Geant4

Como usar

http://cern.ch/geant4
http://geant4.cern.ch

O Geant4 foi escrito em C++
com tecnologia Orientada a Objeto

Se quiser aprender um pouco de C++

http://www.lishep.uerj.br/lishep2004/TutorialsSchedule.html

➔ Esse é o curso de C++ ministrado pelo Prof. Paul Kunz, que ministrou esse mesmo curso em vários laboratórios internacionais, tais como SLAC, CERN e FERMILAB.

➔ Tem as transparências e o vídeo das aulas.

“Toolkit + User application”

- Geant4 é uma caixa de ferramentas (**toolkit**, em inglês)
  - O usuário não pode simplesmente usá-lo como uma “caixa preta”
  - É necessário escrever uma aplicação, um programa, que use as ferramentas do Geant4.
- Consequência
  - Não existe um “Geant4 default”
  - O usuário deve fornecer a informação necessária para configurar a simulação desejada.
  - Ele/Ela deve escolher deliberadamente quais são as ferramentas específicas do Geant4 que quer usar.
- Guia: A colaboração Geant4 fornece vários **exemplos**
  - **Novice Examples**: visão geral de como usar o Geant4
  - **Advanced Examples**: “Geant4 tools” em aplicações da vida real

Conceitos Básicos

➔ **Necessário:**

Descrever o **setup experimental**

Definir quais são as **partículas primárias,** o *input* para a simulação

Decidir quais **partículas** e quais **modelos físicos,** entre aqueles fornecidos pelo Geant4, serão usados, bem como os cortes para produção de partículas secundárias, a energia mínima para cada partícula, etc…

➔ O usuário pode, se desejar,

Interagir com o Geant4 a fim de **controlar** sua simulação

visualizar os resultados

produzir **histograms, tuples** etc. para análise posterior

Interação com o Geant4

- O design do Geant4 foi feito para fornecer ao usuário **tools** para cada aplicação desejada
  - Para dizer qual é a configuração desejada na simulação
  - Para interagir com o Geant4

- "Geant4 tools" para interação com o usuários são **base classes** (classes básicas em C++)
  - O usuário cria **suas próprias classes concretas**, derivadas das classes básicas (base classes)
  - O Geant4 (o kernel) será capaz de incluir as classes do usuário de forma transparente através das classes de interface (polymorphism)

- **Abstract base classes** para interação com o usuário
  - Classes concretas fornecidas pelo usuário são **mandatórias**

- **Concrete base classes** (com "virtual dummy methods") para a interação com o usuário
  - Essas classes são **opcionais**

Classes do Usuário

Classes de Inicialização

Usadas na inicialização

- *G4VUserDetectorConstruction*
- *G4VUserPhysicsList*

Classes de Ação

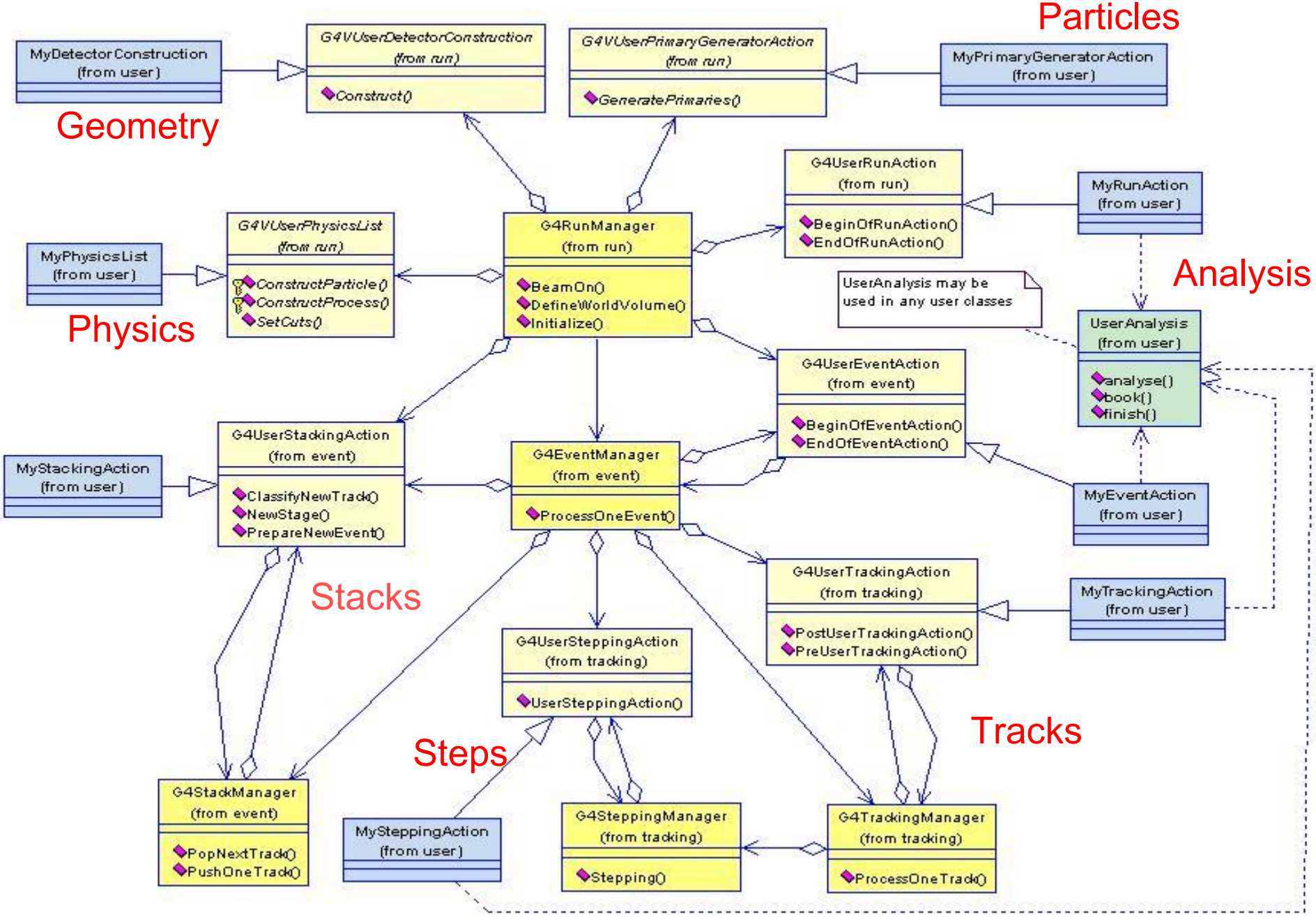
Usadas durante o loop de execução

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

Classes Mandatórias:

- *G4VUserDetectorConstruction*
  descrive o setup experimental
- *G4VUserPhysicsList*
  seleciona os processos físicos desejados
- *G4VUserPrimaryGeneratorAction*
  gera os eventos primários

Diagrama de Classes diagram uma simulação básica utilizando o Geant4



Desenvolvendo uma simulação com o Geant4

- Tentaremos apresentar os conceitos básicos para se fazer uma simulação básica com o Geant4, ou ao menos para se entender como são feitos os exemplos novice.
- O Geant4 recomenda:

*"Your application development will be greatly facilitated, if you adopt a sound **software process***

- *Vision of your simulation, clear user requirements*
- *Documented architecture and detailed software design*
- *Test process at various levels (unit, integration, system…)*
- *Well defined, documented procedures*
- *An iterative and incremental process to achieve your goals*
- *etc.*

We will not teach you software process

- *(but you could learn it in a specific course, if you are interested)"*

"The main function"

- O Geant4 provê várias classes, mas não provê o **main()**
  - O Geant4 é um toolkit, um conjunto de ferramentas!
  - O **main()** é parte a ser fornecida pelo usuário, já que define as necessidades específicas de sua simulação

- No **main()**, o usuário **deve**
  - Instanciar o **G4RunManager** (ou classes derivadas, criadas pelo usuário)
  - Notificar o G4RunManager sobre as classes mandatórias derivadas de
    - ***G4VUserDetectorConstruction***
    - ***G4VUserPhysicsList***
    - ***G4VUserPrimaryGeneratorAction***

- O usuário pode definir no main()
  - Classes opcionais de ação
  - a visualização: VisManager, e a interação com o usuário: (G)UI session

main()

```
{
  …
  // Construct the default run manager
  G4RunManager* runManager = new G4RunManager;

 // Set mandatory user initialization classes
  MyDetectorConstruction* detector = new MyDetectorConstruction;
  runManager->SetUserInitialization(detector);
  MyPhysicsList* physicsList = new MyPhysicsList;
  runManager->SetUserInitialization(myPhysicsList);

  // Set mandatory user action classes
  runManager->SetUserAction(new MyPrimaryGeneratorAction);

  // Set optional user action classes
  MyEventAction* eventAction = new MyEventAction();
   runManager->SetUserAction(eventAction);
  MyRunAction* runAction = new MyRunAction();
   runManager->SetUserAction(runAction);
  …
}
```

Descrevendo o setup experimental

- Derive sua classe concreta (concrete class) a partir das classes abstratas (abstract base class) do ***G4VUserDetectorConstruction***

- Implemente o método de construção (**Construct()** method)
  - defina todos os **materiais necessários**
  - defina as formas e sólidos requeridos para descrever a geometria de seu detector
  - **construa** e **posicione os volumes** do seu detector
  - defina os "**sensitive detectors",** aqueles onde as partículas sofrerão alguma interação, e identifique os volumes associados a eles
  - defina o **campo magnético** associado a determinadas regiões do detector
  - defina os atributos de **visualização** para todos os elementos dos detectores

Como definir os materiais

Tipos diferentes de materiais podem ser definidos:

- Isotopes (isótopos)
- Elements (elementos puros)
- Molecules (moléculas)
- Compounds and mixtures (compostos e misturas)

```
PVPhysicalVolume*  MyDetectorConstruction::Construct()
 {
    …
    a = 207.19*g/mole;
    density = 11.35*g/cm3;
    G4Material* lead = new G4Material(name="Pb", z=82., a, density);

    density = 5.458*mg/cm3;
    pressure = 1*atmosphere;
    temperature = 293.15*kelvin;
    G4Material* xenon = new G4Material(name="XenonGas", z=54.,
                                       a=131.29*g/mole, density,
                                       kStateGas, temperature, pressure);
    ...
 }
```

Lead

Xenon gas

Como definir um material composto

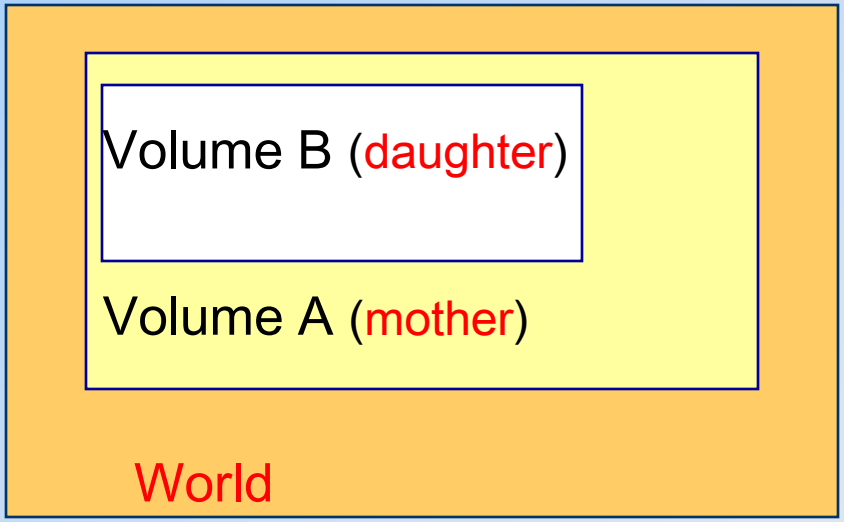
por exemplo, a **cintilador** composto por Hidrogênio e Carbono:

```
G4double a = 1.01*g/mole;
G4Element* H  = new G4Element(name="Hydrogen", symbol="H", z=1., a);

a = 12.01*g/mole;
G4Element* C  = new G4Element(name="Carbon", symbol="C", z=6., a);

G4double density = 1.032*g/cm3;
G4Material* scintillator = new G4Material(name = "Scintillator", density,
numberOfComponents = 2);

scintillator -> AddElement(C, numberOfAtoms = 9);
scintillator -> AddElement(H, numberOfAtoms = 10);
```

Definindo a geometria do detector

- Três camadas conceituais
  - G4VSolid — forma e tamanho
  - G4LogicalVolume — material, sensibilidade, campo magnético, etc.
  - G4VPhysicalVolume — posição, rotação

- Um volume físico único, o MUNDO (the world volume), que representa toda a área experimental, deve ser definido de forma a conter todos os outros componentes da simulação



→ Volume A é mãe (mother) do Volume B

O volume mãe (mother) **DEVE** conter totalmente o volume filha (daughter)

Como construir o mundo - World

```
solidWorld = new G4Box(“World", halfWorldLength, halfWorldLength, halfWorldLength);
logicWorld = new G4LogicalVolume(solidWorld, air, "World", 0, 0, 0);
physicalWorld = new G4PVPlacement(0,                    //no rotation
                                  G4ThreeVector(),      // at (0,0,0)
                                  logicWorld,           // its logical volume
                                  "World",              // its name
                                  0,                    // its mother  volume
                                  false,                // no boolean operations
                                  0);                   // no magnetic field
```

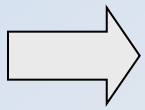
Como construir um volume dentro do World

```
solidTarget = new G4Box(“Target", targetSize, targetSize, targetSize);
logicTarget = new G4LogicalVolume(solidTarget, targetMaterial, "Target",0,0,0);
physicalTarget = new G4PVPlacement(0,                   // no rotation
                                   positionTarget,      // at (x,y,z)
                                   logicTarget,         // its logical volume
                                   "Target",            // its name
                                   logicWorld,          // its mother  volume
                                   false,               // no boolean operations
                                   0);                  // no particular field
```

Selecionando os processos físicos

- O Geant4 não tem nenhum “default” para partículas ou processos físicos
- O usuário **PRECISA** definir todos eles.
- Em linguagem C++, derive sua concrete class da ***G4VUserPhysicsList*** abstract base class
- traduzindo…
  - defina todas as partículas a serem usadas na simulação
  - defina todos os processos necessários, declarando qual deles pertence a qual partícula
  - defina os intervalos de produção
- Falando C++: *Pure virtual methods of G4VUserPhysicsList*

ConstructParticles()
ConstructProcesses()
SetCuts()



Devem ser implementados pelo usuários em suas classes concretas

PhysicsList: partículas e cortes

```
MyPhysicsList :: MyPhysicsList():  G4VUserPhysicsList()
{
    defaultCutValue = 1.0*cm;
}
```

Define os "**production thresholds**" (o mesmo para todas as partículas)

```
void MyPhysicsList :: ConstructParticles()
{
    G4Electron::ElectronDefinition();
    G4Positron::PositronDefinition();
    G4Gamma::GammaDefinition();
}
```

Define **as partículas** a serem usadas na simulação

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

Estabelece (set) o "**production threshold**"

PhysicsList: cortes → +informações

```
MyPhysicsList :: MyPhysicsList():  G4VUserPhysicsList()
{
    // Define production thresholds
    cutForGamma  = 1.0*cm;
    cutForElectron = 1.*mm;
    cutForPositron = 0.1*mm;
} ;
```

```
void MyPhysicsList :: SetCuts()
{
    // Define production thresholds
    SetCutValue(cutForGamma, "gamma");
    SetCutValue(cutForElectron, "e-");
    SetCutValue(cutForPositron, "e+");
}
```

O usuário pode definir cortes diferentes para partículas diferentes ou regiões diferentes

Physics List: processos

Seleciona os processos físicos a serem ativados para cada tipo de partícula

```
void MyPhysicsList :: ConstructParticles()
 {
    if (particleName == "gamma")
      {
            pManager->AddDiscreteProcess(new G4PhotoElectricEffect());
            pManager->AddDiscreteProcess(new G4ComptonScattering());
            pManager->AddDiscreteProcess(new G4GammaConversion());
      }
   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);
      }
   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);
      }
}
```

Este exemplo seleciona os processos do Geant4 *Standard* electromagnetic

Physics List: Builders

```
if (name == "standard_opt3") {
    emName = name;
    delete emPhysicsList;
    emPhysicsList = new
G4EmStandardPhysics_option3();
  } else if (name == "LowE_Livermore") {
    emName = name;
    delete emPhysicsList;
    emPhysicsList = new G4EmLivermorePhysics();
  } else if (name == "LowE_Penelope") {
    emName = name;
    delete emPhysicsList;
    emPhysicsList = new G4EmPenelopePhysics();
```

exemplo de uso em Física Médica

***Builders* (construtores)**
$G4INSTALL/source/physics_lists/builders

Physics List: Reference Lists

Especificamente para o LHC existem várias opções de Physics Lists, que podem ser classificadas em três famílias:

- LHEP or parameterised modelling of hadronic interactions
- QGS, or list based on a model that use the Quark Gluon String model for high energy hadronic interactions of protons, neutrons, pions and kaons
- FTF, based on the FTF (FRITIOF like string model) for protons, neutrons, pions and kaons

http://geant4.cern.ch/support/proc_mod_catalog/physics_lists/referencePL.shtml
http://geant4.slac.stanford.edu/SLACTutorial09/ChoosingPhysicsList.ppt

Recomendação de uso ➔ insira no main

```
include <QGSP_BERT.hh>
int main(int,char**)
{
//....
! runManager->SetUserInitialization( new QGSP_BERT );
}
OR
#include <G4PhysListFactory.hh>
int main(int,char**)
{
//....
 G4PhysListFactory factory;
 G4VModularPhysicsList* physList =
factory.ReferencePhysList();
 runManager->SetUserInitialization( physList );
}
```

Reference Physics Lists:
$G4INSTALL/source/physics_lists/lists

Eventos Primários

➔ Derive suas concrete class a partir das *G4VUserPrimaryGeneratorAction* abstract base class

➔ Defina as partículas primárias, fornecendo:

- Particle type (tipo de partícula)
- Initial position (posição inicial)
- Initial direction (direção inicial)
- Initial energy (energia inicial)

➔ Implemente a virtual member function **GeneratePrimaries()**

Gere as partículas primárias

```
MyPrimaryGeneratorAction:: My PrimaryGeneratorAction()
{
  G4int numberOfParticles = 1;
  particleGun = new G4ParticleGun (numberOfParticles);
  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition* particle = particleTable->FindParticle("e-");
  particleGun->SetParticleDefinition(particle);
  particleGun->SetParticlePosition(G4ThreeVector(x,y,z));
  particleGun->SetParticleMomentumDirection(G4ThreeVector(x,y,z));
  particleGun->SetParticleEnergy(energy);
}

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

Outra sugestão:

G4VUserPrimaryGeneratorAction

```
void MyPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
    {
      G4ParticleDefinition* particle;
      gun->SetParticleDefinition(particle);
      G4double pp = momentum + (G4UniformRand()-0.5)*sigmaP;
      G4double mass = particle->GetPDGMass();
      G4double kineticE = sqrt(pp*pp + mass*mass) - mass;
      gun->SetParticleEnergy(kineticE);
      G4double angle = (G4UniformRand() - 0.5) * sigmaAngle;
      gun->SetParticleMomentumDirection
               (G4ThreeVector(sin(angle), 0., cos(angle));

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

www.ge.infn.it/geant4/training/dresden_2008/primaryParticle.pdf

Classes Opcionais para o Usuário
(Optional User Action classes)

- Cinco concrete base classes cujas virtual member functions podem ser sobre-escritas pelo usuário a fim de controlar os vários estágios da simulação
  - G4User**Run**Action
  - G4User**Event**Action
  - G4User**Tracking**Action
  - G4User**Stacking**Action
  - G4User**Stepping**Action

Somente para usuários experientes

- Cada member function das base classes tem uma implementação vazia (dummy)
  - "Empty implementation": não faz nada
- O usuário pode implementar as member functions que desejar em suas classes derivadas
- Os objetos das user action classes precisam ser registrados no **G4RunManager**

Optional User Action classes

G4UserRunAction

BeginOfRunAction(const G4Run*)

- por exemplo: book histograms

EndOfRunAction(const G4Run*)

- por exemplo: store histograms

G4UserEventAction

BeginOfEventAction(const G4Event*)

- por exemplo: faz seleção de eventos

EndOfEventAction(const G4Event*)

- por exemplo: analisa o evento

G4UserTrackingAction

PreUserTrackingAction(const G4Track*)

- por exemplo: decide se uma dada trajetória deve ser armazenada ou não

PostUserTrackingAction(const G4Track*)

Optional User Action classes

G4UserSteppingAction

UserSteppingAction(const G4Step*)

- por exemplo: “mata”, suspende, “postpone” o traço
- por exemplo: desenha o passo

G4UserStackingAction

PrepareNewEvent()

- por exemplo: “reset” o controle de prioridade

ClassifyNewTrack(const G4Track*)

- Chamado toda vez que um novo traço é tratado (a new track is pushed)
- por exemplo: classifica o novo traço (controle de prioridade)
  - Urgent, Waiting, PostponeToNextEvent, Kill

NewStage()

- Chamado quando a pilha “Urgente” está vazia
- por exemplo: muda o critério de classificação
- por exemplo: filtra o evento, ou aborta o evento

Visualização e Interface com o Usuário - (G)UI and visualisation

- No **main()**, levando em consideração o ambiente computacional usado, instantiate a **G4UIsession** concrete class dada pelo Geant4 e invoque o método **sessionStart()**

- O Geant4 permite usar:
  - G4UIterminal
  - csh or tcsh like character terminal
  - G4GAG
  - tcl/tk or Java PVM based GUI
  - G4Wo
  - Opacs
  - G4UIBatch
  - batch job with macro file
  - …

- No **main()**, levando em consideração o ambiente computacional usado, instantiate a **G4VisExecutive** e invoque o método **initialize()**

- O Geant4 tem interfaces para vários pacotes gráficos:
  - DAWN *(Fukui renderer)*
  - WIRED
  - RayTracer *(ray tracing by Geant4 tracking)*
  - OPACS
  - OpenGL
  - OpenInventor
  - VRML
  - …

Receita para “novice users”, segundo o Geant4

- Faça o diagrama de classes tal qual um *Geant4 Application Example* genérico
- Crie as classes mandatórias
  - My**DetectorConstruction**
  - My**PhysicsList**
  - My**PrimaryGeneratorAction**
- Optional: crie as “derived user action classes”
  - MyUser**Run**Action
  - MyUser**Event**Action
  - MyUser**Tracking**Action
  - MyUser**Stacking**Action
  - MyUser**Stepping**Action
- Crie o main()
  - Instancie **G4RunManager** ou seu derivado My**RunManager**
  - Notifique o **RunManager** de suas classes mandatórias e opcionais
  - Optional: inicializa sua Interface e Visualização preferidos
- É só!

Usuários experientes podem fazer muito mais, porém conceitualmente o processo ainda é o mesmo...

Minha Receita para “novice users”

- Escolha um dos exemplos Novice, de preferência o N02 ou N03, que já possuem os comandos para a interface gráfica.
- Siga o mesmo o diagrama de classes
- Edite as classes, defina o MUNDO e a geometria do detector
  defina as partículas a serem usadas
  defina os processos físicos envolvidos
- Edite o main(), se necessário
- Siga as mesmas instruções para compilar e rodar o programa.
- Lembre-se de copiar o arquivo vis.mac para a área de execução.
- É só!

Sempre é aconselhável trabalhar na conta do próprio usuário, então para testar a instalação:

➔ **entrar na conta do usuário**

➔ **ir para a área onde quer trabalhar com o Geant4**
(por exemplo /home/begalli/Geant4/sprace**)**

➔**fazer**

source /usr/local/src/geant4.9.1.p03/env.sh
cd ../examples/novice/N02
gmake all
echo $G4WORKDIR
cd $G4WORKDIR/bin/Linux-g++/
cp $G4INSTALL/examples/novice/N02/vis.mac .

➔ *no **vis.mac** mudar a linha do OGLIX para OIX a fim de usar o OpenInventor*
quando a janela abrir, entrar no ETC
pedir a visualizacao de mothers and daughters
*rodar novamente **/run/beamOn 10***

É preciso um visualizador…

O Geant4 possibilita o uso de vários visualizadores. Para saber mais

http://www.geant4.org/geant4/collaboration/working_groups/visualization/

Um exemplo:

- para instalar o **OpenInventor**:
  http://oss.sgi.com/projects/inventor/
  - crie uma area teste e copie ali o arquivo rpm mais recente
  - use o comando
    rpm2cpio Inventor-2.1.5-12.el4.kb.i386.rpm | cpio -idmv

- na sub-área do Geant4:

  ➔ precisa editar os arquivos no ***config/sys*** e tirar o "***pedantic***" da opção de compilação

Meus agradecimentos a:

Dr. Maurício Moralles (IPEN)
Dr. Maria Grazia Pia (INFN-Genova)
Dr. Denison de Souza Santos (IRD)
Dr. Pedro Pacheco Queiroz Filho (IRD)

À comissão organizadora

… e aos participantes,
que tiveram a paciência de me ouvir.

Título: Da importância do agradecimento
Autor: não citado
http://adeusavida.blogspot.com/2009/10/da-importancia-do-agradecimento.html