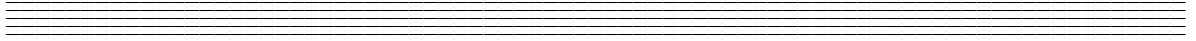


# Taller GEANT4



Escuela de Detectores Uniandes

Universidad de los Andes

(Material distribuible para los participantes de la escuela de detectores.)

# Script Taller GEANT4

## Agenda

---

### Agenda

El taller cubre los siguientes temas

1. Preliminar: Practica con GIT
  2. Preliminar: Practica con ROOT (énfasis en C++)
  3. GEANT4: Generalidades, aspectos generales
  4. GEANT4: Ejemplo 1
  5. GEANT4: Ejemplo 2
  6. GEANT4: Ejemplo 3
- 

### Prerrequisitos

Conocimientos básicos de C++

**GitHub:** <https://github.com/aosorioUniandes>

**Username:** aosorioUniandes **Password:** edu2016

**Repositorio:** <https://github.com/aosorioUniandes/Geant4Tutorials>

USB con el software listo para trabajar la sesión del laboratorio

- Utilizar la cuenta de usuario “Detectores” y Password “edu2016”.
-

## Requerimientos 1

*El tiempo es breve y hay varios mensajes que queremos que queden en los estudiantes. 1. Herramientas de desarrollo colaborativo 2. Para trabajar con Geant4 hay que tener un buen nivel de C++.*

¿Qué es GIT? <https://git-scm.com/> (abrir la página en el explorador web)

- Sistema de Control de Versiones (VCS)
- Almacena historia y lleva registro de cada versión de un archivo o un conjunto de archivos.
- Permite revertir cambios de un archivo o del proyecto completo.
- Compara cambios a través del tiempo.
- Permite crear líneas de desarrollo independientes a través de ramas (*branches*).

¿Qué es GitHub? <https://github.com> (abrir la página en el explorador web)

- Repositorio remoto de Git.
- Ofrece una interfaz gráfica web, adicional a la herramienta de línea de comandos.
- Permite registro de defectos, administración de tareas y Wikis para cada proyecto.
- Genera gráficas estadísticas de cada proyecto, incluyendo contribuidores, confirmaciones (*commits*), frecuencia de codificación, etc.

Practica:

1. Clonar un repositorio
2. Hacer una modificación del código
3. Subir los cambios

Clonar repositorio para esta práctica (ingresar a la cuenta “Detectores” de la imagen de la distribución de Linux para la Escuela ):

```
$ git clone https://github.com/aosorioUniandes/Geant4Tutorials
```

Pasarse a la “rama” “practica”

```
$ cd Geant4Tutorials
$ git checkout practica
$ cd GitPractice/src
```

Editar HolaGit.cc ... usar para ello *Visual Studio Code*:

```
$ code HolaGit.cc
```

Agregar su sombra (siguiendo el comentario y la sintaxis de C++).  
Probar que corre el código y subir el cambio al repositorio:

```
$ g++ HolaGit.cc  
$ ./a.out  
$ git add HolaGit.cc  
$ git commit -m "[AO] presentación de AOsorio"  
$ git push origin practica
```

Esperar que los demás compañeros hayan subido su presentación. Descargar la versión más reciente con “*pull*”. Atención: ¡aquí puede haber conflictos, lo cual hace la práctica más interesante!

```
$ git pull
```

Cerrar ejercicio cuando todas las inquietudes se hayan aclarado.

Referencias:

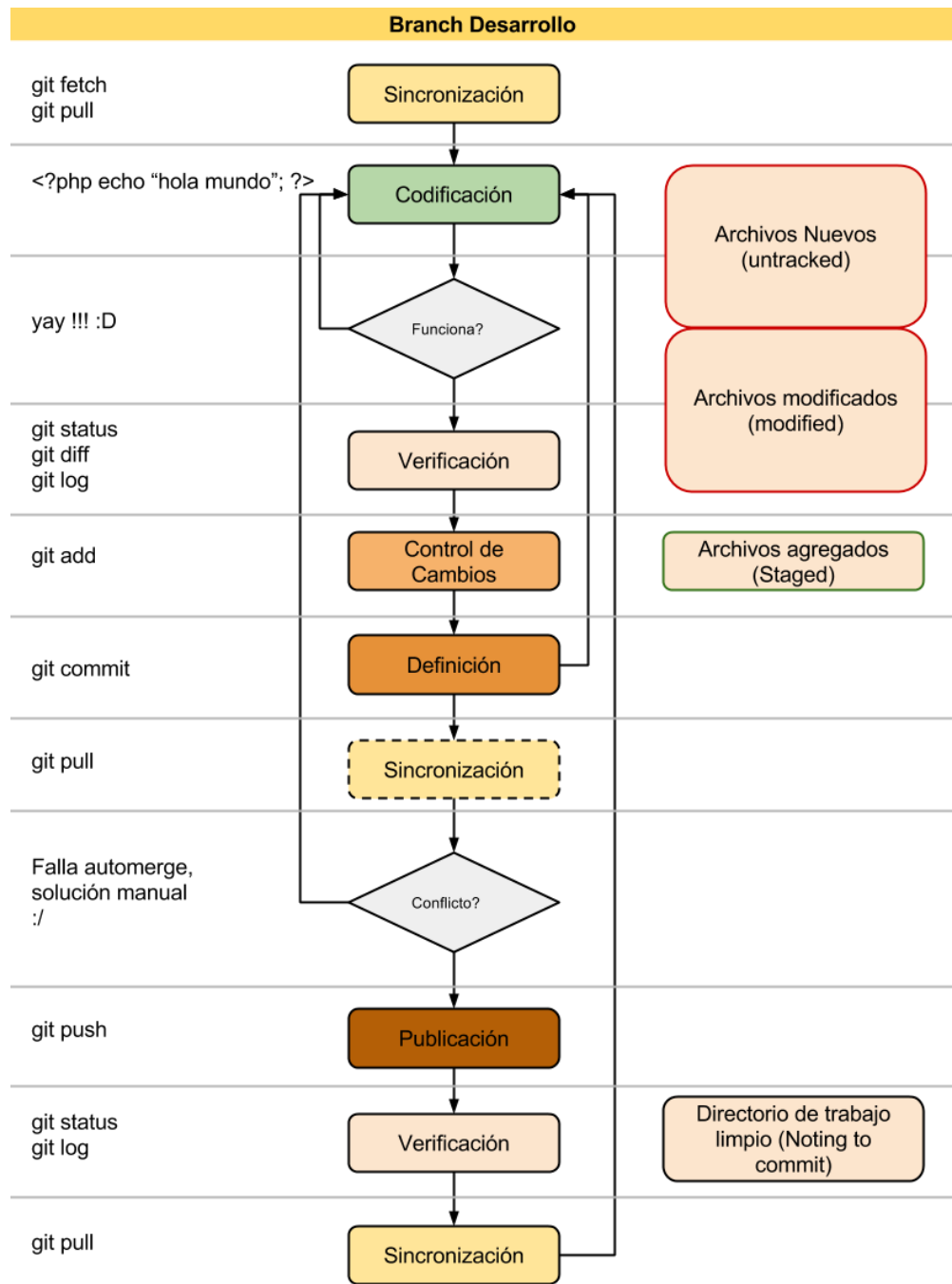
Manos la Rama (documento temporal):

<http://codeaholics.dynns.com/do/view/Main/ManosAlaRama>

Tutoriales en línea:

<http://nvie.com/posts/a-successful-git-branching-model/>

<http://rogerdudler.github.io/git-guide/>



## Requerimientos 2      ¿Qué es ROOT? <http://root.cern.ch>

ROOT es un framework científico usado para el análisis de datos. El framework provee las herramientas necesarias para:

- Procesamiento de grandes volúmenes de datos (Big data)
- Hacer análisis de datos
- Visualización
- Almacenamiento

Fue desarrollado en C++, aunque hoy en día se integra con otros lenguajes como Python y R (estadísticas). Nota: Mencionar artículo reciente en blog de EDX (<http://blog.edx.org/highest-paying-programming-languages-2016>)

Practica:

1. Clonar un repositorio (*branch*: master) o hacer `$ git checkout master` (para pasar a la rama master)
2. Correr el ejemplo de ROOT (fillrandom.C) y seguir los comentarios

```
$ cd RootPractice/src  
$ root fillrandom.C
```

3. Discusión acerca de ROOT y sus capacidades

## Introducción

En un detector de partículas, las partículas interactúan con el material del detector, depositando energía. Geant4 es **una caja de herramientas** (*toolbox*) para simular el paso de partículas a través de la materia. Según la documentación de sus desarrolladores, el área de aplicación incluye la física (de altas energías, nuclear y de aceleradores), al igual que el área de la medicina y la industria aeroespacial.

En este taller, queremos darles una introducción a lo que es trabajar con Geant4. Sin embargo, el tiempo es muy breve para tocar todos los temas que de por sí darían para una escuela sola (alcance de este Laboratorio).

Página web: <http://geant4.web.cern.ch/geant4/>

Aspectos generales:

Conceptos de OO:

- Búsqueda dinámica
- Abstracción
- Subtipos
- Herencia

Geant4 está escrito en C++, el cual es en lenguaje que adopta la filosofía o metodología de la Programación Orientada a Objetos (preguntar por las 4 propiedades de OO). Las características generales de un sistema de simulación de detectores son:

- El usuario elige el montaje del experimento
- El software se encarga de transportar las partículas que usted dispara al sistema, simulando su interacción con la materia usando el método de Monte Carlo.
- El método de Monte Carlo, es un método que resuelve un problema matemático usando muestras de números aleatorios.

Geant4 transporta partículas paso a paso, teniendo en cuenta interacciones con materiales y campos electromagnéticos externos, hasta que las partículas:

- Pierden toda su energía cinética
- Desaparecen debido a alguna interacción
- Llegan a la frontera de un volumen simulado

¿Que debe proveer el usuario?

- Descripción de la Geometría y material del montaje experimental
- Información cinemática de las partículas primarias que va a ser seguidas
- Elección de los procesos de física que pueden ocurrir
- Además: campos (E/M, acciones especiales, entre otros)

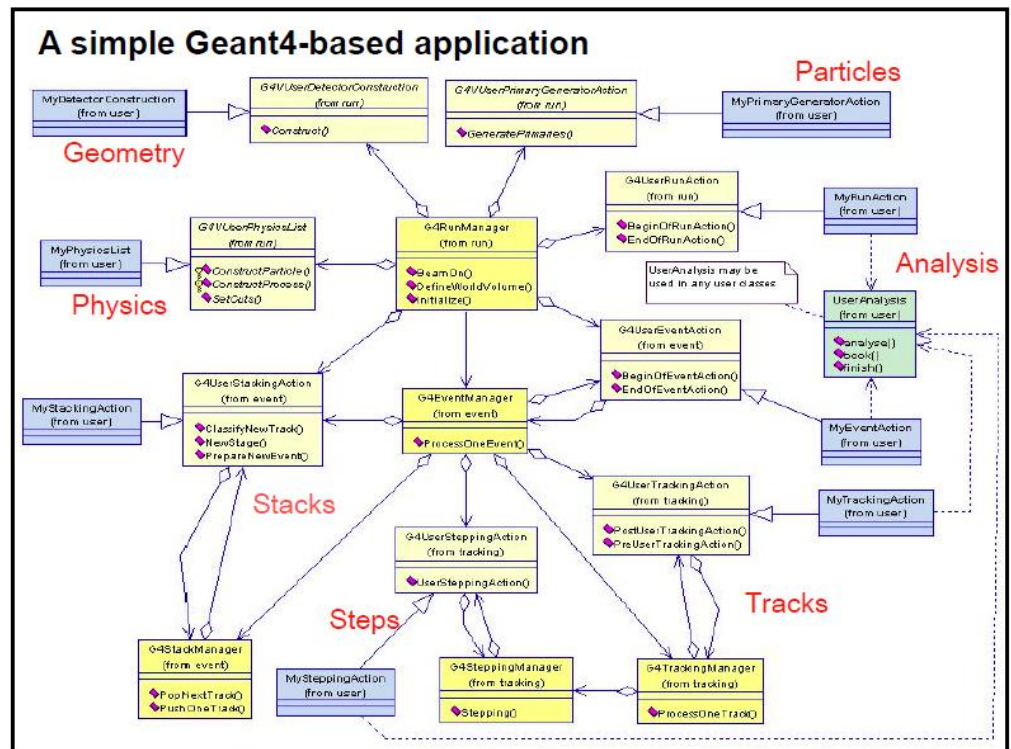


Figura 1 Modelo de una aplicación típica de Geant4 (Ref: [https://www.ge.infn.it/geant4/training/nss2013/geant4course\\_handouts.pdf](https://www.ge.infn.it/geant4/training/nss2013/geant4course_handouts.pdf) )

## Ejemplo 1

1. En el repositorio que ya se ha descargado, bajar hasta la carpeta Geant4
2. Allí se encuentra el ejemplo B1
3. ¿Qué podemos ver en esta estructura en el ejemplo B1? ¿Cómo es el esqueleto de una aplicación de Geant4 (ver GitHub)? Explorar los distintos elementos que constituyen un programa de Geant4
4. Compilar y correr (en el repo hay un archivo HowTo.txt que contiene estos pasos):

```
$ ls
$ B1
$ mkdir B1-build
$ cd B1-build/
$ cmake -DGeant4_DIR=/opt/hepsw/geant4/install/share/Geant4-10.2.2 \
/home/detectores/Geant4Tutorials/Geant4/B1
...
$ make
...
$ ./exampleB1
```

Observación: aquí se visualiza el ejemplo B1. Descripción de las distintas secciones de la ventana.

## Ejemplo 2



Objetivos:

- Definición de materiales
- Definición de una geometría muy sencilla

Pasemos al segundo ejemplo (HandsOn2<sup>1</sup>). Abramos desde *Visual Studio Code* toda la carpeta:

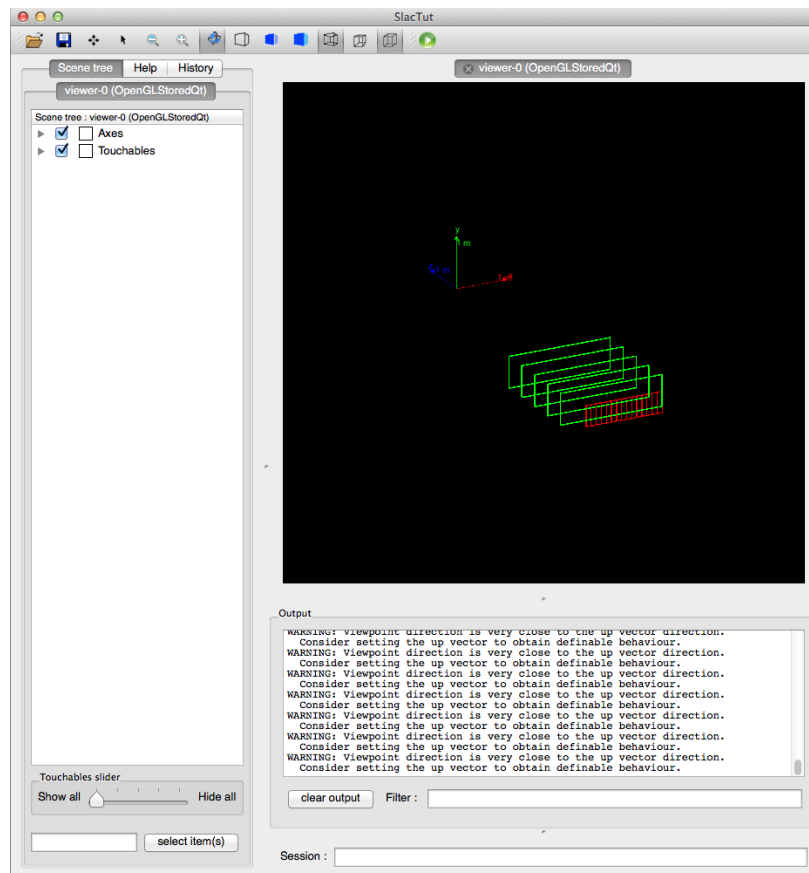
Vamos a proceder primero a compilarlo, recuerde entonces que toca crear una carpeta adicional:

```
$ mkdir HandsOn2-build
$ cd HandsOn2-build
```

Y ejecutar **cmake** para que genere los scripts de compilación:

```
$ cmake <Opcional -DGeant4_DIR> ../HandsOn2-build
...
$ make
$ ./SlacTut
```

Familiarizarse con la GUI.



<sup>1</sup> Tomado del tutorial de Geant4 ocurrido en SLAC en 2014

Geant4 tiene básicamente 4 modos de ejecución:

1. Totalmente en batch
2. En batch simple
3. Interactivamente ejecutando macros
4. Interactivamente desde una GUI

Ejercicio: encontrar en que parte del código, los macros (\*.mac) son ejecutados.

(Localizar la función o el método main() en tutorial.cc). Pregunta: ¿Cuál es el rol de gui.mac? ¿Cómo cambiar el número de eventos?

Ejercicio, crear los siguientes materiales:

- For Cs:  $A=55$   $Z_{\text{eff}}=132.9 \text{ g/mol}$
- For I :  $A=53$  ,  $Z_{\text{eff}}=126.9 \text{ g/mol}$
- Density of crystal of CsI is:  $\rho=4.51 \text{ g/cm}^3$

Para ello, modificar el siguiente archivo DetectorConstruction.cc (buscar los comentarios que tienen Exercise):

```
G4Element* el_i = new G4Element("Iodine","I", 53,126.9*g/mole);
G4Element* el_cs = new G4Element("Cesium","Cs",55,132.9*g/mole);
G4Material* mat_csi = new G4Material("CsI",4.51*g/cm3,2);
mat_csi->AddElement(el_i,1); // 1 = fraccion
mat_csi->AddElement(el_cs,1); // 1 = fraccion
```

Notas: G4Material revisar documentación *doxygen*.

## Materials

- Different kinds of materials can be defined
  - isotopes ↔ G4Isotope
  - elements ↔ G4Element
  - molecules ↔ G4Material
  - compounds and mixtures ↔ G4Material
- Associated attributes:
  - temperature
  - pressure
  - state
  - density

```
G4double density = 1.390*g/cm3;
G4double a = 39.95*g/mole;
G4Material* lAr =
new G4Material("liquidArgon",z=18.,a,density);
```

Geant 4

26



Compilar y revisar la salida de la nuestra modificación en tutorial.out

Ejercicio: crear una geometría muy sencilla. Para ello, queremos construir en Geant4, un volumen del material que adicionamos el punto anterior.

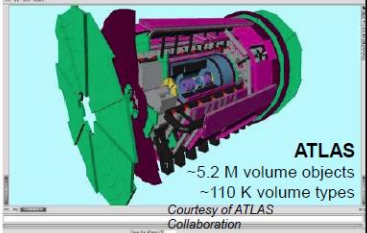
Nuevamente nos vamos a concentrar en el código fuente que se encuentra en DetectorConstruction.cc. Adicionamos las siguientes líneas:

```
G4Material* material = G4Material::GetMaterial("CsI");
G4VSolid* hadCalorimeterSolid = new G4Box("HadCalorimeterBox",1.5*m,30.*cm,50.*cm);
G4LogicalVolume* hadCalorimeterLogical = new
G4LogicalVolume(hadCalorimeterSolid,material,"HadCalorimeterLogical");
new
G4PVPlacement(0,G4ThreeVector(0.,0.,3.*m),hadCalorimeterLogical,"HadCalorimeterPhysical",secondArmLogical,false,0,checkOverlaps);
```

## Geometry

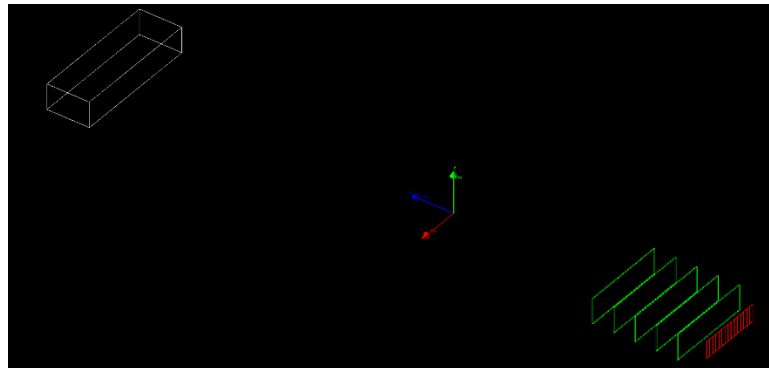
- Role
  - detailed detector description
  - efficient navigation
- Three conceptual layers
  - **Solid**: shape, size
  - **LogicalVolume**: material, sensitivity, daughter volumes, etc.
  - **PhysicalVolume**: position, rotation
- One can do fancy things with geometry...
  - Boolean operations 
  - Transparent solids 

Geant 4



ATLAS  
~5.2 M volume objects  
~110 K volume types  
Courtesy of ATLAS Collaboration

Compilar nuevamente el código. Ejecutar el ejemplo con el visualizador y observar nuestra creación.



Practica libre: modificar el material del detector y cambiar el *beam* por otro tipo de partículas.

## Ejemplo 3

El último ejercicio es un poco más complejo, pero nos sirve para ilustrar como cerrar el ciclo en el desarrollo de nuestra simulación. Una simulación tiene por objetivo obtener unos datos. En este caso vamos a indagar por los depósitos de energía, numero de pasos de gammas y usar algunos filtros.

Podemos usar la GUI para crear una grilla localizada arriba de la caja que creamos en el punto anterior. Esta grilla debe las mismas dimensiones de la caja anterior. Score the following quantities:

- Energy deposits
- Number of steps for gammas
- Number of steps for e-
- Number of steps for e+
- Dump the scored quantities to files and verify the content.

Solución:

El primer paso es instanciar el “scoring manager” en el main() (abrir tutorial.cc).

```
// Activate UI-command base scorer
G4ScoringManager * scManager = G4ScoringManager::GetScoringManager();
scManager->SetVerboseLevel(1);
```

El macro “scoring.mac”, muestra los pasos que debemos seguir para este ejercicio.

```
$ ./SlacTut scorer.mac
```

Nota: Hay que reducir el número de eventos simulados, sino la simulación se tomará mucho tiempo en ejecutar.

scoring.mac:

```
/run/initialize
#####
#
# define scoring mesh
#
/score/create/boxMesh boxMesh_1
#
#Create a mesh large as the box
/score/mesh/boxSize 150. 30. 50. cm
#Position it over the box
/score/mesh/translate/xyz 0 0 8 m
#mesh voxel size of 5cm
/score/mesh/nBin 30 6 10
#
/score/quantity/energyDeposit eDep
/score/quantity/nOfStep nOfStepGamma
/score/filter/particle gammaFilter gamma
/score/quantity/nOfStep nOfStepEMinus
/score/filter/particle eMinusFilter e-
/score/quantity/nOfStep nOfStepEPlus
/score/filter/particle ePlusFilter e+
#
/score/close
#
/run/verbose 1
/gun/particle e-
/run/beamOn 1
```

```
#####  
#  
# Dump scores to a file  
#  
/score/dumpQuantityToFile boxMesh_1 nOfStepGamma nOfStepGamma.txt
```

## Notas finales

Geant4 ofrece muchas otras capacidades, que por tiempo no alcanzamos a cubrir:

- Partículas: se puede usar todo lo que existe definido en el PDG (Particle Data Group)
- Hit & Digitalizacion
- Simulación rápida
- Persistencia
- Paralelización
- Y muchas cosas más, es una caja de herramientas muy completa y utilizada en varios contextos
- Tiene implementado mucha fisica

- Multiple scattering
- Bremsstrahlung
- Ionisation
- Annihilation
- Photoelectric effect
- Compton scattering
- Rayleigh scattering
- $\gamma$  conversion
- $e^+e^-$  pair production
- Synchrotron radiation
- Transition radiation
- Cherenkov
- Refraction
- Reflection
- Absorption
- Scintillation
- Fluorescence
- Auger emission

