Monte Carlo ゼミ 3

K. Miwa

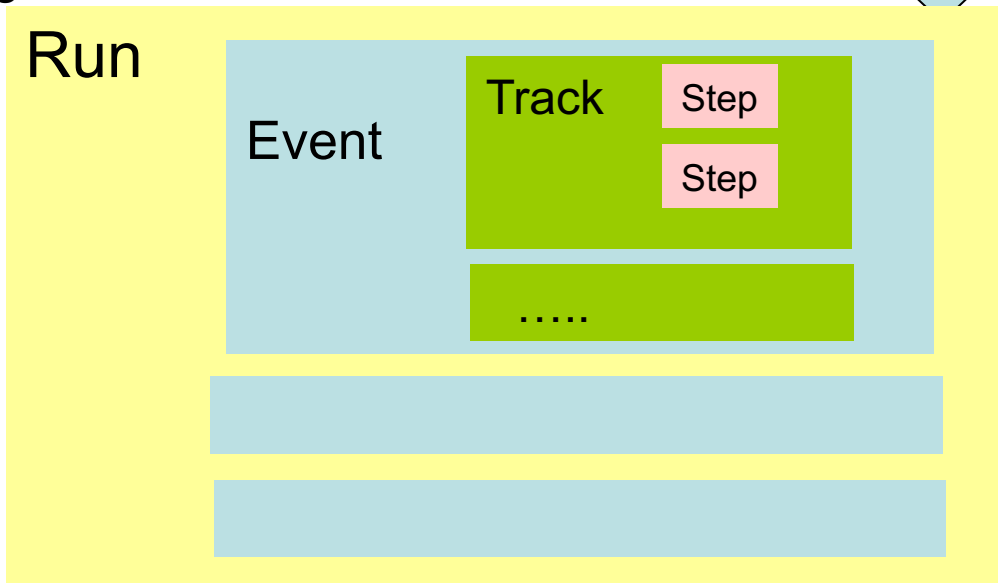
Physics process

- Geant4の中で物理プロセスに関する情報をインプットする必要がある
  - 使用する粒子を定義
  - 粒子に対して起こりうる物理プロセスの定義
    - 粒子の崩壊
    - 物質との相互作用
    - など

基本的にはGeant4の中にプロセスが定義してあり、自分が行うsimulationに対して必要なものを過不足なくプログラムの中に組み込む必要がある。
それらの物理プロセスを取り扱うのが**PhysicsList.ccに対応する。

Overview of program (復習)

Geant4のプログラムで必須となるクラスの一つ
必ずRunManagerに登録する必要がある



Start simulation
> beamOn #eventNv

Start simulation
- Detector set up
- Physics list taken into account

- One event consists of many tracks
- One Tracks consists of many steps

Physics processの登録の仕方

- 大きく分けて3つのやり方がある
- 必要なプロセスをすべてプログラムに自分で記述する
  - 正しく入れないと想定しているプロセスを正しくsimulationされない可能性がある
- 物理プロセス毎にモジュール的にまとめられているものの中から必要なものを取り入れて用いる
  - これが使いやすいのかもしれない
- Geant4が既にパッケージとなっているものを用いる
  - 簡単ではあるが、カスタマイズが出来ない(難しい)

Physics processの登録の方法1（自分ですべて記述する）

sks@dhcp2576: /home/sks/user/miwa/MonteCarlo/geant4/N00
が参考プログラムです

- Simulationで用いられるphysics processはすべて自分で取り入れる必要があります。
  - 用いる粒子の定義
  - 用いる物理プロセス
    - 例えばdecay processを考えたときには、崩壊後に生じる粒子をすべて定義しておく必要があります。
- PhysicsList.cc と PhysicsList.hh を編集することが一般的です。
  - その中でG4VuserPhysicsListというクラスを継承したPhysicsListクラスを自分で定義する必要があります。
  - そしてその中には以下の関数が必ず定義される必要があります
    - ConstructParticle() : construction of particles
    - ConstructProcess(): construct processes and register them to particles
    - SetCuts() : setting a cut value in range to all particles

Particle definition

- Geant4 provides various types of particles used in simulations
  - Ordinary particles (electron, proton, gamma etc.)
  - Resonant particles with very shot life (vector mesons, delta baryons etc.)
  - Nuclei (deuteron, alpha, heavy ions etc.)
  - Quarks, di-quarks, and gluon
- There are 6 major particle categories
  - Lepton
  - Meson
  - Baryon
  - Boson
  - Shortlived
  - ion

```
G4KaonMinus* G4KaonMinus::Definition()
{
  if (theInstance !=0) return theInstance;
  const G4String name = "kaon-";
  // search in particle table]
  G4ParticleTable* pTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition* anInstance = pTable->FindParticle(name);
  if (anInstance ==0)
  {

   anInstance = new G4ParticleDefinition(
              name,    0.493677*GeV,   5.315e-14*MeV,    -1.*eplus,
                0,            -1,            0,
                1,            -1,            0,
            "meson",            0,           0,       -321,
             false,      12.3785*ns,         NULL,
             false,      "kaon");

 //create Decay Table
  G4DecayTable* table = new G4DecayTable();

 // create decay channels
  G4VDecayChannel** mode = new G4VDecayChannel*[6];
  // kaon- -> mu- + anti_nu_mu
  mode[0] = new G4PhaseSpaceDecayChannel("kaon-",0.6339,2,"mu-
","anti_nu_mu");
  // kaon- -> pi- + pi0
  mode[1] = new G4PhaseSpaceDecayChannel("kaon-",0.2103,2,"pi-","pi0");
  // kaon- -> pi- + pi+ + pi-
  mode[2] = new G4PhaseSpaceDecayChannel("kaon-",0.0559,3,"pi-","pi+","pi-");
  // kaon- -> pi- + pi0 + pi0
  mode[3] = new G4PhaseSpaceDecayChannel("kaon-",0.01757,3,"pi-","pi0","pi0");
  // のこりは省略

  for (G4int index=0; index <6; index++ ) table->Insert(mode[index]);
  delete [] mode;

   anInstance->SetDecayTable(table);
  }
  theInstance = reinterpret_cast<G4KaonMinus*>(anInstance);
  return theInstance;
}
```

G4ParticleDefinition

- Properties to characterize individual particles
  - Name
  - Mass
  - Width
  - Spin, parity
  - Decay mode (decay table)
- More than 100 types of particles are already provided by default.
- In normal application, users do not need to define particles for themselves.

自分で特別な粒子を定義することも可能
(例えば、K^- → $\pi^-\pi^0$にしか崩壊しないようなK-など)

Construct particles

- In ConstructParticle() method, you should call Definition functions for all the particles you want. This ensures that objects of these particles will be created.

```
void ExN01PhysicsList::ConstructParticle()
{
  G4Geantino::GeantinoDefinition();
  G4Proton::ProtonDefinition();                  Proton
  G4PionPlus::PionPlusDefinition();              π+
  G4PionMinus::PionMinusDefinition();            π-
  G4PionZero::PionZeroDefinition();              π0
}
```

- When you produce K- in PrimaryGeneratorAction and add decay process, you have to also register decay products of K-
- If you add electromagnetic process, e-,e+,gamma are produced. Therefore you must also register these particles

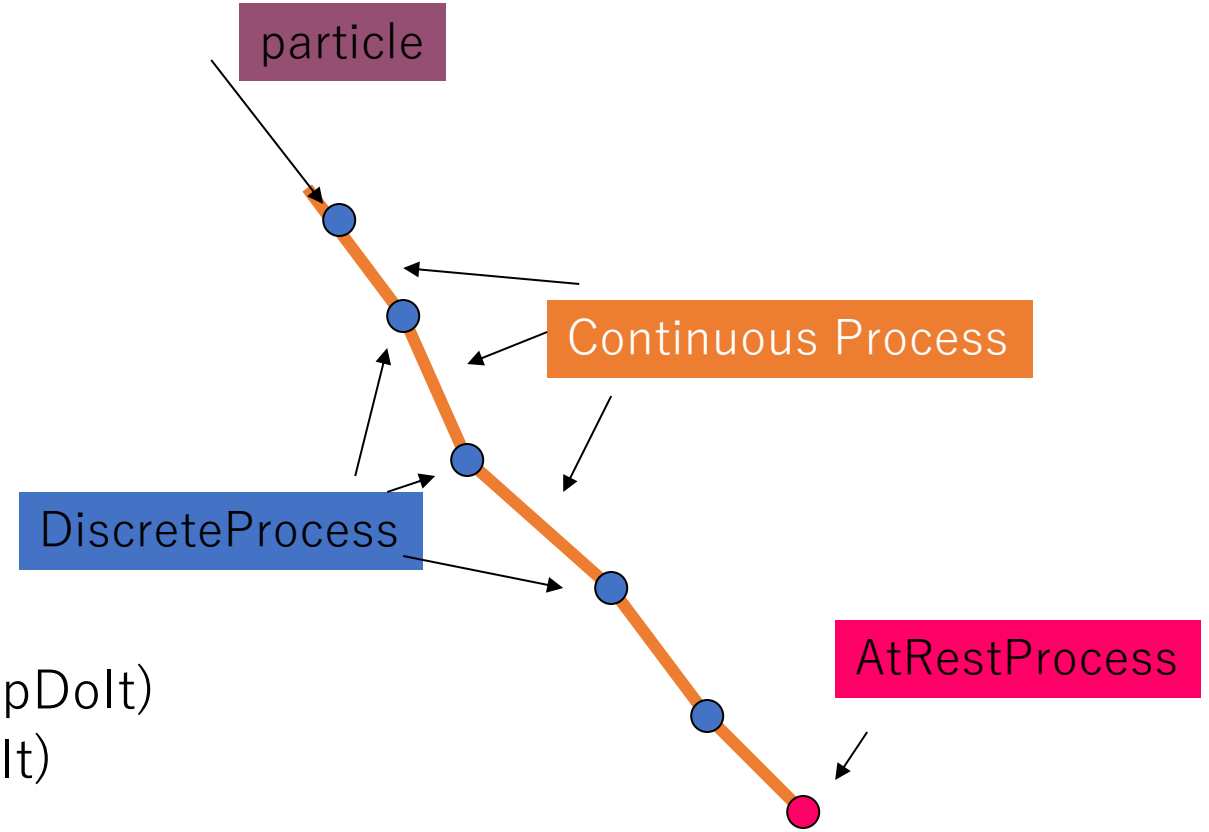
粒子の定義 (コンストラクタを使用する方法)

- 一つずつ粒子を定義するのは面倒なので、粒子の種類毎に粒子を登録するためのコンストラクタが用意してあるので、これを使用するのが一般的だと思う。

```
void HadronPhysics::ConstructParticle()
{
  // Purpose and Method: this method is invoked by G4 to build the particles
  //                     classes

  //  Construct all mesons
  G4MesonConstructor pMesonConstructor;
  pMesonConstructor.ConstructParticle();

  //  Construct all barions
  G4BaryonConstructor pBaryonConstructor;
  pBaryonConstructor.ConstructParticle();

  //  Construct  resonaces and quarks
  G4ShortLivedConstructor pShortLivedConstructor;
  pShortLivedConstructor.ConstructParticle();

}
```

Physical Processes

- Physics processes describe how particles interact with materials. Geant4 provides seven major categories of processes
  - Electromagnetic,
  - Hadronic,
  - Transportation,
  - Decay,
  - Optical,
  - Photolepton_hadron,
  - parameterisation.
- Three kinds of process
  - AtRestProcess (AtRestDoIt)
  - ContinuousProcess (AlongStepDoIt)
  - DiscreteProcess (PostStepDoIt)



```
void ExN00PhysicsList::ConstructProcess()
{
  AddTransportation();
  ConstructEM();
}
void ExN00PhysicsList::ConstructEM()
{
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle =
                        theParticleIterator->value();
    G4ProcessManager* pmanager =
                        particle->GetProcessManager();
    G4String particleName =
                        particle->GetParticleName();

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

Specifying Physics Processes

- In the ConstructProcess() method, physics processes should be created and registered with each particle's instance of G4ProcessManager
- The AddTransportation() method is provided to register the G4Transportation class with all particles classes.
- The G4ProcessManager class contains a list of processes that a particle can undertake.
  - The order of invocation
  - Kind of DoIt method
    - AtRestProcess
    - ContinuousProcess
    - DiscreteProcess

Managing Processes

ConstructEM()の中

```
  //electron
    G4VProcess* theeminusMultipleScattering = new G4MultipleScattering();
    G4VProcess* theeminusIonisation         = new G4eIonisation();
    G4VProcess* theeminusBremsstrahlung     = new G4eBremsstrahlung();
    //
    // add processes
    pmanager->AddProcess(theeminusMultipleScattering);
    pmanager->AddProcess(theeminusIonisation);
    pmanager->AddProcess(theeminusBremsstrahlung);
    //
    // set ordering for AlongStepDoIt
    pmanager->SetProcessOrdering(theeminusMultipleScattering,
idxAlongStep,1);
    pmanager->SetProcessOrdering(theeminusIonisation,       idxAlongStep,2);
    pmanager->SetProcessOrdering(theeminusBremsstrahlung,
idxAlongStep,3);

    //
    // set ordering for PostStepDoIt
    pmanager->SetProcessOrdering(theeminusMultipleScattering,
idxPostStep,1);
    pmanager->SetProcessOrdering(theeminusIonisation,       idxPostStep,2);
    pmanager->SetProcessOrdering(theeminusBremsstrahlung,
idxPostStep,3);
```

- In order to validate processes, processes should be registered with the particle's G4ProcessManager.
- Processes are added with AddProcess() method
- The order of invocation
- Kind of DoIt method
  - AtRestProcess
  - ContinuousProcess
  - DiscreteProcess

Decay process

```
#include "G4Decay.hh"

void ExN00PhysicsList::ConstructGeneral()
{
  // Add Decay Process
   G4Decay* theDecayProcess = new G4Decay();
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
    if (theDecayProcess->IsApplicable(*particle)) {
      pmanager ->AddProcess(theDecayProcess);
      // set ordering for PostStepDoIt and AtRestDoIt
      pmanager ->SetProcessOrdering(theDecayProcess, idxPostStep);
      pmanager ->SetProcessOrdering(theDecayProcess, idxAtRest);
    }
  }
}
```

Physics processの登録の方法3 (すべてGeantまかせ)

- Pre-packaged PhysicsListを用いる
  - シミュレーションで一般的に用いられるプロセスはEM, Decay, Hadronicプロセスになる。特にHadronicプロセスはどれを使ったら良いかを選ぶのにはかなりの経験と知識が必要になる。
  - そのため、Geantがお勧めするプロセスがパッケージ化されたPhysicsListが用意されている。

Geant4のexampleの多くのプログラムはこのpre-packaged Physics Listを用いている
B01, B02, B04, B05

- **Some Hadronic options:**
  - "QGS" Quark Gluon String model (> ~15 GeV)
  - "FTF" FRITIOF String model (> ~5 GeV)
  - "BIC" Binary Cascade model (< ~10 GeV)
  - "BERT" Bertini Cascade model (< ~10 GeV)
  - "P" `G4Precompound` model used for de-excitation
  - "HP" High Precision neutron model (< ~20 MeV)
- **Some EM options:**
  - No suffix: standard EM i.e. the default `G4EmStandardPhysics` constructor
  - "EMV" `G4EmStandardPhysics_option1` CTR: HEP, fast but less precise
  - "EMY" `G4EmStandardPhysics_option3` CTR: medical, space sci., precise
  - "EMZ" `G4EmStandardPhysics_option4` CTR: most precise EM physics
- **Name decoding:** String(s)_Cascade_Neutron_EM
- **The complete list of pre-packaged physics list with detailed description can be found in the documentation ("*Guide for Physics Lists*"):**

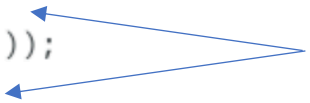
✦ **http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html**

- **FTFP_BERT:**
  - Recommended by Geant4 developers for HEP applications
  - Includes the standard EM physics i.e. `G4EmStandardPhysics` CTR
  - "FTF" FRITIOF string model (> 4 GeV)
  - "BERT" Bertini Cascade model (< 5 GeV)
  - "P" `G4Precompound` model used for de-excitation

- **QGSP_BIC_HP(_EMZ):**
  - Recommended for medical applications (experimental QGSP_BIC_AllHP)
  - "QGS" Quark Gluon String model (> 12 GeV)
  - "FTF" FRITIOF String model (9.5 - 25 GeV)
  - "P" `G4Precompound` model used for de-excitation
  - "BIC" Binary Cascade model (200 MeV - 9.9 GeV)
  - "HP" High Precision neutron model (< ~20 MeV)
  - "EMZ" `G4EmStandardPhysics_option4` CTR (or EMY that's a bit less precise)

RunManagerへの登録

```
int main(int argc,char** argv)
{
  // Detect interactive mode (if no arguments) and define UI session
  //
  G4UIExecutive* ui = 0;
  if ( argc == 1 ) {
    ui = new G4UIExecutive(argc, argv);
  }

  // Choose the Random engine
  G4Random::setTheEngine(new CLHEP::RanecuEngine);

  // Construct the default run manager
  //
#ifdef G4MULTITHREADED
  G4MTRunManager* runManager = new G4MTRunManager;
#else
  G4RunManager* runManager = new G4RunManager;
#endif

  // Set mandatory initialization classes
  //
  runManager->SetUserInitialization(new B2aDetectorConstruction());

  G4VModularPhysicsList* physicsList = new FTFP_BERT;
  physicsList->RegisterPhysics(new G4StepLimiterPhysics());
  runManager->SetUserInitialization(physicsList);
```



ここでPhysicsListとして呼び出して、
RunManagerに登録するだけ

Physics processの登録の方法2 (必要なプロセスをモジュール的に登録)

- どうしてモジュールとして分けるのか？
  - 先ほどの登録の仕方は、はっきり言って難しいし、ユーザーが定義するので正しく登録できているかどうかが判断しづらいところがある。
  - すべてのプロセスを正しく入れようとすると非常にプログラムが長くなり、またメンテナンス性も良くない。
- 必要な(または良く使用される)プロセスをモジュール化しておいた方が使いやすいだろうとのこと。
  - G4VModularPhysicsList を継承したクラスをPhysicsListとして用いる。
    - 自動的にtransportationなどは実行される
    - EM, Hadronic, Decayなどの区別しやすい分類でモジュール化されている

sks@dhcp2576: /home/sks/user/miwa/MonteCarlo/geant4/Si_test4
およびB3が参考プログラムです

Physics processの登録の方法2 (必要なプロセスをモジュール的に登録)

```
class B3PhysicsList: public G4VModularPhysicsList
{
public:
  B3PhysicsList();
  virtual ~B3PhysicsList();

  void ConstructSpecialParticle();
  virtual void SetCuts();
};
```

.hh ファイル
クラスの定義
G4VModularPhysicsListを継承する

```
B3PhysicsList::B3PhysicsList()
: G4VModularPhysicsList(){
  SetVerboseLevel(1);

  // Default physics
  RegisterPhysics(new G4DecayPhysics());

  // EM physics
  RegisterPhysics(new G4EmStandardPhysics());

  // Radioactive decay
  RegisterPhysics(new G4RadioactiveDecayPhysics());

  ConstructSpecialParticle();
}
```

.cc ファイル
実際にやっていること

RegisterPhysics()という関数で自分が登録したいプロセスを登録する

自分で粒子を定義したい時の例

- **Some "standard" EM physics constructors:**
  - `G4EmStandardPhysics` - default
  - `G4EmStandardPhysics_option1` - for HEP, fast but not precise settings
  - `G4EmStandardPhysics_option2` - for HEP, experimental
  - `G4EmStandardPhysics_option3` - for medical and space science applications
  - `G4EmStandardPhysics_option4` - most accurate EM models and settings
- **Some "low energy" EM physics constructors:**
  - `G4EmLivermorePhysics`
  - `G4EmLivermorePolarizedPhysics`
  - `G4EmPenelopePhysics`
  - `G4EmDNAPhysics`
- **The complete list can be found in your toolkit:**
  `geant4/source/physics_lists/constructors/` ==> all built in CTR
  `geant4/source/physics_lists/constructors/electromagnetic`
  `geant4/source/physics_lists/constructors/hadron_elastic`
  `geant4/source/physics_lists/constructors/hadron_inelastic`
- **More information at:**
  `geant4/source/physics_lists/constructors/xxx/README`
  http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html

自分で粒子を定義する時の例

```
void B3PhysicsList::ConstructSpecialParticle()
{
  G4DecayTable* decayTable;
  G4VDecayChannel* mode;
  G4ParticleDefinition* particle;


  // skaon- non-decay kaon-
  particle = new G4ParticleDefinition(
                                  "skaon-",    0.493677*GeV,   5.315e-14*MeV,    -1.*eplus,
                                  0,              -1,             0,
                                  1,              -1,             0,
                                  "meson",                0,              0,        -321,
                                  true,       0,          NULL);

  // usigma- life time is 0
  particle = new G4ParticleDefinition(
                                  "usigma-",    1.19744*GeV,   4.45e-12*MeV,    -1*eplus,
                                  1,              +1,             0,
                                  2,              -2,             0,
                                  "baryon",               0,              +1,        3112,
                                  false,       0.0*ns,          NULL);

  //create Decay Table
  decayTable = new G4DecayTable();
  // create decay channels
  // sigma- -> neutron + pi-
  mode = new G4PhaseSpaceDecayChannel("usigma-",1.00,2,"neutron","pi-");
  decayTable->Insert(mode);
  particle->SetDecayTable(decayTable);
```

ここがtrueだと安定な粒子となり、崩壊しない

安定なK-

粒子の定義

decay tableの作成

粒子に対してdecay tableの登録