

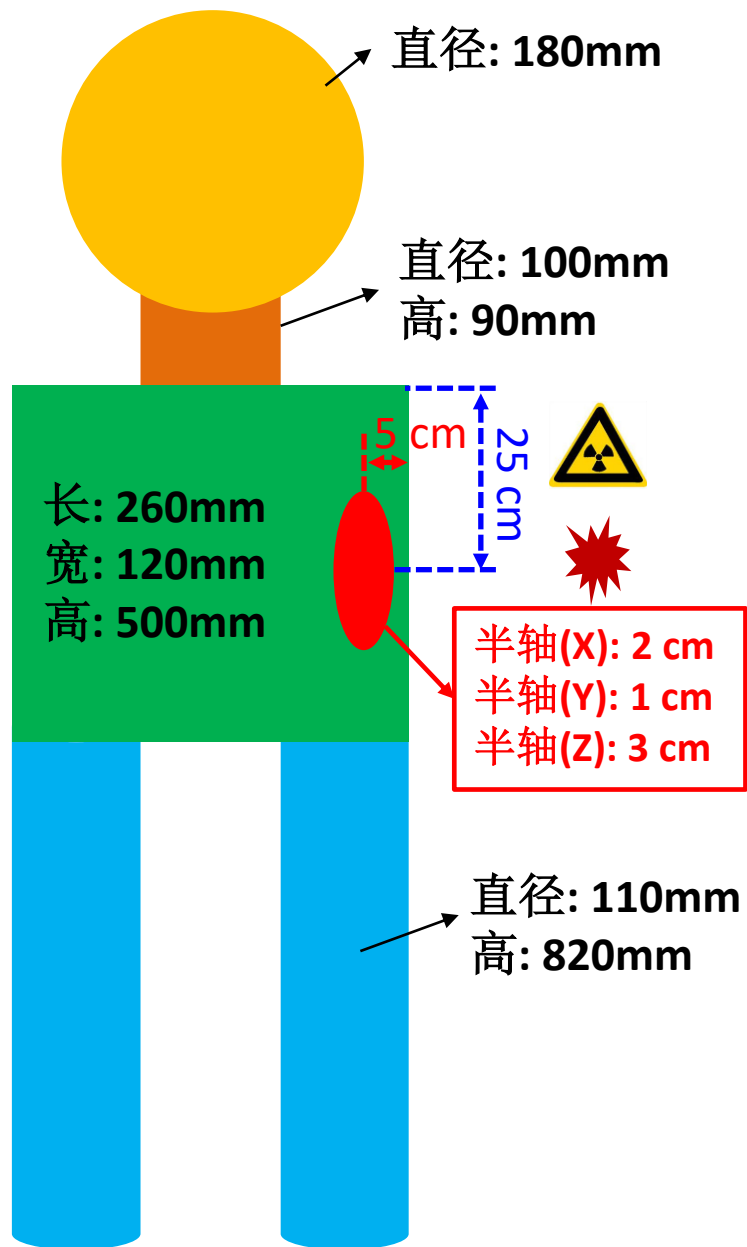
# 探测器模拟与数据分析

## --探测器模拟基础

曹国富

中国科学院高能物理研究所





- ❖ 放疗是肿瘤治疗的一种常用手段，射线种类有伽马/质子/中子/离子等。
- ❖ 请设计并实现一个探测器模拟程序，从能量沉积的大小、尺度和位置几方面，对伽马和质子射线能量进行优化，对比使用这两种射线进行放疗的优劣。
- ❖ 硼中子俘获疗法(BNCT)是一种新的疗法，2020年全球首个BNCT设备在日本获批上市。假如使用0.5 eV的热中子束流，请研究预期治疗效果与肿瘤区内<sup>10</sup>B原子浓度的关系，以及束流截面大小的影响。最后，与其它射线的治疗效果进行对比。
- ❖ 肿瘤大小和位置如图红色区域所示。

## ❄ 用户使用事例产生子时需要继承于其基类

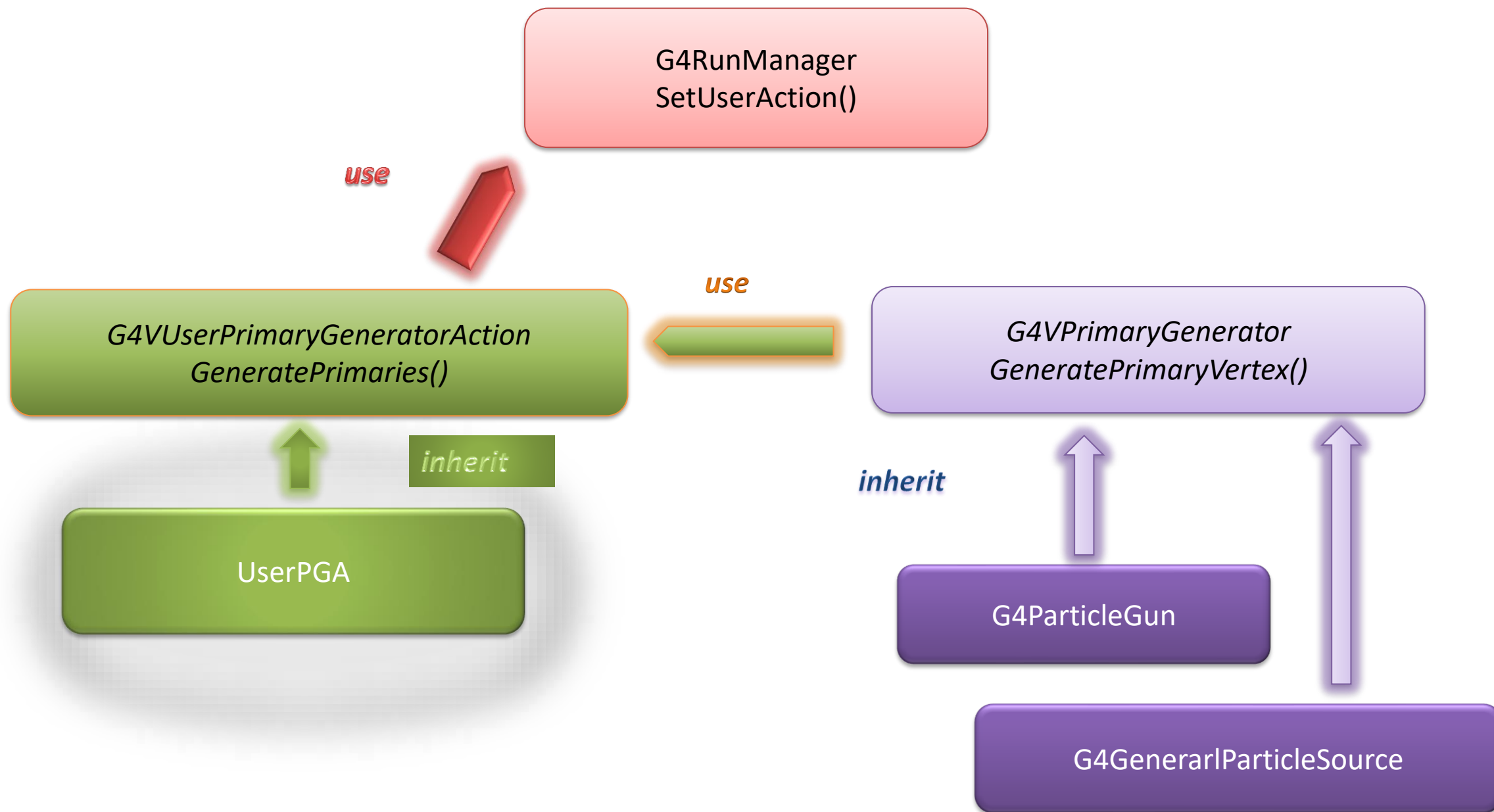
**G4VUserPrimaryGeneratorAction**

- 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**)



- ❄ 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 MyPrimaryGeneratorAction::MyPrimaryGeneratorAction
{
    // particleGun is a member of this class.
    particleGun = new G4ParticleGun();
}

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

## ❄ *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

```
void MyPrimaryGeneratorAction::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。



```
// 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](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

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

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

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

## ❄ 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



## ❄ 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

## ❄ 电磁相互作用

- 标准电磁过程, 从  $\sim 1 \text{ keV}$  到  $\sim \text{PeV}$
- 低能电磁过程, 从  $250 \text{ eV}$  到  $\sim \text{PeV}$
- 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  $\sim \text{TeV}$
- 电子和伽玛与核的相互作用, 从  $10 \text{ MeV}$  到  $\sim \text{TeV}$

## ❄ 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()

```
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();  
}
```

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 ( $\gamma$ , X)	G4Gamma	gamma	22
photon (optical)	G4OpticalPhoton	opticalphoton	(0)
geantino	G4Geantino	geantino	(0)
charged geantino	G4ChargedGeantino	chargedgeantino	(0)



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

```
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();  
}
```

```
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;
            G4UrbanMscModel195* msc1 = new G4UrbanMscModel195();
            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
    }
}
```

```
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
    }
}
```

```
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);
        }
    }
}
```



```
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  
}
```

```
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();
}
```

## 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/](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](https://geant4.kek.jp/lxr/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/](https://geant4.kek.jp/lxr/source/physics_lists/lists/)

## ❄ **Geant4一共有多个打包好的物理列表**

- **FTFP\_BERT**
- **QBBC**
- **QGSP\_BERT**
- **QGSP\_BIC**
- **Shielding**

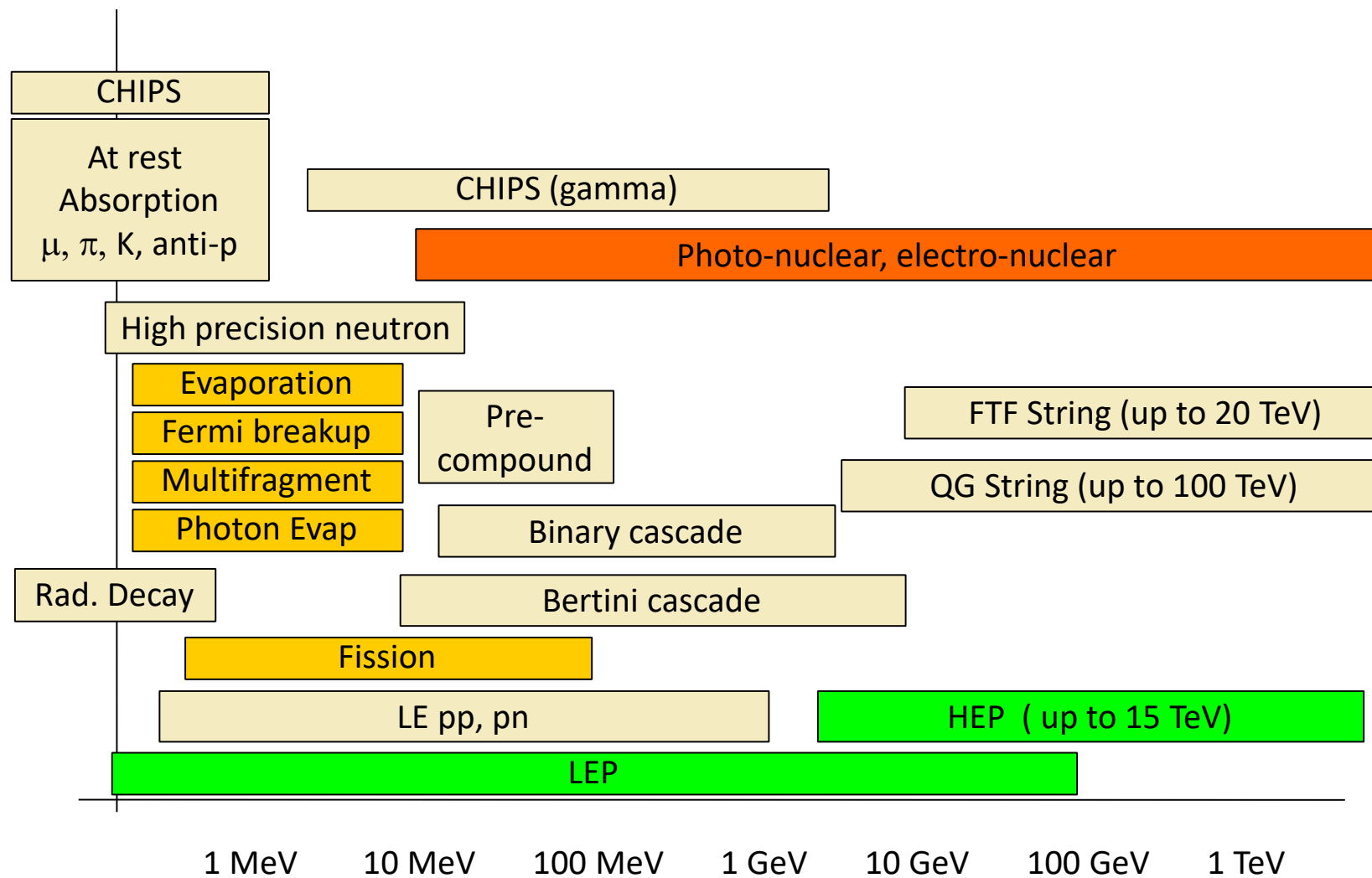
<https://geant4-userdoc.web.cern.ch/UsersGuides/PhysicsListGuide/html/index.html>

## ❄ **强子过程命名规则**

- **QGS** → Quark Gluon String model ( $> \sim 20$  GeV)
- **FTF** → Fritiof string model ( $> \sim 5$  GeV)
- **BIC** → Binary Cascade ( $< \sim 10$  GeV)
- **BERT** → Bertini-style cascade ( $< \sim 10$  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





## ❄ 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/>

❄ **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})
```