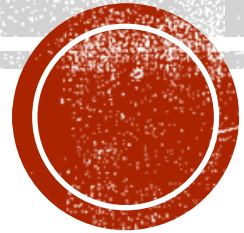
超级陶粲装置
Super Tau-Charm Facility

OSCAR框架下探测器模拟介绍

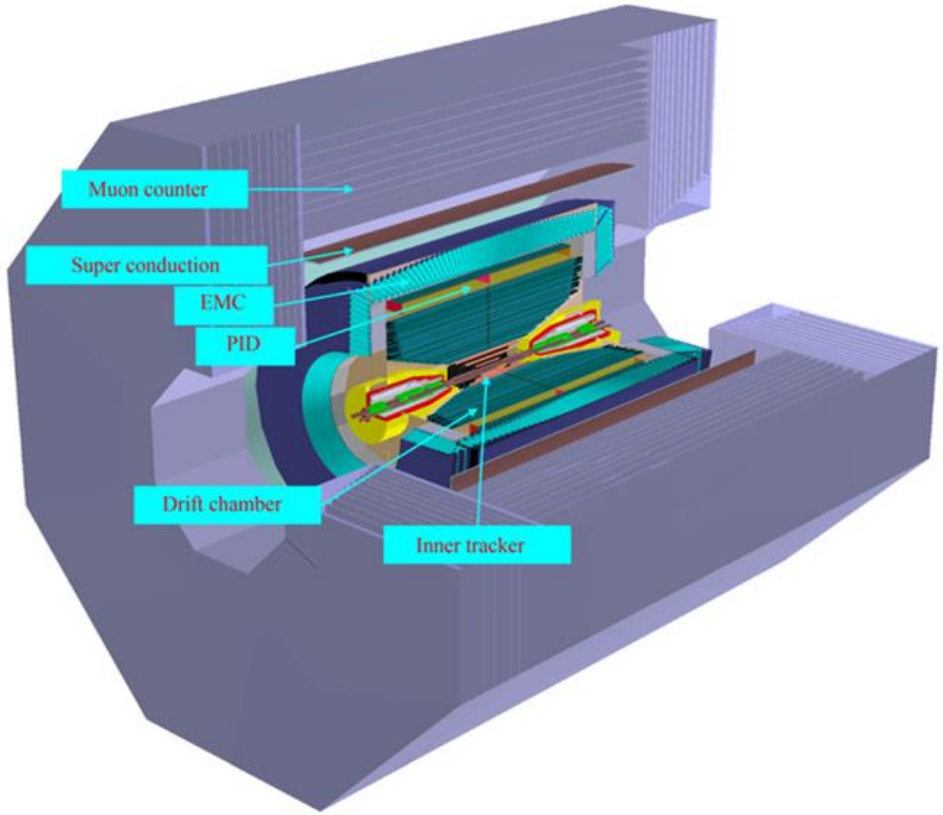


齐斌斌

中国科学技术大学

主要内容

- Geant4模拟
  - Geant4简介
  - 探测器材料与几何
  - 粒子和物理过程
  - Run、Event、Track和Step等基本概念
  - 事例产生子接口
- OSCAR框架下探测器模拟
  - 产生子设置
  - EDM定义与数据获取



什么是GEANT4?

- Geant4 是模拟粒子穿过物质的工具包。
- 提供了探测器模拟的完整工具：几何，探测器响应，运行、事例以及径迹管理、图形显示、用户接口等。
- 提供了极为丰富的可供选择的物理过程。
- Geant4 发展于 Geant3,充分利用了C++语言中的优势，是粒子与核物理实验模拟最好的工具之一。
- Geant4 是最早成功使用面向对象环境重新设计的粒子与核物理软件包，并应用于新一代实验。
- 网页：http://geant4.cern.ch

✓ 帮助设计探测器，研究预期性能
✓ 模拟与实验对比，检验分析结果是否正确

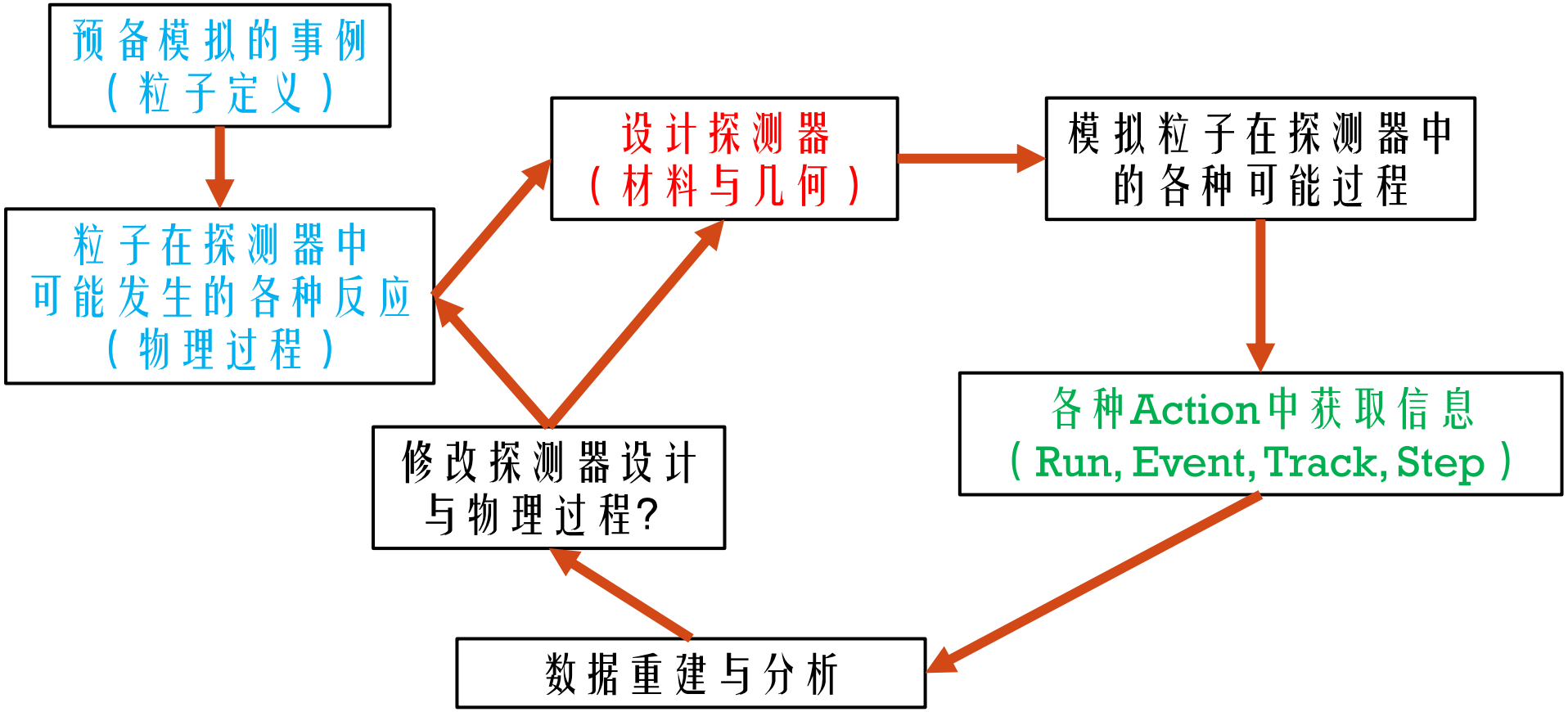
模拟需要做的主要工作

- 定义探测器几何(具有有限接受范围的探测器)
- 指定探测器的物质(有用的/没用的各种材料)
- 指定物理过程与粒子(什么东西在探测器中干了什么)
- 产生原始事例(模拟重复实验n次)
- 数据收集和分析(取出探测器可以获得的信息并处理分析)

➔ 除了最后的分析，其它部分都可以由Geant4负责处理

预备课程：粒子物理导论，粒子探测技术

GEANT4模拟基本流程



材料与几何

- 定义材料：
  - G4Isotope：同位素
  - G4Element：元素
  - G4Material：单质，化合物，混合物……

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

- 定义几何：
  - G4VSolid – 形状，尺寸
  - G4LogicalVolume – 赋予体积材料定义，灵敏区，磁场，用户限制等等。
  - G4VPhysicalVolume – 位置，转动

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

注意：材料与几何描述的多层面设计是为了最大限度的信息再利用，以便减小内存空间。

粒子定义

- 粒子的定义在相应的PhysicsList类中
- 定义粒子：
  - G4ParticleDefinition：粒子的“静态”特征量，如电荷、质量、寿命等等。没有能量、方向等信息
  - G4DynamicParticle：赋予粒子运动学(动态)属性，如动量，能量，自旋方向等等。
  - G4Track：将动态粒子放到具体环境中，给出位置，几何信息等

```
void PhysicsList::ConstructParticle()
{
    // 单独定义不同粒子
    G4Proton::ProtonDefinition(); //定义质子
    G4Positron::PositronDefinition(); //正电子
    G4MuonPlus::MuonPlusDefinition(); //μ+
}
```

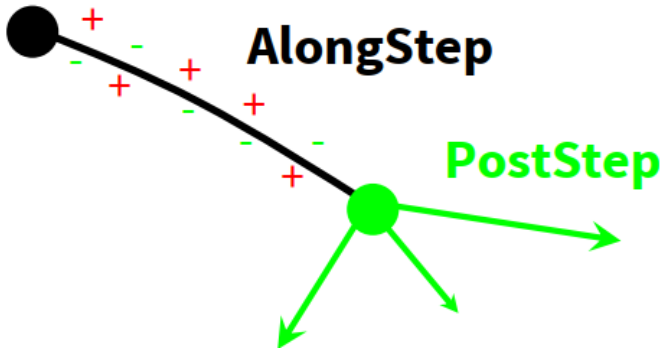
- Geant4提供了各种类型的粒子：
  - 普通粒子：如电子、质子、光子等；共振态粒子：寿命短，如矢量介子等；核子：如氘核、氦核及重离子等；夸克、胶子等
  - 定义的附带了粒子的各种信息：如名称、质量、电荷、自旋、寿命、衰变道等
- Geant4对不同类型粒子的处理不同，如：
  - 稳定/长寿命粒子：径迹模拟；K0:直接被重定义为$K0_L$或$K0_S$，然后模拟径迹；短寿命粒子：直接衰变，而不模拟径迹

```
void PhysicsList::ConstructParticle()
{
    // 分类批量定义粒子
    // 定义所有轻子
    G4LeptonConstructor pConstructor;
    pConstructor.ConstructParticle();
    // 定义所有玻色子
    G4BosonConstructor pConstructor;
    pConstructor.ConstructParticle();
}
```

物理过程

- 要模拟真实的物理，必须首先知道粒子在物质中哪些相互作用是最主要的，或者说哪些物理过程是重要的。
- Geant4提供了7大类物理过程描述粒子与物质的相互作用。$G4INSTALL/data目录存放物理模型的数据
  - electromagnetic:电磁相互作用过程(标准的和低能的)
  - hadronic:强子相互作用过程
  - decay:衰变过程
  - photolepton-hadron:光轻子与强子的相互作用过程
  - optical:光学的光子过程
  - parameterization:参数化过程（即fast simulation）
  - transportion:输运过程
- 要根据事例中的粒子以及材料，指定必要的物理过程，其中输运过程是必须添加的过程

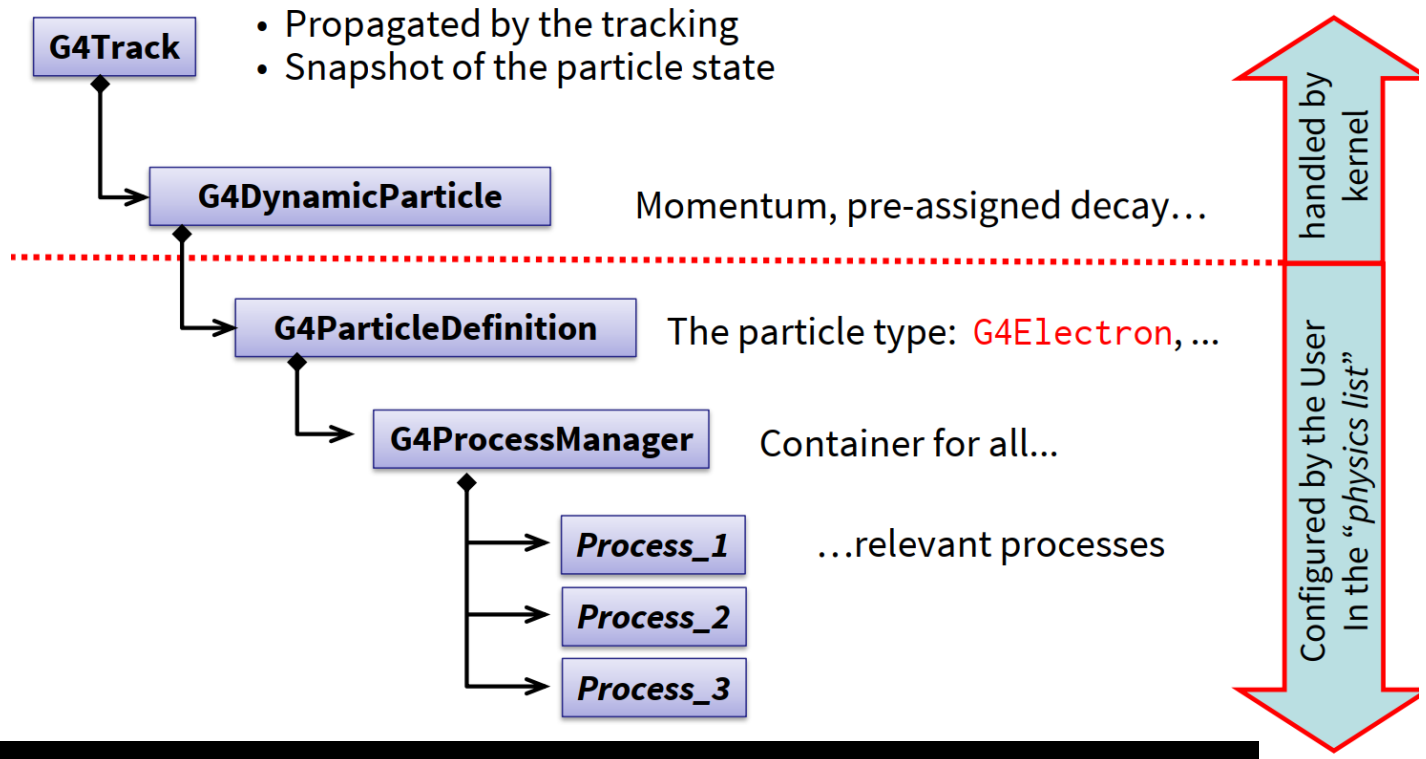
物理过程分类



- 三种不同物理过程：
- AtRest：在粒子停止运动时发生，如静止的缪子俘获过程
- AlongStep：描述连续作用过程，与径迹长度相关，如切伦科夫辐射
- PostStep：描述不连续作用过程，由截面决定，发生在Step结尾，比如各种散射

- 一些物理过程在不同状态下都发生
- 在飞行中和静止时都发生，如湮灭过程，衰变
- 同时包含连续和离散过程，比如电离能损是连续的，但是电离产生δ电子过程是离散过程

物理过程与粒子关联




```
G4ProcessManager *elManager = G4Electron::ElectronDefinition()->GetProcessManager();
elManager->AddProcess(new G4eMultipleScattering, -1, 1, 1);
elManager->AddProcess(new G4eIonisation, -1, 2, 2);
elManager->AddProcess(new G4eBremsstrahlung, -1, -1, 3);
elManager->AddDiscreteProcess(new G4StepLimiter);
AddTransportation();
```

如何定义PHYSICSLIST

- 定义PhysicsList

```
class MyPhysicsList : public G4VModularPhysicsList {
        public:
        MyPhysicsList();
        void ConstructParticle();
        void ConstructProcess();
        void SetCuts();     }
```

- 使用Physics constructor

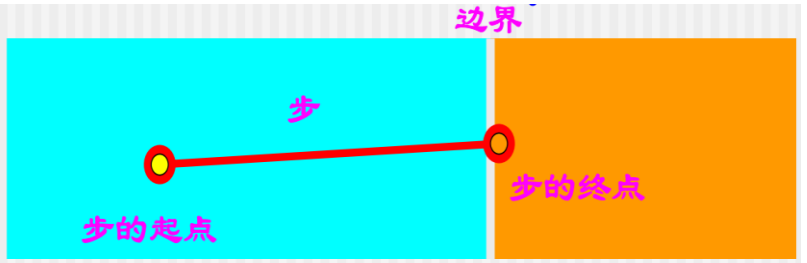
```
MyPhysicListList ::MyPhysicListList() {
        // Hadronic physics
        RegisterPhysics(new G4HadronElasticPhysics());
        RegisterPhysics(new G4HadronPhysicsFTFP_BERT_TRV());
        // EM physics
        RegisterPhysics(new G4EmStandardPhysics());       }
```

- 直接使用预定义Physics Lists

```
Int main() {
        …
        FTFP_BERT* physlist = new FTFP_BERT();
        runManager->SetUserInitialization(physlist);          }
```

- *FTFP_BERT*
- FTFP_BERT_ATL
- FTFP_BERT_HP
- FTFP_BERT_TRV
- FTFP_INCLXX
- FTFQGSP_BERT
- FTF_BIC
- *QBBC*
- *QGSP_BERT*
- QGSP_BERT_HP
- *QGSP_BIC*
- QGSP_BIC_AllHP
- QGSP_BIC_HP
- QGSP_FTFP_BERT
- QGSP_INCLXX
- QGS_BIC
- *Shielding*
- ShieldingLEND
- LBE
- NuBeam

自定义用户作用类



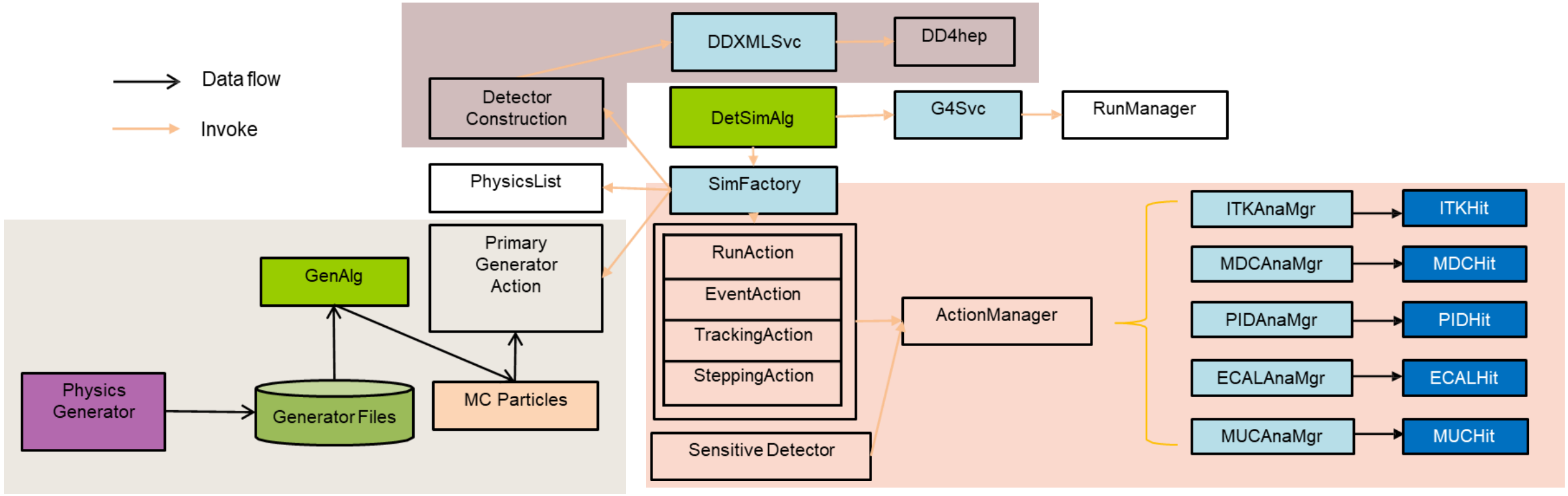
- 在调用BeanOn()的过程中，将调用各种(如存在)用户作用类
  - 跟踪径迹，提取，保存模拟信息
- G4UserRunAction，一个Run包含多个Event
  - G4UserRunActio类中有BeginOfRunAction()和EndOfRunAction()
  - 前者主要用于进行run号设定、直方图或TTree，TFile定义等，后者主要进行存储直方图或者文件等
- G4UserEventAction，一个Event包含多个Track
  - G4UserEventAction类中有BeginOfEventAction()和EndOfEventAction()
  - 前者可以作事例开始的预备工作，后者可以将事例的有用信息提取出来，填充到直方图或者TTree中
- G4UserTrackingAction，径迹是粒子在探测器中留下的痕迹
  - 只体现出当时粒子的位置和物理量
  - G4UserEventAction类中有PreUserTrackingAction()和PostUserTrackingAction()
- G4UserSteppingAction，G4Step是粒子径迹的一小段
  - 每一步都有两个点和粒子的 “Δ”信息(在该步的能损，所需的飞行时间，等等)。
  - 在每一点上，都应该知道其所处在的体积(与材料)内。
  - 如有一步跨越边界，该步的截止点物理上就设在该边界上，逻辑上该点属于下一个体积（为什么？）

产生事例

- 必须指定如何产生一个事件，才能进行模拟，在G4VUserPrimaryGeneratorAction的具体类中用G4VPrimaryGenerator的具体类来完成。
- 不同PrimaryGenerator
  - G4ParticleGun:发射指定能动量的特定粒子
  - G4GeneralParticleSource：可用mac文件控制设置参数的粒子枪
  - G4HEPEvtInterface:利用提供的接口，读取外部产生子产生的事例。
- 一个G4Event包含哪些东西?
  - 一个或多个G4PrimaryVertex，包含粒子位置、时间信息
  - 一个顶点包含多个G4PrimaryParticle，包含粒子定义，四动量等信息

```
/gps/source/add 1
/gps/particle mu-
/gps/pos/type Point
/gps/pos/centre 0. 0. 0. cm
/gps/ang/type iso
/gps/ang/mintheta 22 deg
/gps/ang/maxtheta 90 deg
/gps/ang/minphi 0 deg
/gps/ang/maxphi 360 deg
/gps/ene/type User
#false 表示动量
/gps/ene/emspec false
/gps/hist/type energy
# cannot delete this line
/gps/hist/point 1000 0.
/gps/hist/point 1000 1.
/run/beamon 100
```

如何在OSCAR框架下开发模拟程序?



- 物理表，产生子，几何，用户作用类

模拟脚本

```
import Sniper
task = Sniper.Task("task")
task.setLogLevel(4) #higher is less
import os
topdir = os.getenv('OFFLINETOP')
import StcfRndmGenSvc
sSvc=task.createSvc("StcfRndmGenSvc/hSvc")
sSvc.property("RndmSeed").set(28133)
import PodioDataSvc
dsvc = task.createSvc("PodioDataSvc")
import PodioSvc
oSvc = task.createSvc("PodioOutputSvc/OutputSvc")
oSvc.property("OutputFile").set("Sim.root")
reduced_coll_list =
["EventHeaderCol","MCParticleCol","ITKHitCol","MDCHitCol","DTOFBa
rHitCol","DTOFPDHitCol","RICHHitCol","ECALHit3Col","ECALTPointCol
","MUDPointCol"]
oSvc.property("OutputCollections").set(reduced_coll_list)
import MdcGeomSvc
Mdcgeom1 = task.createSvc("MdcGeomSvc")
Mdcgeom1.property("geomDataFilePath").set(topdir+"/CommonSvc/MDCS
vc/MdcGeomSvc/dat/mdcGeomData.dat")
Mdcgeom1.property("geomDataSource").set(0)
import GeometrySvc
myxmlsvc = task.createSvc("GeometrySvc")
myxmlsvc.property("GeoCompactFileName").set(topdir+"/Geometry/Ful
lGeometry/compact/STCF.xml")
import RICHGeoSvc
Richgeom = task.createSvc("RICHGeoSvc")
import DTOFGeoSvc
DTOFgeom = task.createSvc("DTOFGeoSvc")
import MUDIDSvc
mudsvc= task.createSvc("MUDIDSvc")
```

数据输出

读取几何

几何Svc

```
import G4Svc
g4svc = task.createSvc("G4Svc")
#g4svc.property("VisMac").set("vis.mac")
g4svc.property("RunMac").set("run.mac")
import DetSimAlg
simalg = task.createAlg("DetSimAlg/DetSimAlg")
simalg.property("DetFactory").set("FullFactory")
import FullSim
genTool = simalg.createTool("GeneratorMgr")
genTool.property("Translation").set([0,0,0]) #generator translation [mm]
#genTool.property("ParticleSource").set("Generator") #Generator or ParticleGun
genTool.property("ParticleSource").set("ParticleGun") #Generator or ParticleGun
genTool.property("Boost").set(False) #only work when ParticleSource == Generator
geoTool = simalg.createTool("GeoAnaMgr")
geoTool.property("GdmlEnable").set(False) #exports geometry into a gdml file
geoTool.property("GdmlOutput").set("stcfGeo.gdml")
mcwTool = simalg.createTool("MCTruthWriter")
mcwTool.property("saveDecay").set(True)
mdcsim = simalg.createTool("MDCAnaMgr")
mdcsim.property("InducedQSim").set(True)#Set this to false if you cannot
tolerate the simulation speed.
itksim = simalg.createTool("ITKAnaMgr")
itksim.property("UsePAI").set(True)
factory = task.createSvc("FullSimFactory/FullFactory")
factory.property("G4Verbose").set(0) #smaller ==> less output
factory.property("OpticalSimOn").set(True)
factory.property("PAISimOn").set(True)#Set this to false if you cannot tolerate
the simulation speed.
#note: If you want to use dE/dX particle identification, it cannot be set to
false.(20240402)
factory.property("ITKPAISimOn").set(True)
#important!! Incorrect seed usage may result in identical outcomes across
multiple simulations.
factory.property("useRandom").set(False) #If this is set to true, time will be
used as the random seed;
#otherwise, the seed set in the next line will be used.
factory.property("RandomSeed").set(304334425)
factory.property("AnaMgrList").set(["GeoAnaMgr","GeneratorMgr","MCTruthWriter","
ITKAnaMgr","MDCAnaMgr","ECALAnaMgr","DTOFAnaMgr","RICHAnaMgr","MUDAnaMgr"])
import MUDIDSvc
mudsvc= task.createSvc("MUDIDSvc")
task.setEvtMax(100)
task.show()
task.run()
```

粒子枪设置

全模拟设置

产生子工具设置

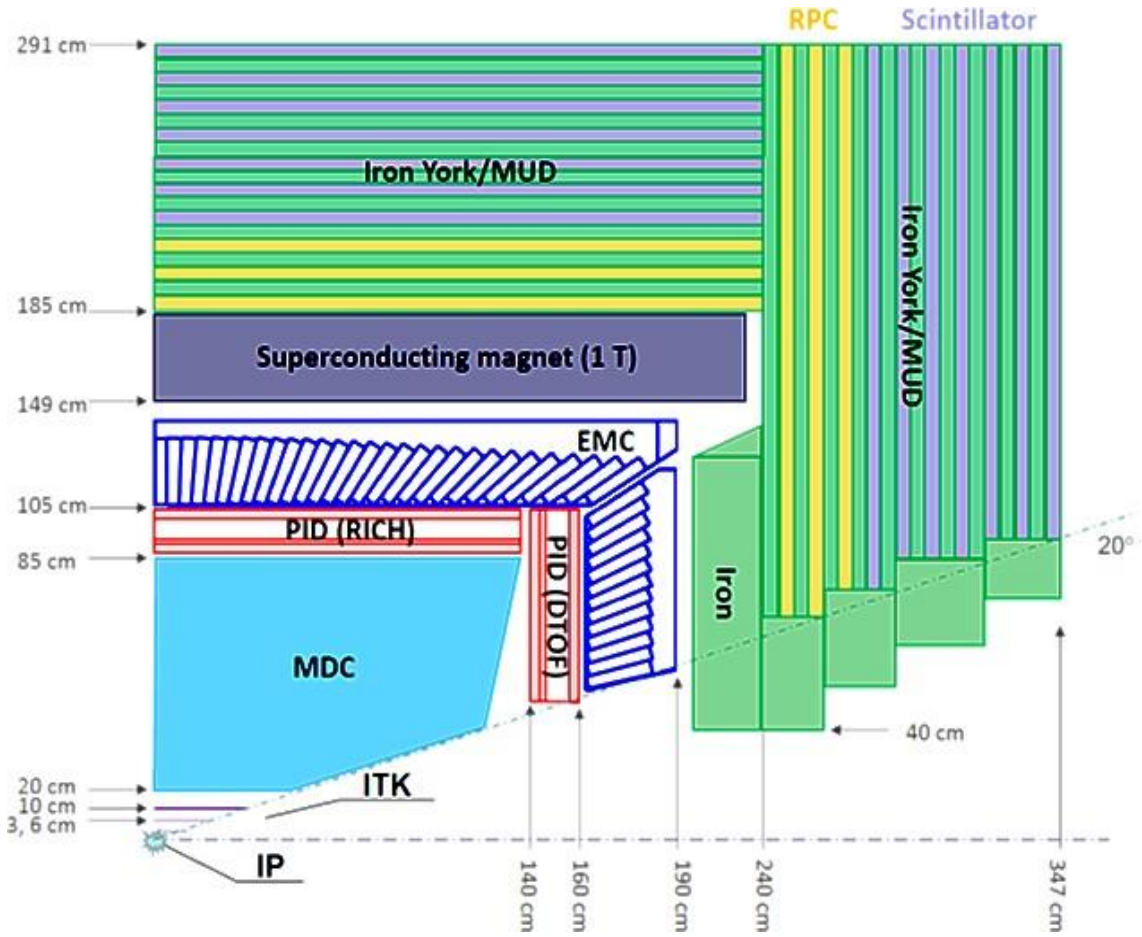
几何工具设置

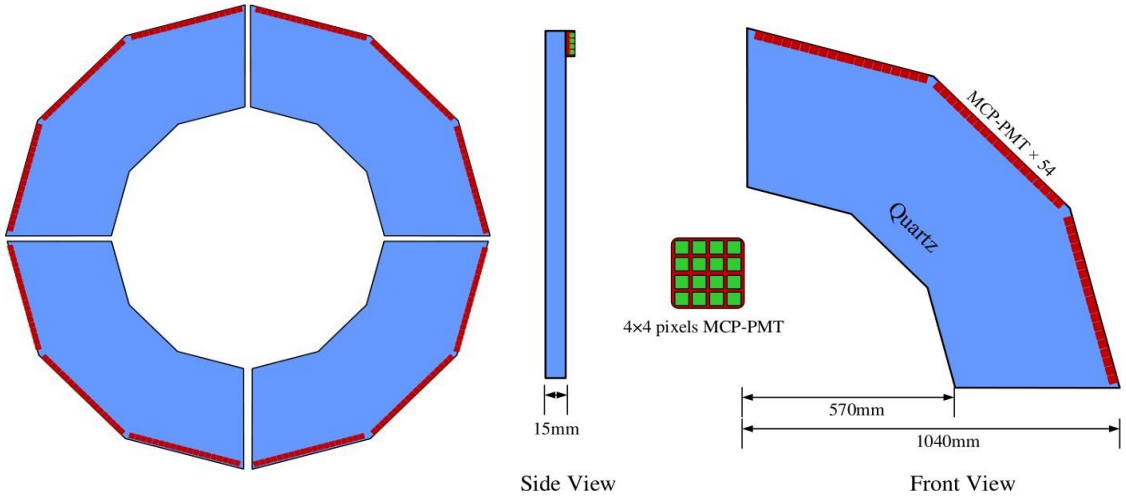
保存MCTruth

探测器定制模拟设置

通用模拟设置

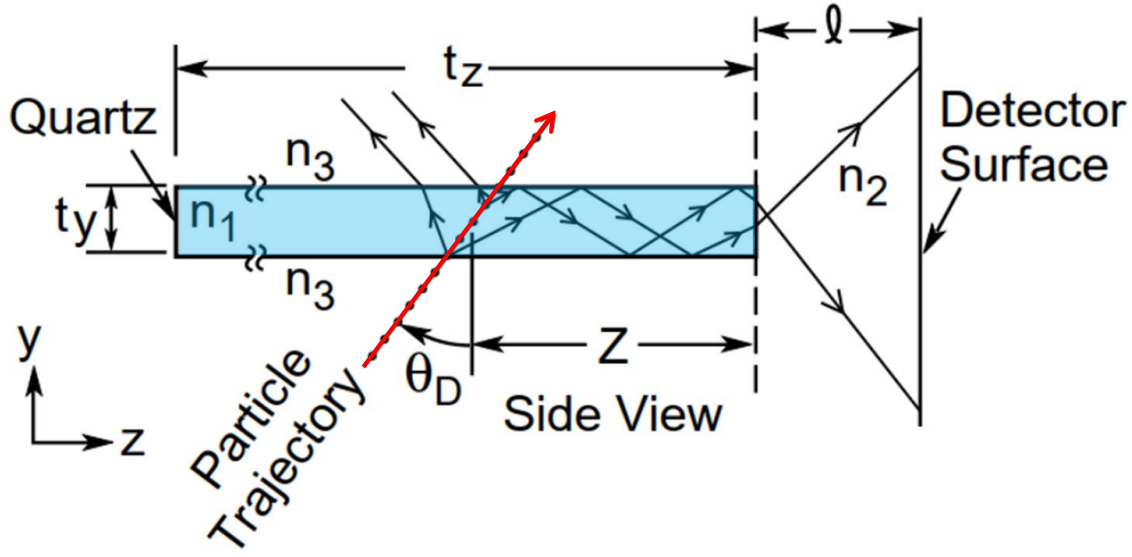
以DTOF为例

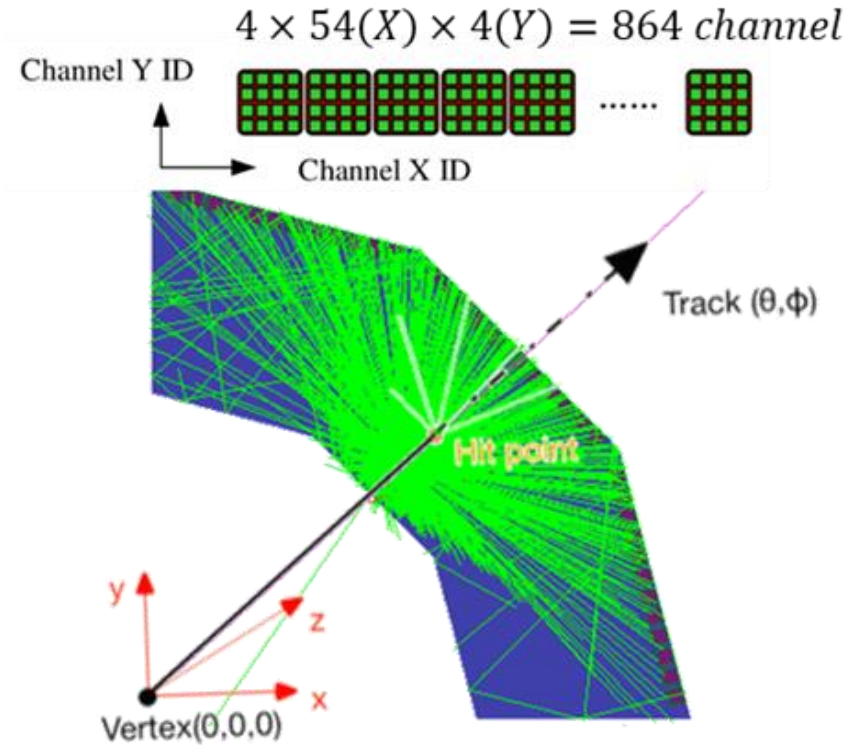




- STCF端盖粒子鉴别探测器
  - DIRC-like飞行时间探测器
- 极角覆盖范围 20°-35°
- 多阳极MCP-PMT与高精度定时电路
- 系统定时精度<50 ps（包含σ_{T0}=40 ps）

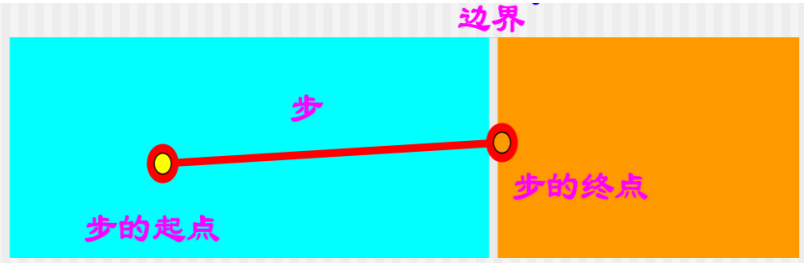
DIRC原理





- DIRC：Detection of Internally Reflected Cherenkov light
- 石英折射率~1.47，大于空气折射率
- 光在石英表面发生全反射，传输到晶体末端被探测

定义HIT



- 基于Podio定义EDM，位于$OFFLINETOP/DataModel/datalayout.yaml

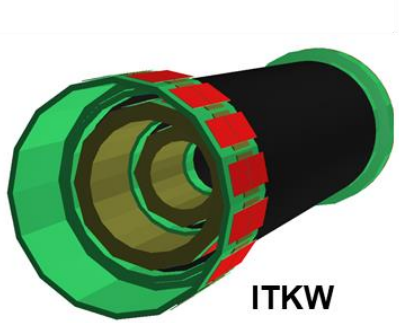
```
DTOFBarHit:
  Description: "Data class for Truth of Charged Track in DTOF"
  Author: "SDU"
  Members:
    - int type          // Hit type
    - int trackID       // Track index
    - unsigned int eventID   // MC Event id
    - Vector3d momentum    // Momentum components [Mev]
    - double energy       // energy of photon
    - double time         // Time since event start [ns]
    - double length       // Track length since creation [mm]
    - double eloss        // Energy loss at this point [Mev]
    - int detectorID      // Detector unique identifier
    - Vector3d position    // Position of hit [mm]
    - int sectorID  // sector index
    - double mass //  particle mass
    - int pdgID    // PDG code
  OneToOneRelations:
    - MCParticle Particle // Primary MCParticle that caused the hit.
```

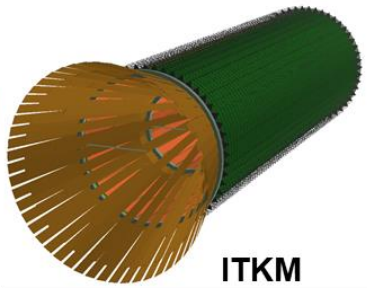
```
DTOFPDHit:
  Description: "Data class for Photon in DTOF"
  Author: "SDU"
  Members:
    - int type          // Hit type
    - int trackID       // Parent Track index
    - Vector3d momentum    // Momentum components [Mev]
    - double energy       // energy of photon
    - double time         // Time since event start [ns]
    - double length       // Track length since creation [mm]
    - int detectorID      // Detector unique identifier
    - Vector3d position    // Position of hit [mm]
    - int sectorID // Sector index
    - int channelX // Channel index X
    - int channelY // Channel index Y
    - int pdgID // PDG code
    - double timeStart // Time of the photon creating
  OneToOneRelations:
    - MCParticle time Start // Primary MCParticle that caused the hit.
```

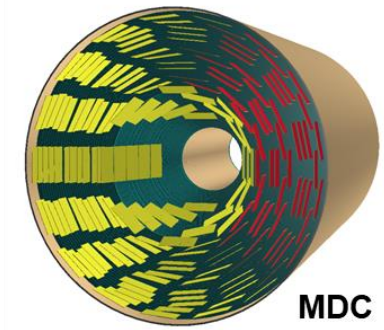
- DTOFBarHit，虚拟Hit
  - 粒子进入辐射体点相关信息，可选择不存
- DTOFPDHit，真实Hit
  - 光子被探测点相关信息，必要信息

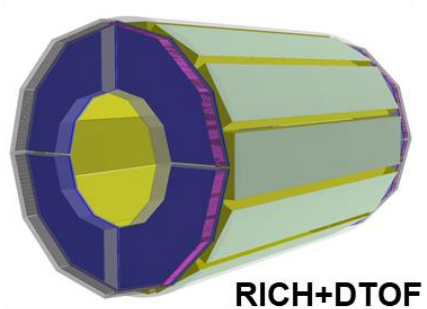
几何设置

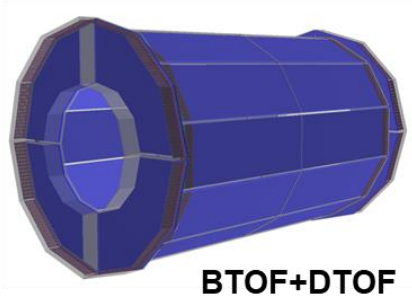
- 基于DD4hep定义几何
  - 几何参数：$OFFLINETOP/ Geometry/FullGeometry/compact/DTOF/04/detectorDTOF.xml
  - 解析参数构建几何：$OFFLINETOP/Geometry/FullGeometry/src/detectorDTOF_03.cpp
- DD4hep会解析xml文件，构建ROOT几何，通过dd4hep::sim::Geant4Converter转换成Geant4几何

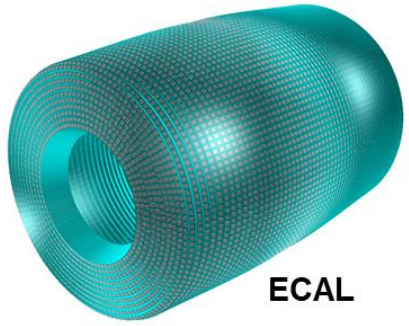


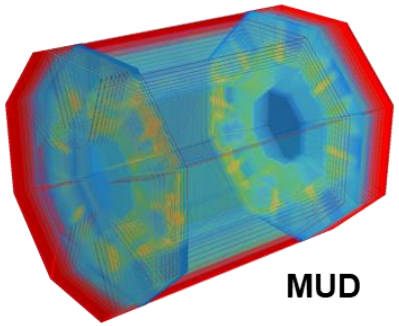


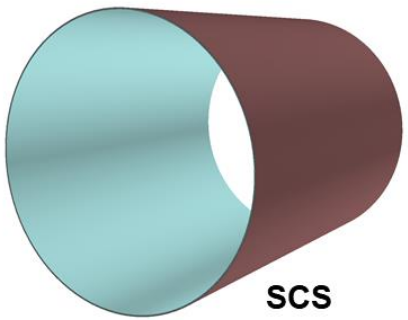


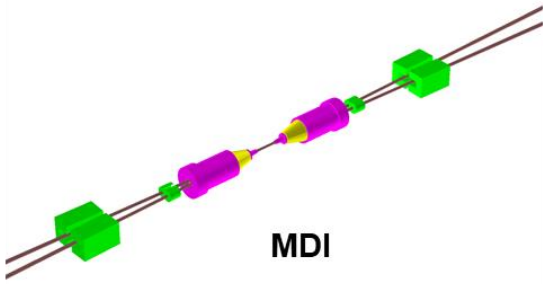












如何设置光学参数

- 两类不同属性：
  - 材料属性（体）：折射率，吸收长度，荧光产额等
  - 表面属性：反射系数，粗糙度，量子效率等

```
<properties>
   <matrix name="DTOF_RINDEX_Air" coldim="2" values="0.4*eV 1.0 6.02*eV 1.0"/>
</properties>
<materials>
   <material name="DTOF_OpticalAir" state="gas">
     <D unit="g/cm3" value="0.00120479"/>
     <fraction n="1" ref="G4_AIR"/>
     <property name="RINDEX" ref="DTOF_RINDEX_Air"/>
   </material>
 </materials>
```

定义空气折射率

```
<properties>
   <matrix name="DTOF_REFLECTIVITY_QuartzAir" coldim="2" values="
     1.909*eV 0.9999 1.939*eV 0.9999 1.970*eV 0.9999 2.001*eV 0.9999 2.033*eV 0.9999
     ……
     6.216*eV 0.9926"/>
</properties>
<surfaces>
   <opticalsurface name="DTOFQuartzAirSurface" finish="ground" model="unified" type="dielectric_dielectric" value="0.1*deg">
     <property name="REFLECTIVITY" ref="DTOF_REFLECTIVITY_QuartzAir"/>
   </opticalsurface>
</surfaces>
```

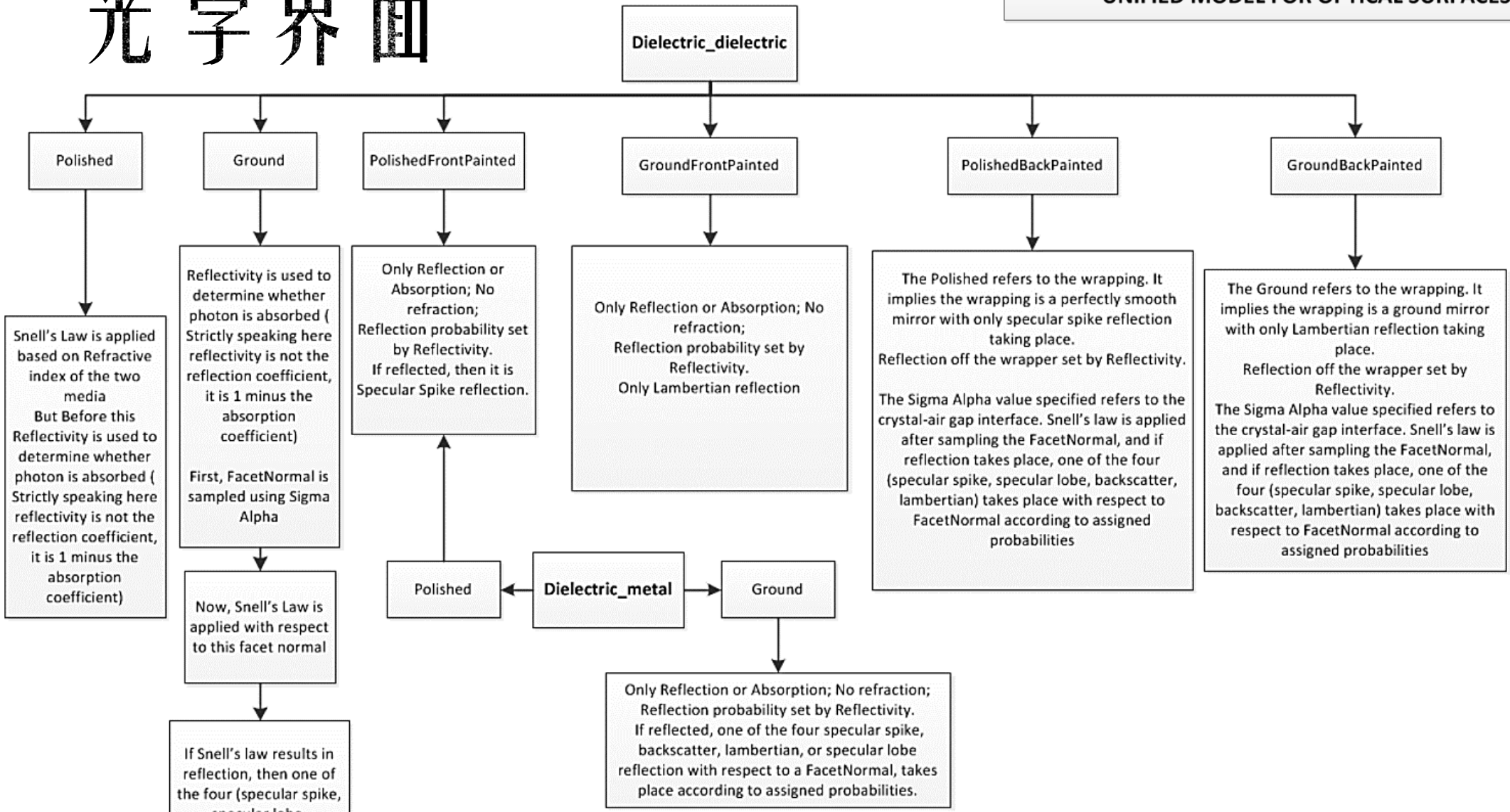
定义辐射体与空气界面反射率

```
OpticalSurfaceManager surfMgr = description.surfaceManager();
OpticalSurface QuartzAirSurf = surfMgr.opticalSurface("DTOFQuartzAirSurface");
for (int i = 0; i < SectorNu * 2; i++)
   BorderSurface(description, Dtof, _toString(i, "DTOFQuartzAirSurf_%d"), QuartzAirSurf, pvQuartz, pvSector[i]);
```

调用界面定义

光学界面

UNIFIED MODEL FOR OPTICAL SURFACES



NOTE: Applying Snell's Law includes applying Fresnel's equations of reflection and refraction, and these combined may result in Fresnel Refraction or Total Internal Reflection or Fresnel Reflection

PHYSICSLIST

```
G4VUserPhysicsList* FullSimFactory::createPhysicsList()
{
  //return new QGSP_BERT;
  if(m_UseQGSP) return new QGSP_BERT;
  //return new PhysicsList;
  return new PhysicsList(m_G4Verbose,m_PAI,m_Optical,m_PAIITK,m_ITKMStepLimiter, m_ITKMMaximumStep);
}
```

使用预定义QGSP_BERT或者自定义PhysicsList

```
PhysicsList::PhysicsList(int ver, bool PAIon, bool OpticalOn, bool PAIITKOn, bool ITKMStepLimiter,
 double ITKMMaximumStep): G4VModularPhysicsList(),m_PAIOn(PAIon), m_OpticalOn(OpticalOn), m_PAIITKOn(PAIITKOn),
                          m_ITKMStepLimiter(ITKMStepLimiter), m_ITKMMaximumStep(ITKMMaximumStep)
{      加载不同类物理过程，部分过程自定义
 m_STCFPhysics.push_back(new G4EmStandardPhysics(ver)); // EM Physics
 m_STCFPhysics.push_back(new G4EmExtraPhysics(ver)); // Synchroton Radiation & GN Physics
 m_STCFPhysics.push_back(new G4DecayPhysics(ver)); // Decays
 m_STCFPhysics.push_back(new G4HadronElasticPhysics(ver)); // Hadron Elastic scattering
 m_STCFPhysics.push_back(new G4HadronPhysicsQGSP_BERT(ver)); // Hadron Physics
 m_STCFPhysics.push_back(new G4StoppingPhysics(ver)); // Stopping Physics
 m_STCFPhysics.push_back(new G4IonPhysics(ver)); // Ion Physics
 m_STCFPhysics.push_back(new G4NeutronTrackingCut(ver)); // Neutron tracking cut
 if (m_OpticalOn) m_STCFPhysics.push_back(new OpticalPhysics("optical", ver)); // Optical Physics
 G4EmParameters::Instance()->SetVerbose(ver);
 if (m_PAIOn) G4ProductionCutsTable::GetProductionCutsTable()->SetEnergyRange(1 * eV, 5 * GeV);
G4Region *region = G4RegionStore::GetInstance()->GetRegion("ITKMVolume",false);
 if (m_ITKMStepLimiter && region) {
       region->SetUserLimits(new G4UserLimits(m_ITKMMaximumStep * CLHEP::um));
       m_STCFPhysics.push_back(new G4StepLimiterPhysics());
}}
```

```
void PhysicsList::ConstructParticle()
{
  // create particles
  for (size_t i = 0; i < m_STCFPhysics.size(); i++)
    m_STCFPhysics[i]->ConstructParticle();
}
```

创建粒子和物理

```
void PhysicsList::ConstructProcess()
{
  AddTransportation();
  for (size_t i = 0; i < m_STCFPhysics.size(); i++)
    m_STCFPhysics[i]->ConstructProcess();
}
```

更多细节参考$OFFLINETOP/Simulation/FullSim/src/PhysicsList.cc

用户作用类

- OSCAR已定义好的OSCAREventAction，OSCARTrackingAction等类，自动调用不同功能的用户作用类xxxAnaMgr
- 开发者只需根据需求开发的xxxAnaMgr
  - 路径$OFFLINETOPSimulation/FullSim/src/xxxAnaMgr.cc
  - 包含BeginOfRunAction(), EndOfRunAction(), BeginOfEventAction(), EndOfEventAction(), PreUserTrackingAction(), PostUserTrackingAction(), UserSteppingAction()七个固定函数

```
void DTOFAnaMgr::BeginOfEventAction(const G4Event *aEvent)
{
  m_barHit = getRWColl(DTOFBarHitCollection, "DTOFBarHitCol");
  m_pdHit = getRWColl(DTOFPDHitCollection, "DTOFPDHitCol");
}
```

获取HitCollection指针

```
void DTOFAnaMgr::UserSteppingAction(const G4Step *aStep)
{

  G4Track *track = aStep->GetTrack();
  const G4DynamicParticle *particle = track->GetDynamicParticle();
  G4StepPoint *preStepPoint = aStep->GetPreStepPoint();
  G4StepPoint *postStepPoint = aStep->GetPostStepPoint();
  G4VPhysicalVolume *prePhyVolume = preStepPoint->GetPhysicalVolume();
  G4VPhysicalVolume *postPhyVolume = postStepPoint->GetPhysicalVolume();
  if (!postPhyVolume)  return;
  ……
}
```

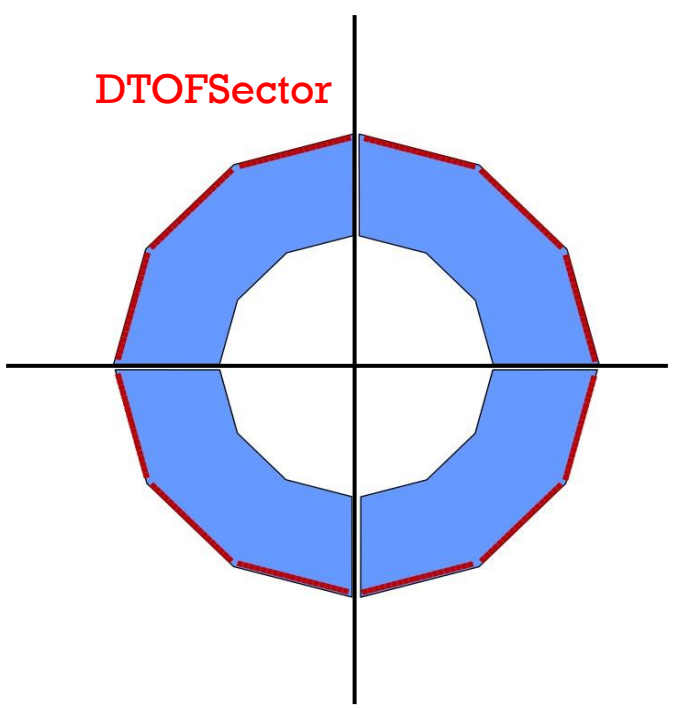
通过G4Step拿到Track、Particle、Volume等

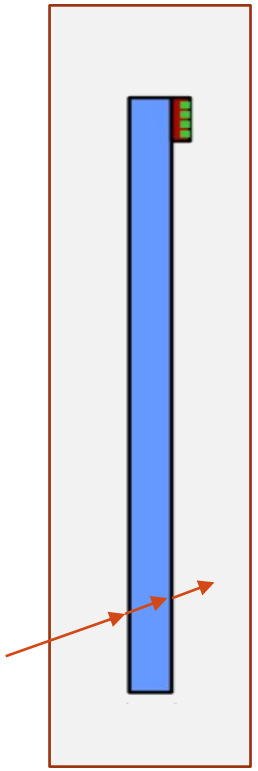
如何获取HIT信息？

```
// collect points of charge particle entering quartz
 if (postPhyVolume->GetLogicalVolume()->GetName() == "DTOFradiator" &&
     prePhyVolume->GetLogicalVolume()->GetName() == "DTOFSector" &&
     TMath::Abs(particle->GetCharge()) != 0)
 {
   auto barHit = m_barHit->create(); // MutableDTOFBarHit*

   barHit.setTrackID(track->GetTrackID());
   barHit.setPdgID(particle->GetPDGcode());
   G4ThreeVector momentum = postStepPoint->GetMomentum();
   barHit.setMomentum(Vector3d(momentum.x(), momentum.y(), momentum.z()));
   barHit.setEnergy(particle->GetKineticEnergy());
   barHit.setTime(track->GetGlobalTime());
   barHit.setLength(track->GetTrackLength());
   barHit.setEloss(aStep->GetTotalEnergyDeposit());
   barHit.setDetectorID(prePhyVolume->GetCopyNo());
   G4ThreeVector position = postStepPoint->GetPosition();
   barHit.setPosition(Vector3d(position.x(), position.y(), position.z()));

   barHit.setSectorID(prePhyVolume->GetCopyNo());
   barHit.setMass(particle->GetMass());
 }
```

判断是否是否在界面

从collection初始化新的hit





- 先定义DTOFSector空气体积，再往sector里面放置DTOFradiator，暗盒等结构。

如何获取HIT信息？

```
// check if we are in scoring volume
 if (postStepPoint->GetStepStatus() == fGeomBoundary &&
     particle->GetDefinition()->GetParticleName() == "opticalphoton" &&
     postPhyVolume->GetLogicalVolume()->GetName() == "DTOFsens")
 {
   G4ProcessManager *pm = G4OpticalPhoton::OpticalPhoton()->GetProcessManager();
   G4int nprocesses = pm->GetProcessListLength();
   G4ProcessVector *pv = pm->GetProcessList();
   G4OpBoundaryProcess *boundary = NULL;
   for (int i = 0; i < nprocesses; i++)
   {
     if ((*pv)[i]->GetProcessName() == "OpBoundary") {
       boundary = (G4OpBoundaryProcess *)(*pv)[i];   break; }
   }
   if (boundary) {
     G4OpBoundaryProcessStatus theStatus = boundary->GetStatus();
     switch (theStatus){
       case Absorption:
         break;
       case Detection:
       {
         // do something to save PDHit
         break;
       }
       default:
         break;
     }
   }
 }
```

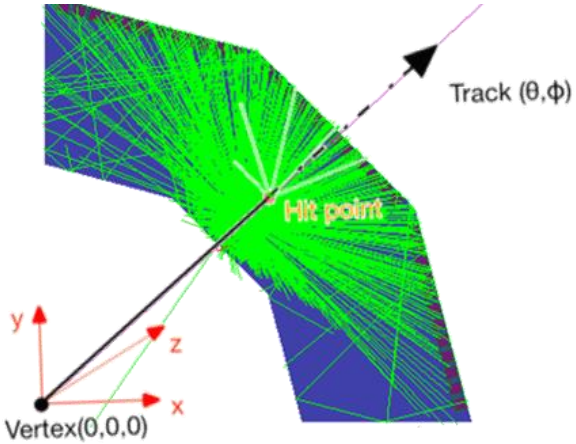
判断是否光学粒子，是否在界面

判断在界面是否被探测

```
auto PDHit = m_pdHit->create();

G4int CathodeID = postStepPoint->GetTouchable()
                        ->GetVolume(0)->GetCopyNo();
G4int PhotonDetID = postStepPoint->GetTouchable()
                        ->GetVolume(1)->GetCopyNo();
G4int SectorID = postStepPoint->GetTouchable()
                        ->GetVolume(2)->GetCopyNo();

PDHit.setType((track->GetCreatorProcess()->GetProcessName())
                        == "Cerenkov" ? 0 : 1);

// parentID, to match peHit to charged particle
PDHit.setTrackID(track->GetParentID());

G4ThreeVector momentum = postStepPoint->GetMomentum();
PDHit.setMomentum(Vector3d(momentum.x(), momentum.y(), momentum.z()));
PDHit.setEnergy(particle->GetKineticEnergy());
PDHit.setTime(track->GetGlobalTime());
PDHit.setLength(track->GetTrackLength());
PDHit.setDetectorID(SectorID);

G4ThreeVector position = postStepPoint->GetPosition();
PDHit.setPosition(Vector3d(position.x(), position.y(), position.z()));
PDHit.setPdgID(particle->GetPDGcode());
PDHit.setTimeStart(track->GetLocalTime());
PDHit.setSectorID(SectorID);
PDHit.setChannelX(PhotonDetID * nPixel + CathodeID / nPixel);
PDHit.setChannelY(CathodeID % nPixel);
```

如何加快模拟

- 单个带电粒子在DTOF辐射体中产生几百个光学光子，仅有约30个被探测（主要由量子效率决定）
- 终止不被探测的光子模拟可以加块整体模拟所需时间
- 将量子效率模拟由Geant4调用改为自主调用



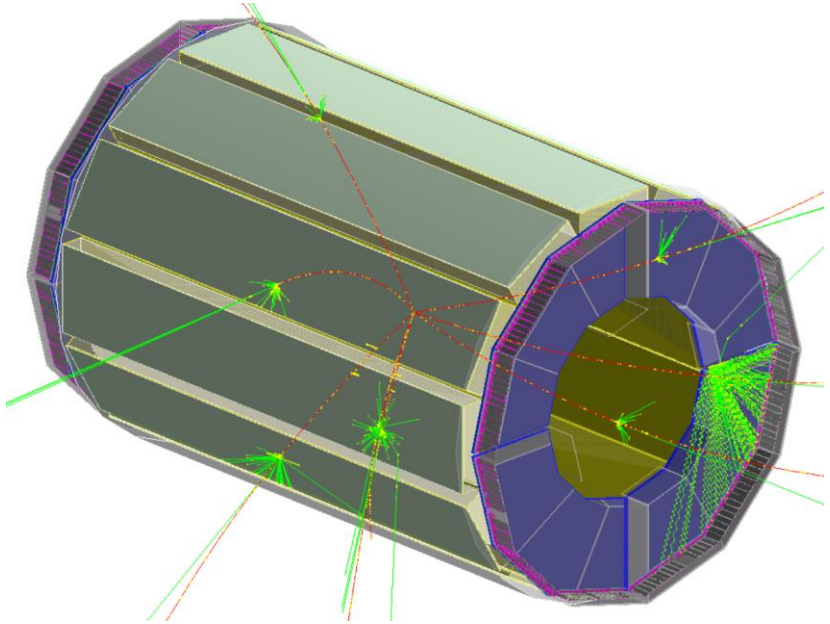

```
if (fFastSim)
 {                                                          获取次级粒子
   const G4TrackVector *secondary = aStep->GetSecondary();
   if (secondary && secondary->size() != 0) {
     int Nsecondary = secondary->size();
     int NsecondaryCurrentStep = aStep->GetNumberOfSecondariesInCurrentStep();
     G4ThreeVector direction = aStep->GetDeltaPosition();
     for (int i = Nsecondary - NsecondaryCurrentStep; i < Nsecondary; i++) {
       G4Track *sectrack = secondary->at(i);
       if (sectrack->GetDefinition()->GetParticleName() != "opticalphoton")  continue;
       double SecE = sectrack->GetTotalEnergy();
       if (prePhyVolume->GetLogicalVolume()->GetName() == "DTOFradiator")   QE抽样
       {
         double CE = GetQE10754(SecE / CLHEP::eV) * GetTGrease(SecE / CLHEP::eV);
         double rand = G4UniformRand();
         if (rand > CE)
         {
           sectrack->SetTrackStatus(fStopAndKill); ///*****//     Kill 粒子
           continue;
         }
}}}}
```

一些小技巧

```
// 局部坐标转换成全局坐标
G4Navigator* navigator = G4TransportationManager::GetTransportationManager()->GetNavigatorForTracking();
G4ThreeVector globalPosition = navigator->GetLocalToGlobalTransform().TransformPoint(G4ThreeVector(0, 0, 0));

// 全局坐标转换成局部坐标
G4ThreeVector position = postStepPoint->GetPosition();
G4ThreeVector localPosition = navigator->GetGlobalToLocalTransform().TransformPoint(position);

// 获取不同Volume ID
G4int CathodeID = postStepPoint->GetTouchable()->GetVolume(0)->GetCopyNo();
G4int PhotonDetID = postStepPoint->GetTouchable()->GetVolume(1)->GetCopyNo();
G4int SectorID = postStepPoint->GetTouchable()->GetVolume(2)->GetCopyNo();
```

模拟脚本代码

- 脚本路径$OFFLINETOP/Examples/RunCheck/FullChain/sim_xxx.py
  - sim_particleGun.py、sim_generator.py、sim_particleGun_ITKM.py、sim_particleGun_BTOF.py
  - 其他所需vis.mac, run.mac, decay card也都在相同路径



脚本中打开下面开关可以可视化显示
（仅适用粒子枪，不推荐，很卡）
g4svc.property("VisMac").set("vis.mac")