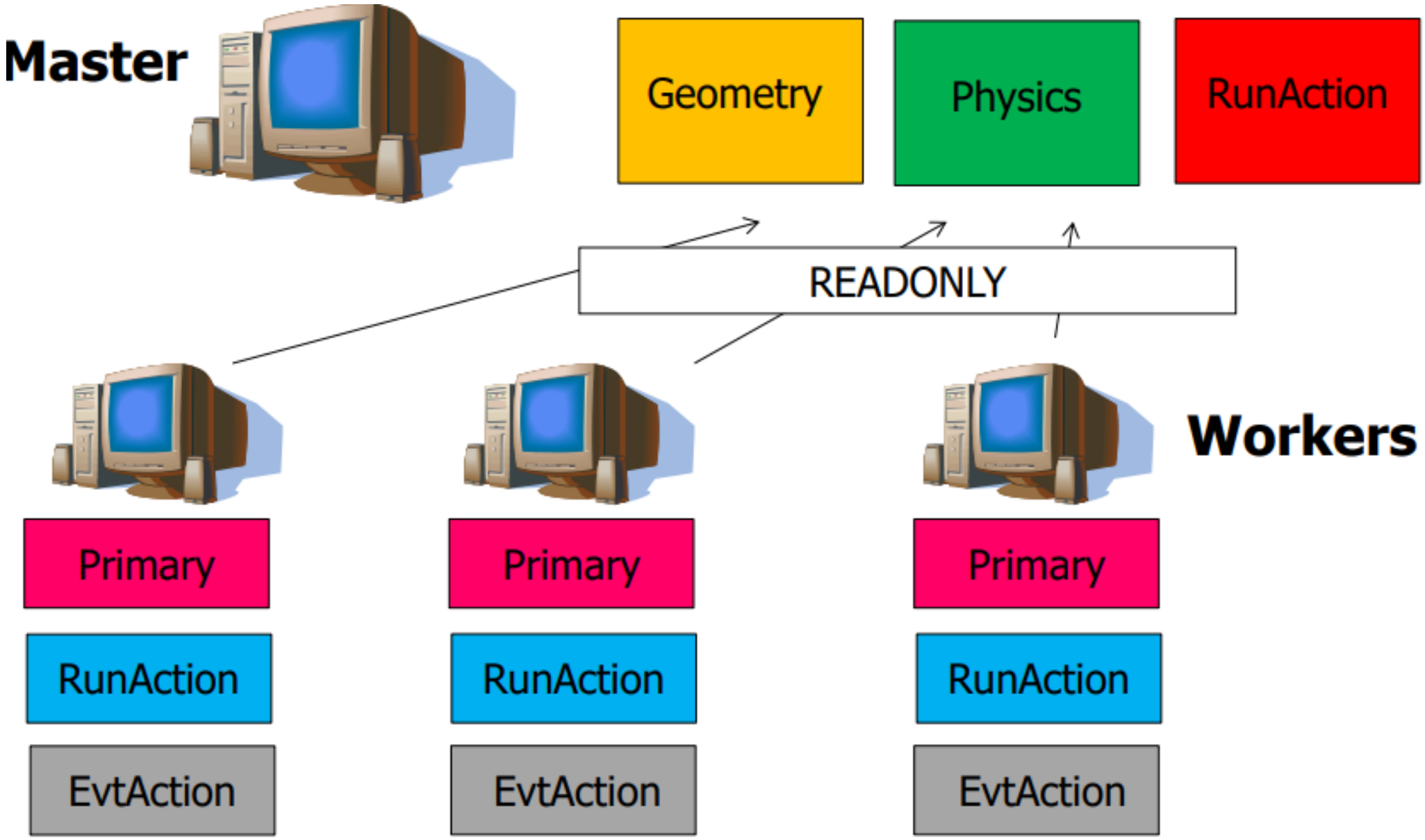
History of Geant4 multithreading

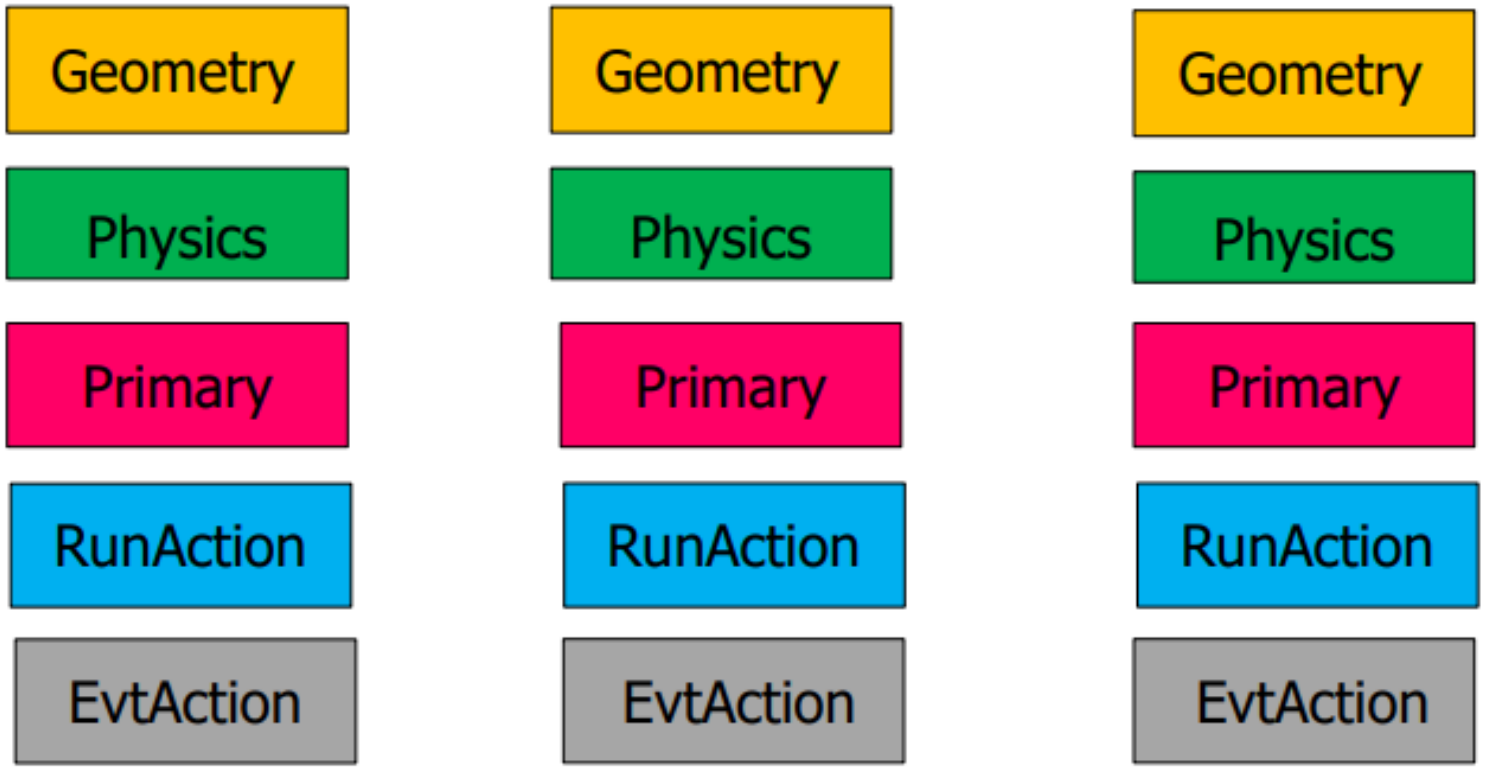


- Proof of principle
- Identify objects to be shared
- First testing

- MT code integrated into G4
- API re-design
- Example migration
- Further testing
- First optimizations

- Public release
- All functionalities ported to MT
- Further Refinements
- Focus on further performance improvements

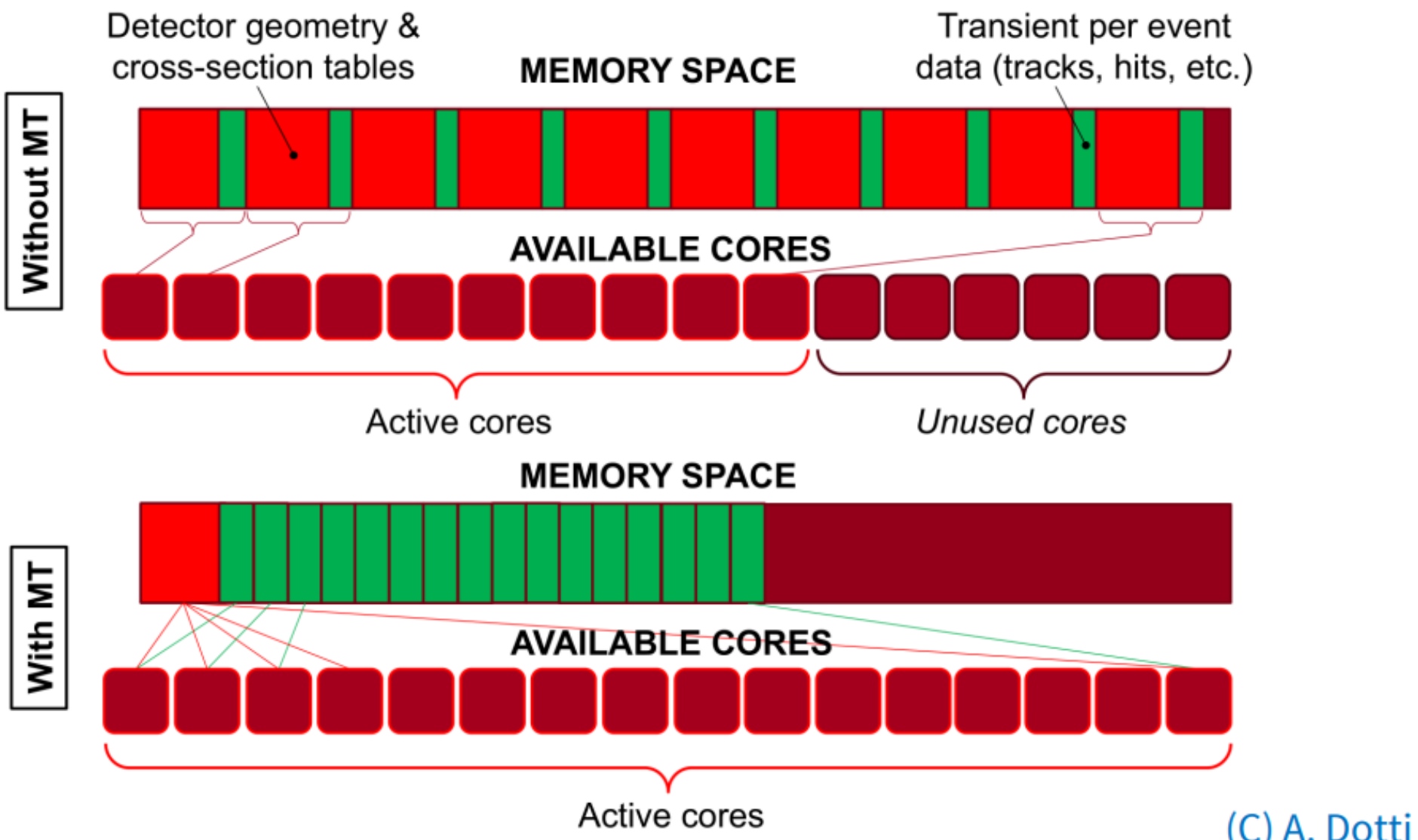

Master
Geometry
Physics
RunAction
READONLY
Workers
Primary
Primary
Primary
RunAction
RunAction
RunAction
EvtAction
EvtAction
EvtAction



Node

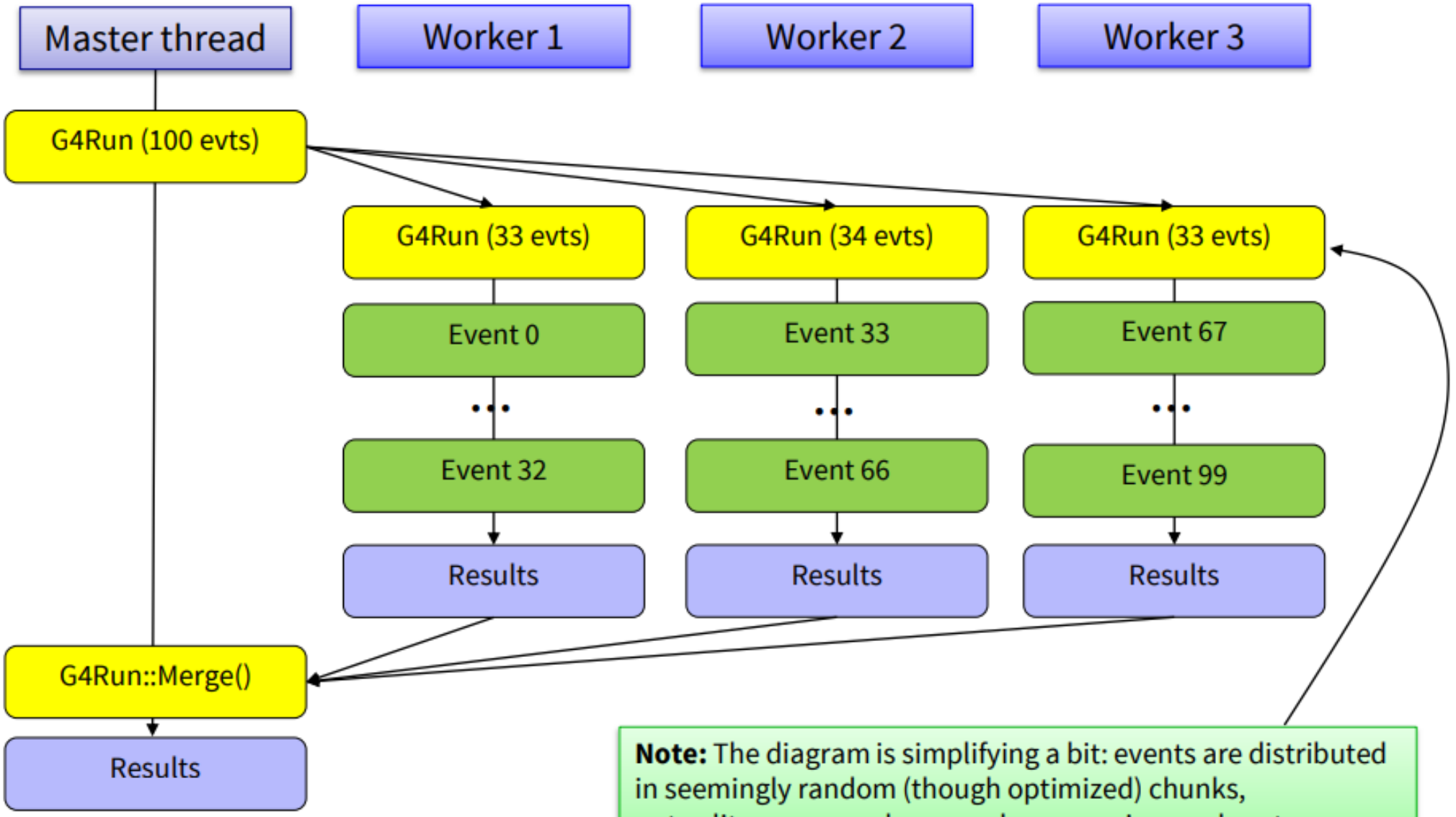
- Each node hosts a complete simulation
- Many copies of geometry and physics tables
- More memory-thirsty

Memory in MT application



(C) A. Dotti

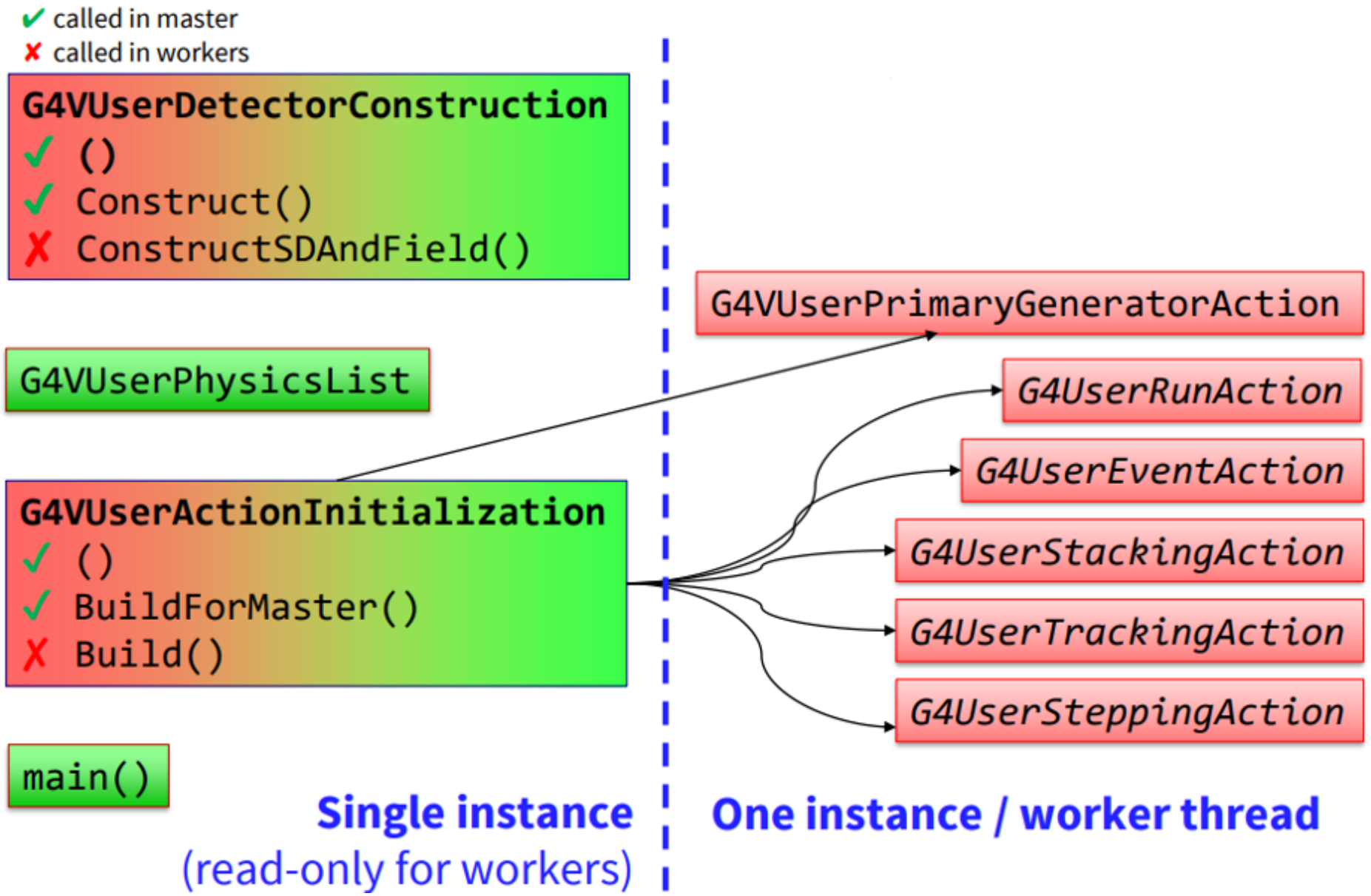
General design



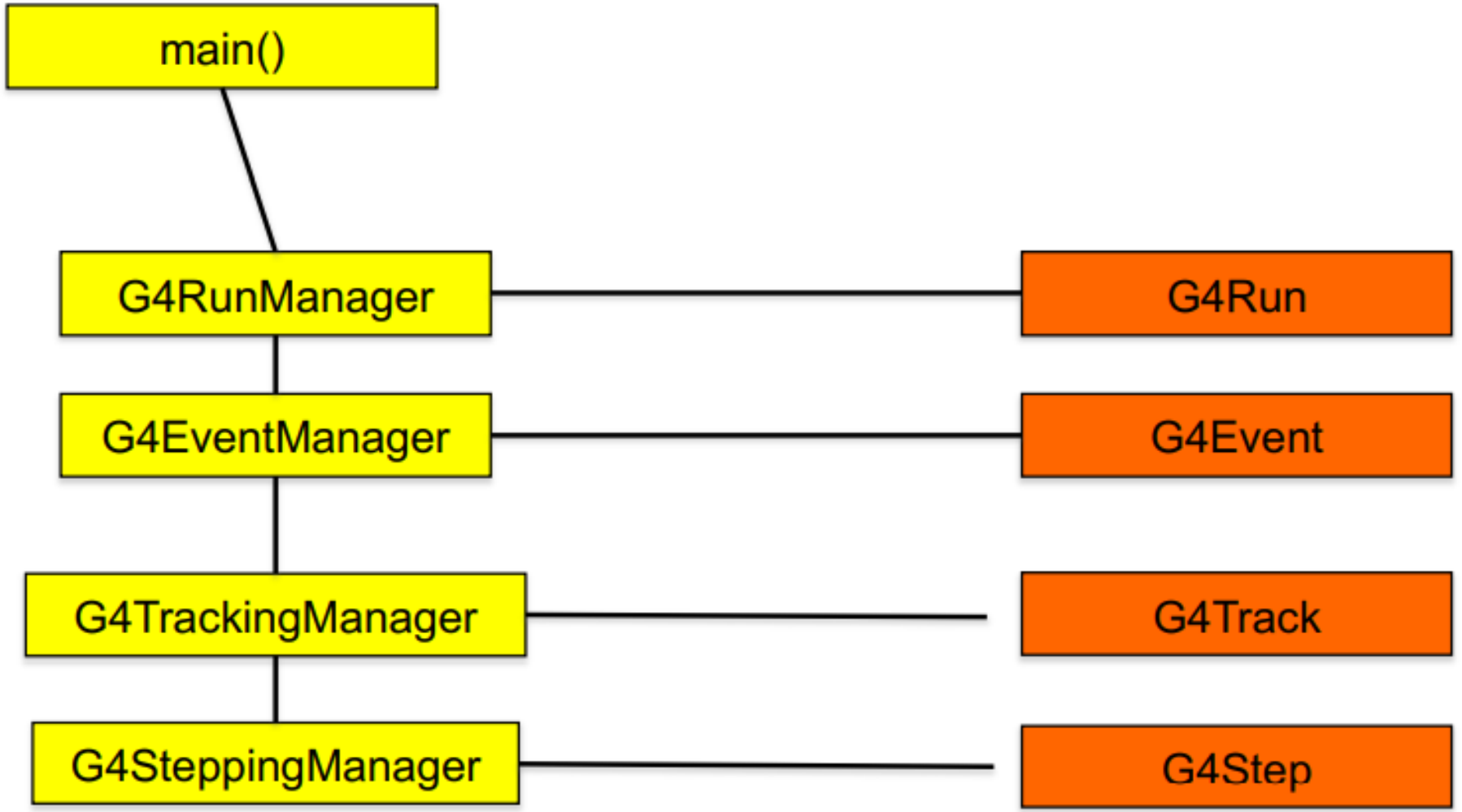
Shared and thread-local objects

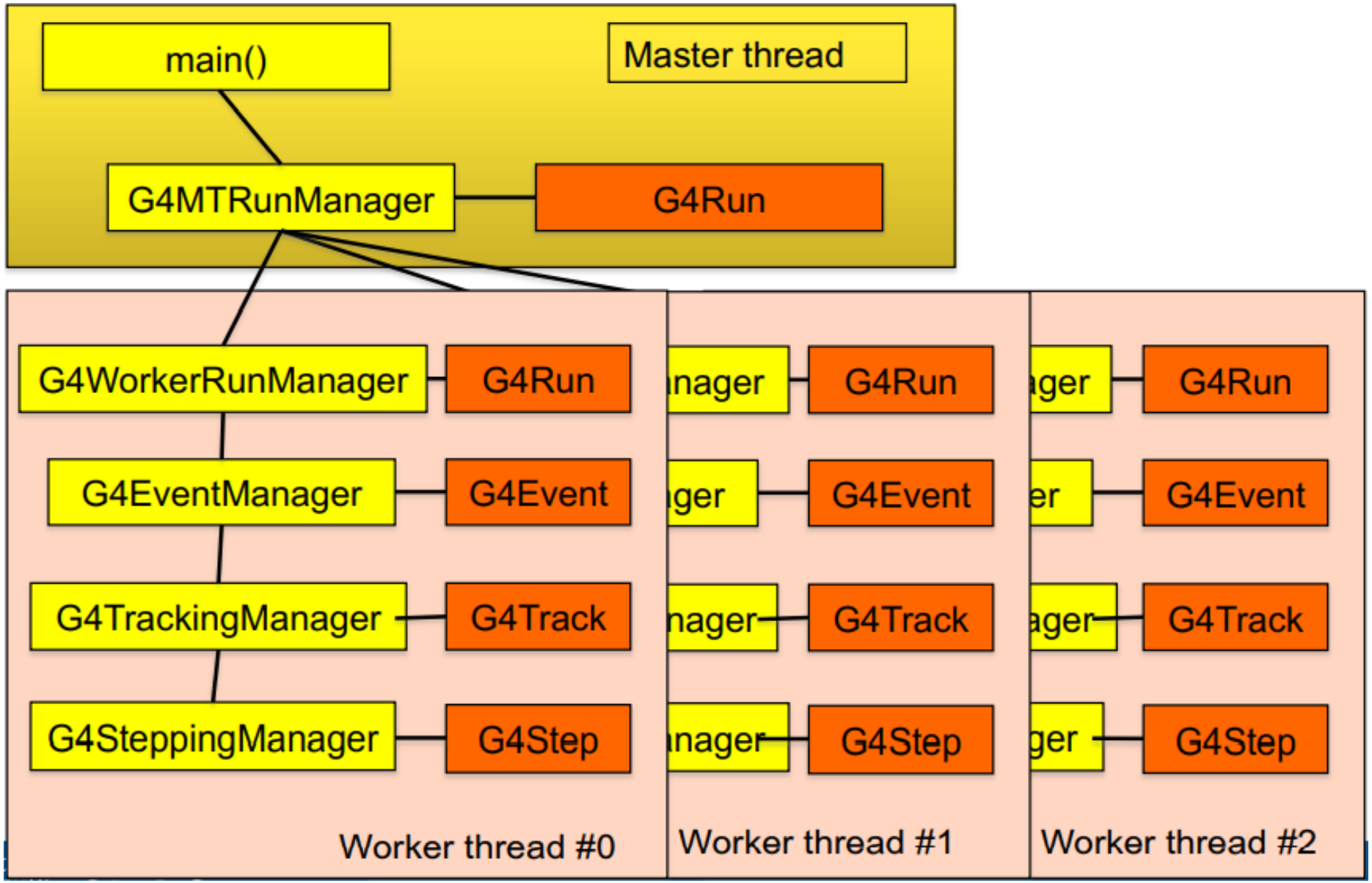
- The geometry and physics tables are shared
- The event, track, step, trajectory, hits, etc., as well as several Geant4 manager classes (EventManager, TrackingManager, SteppingManager, TransportationManager, FieldManager, SensitiveDetectorManager) and Navigator are thread-local
- Among the user classes, the initialization classes (G4VUserDetectorConstruction, G4VUserPhysicsList and the new G4VUserActionInitialization) are shared, whereas all user action classes and sensitive detector classes are thread-local
  - It is not recommended to access a thread-local object from a shared class object, e.g. from detector construction to stepping action.
  - Please note that thread-local objects are instantiated and initialized at the first *BeamOn*.
- To avoid potential errors, try to keep in mind which classes are shared and which classes are thread-local

User classes in MT and thread safety

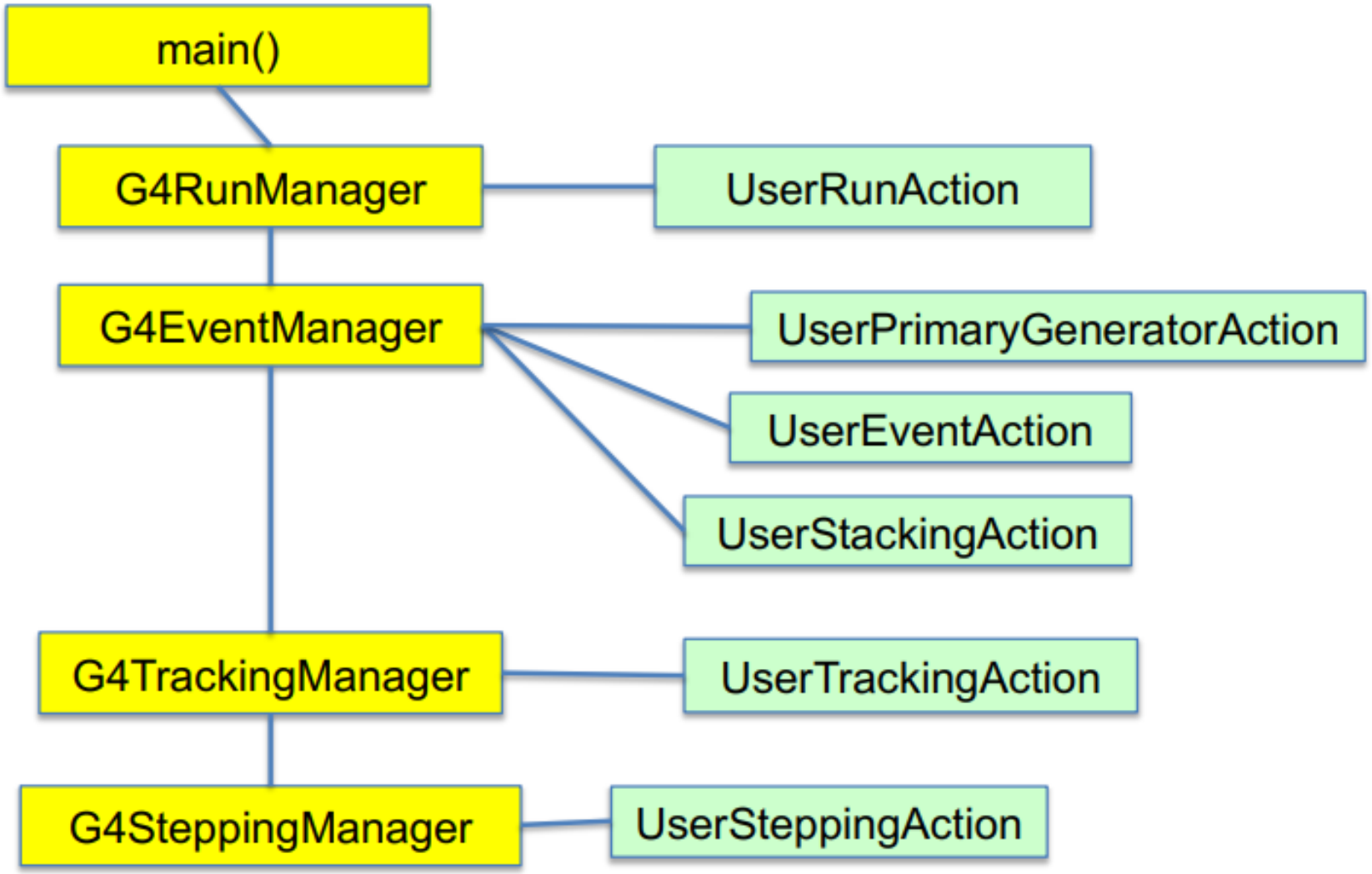


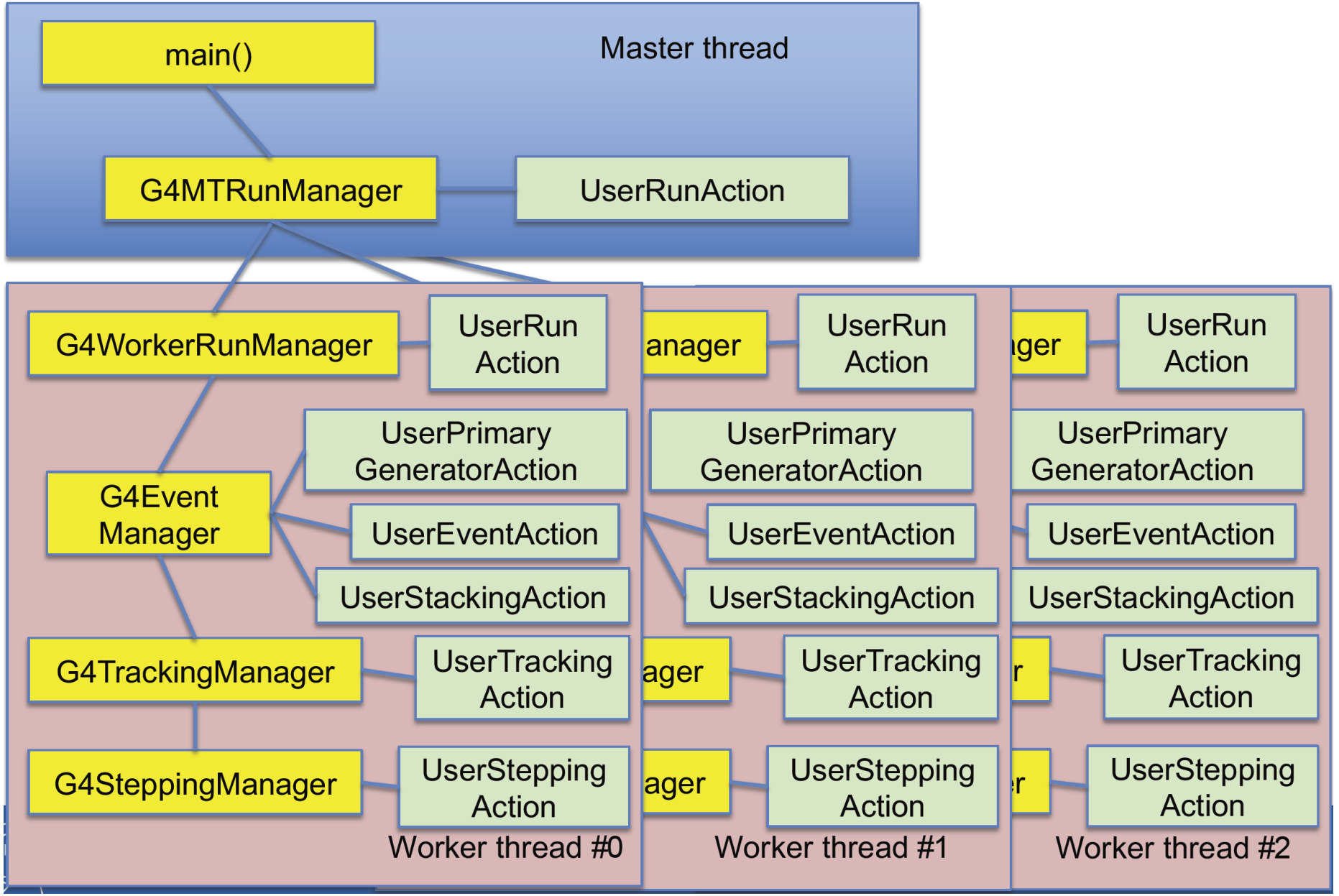
Sequential mode




main()
Master thread
G4MTRunManager
G4Run
G4WorkerRunManager
G4Run
G4EventManager
G4Event
G4TrackingManager
G4Track
G4SteppingManager
G4Step
Worker thread #0
Worker thread #1
Worker thread #2

Sequential mode




main()
Master thread
G4MTRunManager
UserRunAction
G4WorkerRunManager
UserRun Action
UserPrimary GeneratorAction
G4Event Manager
UserEventAction
UserStackingAction
G4TrackingManager
UserTracking Action
G4SteppingManager
UserStepping Action
Worker thread #0
anager
UserRun Action
UserPrimary GeneratorAction
UserEventAction
UserStackingAction
ager
UserTracking Action
ager
UserStepping Action
Worker thread #1
ger
UserRun Action
UserPrimary GeneratorAction
UserEventAction
UserStackingAction
r
UserTracking Action
r
UserStepping Action
Worker thread #2

G4UserRunAction in MT mode

- **This action (unlike the rest) can apply in both worker and master threads:**
  - **To distinguish where you are, use IsMaster() method**
  - **If you have behavior for master, register the instance in G4VUserActionInitialization::BuildForMaster()**

```
void MyActionInitialization::Build() const {
    SetUserAction(new MyRunAction());
    // ...other actions
}

void MyActionInitialization::BuildForMaster() const {
    SetUserAction(new MyRunAction());
    // Only run action
}
```

Note: This, in principle, can be a different class

Set the number of threads

- Default number of threads: 2
- Change this using:
  - UI command
    - /run/numberOfThreads 6
    - /run/useMaximumLogicalCores
  - C++ code
    - runManager->SetNumberOfThreads(4)
  - Environment variable (highest priority)
    - G4FORCENUMBEROFTHREADS=4
- G4Threading::G4GetNumberOfCores() tells the actual number of logical cores
- Note: Must be done in pre-initialize stage

G4cout in MT mode

- G4cout is relatively synchronized and each message is prepended with the thread number (this is not true for std::cout)

```
### Run 0 starts.
G4WT1 > EventAction: absorber energy/time scorer ID: 0
G4WT1 > EventAction: scintillator energy/time scorer ID: 1
G4WT0 > EventAction: absorber energy/time scorer ID: 0
G4WT0 > EventAction: scintillator energy/time scorer ID: 1
 Run terminated.
Run Summary
  Number of events processed : 10000
  User=21s Real=11.36s Sys=1.59s
```

- To buffer the output from each thread at a time, so that the output of each thread is grouped and printed at the end of the job
  - /control/cout/useBuffer true|false
- To limit the output from threads to one selected thread only:
  - /control/cout/ignoreThreadsExcept 0
- To redirect the output from threads in a file:
  - /control/cout/setCoutFile coutFileName
  - /control/cout/setCerrFile cerrFileName

Use native Geant4 scoring

- Geant4 provides a number of primitive scorers, each one accumulating one physics quantity (e.g. total dose) for an event.

- G4MultiFunctionalDetector is a concrete class derived from G4VSensitiveDetector
  - It should be assigned to a logical volume as a kind of sensitive detector
  - It takes an arbitrary number of G4VPrimitiveScorer classes, to define the scoring quantities that you need
  - Each G4VPrimitiveScorer accumulates one physics quantity for each physical volume
    - E.g. G4PSDoseScorer (a concrete class of G4VPrimitiveScorer provided by Geant4) accumulates dose for each cell
  - Each G4VPrimitiveScorer is automatically named as <MultiFunctionalDetectorName>/<PrimitiveScorerName>

- By using this approach, no need to implement sensitive detector and hit classes!

Example

```
MyDetectorConstruction::ConstructSDandField()
{
   G4MultiFunctionalDetector* myScorer = new
   G4MultiFunctionalDetector("myCellScorer");

   myCellLog->SetSensitiveDetector(myScorer);

   G4VPrimitiveScorer* totalSurfFlux = new
     G4PSFlatSurfaceFlux("TotalSurfFlux");
   myScorer->RegisterPrimitive(totalSurfFlux);
   G4VPrimitiveScorer* totalDose =   new
     G4PSDoseDeposit("TotalDose");
   myScorer->RegisterPrimitive(totalDose);
}
```

instantiate multi-functional detector

attach to volume

create a primitive scorer (surface flux) and register it

create a primitive scorer (total dose) and register it

List of some primitive scorers

❄ Concrete Primitive Scorers (→ Application Developers Guide 4.4.5)

- **Track length**
  - **G4PSTrackLength, G4PSPassageTrackLength**
- **Deposited energy**
  - **G4PSEnergyDepsit, G4PSDoseDeposit**
- **Current/Flux**
  - **G4PSFlatSurfaceCurrent,**
  - **G4PSSphereSurfaceCurrent,G4PSPassageCurrent,**
  - **G4PSFlatSurfaceFlux, G4PSCellFlux, G4PSPassageCellFlux**

❄ Others

- **G4PSMinKinEAtGeneration, G4PSNofSecondary, G4PSNofStep, G4PSCellCharge**

- **A G4VSDFilter can be attached to G4VPrimitiveScorer to define which kind of tracks have to be scored (e.g. one wants to know surface flux of protons only)**
  - **G4SDChargeFilter (accepts only charged particles)**
  - **G4SDNeutralFilter (accepts only neutral particles)**
  - **G4SDKineticEnergyFilter (accepts tracks in a defined range of kinetic energy)**
  - **G4SDParticleFilter (accepts tracks of a given particle type)**
  - **G4VSDFilter (base class to create user-customized filters)**

Example

```
MyDetectorConstruction::ConstructSDandField()
{
   G4VPrimitiveScorer* protonSurfFlux
   = new G4PSFlatSurfaceFlux("pSurfFlux");
    G4VSDFilter* protonFilter = new
       G4SDParticleFilter("protonFilter");
    protonFilter->Add("proton");

    protonSurfFlux->SetFilter(protonFilter);

    myScorer->RegisterPrimitive(protonSurfFlux);
}
```

create a primitive scorer (surface flux), as before

create a particle filter and add protons to it

register the filter to the primitive scorer

register the scorer to the multifunc detector (as shown before)

Retrieve the information


```
//needed only once
G4int collID = G4SDManager::GetSDMpointer()
  ->GetCollectionID("myCellScorer/TotalSurfFlux");
```

Get **ID** for the collection (given the name)

```
G4HCofThisEvent* HCE = event->GetHCofThisEvent();
```

Get **all HC** available in this event

```
G4THitsMap<G4double>* evtMap =
    static_cast<G4THitsMap<G4double>*>
    (HCE->GetHC(collID));
```

Get the HC with the **given ID** (need a cast)

```
for (auto pair : *(evtMap->GetMap())) {
    G4double flux = *(pair.second);
    G4int copyNb  = *(pair.first);
}
```

Loop over the **individual entries** of the HC: the key of the map is the copyNb, the other field is the real content

Retrieve the information

```
//needed only once
G4int collID = G4SDManager::GetSDMpointer()
  ->GetCollectionID("myCellScorer/TotalSurfFlux");
```

Get **ID** for the collection (given the name)

```
G4HCofThisEvent* HCE = event->GetHCofThisEvent();
```

Get **all HC** available in this event

```
G4THitsMap<G4double>* evtMap =
    static_cast<G4THitsMap<G4double>*>
    (HCE->GetHC(collID));
```

Get the HC with the **given ID** (need a cast)