Physics List

歳藤利行
KEK/JST

Physics List

- Physics List においてシミュレーションに必要な
  - 粒子の種類
  - それぞれの粒子がもつ物理プロセス
    - プロセスの必要とするパラメータ
  - Cut in Range

  をユーザーが設定する

? なぜユーザーが設定する必要があるの
- それぞれの物理プロセスには適用限界がある
- ユーザーによって
  - 必要とするエネルギーレンジ
  - 観測物理量に要求される精度

  が異なる
- 計算時間も重要

- G4VUserPhysicsList class
- How to define physics list
- Modular physics list
- Packaged physics lists

G4VUserPhysicsList

- ユーザーはG4VUserPhysicsListを継承したクラスを作りその中でphysics listを実装する
  - main関数の中でrun managerに登録する

- ここでの例:

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

- ユーザーはConstructParticle, ConstructProcess および SetCuts関数を実装しなければならない

G4VUserPhysicsList: 実装が要求される関数

- ConstructParticle()　- シミュレーションに必要なすべての粒子を選択し、定義する
- ConstructProcess()　- 各粒子に対して必要なprocess(物理過程)を割り当てる
  - Processの例
  - => ガンマ線のコンプトン散乱、陽子の弾性散乱、崩壊、、、
- SetCuts()　- 2次粒子生成のレンジカットを設定する

ConstructParticle()

```
void MyPhysicsList::ConstructParticle()
{
    G4BaryonConstructor   baryonConstructor;
    baryonConstructor.ConstructParticle();

    G4BosonConstructor bosonConstructor;
     bosonConstructor.ConstructParticle();
      ....
    ....
}
```

これでbaryonが使える

これでboson（実質的にはphoton)が使える

他にもG4LeptonConstructor, G4MesonConstructorなどがある

ConstructParticle()
(別のやり方)

```
void MyPhysicsList::ConstructParticle()
{
    G4Electron::ElectronDefinition();
    G4Proton::ProtonDefinition();
    G4Neutron::NeutronDefinition();
    G4Gamma::GammaDefinition();
    ....
    ....
}
```

ConstructProcess()

電磁相互作用とその他一般を別々の関数で記述する例

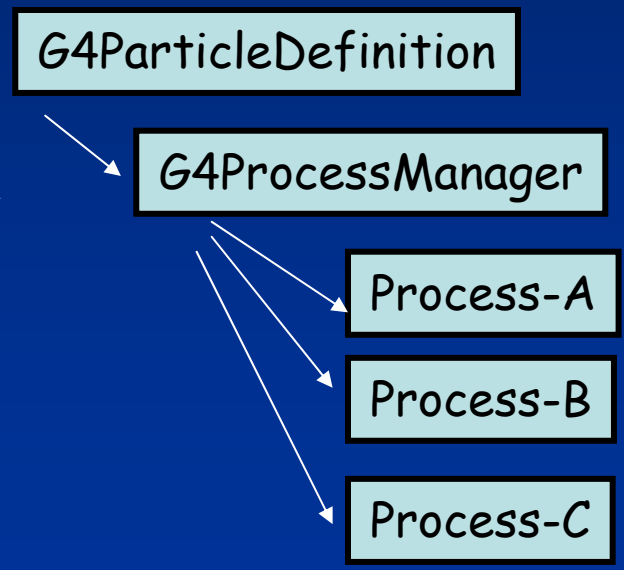
```
void MyPhysicsList::ConstructProcess()
{
        AddTransportation();
                //粒子をトラッキングするために必須

        ConstructEM();
        ConstructGeneral();
}
```

プロセスの登録

- 各粒子がPM(*ProcessManager*)を持っていて、PMにその粒子が扱うプロセスが登録される。

```
G4ParticleDefinition* gamma=
    G4Gamma::GammaDefinition()
// 粒子は、singleton
G4ProcessManager* pm= gamma->GetProcessManager();
//   PMは粒子毎にある
pm-> AddProcess(....);
// その粒子の考慮すべきプロセスを登録
```



ConstructEM()

```
void MyPhysicsList::ConstructEM()
{
    theParticleIterator->reset();
    while( (*theParticleIterator)() ) {
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        G4String particleName = particle->GetParticleName();
```

この部分の詳細

```
    }
}
```

example of PhysicsList

ordAtRestDoIt ordPostStepDoIt

```
if (particleName == "e-") {
pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
pmanager->AddProcess(new G4eIonisation,        -1, 2,2);
pmanager->AddProcess(new G4eBremsstrahlung,    -1,-1,3);
}

else if (particleName == "e+") {
pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
pmanager->AddProcess(new G4eIonisation,        -1, 2,2);
pmanager->AddProcess(new G4eBremsstrahlung,    -1,-1,3);
pmanager->AddProcess(new G4eplusAnnihilation,   0,-1,4);
}
```

ordAlongSteptDoIt

-1 - parameter to indicate InActive DoIt

```
if (particleName == "mu+" || particleName == "mu-") {
pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);
pmanager->AddProcess(new G4MuIonisation,       -1, 2,2);
pmanager->AddProcess(new G4MuBremsstrahlung,   -1,-1,3);
pmanager->AddProcess(new G4MuPairProduction,   -1,-1,4);
}

if ((particle->GetPDGCharge() != 0.0) &&
(!particle->IsShortLived()) &&
(particle->GetParticleName() != "chargedgeantino")) {
pmanager->AddProcess(new G4MultipleScattering, -1,1,1);
pmanager->AddProcess(new G4hIonisation,        -1,2,2);
}
```

```
if (particleName == "gamma") {
pmanager->AddDiscreteProcess(new G4PhotoElectricEffect);
pmanager->AddDiscreteProcess(new G4ComptonScattering);
pmanager->AddDiscreteProcess(new G4GammaConversion);
}
```

is a shortcut for :

```
pmanager->AddProcess(new G4PhotoElectricEffect, -1,-1,1);
pmanager->AddProcess(new G4ComptonScattering,   -1,-1,2);
pmanager->AddProcess(new G4GammaConversion,     -1,-1,3);
```

For processes which have only PostStepAction, the ordering is not important.

ConstructGeneral()

```
void MyPhysicsList::ConstructGeneral()
{
      // ここでは崩壊過程を登録する
       G4Decay* theDecayProcess = new G4Decay();
      theParticleIterator->reset();
      while( (*theParticleIterator)() ) {
          G4ParticleDefinition* particle = theParticleIterator->value();
          G4ProcessManager* pmanager = particle->GetProcessManager();
          if (theDecayProcess->IsApplicable(*particle) ) {
             pmanager->AddProcess(theDecayProcess);
             pmanager->SetProcessOrdering(theDecayProcess,idxPostStep);
             pmanager->SetProcessOrdering(theDecayProcess,idxAtRest);
          }
      }
}
```

SetCuts()

- 電磁相互作用でのproduction cutを設定
- rangeで与える
  - 各物質内でrange以上の距離を走る粒子のみが生成される
- */run/setCutコマンドで変更可能*

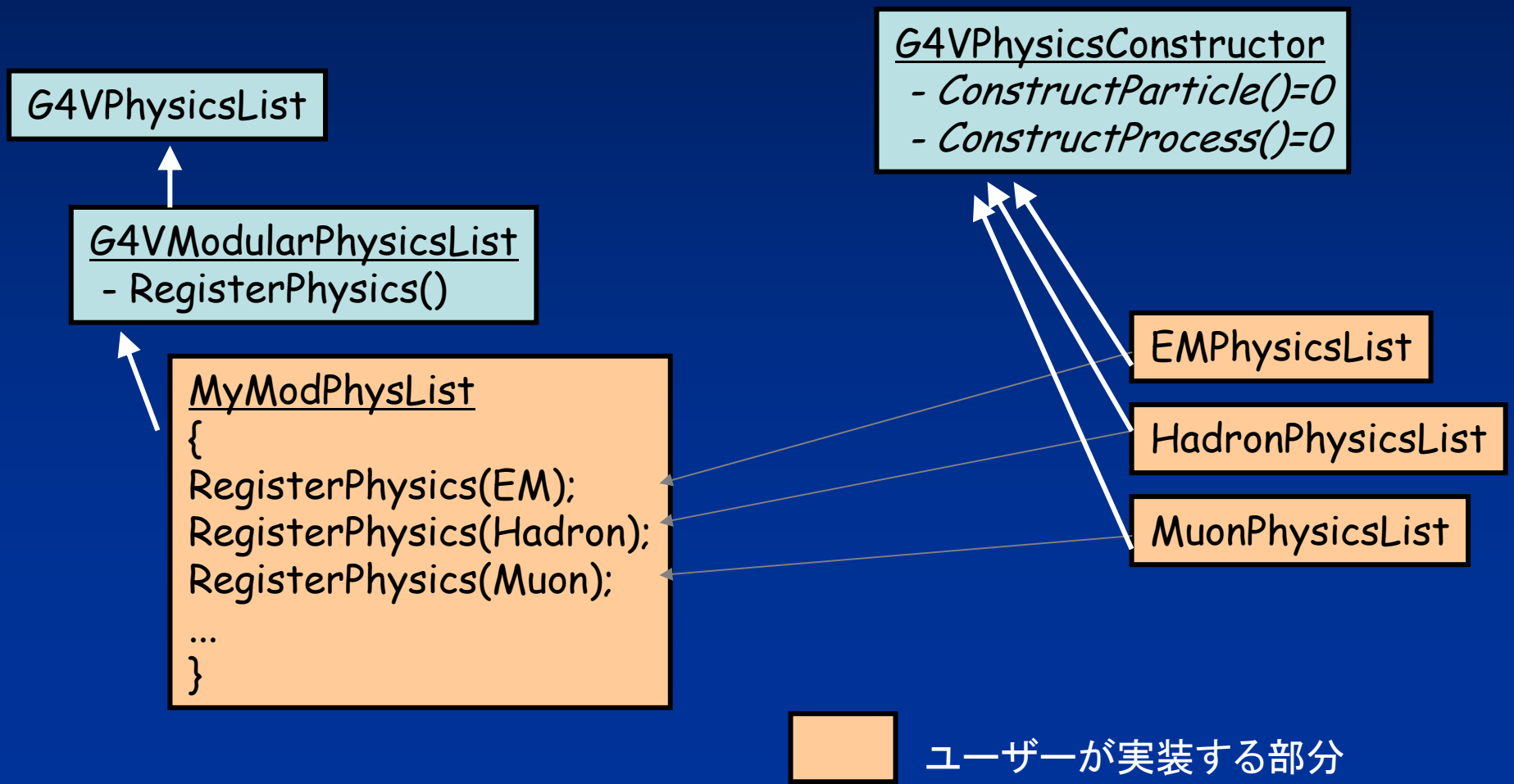
SetCuts()

```
void MyPhysicsList::SetCuts()
{
    defaultCutValue = 1.0*mm;
    SetCutValue(defaultCutValue, “gamma”);
    SetCutValue(defaultCutValue, “e-”);
    SetCutValue(defaultCutValue, “e+”);

    //  これで全て。他の粒子については不要。
}
```

Modular Physics List

- Modular Physics Listの利用
  - 1つのPhysicsListクラスにすべてを長々書いていくのは分かりにくいので、カテゴリに分けてPLを登録できるようにしたもの。
  - カテゴリ分けは自由



G4VModularPhysicsList

- 特徴
  - G4VUserPhysicsListから継承されている
  - AddTransportation() は全ての登録した粒子に対して自動的に呼ばれる

G4VModularPhysicsListの例

- Constructor:

```
MyModPhysList::MyModPhysList():  G4VModularPhysicsList()
{
  defaultCutValue = 1.0*mm;
  RegisterPhysics( new ProtonPhysics() );
        // 陽子に関するすべての物理過程
  RegisterPhysics( new ElectronPhysics() );
        // 電子に関するすべての物理過程
  RegisterPhysics( new DecayPhysics() );
        // 不安定粒子に関する物理
}
```

- Set Cuts:

```
void MyModPhysList::SetCuts()
{
  SetCutsWithDefault() ;   // defaultCutValueでproduction cutが設定される
}
```

Packaged Physics Lists

- 実装ずみのUserDetectorConstructionクラス

例えば QGSP (Quark gluon string modelがベース) を使うには

```
#include "QGSP.hh"
```

main()関数の中で

```
G4VUserPhysicsList* physics = new QGSP;
runManager->SetUserInitialization(physics);
```

とするだけでよい

/examples/novice/N04を参照

他にも
LHEP LEP,HEP parametrized model
QGSP_BERT (Bertini cascade below 3GeV)
QGSP_BIC (Binary cascade below 3GeV)
などがある
/source/physics_lists/lists/includeの中のヘッダファイルを参照

まとめ

- アプリケーションで必要な粒子、物理過程、production cut は physics listで設定する
- ユーザーが継承してphysics listを実装するためのクラスが2種類ある
  - **`G4VUserPhysicsList`** – 単純なphysics list用
  - **`G4VModularPhysicsList`** – 詳細なphysics用
- Geant4の中であらかじめパッケージ化されたphysics listもある
  - QGSP,LHEPなど