

The Geant4 Simulation Toolkit – Short Course, IEEE NSS-MIC 2013

■ Presentation of the course

■ Overview of Geant4

- Geant4 vision: scope, fundamental concepts
- Geant4 architecture
- Overview of Geant4 simulation capabilities
- Overview of experimental applications
- Introduction to Geant4-based simulation

■ Refresher of basic concepts of Object Oriented Programming and C++ features *(optional)*

- This lecture provides a brief overview of basic programming concepts propaedeutic to the following lectures. If the majority of the audience is already familiar with such technical background, it will be skipped

■ Concepts of a Geant4 user application

- How Geant4 is used: fundamental concepts of a Geant4-based simulation
- Design of a basic user application
- How a user interacts with Geant4 kernel

■ Elements of Geant4 kernel

- Run and Event
- Tracking and related concepts

■ Modelling an experimental set-up with Geant4

- Primary particle generation
- Materials
- Geometry
- Read-out, detector response, hits and digitization

■ Geant4 physics

- Particles and physics processes in Geant4 design
- Electromagnetic physics
- Hadronic physics
- Validation of Geant4 physics is covered in a dedicated Refresher Course: [Geant4 Physics Validation](#) *(free of charge to NSS-MIC participants)*

■ Overview of additional capabilities

- Visualisation
- User interface
- Event biasing
- Parameterisation (fast simulation)
- Parallelisation and integration in a distributed computing environment
- Data analysis in a Geant4-based application

■ Geant4 in practice

- Geant4 installation, supported platforms
- Guided tour of a simple application

■ How to learn more

- Documentation
- User support

■ Conclusion and outlook

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

*This course encompasses training material developed by several Geant4
members:
thanks to all of them!*

Geant 4

Introduction

- This Short Course provides an overview of Geant4, its capabilities and how they can be used in an experimental simulation application
- **Geant4 is a complex and powerful toolkit**
 - Impossible to teach everything in one day!
 - Impossible to learn everything in one day either...
- This course provides you with a **vision** of Geant4 and a **method** for how to use it
 - Finding your way in the complexity of Geant4 is not easy
 - A **clear vision** will guide you in your further learning
 - Getting familiar with Geant4 **concepts** and **tools** will facilitate your learning path through Geant4 documentation and examples

Geant 4

2

1) Geant4 at a glance

• Overview of Geant4

- Geant4 vision: scope, fundamental concepts
- Geant4 architecture
- Geant4 kernel (Run, Event, Tracking)
- Overview of the main simulation domains in Geant4 and their capabilities
- Detector modelling
- Physics

Geant 4

3

2) Using Geant4

• Refresher of basic concepts of Object Oriented Programming and C++ features *(optional)*

- Brief overview of basic technical concepts propaedeutic to the following lectures
- Please let me know if you wish it
 - if the majority of the audience is familiar with this technical background, it will be skipped

• Concepts of a Geant4 user application

- How Geant4 is used: a Geant4-based application
- Design of a basic user application
- User initialisation and actions

Geant 4

4

3) Geant4 in detail

- **Geant4 kernel**
 - Basic concepts
- **Modelling an experimental set-up with Geant4**
 - Primary particles generation
 - Materials
 - Geometry
 - Read-out and detector response
- **Geant4 physics** [Geant4 Physics Validation → Refresher Course](#)
 - Particles and physics processes in Geant4 design
 - Electromagnetic physics
 - Hadronic physics
- **Overview of additional capabilities**
 - Visualisation
 - User interface
 - Event biasing
 - Parameterisation (fast simulation)
 - Parallelisation and execution in a distributed computing environment
 - Data analysis in a Geant4-based application

Geant 4

5

4) Geant4 in practice

- **Geant4 in practice**
 - Geant4 installation, supported platforms
 - Guided tour of a simple application code
- **How to learn more**
 - Documentation
 - User support
- **Conclusion and outlook**
- *Questions, feedback and how to keep in touch*

Geant 4

6

Course material

- All course material can be downloaded from <http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>
- This course is a brief introduction to the functionality and use of the Geant4 Toolkit
 - it does not replace the study of Geant4 User Documentation
- Geant4 User Documentation and further information can be found at Geant4 web site: <http://cern.ch/geant4>

Geant 4

7

Questions and feedback

- Your feedback is important
 - Please let us know if the pace is appropriate, if you need further clarifications etc.
- Geant4 is huge!
 - Despite some of us have been Geant4 developers since its early R&D days, our own expertise is not infinite...
 - We appreciate your patience, if we cannot answer all your questions
 - We'll put you in touch with other Geant4 collaborators, should you need specific expertise on any of the topics outlined in this course
- Your feedback is important also after the course
 - We would like hearing from you!

Geant 4

8

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

*This course encompasses training material developed by several Geant4
members:
thanks to all of them!*

Geant 4

Introduction

Basic concepts of simulation with Geant4 as
a general-purpose Monte Carlo toolkit

Geant 4

The role of simulation

- Simulation plays a fundamental role in various domains and phases of an experimental physics project
 - design of the experimental setup
 - evaluation and definition of the potential physics output of the project
 - evaluation of potential risks to the project
 - assessment of the performance of the experiment
 - development, test and optimisation of reconstruction and physics analysis software
 - contribution to the calculation and validation of physics results
- The scope of these lectures (and of Geant4) encompasses the [simulation of the passage of particles through matter](#)
 - there are other kinds of simulation components, such as *physics event generators*, *electronics response* generation, etc.
 - often the simulation of a complex experiment consists of several of these components interfaced to one another

Geant 4

11

Detector Simulation - General

- General characteristics of a detector simulation system
 - You specify the set-up of an experimental system
 - Then the software system automatically transports the particle you shoot into the defined system by simulating their interactions in matter based on the Monte Carlo method
- The heart of the simulation: the Monte Carlo method
 - A method to search for solutions to a mathematical problem using statistical sampling with random numbers

Geant 4

12

Basic requirements for a simulation system

- Modeling the experimental set-up
- Tracking particles through matter
- Interaction of particles with matter
- Modeling the detector response
- Run and event control
- Accessory utilities (*random number generators, PDG particle information etc.*)
- Interface to event generators
- Visualisation of the set-up, tracks and hits
- User interface
- Persistency

Geant 4

13

The zoo

EGS4, EGS5, EGSnrc
Geant3, Geant4
MARS
MCNP, MCNPX, A3MCNP, MCNP-DSP
MVP, MVP-BURN
Penelope
Peregrine
Tripoli-3, Tripoli-3 A, Tripoli-4

...and I probably forgot some more

Many codes not publicly distributed

A lot of business around MC

DPM
EA-MC
FLUKA
GEM
HERMES
LAHET
MCBEND
MCU
MF3D
NMTC
MONK
MORSE
RTS&T-2000
SCALE
TRAX
VMC++

Monte Carlo codes presented at the MC200 Conference, Lisbon, October 2000

Geant 4

14

What can Geant4 do for you?

- Transports a particle step-by-step, taking into account interactions with materials and external electromagnetic fields, until the particle
 - loses all its kinetic energy,
 - disappears due to an interaction,
 - reaches the boundary of the simulation volume
- Provides means for the user to interact with the transport process and access to the simulation results
 - at the beginning and end of transport
 - at the end of each step in transport
 - at the time when the particle is in a sensitive volume of the setup
 - etc.

Geant 4

15

What should you do for Geant4?

- Three **essential** pieces of information you must provide:
 - **Geometry and material** description of your experimental setup
 - Kinematical information of the **primary particles** to be tracked
 - Choice of **physics processes** which may occur
- **Additional** matter you may provide, if you wish:
 - Magnetic and electric field
 - Actions you want to take when you access particle transport
 - Actions you want to take when a particle goes into a sensitive volume of the experimental setup
 - etc.

Geant 4

16

Computing background knowledge to use Geant4

- **C++**
 - Geant4 is implemented in C++, therefore a basic knowledge of C++ is required
 - C++ is a complex language, but you are not required to be a C++ expert to use Geant4
- **Object Oriented Technology**
 - basic concepts
 - in-depth knowledge needed only for the development of complex applications
- **Unix/Linux**
 - it is a standard work environment for Geant4, therefore some minimum knowledge/experience is recommended
 - How to use basic Unix command
 - How to compile a C++ code
- **Windows**
 - You can use Visual C++
 - Though you still need some knowledge of Unix (cygwin) for installation
 - *Beware: most Geant4 developers are not familiar with Geant4 on Windows!*

Geant 4

17

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it


Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

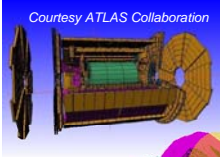
This course encompasses training material developed by several Geant4 members:

thanks to all of them!


Geant 4



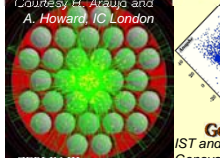
Courtesy CMS Collaboration



Courtesy ATLAS Collaboration



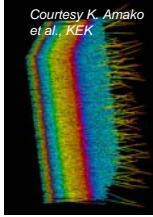
Courtesy R. Narfallo et al., ESA



Courtesy H. Araujo and A. Howard, IC London

Geant 4

Born from the requirements of large scale HEP experiments

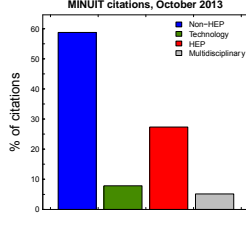


Courtesy K. Amako et al., KEK

Widely used also in

- Space science and astrophysics
- Medical physics, nuclear medicine
- Radiation protection
- Accelerator physics
- Pest control, food irradiation
- Homeland security
- etc...

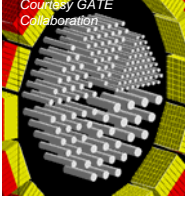
S. Agostinelli et al., Geant4: a simulation toolkit, *NIM A* 506, pp. 250-303, 2003



MINUIT citations, October 2013

> 4000 citations

Thomson-Reuters, ISI Web of Science,
Most cited paper
Nuclear Science and Technology,
Instruments and Instrumentation,
Particle and Fields Physics



Courtesy GATE Collaboration

Outline

Geant4 in one hour...

- **A little bit of software**
 - Basic concepts of Geant4 use: application, PhysicsList
- **Overview of Geant4 physics functionality**
 - Electromagnetic and hadronic physics
- **Validation**
 - Concepts and a few results
- **Outlook**

Geant 4

Geant4 users

- What is the difference between Standard and low energy?
- What is the difference between Geant4 and MCNP? (*Geant4* and *EGS*, *Geant4* and *FLUKA*...)
- Which PhysicsList should I use?

TNS editor

- Could you please document the validation of your simulation?
- Could you please quantify the accuracy of your simulation?
- Why did you use model X in your simulation?

What is Geant 4 ?

OO Toolkit

for the simulation of next generation HEP detectors

...of the current generation

...not only of HEP detectors

Born from **RD44**, 1994 – 1998 (R&D phase)

1st release: 15 December 1998

1-2 new releases/year since then

RD44 was also an experiment of

- **distributed software production** and management
- application of rigorous **software engineering** methodologies
- introduction of the **object oriented technology** in the HEP environment

Geant 4

21

Strategic vision

Toolkit

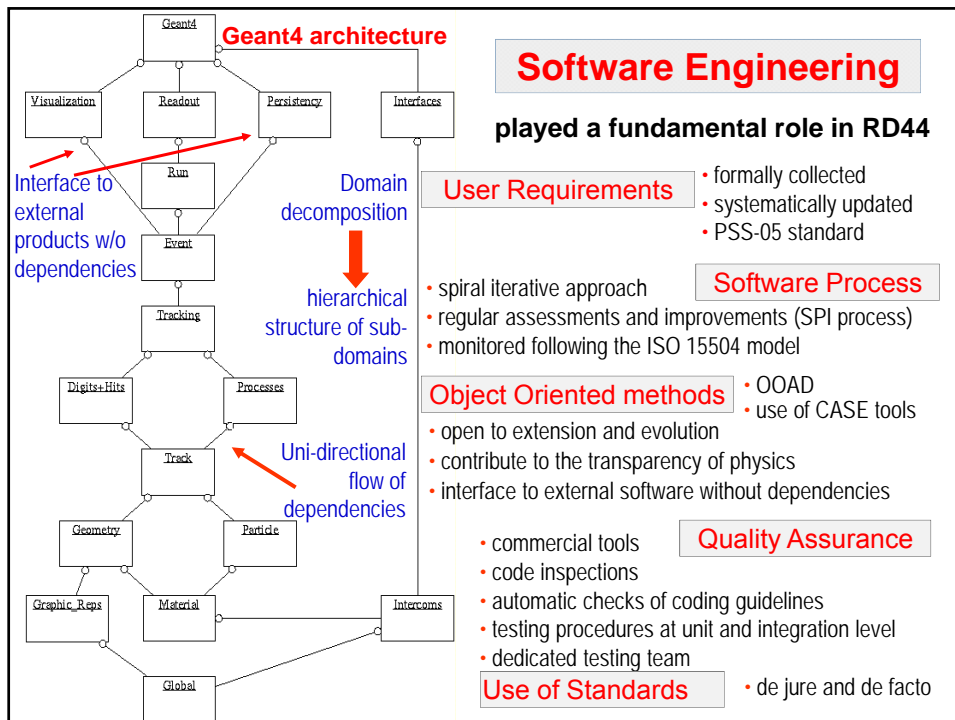
- A set of compatible components
 - each component is specialised for specific functionality
 - each component can be refined independently
- Components can cooperate at any degree of complexity
- Providing (and using) alternative components is easy
- User applications can be customised as needed

Object-oriented technology

- Open to extension and evolution
 - New implementations can be added without changing existing code
- Robustness and ease of maintenance
- Protocols and well defined dependencies minimize coupling

Geant 4

22



Geant4 kernel: Run and Event

- Conceptually, a **run** is a collection of **events** that share the *same detector conditions*
 - Detector and physics settings are frozen in a run
- An **event** initially contains the *primary particles*; they are pushed into a stack and further processed

Geant4 kernel: Tracking

- Decoupled from physics
 - all processes handled through the same abstract interface
- Independent from particle type
- New physics processes can be added to the toolkit without affecting tracking
- Geant4 has only **secondary production thresholds, no tracking cuts**
 - all particles are tracked down to zero range
 - energy, TOF ... cuts can be defined by the user

Geant 4

25

Materials

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

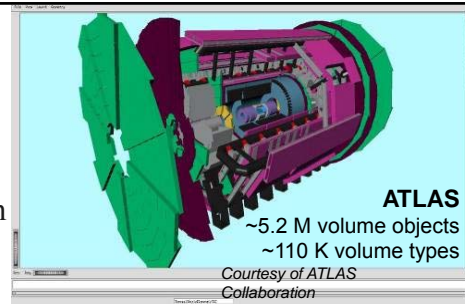
```
G4double density = 1.390*g/cm3;  
G4double a = 39.95*g/mole;  
G4Material* lAr =  
  new G4Material("liquidArgon", z=18., a, density);
```

Geant 4

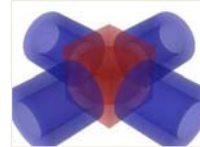
26

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



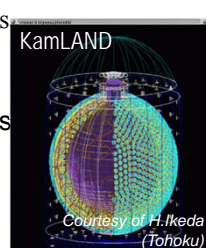
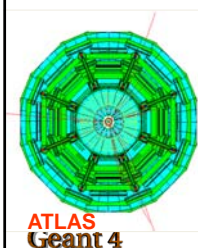
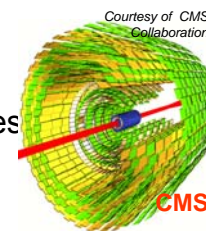
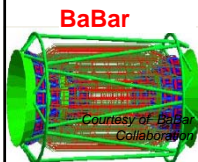
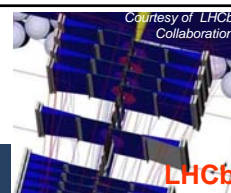
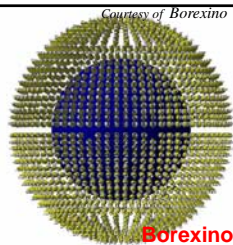
Transparent solids

Geant 4

27

Solids

Multiple representations
Same abstract interface



CSG (Constructed Solid Geometries)

- simple solids

STEP extensions

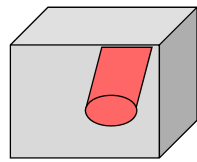
- polyhedra, spheres, cylinders, cones, toroids, etc.

BREPS (Boundary REPresented Solids)

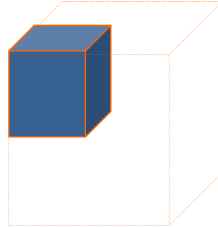
- volumes defined by boundary surfaces

CAD exchange

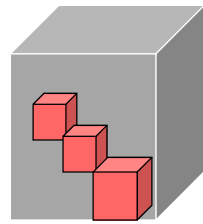
Physical Volumes



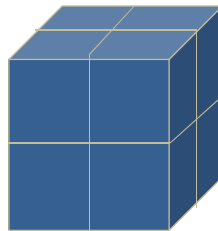
placement



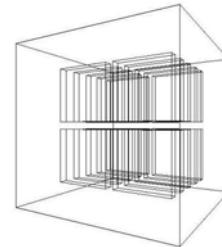
replica



parameterised



assembled

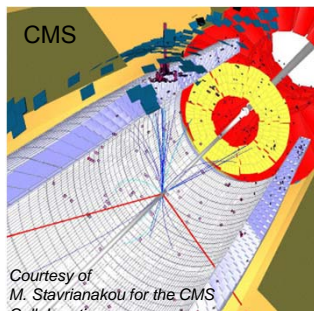


Versatility to describe complex geometries

Geant 4

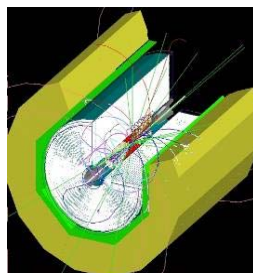
29

Electric and magnetic fields



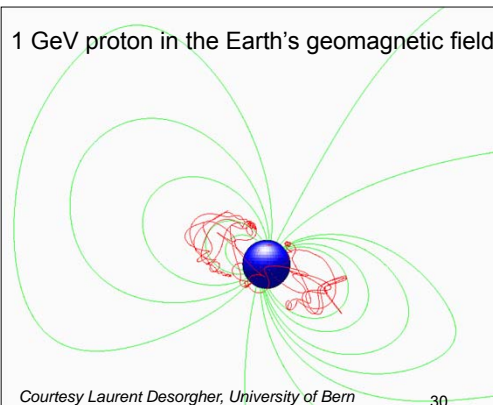
Courtesy of
M. Stavrianakou for the CMS
Collaboration

MOKKA
Linear
Collider
Detector



of variable non-uniformity
and differentiability

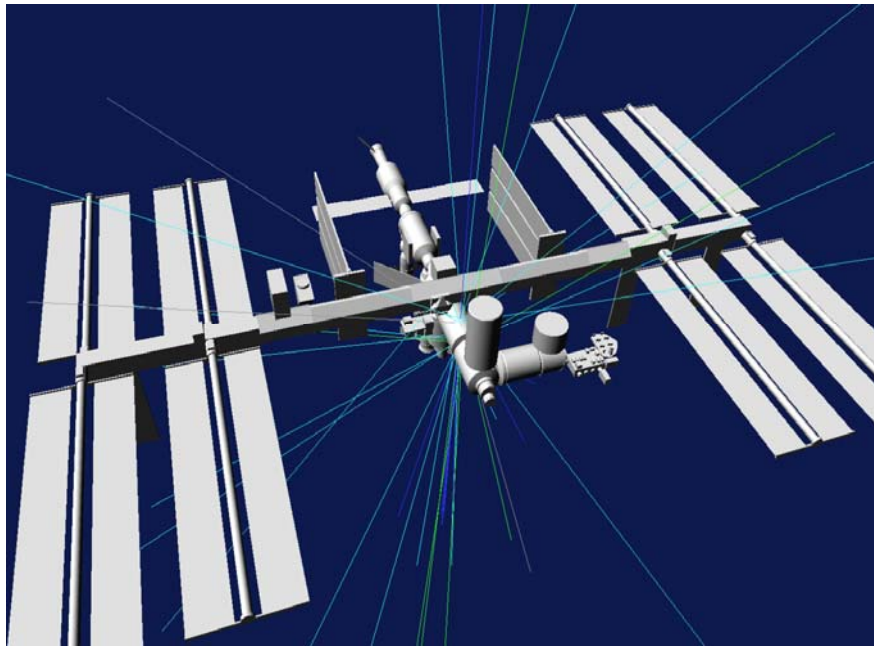
1 GeV proton in the Earth's geomagnetic field



Courtesy Laurent Desorgher, University of Bern

30

Geant 4



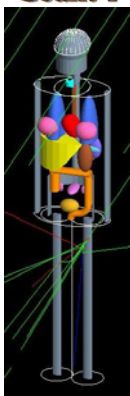
Courtesy T. Ersmark, KTH Stockholm

Geant 4

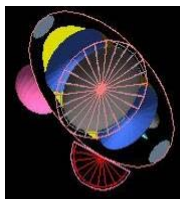
31

**Not only large scale,
complex detectors...**

Geant 4



Analytical
breast

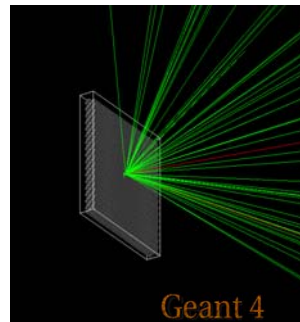


Voxel breast



Geant4 anthropomorphic phantoms

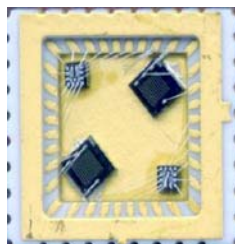
Geant 4



Geant 4

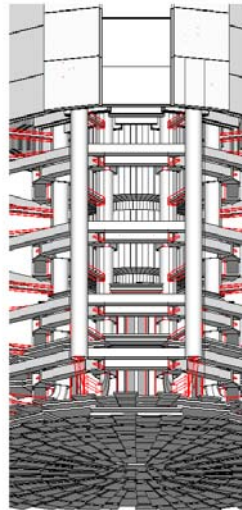
simple geometries

small scale components



Geant 4

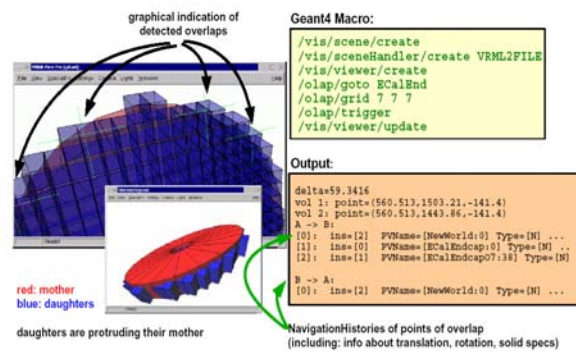
One may also do it wrong...



DAVID

Geant 4

Tools to detect badly defined geometries



33

Other features

- ◆ **Particles**
 - all PDG data and more for specific Geant4 use, like ions
- ◆ **Hits & Digitization**
 - to describe detector response
- ◆ **Primary event generation**
 - some general purpose tools provided in the toolkit
- ◆ **Event biasing**
- ◆ **Fast simulation**
- ◆ **Persistency**
- ◆ **Parallelisation**
- ◆ No time to review them in detail
 - Geant4 user documentation

Geant 4

34

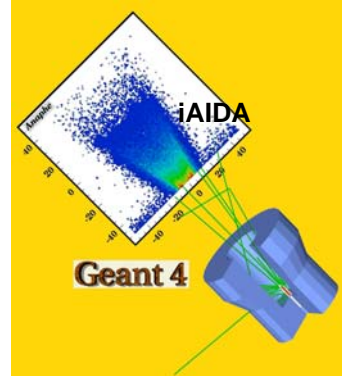
Interface to external tools

through abstract interfaces

no dependency
minimize coupling of components

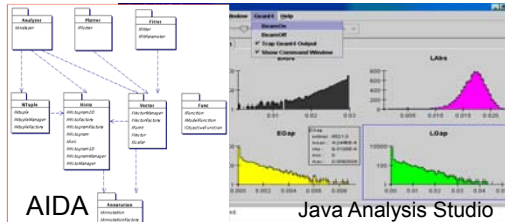
Similar approach

- Visualisation
- (G)UI
- Persistency
- Analysis



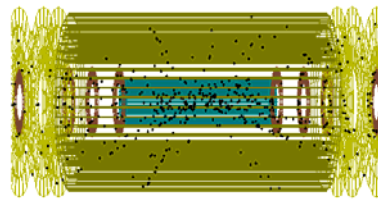
The user is free to choose
the concrete system
he/she prefers for each
component

Geant 4



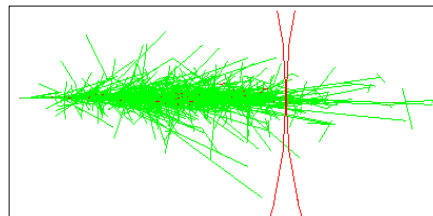
Visualisation

- Control of several kinds of visualisation
 - detector geometry
 - particle trajectories
 - hits in detectors



- Various **drivers**

- OpenGL
- OpenInventor
- X11
- Postscript
- DAWN
- OPACS
- HepRep
- VRML...

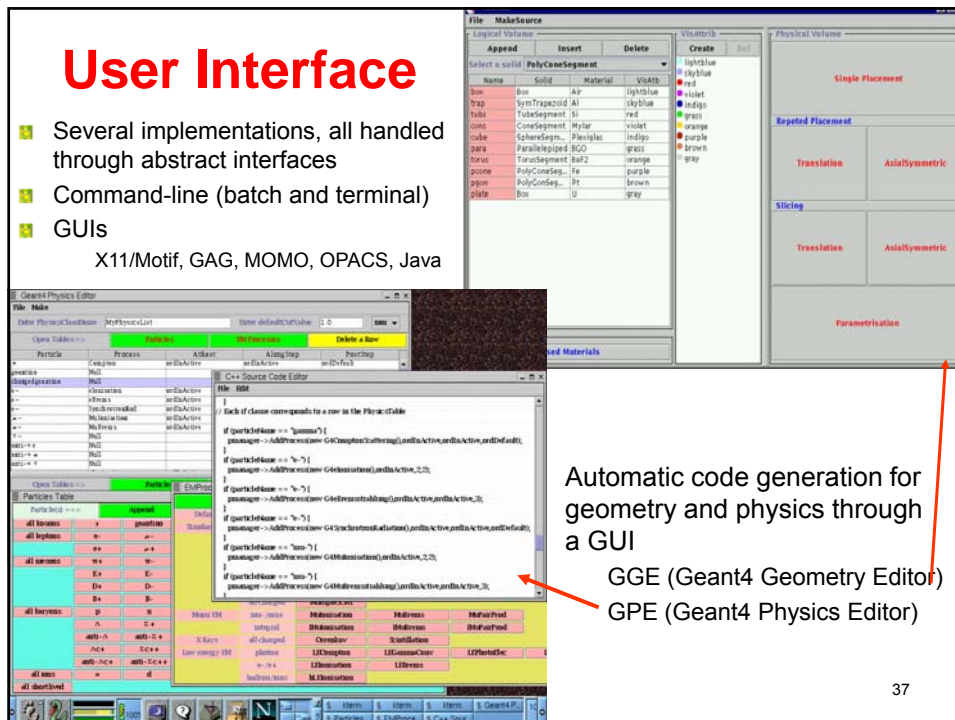


- all handled through abstract interfaces
- Geant 4**

36

User Interface

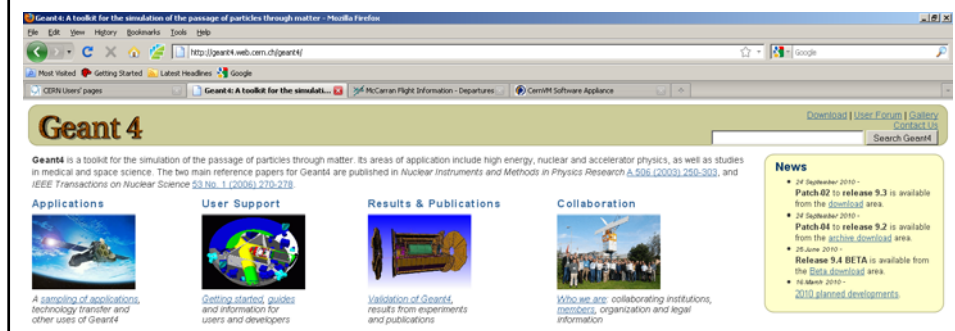
- Several implementations, all handled through abstract interfaces
- Command-line (batch and terminal)
- GUIs
 - X11/Motif, GAG, MOMO, OPACS, Java



37

Distribution

- Geant4 is **open-source**
- Freely available**
 - Source code, libraries, associated data files and documentation can be downloaded from <http://cern.ch/geant4>
- User support provided on a best effort basis
 - User Forum: mutual support within the user community



Physics

“It was noted that experiments have requirements for **independent, alternative physics models**. In Geant4 these models, *differently from the concept of packages*, allow the user to **understand** how the results are produced, and hence improve the **physics validation**. Geant4 is developed with a modular architecture and is the ideal framework where existing components are integrated and new models continue to be developed.”

Minutes of LCB (LHCC Computing Board) meeting, 21/10/1997

Geant 4

39

Physics: general features

- Ample variety of physics functionality
- **Abstract interface** to physics processes
 - Tracking **independent** from physics
- Open system
 - Users can easily create and use their own models
- Distinction between **processes** and **models**
 - often multiple models for the same physics process
 - complementary/alternative

Geant 4

40

Electromagnetic physics

- electrons and positrons
- photons (including optical photons)
- muons
- charged hadrons
- ions

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

- Comparable to GEANT 3 already in α release 1997
- Further extensions (*facilitated by OO technology*)
- High energy extensions
 - Motivated by LHC experiments, cosmic ray experiments...
- Low energy extensions
 - motivated by space and medical applications, dark matter and ν experiments, antimatter spectroscopy, radiation effects on components etc.
- Alternative models for the same process

Geant 4

41

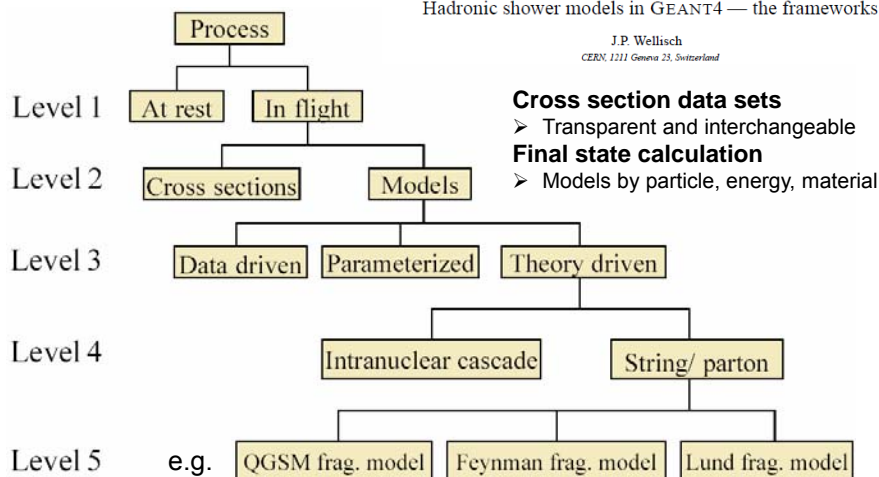
Hadronic physics

Computer Physics Communications 140 (2001) 65–75

Computer Physics
Communications
www.elsevier.com/locate/cpc

Hadronic shower models in GEANT4 — the frameworks

J.P. Wellisch
CERN, 1211 Geneva 23, Switzerland



Ample variety of **alternative/complementary** models

Geant 4

42

Validation

Refresher Course dedicated to

Geant4 Physics Validation

Wednesday, 30 October, 12:40-14:00
Grand Ballroom 105

Open to NSS-MIC participants
(no registration, no fee)

Geant 4

43

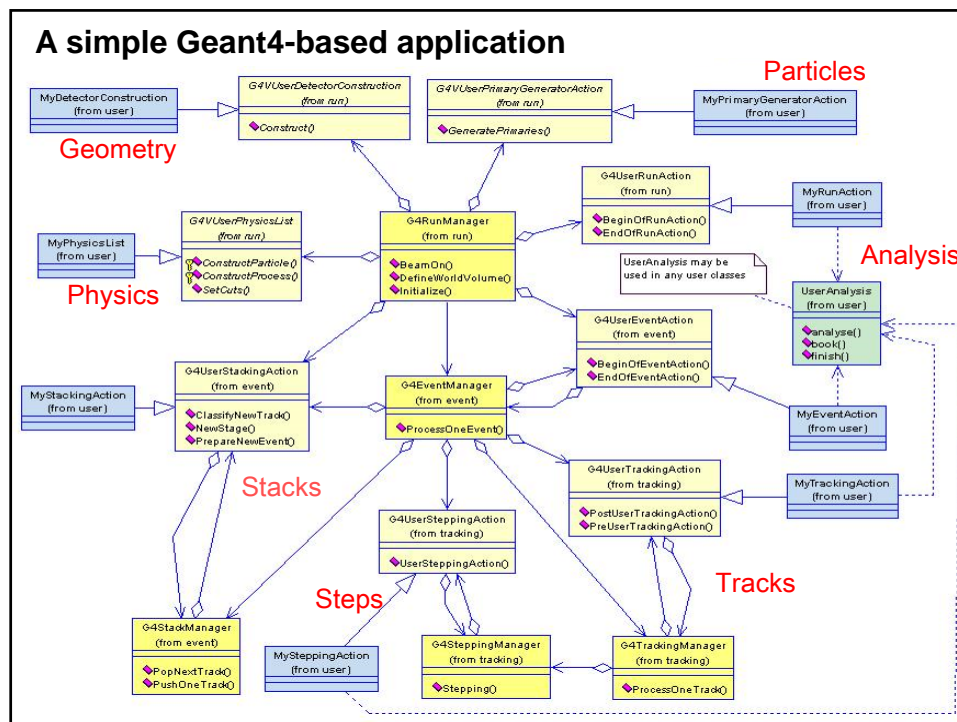
Toolkit + User application



- ♦ Geant4 is a **toolkit**
 - i.e. one cannot “run” Geant4 out of the box
 - One must write an application, which uses Geant4 tools
- ♦ Consequences
 - There is no such concept as “**Geant4 defaults**”
 - One must provide the necessary information to configure one’s own simulation
 - The user must deliberately **choose** which Geant4 tools to use
- ♦ Guidance: many **examples** are distributed with Geant4

Geant 4

44



Basic actions

- What a user **must** do:
 - Describe the **experimental set-up**
 - Input **primary particles** into the simulation
 - Decide which **particles** and **physics models** one wants to use out of those available in Geant4 and the desired precision of the simulation (*cuts to produce secondary particles to be further tracked*)
- One may also **want**
 - To interact with Geant4 kernel to **control** the simulation
 - To **visualise** the simulation configuration or results
 - To produce and to access **objects encoding simulation results** for further analysis

Wide user community

- Geant4 1st reference paper is the most cited paper of Thomson-Reuters'

- Nuclear Science & Technology
- Instruments & Instrumentation
- Physics, Particles & Fields (2013)

S. Agostinelli et al.

Geant4: a simulation toolkit

Nucl. Instr. Meth. A, vol. 506, pp. 250-303, 2003

- Used in particle and nuclear physics, astrophysics, space science, medical physics, medical imaging, industry etc.

Through the narrow gate

- **Think**

- **Learn**

- Master the technology
- Study the literature
- Read Geant4 documentation

- **Work**

- You do not run Geant4, you run your own application
- Master your experimental problem
- Understand what Geant4 does
- Understand what you are doing



Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

*This course encompasses training material developed by several Geant4
members:*

thanks to all of them!

Geant 4

**(Minimal)
(Practical)**

Introduction to C++ and OOP

For use in the Geant4 course

Largely incomplete

Not meant to replace good C++, UML, and OOP
books!

Geant 4

C++ basics

C++ is **not** an object oriented language

A “superset” of C
You can write procedural code in C++

Geant 4

Getting started

```
// my first program in C++
#include <iostream>
int main ()
{
    std::cout << "Hello World!";
    return 0;
}
```

// This is a **comment** line

#include <iostream>

- directive for the **preprocessor**

int main ()

- beginning of the definition of the **main function**
- the main function is the point by where all C++ programs start their execution
- all C++ programs must have a main function
- body enclosed in braces {}

cout << "Hello World";

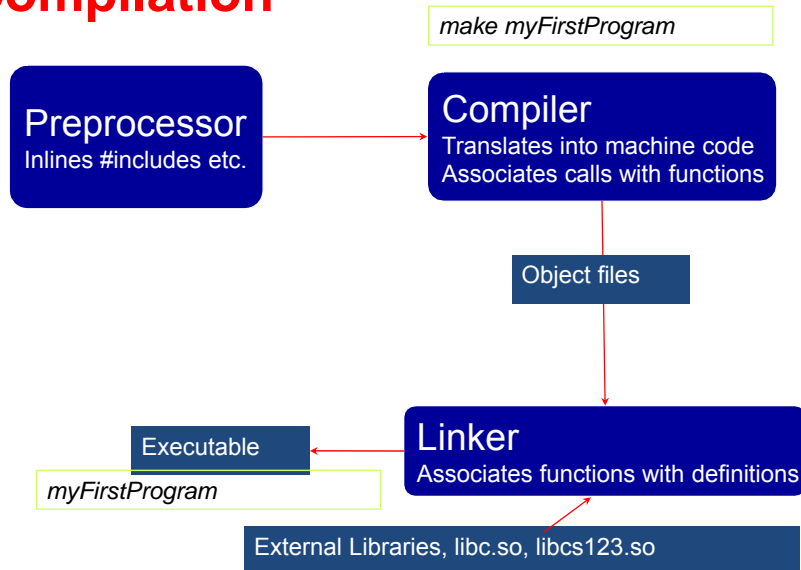
- C++ statement
- cout is declared in the iostream standard file within the std namespace
- cin
- semicolon (;) marks the end of the statement

return 0;

- the return statement causes the main function to finish
- return may be followed by a return code (here: 0)
 - return code 0 for the main function is generally interpreted as the program worked OK

Geant 4

Compilation



Geant 4

53

Using namespace

```
#include <iostream>
#include <string>
...
std::string question = "What do I learn this week?";
std::cout << question << std::endl;

using namespace std;
...
string answer = "How to use Geant4";
cout << answer << endl;
```

Geant 4

54


```
#include <iostream>
#include <string>
using namespace std;
int main ()
{
    // declaring variables:
    int a, b; // declaration
    int result;
    // process:
    a = 5;
    b = 2;
    a = a + 1;
    result = a - b;
    // print out the result:
    cout << result << endl;
    string myString = "This is a string";
    cout << myString << endl;
    const int neverChangeMe = 100;
    // terminate the program:
    return 0;
}
```

Variables

Scope of variables

- global variables can be referred from anywhere in the code
- local variables: limited to the block enclosed in braces ({})

Initialization

```
int a = 0; // assignment operator
int a(0); // constructor
```

const

the value cannot be modified after definition

55

References and Pointers

The address that locates a variable within memory is what we call a **reference** to that variable

```
x = &y;    // reference operator & "the address of"
```

Reference is an alias

```
int i = 10;
int& ir = i;    // reference (alias)
ir = ir + 1;    // increment i
```

A variable which stores a reference to another variable is called a **pointer**. Pointers are said to "point to" the variable whose reference they store

```
z = *x;    // z equal to "value pointed by" x
```

```
double* z; // z is a pointer to a double
double x = 35.7;
z = &x;    // therefore *z is 35.7
```

```
z = 0;    // null pointer (not pointing to any valid reference or memory address)
```

Geant 4

56

Read pointer declarations right to left

```
// A const River  
const River nile;
```

```
// A pointer to a const River  
const River* nilePc;
```

```
// A const pointer to a River  
River* const nileCp;
```

```
// A const pointer to a const River  
const River* const nileCpc;
```

Geant 4

57

Dynamic memory

Operator new

pointer = new type

```
Student* paul = new Student;
```

If the allocation of this block of memory failed, the failure could be detected by checking if paul took a null pointer value:

```
if (paul == 0) {  
    // error assigning memory, take measures  
};
```

Operator delete

```
delete paul;
```

Dynamic memory should be freed once it is no longer needed, so that the memory becomes available again for other requests of dynamic memory
Rule of thumb: every **new** must be paired by a **delete**
Failure to free memory: **memory leak**

Geant 4

58

C++ Gotcha

*Do not return pointers (or references)
to local variables!*

```
double* myFunction(void) {  
    double d;  
    return &d;  
}  
int main() {  
    double* pd = myFunction();  
    *pd = 3.14;  
    return 0;  
}
```

Boom! (maybe)

Geant 4

59

C++ “Gotcha”

Uninitialized pointers are bad!

```
int* i;  
  
if ( someCondition ) {  
    ...  
    i = new int;  
} else if ( anotherCondition ) {  
    ...  
    i = new int;  
}  
  
*i = someVariable;
```

“null pointer exception”

Geant 4

60

Memory allocation jargon

- “on the stack”

- scope: block delimited by {}
- object alive till it falls out of scope
- calls constructor / destructor

- “on the heap”

- **new** and **delete** operators
- new calls constructor, delete calls destructor
- object exists independently of scope in which it was created
- also “on the free store” or “allocated in dynamic memory”
- be careful: new → delete, new[] → delete[]
- for safety, same object should both allocate and deallocate

Geant 4

61

Operators (most common ones)

Assignment =

Arithmetic operators +, -, *, /, %

Compound assignment +=, -=, *=, /=, ...

```
a+=5; // a=a+5;
```

Increase and decrease ++, --

```
a++; // a=a+1;
```

Relational and equality operators ==, !=, >, <, >=, <=

Logical operators ! (not), && (and), || (or)

Conditional operator (?)

```
a>b ? a : b  
// returns whichever is greater, a or b
```

Explicit type casting operator

```
int i;    float f = 3.14;    i = (int) f;
```

Geant 4

62

Control structures

```
if (x == 100)
{
    cout << "x is ";
    cout << x;
}
```

```
if (x == 100)
    cout << "x is 100";
else
    cout << "x is not 100";
```

```
while (n>0) {
    cout << n << " ";
    --n;
}
```

```
do {
    cout << "Enter number (0 to end): ";
    cin >> n;
    cout << "You entered: " << n << endl;
} while (n != 0);
```

for (*initialization; condition; increase*) *statement*;

```
for (int n=10; n>0; n--) {
    cout << n << " ";
}
```

```
for (n=10; n>0; n--)
{
    cout << n << " ";
    if (n==3)
    {
        cout << "countdown aborted!";
        break;
    }
}
```

```
loop:
cout << n << " ";
n--;
if (n>0) goto loop;
cout << "Procedural programming!" ;
```

Functions

In C++ *all* function parameters are passed by **copy**

```
Type name(parameter1, parameter2, ...)
{
    statements...;
    return somethingOfType;
}
```

void

```
void printMe(double x)
{
    std::cout << x << std::endl;
}
```

Arguments passed by value and by reference

```
int myFunction (int first, int second);
```

Pass a **copy** of parameters

```
int myFunction (int& first, int& second);
```

Pass a **reference** to parameters
They may be **modified** in the function!

```
int myFunction (const int& first, const int& second);
```

const reference

They may **not** be **modified** in the function!

Geant 4

More on Functions

Default values in parameters

```
double divide (double a, double b=2. )  
{  
    double r;  
    r = a / b;  
    return r;  
}
```

```
int main ()  
{  
    cout << divide (12.) << endl;  
    return 0;  
}
```

Overloaded functions

Same name, different parameter type

A function cannot be overloaded only by its return type

```
int operate (int a, int b)  
{  
    return (a*b);  
}
```

```
double operate (double a, double b)  
{  
    return (a/b);  
}
```

Geant 4

65

OOP basics

Geant 4

OOP basic concepts

- **Object, Class**

- A class defines the abstract characteristics of a thing (object), including the thing's attributes and the thing's behaviour

- **Inheritance**

- “Subclasses” are more specialized versions of a class, which *inherit* attributes and behaviours from their parent classes (and can introduce their own)

- **Encapsulation**

- Each object exposes to any class a certain *interface* (i.e. those members accessible to that class)
- Members can be **public**, **protected** or **private**

- **Abstraction**

- Simplifying complex reality by modelling classes appropriate to the problem
- One works at the most appropriate level of inheritance for a given aspect of the problem

- **Polymorphism**

- It allows one to treat derived class members just like their parent class' members

Geant 4

67

Class and Object

Object: is characterized by attributes (which define its state) and operations

A **class** is the blueprint of objects of the same type

```
class Rectangle {
public:
    Rectangle (double,double); // constructor
    ~Rectangle() { // empty; } // destructor
    double area () { return (width * height); } // member function
private:
    double width, height; // data members
};
```

```
Rectangle rectangleA (3.,4.); // instantiate an object of type "Rectangle"
Rectangle* rectangleB = new Rectangle(5.,6.);
cout << "A area: " << rectangleA.area() << endl;
cout << "B area: " << rectangleB->area() << endl;
delete rectangleB; // invokes the destructor
```

68

The class interface in C++

Usually defined in a header (.h or .hh) file:

```
class Car {  
  public:  
    // Members can be accessed by any object  
  
  protected:  
    // Can only be accessed by Car and its derived objects  
  
  private:  
    // Can only be accessed by Car for its own use.  
};
```

Geant 4

69

Constructor and assignment

```
class String {  
  public:  
    String( const char* value ); // constructor  
    String( const String& rhs ); // copy constructor  
    ~String();  
    String& operator=( const String& rhs); // assignment operator  
  private:  
    char* data;  
};  
  
  int main() {  
    String s1 = "anton";  
    String s2( "luciano" );  
    s2 = s1;  
  };
```

Geant 4

70

Classes: Basic Design Rules

- Hide all member variables
- Hide implementation functions and data
- Minimize the number of public member functions
- Avoid default constructors
- Use **const** whenever possible / needed

OK:

A invokes a function of a B object
A creates an object of type B
A has a data member of type B

Bad:

A uses data directly from B
(without using B's interface)

Even worse:

A directly manipulates data in B

Geant 4

71

Inheritance

- A key feature of C++
- Inheritance allows to create classes derived from other classes
- Public inheritance defines an “is-a” relationship
 - In other words: what applies to a base class applies to its derived classes

```
class Base {  
public:  
    virtual ~Base() {}  
    virtual void f() {...}  
protected:  
    int a;  
private:  
    int b; ...  
};
```

```
class Derived : public Base {  
public:  
    virtual ~Derived() {}  
    virtual void f() {...}  
    ...  
};
```

Geant 4

72

Polymorphism

- Mechanism that allows a derived class to modify the behaviour of a member declared in a base class

```
Base* b = new Derived;  
b->f();  
delete b;
```

Which f() gets called?

Geant 4

73

Liskov Substitution Principle

- One way of expressing the notion of subtype (or “is-a”)

If Derived is a subtype of Base, then

Base can be replaced everywhere with Derived,
without impacting any of the desired properties of the program

- In other words, you can substitute Base with Derived, and nothing will “go wrong”

Geant 4

74

Inheritance and virtual functions

```
class Shape
{
public:
    Shape();
    virtual void draw();
};
```

A virtual function defines the interface and provides an implementation; derived classes may provide alternative implementations

```
class Circle : public Shape
{
public:
    Circle (double r);
    void draw();
private:
    double radius;
};
```

```
class Rectangle : public Shape
{
public:
    Rectangle(double h, double w);
private:
    double height, width;
};
```

Geant 4

75

Abstract classes, Abstract interfaces

Abstract class, cannot be instantiated

```
class Shape
{
public:
    Shape();
    virtual area() = 0;
};
```

Abstract Interface

a class consisting of pure virtual functions only

A pure virtual function **defines the interface** and delegates the implementation to derived classes

```
class Circle : public Shape
{
public:
    Circle (double r);
    double area();
private:
    double radius;
};
```

Geant 4

```
class Rectangle : public Shape
{
public:
    Rectangle(double h, double w);
    double area();
private:
    double height, width;
};
```

Concrete class 76

Inheritance and Virtual Functions

	Inheritance of the interface	Inheritance of the implementation
Non virtual function	Mandatory	Mandatory
Virtual function	Mandatory	By default Possible to reimplement
Pure virtual function	Mandatory	Implementation is mandatory

Geant 4

77

More C++

Geant 4

Templates

Minimal introduction, only to introduce STL

- A C++ template is just that, a template
- A single template serves as a pattern, so it can be used multiple times to create multiple instantiations

```
template <typename T>
void f(T i) { ... }
```

One function in source code

Compilation & instantiation

```
f:
save_regs
ld r0, sp(0)
add 4
ret
```

```
f:
save_regs
ld r0, sp(4)
add 8
ret
```

```
f:
save_regs
ld r0, sp(8)
add 16
ret
```

- Function templates
- Class templates
- Member templates

Multiple functions in assembly language

79

Instantiation

Compilation

```
template
<typename T>
void
f(T i)
{ ... }
```

```
void
f(char i)
{ ... }
```

```
void
f(int i)
{ ... }
```

```
void
f(double i)
{ ... }
```

```
f<char>:
save_regs
ld r0, sp(0)
add 4
ret
```

```
f<int>:
save_regs
ld r0, sp(4)
add 8
ret
```

```
f<double>:
save_regs
ld r0, sp(8)
add 16
ret
```

Geant 4

Multiple functions in assembly language

80

Standard Template Library (STL)

Containers

- **Sequence**
 - **vector**: array in contiguous memory
 - **list**: doubly-linked list (fast insert/delete)
 - **deque**: double-ended queue
 - stack, queue, priority queue
- **Associative**
 - **map**: collection of (key,value) pairs
 - **set**: map with values ignored
 - multimap, multiset (duplicate keys)
- **Other**
 - **string**, **basic_string**
 - **valarray**: for numeric computation
 - **bitset**: set of N bits

Algorithms

- **Non-modifying**
 - find, search, mismatch, count, for_each
- **Modifying**
 - copy, transform/apply, replace, remove
- **Others**
 - unique, reverse, random_shuffle
 - sort, merge, partition
 - set_union, set_intersection, set_difference
 - min, max, min_element, max_element
 - next_permutation, prev_permutation

Geant 4

81

std::string

Example:

```
#include <string>

void FunctionExample()
{
    std::string s, t;
    char c = 'a';
    s.push_back(c); // s is now "a";
    const char* cc = s.c_str(); // get ptr to "a"
    const char dd[] = 'like';
    t = dd; // t is now "like";
    t = s + t; // append "like" to "a"
}
```

Geant 4

82

std::vector

Example:

use std::vector,
not built-in C-style array,
whenever possible

```
#include <vector>
void FunctionExample()
{
    std::vector<int> v(10);
    int a0 = v[3];      // unchecked access
    int a1 = v.at(3);   // checked access
    v.push_back(2);     // append element to end
    v.pop_back();       // remove last element
    size_t howbig = v.size(); // get # of elements
    v.insert(v.begin()+5, 2); // insert 2 after 5th element
}
```

Geant 4

83

std::vector (more)

Example:

```
#include <vector>
#include <algorithm>

void FunctionExample()
{
    std::vector<int> v(10);
    v[5] = 3; // set fifth element to 3
    std::vector<int>::const_iterator it
        = std::find(v.begin(), v.end(), 3);
    bool found = it != v.end();
    if (found) {
        int three = *it;
    }
}
```

Geant 4

84

Iterators

- *iterator* – kind of generalized pointer
- Each container has its own type of iterator

```
void FunctionExample() {  
    std::vector<int> v;  
    std::vector<int>::const_iterator it = v.begin();  
    for (it = v.begin() ; it != v.end() ; it++) {  
        int val = *it;  
    }  
}
```

Geant 4

85

A few practical issues

Geant 4

86

Organizational Strategy

image.hh

Header file: Class definition

```
void SetAllPixels(const Vec3& color);
```

image.cc

.cc file: Full implementation

```
void Image::SetAllPixels(const Vec3& color) {  
    for (int i = 0; i < width*height; i++)  
        data[i] = color;  
}
```

main.cc

Main function

```
myImage.SetAllPixels(clearColor);
```

Geant 4

87

How a Header File looks like

header file

begin header guard

forward declaration

class declaration

constructor

destructor

member functions

member variables

need semi-colon

end header guard

Segment.h

```
#ifndef SEGMENT_HEADER  
#define SEGMENT_HEADER  
  
class Point;  
class Segment  
{  
public:  
    Segment();  
    virtual ~Segment();  
    double length();  
private:  
    Point* p0,  
    Point* p1;  
}  
#endif // SEGMENT_HEADER
```

Geant 4

88

Forward Declaration

Gui.hh

Class Gui

{

//

};

Controller.hh

//Forward declaration

class Gui;

class Controller

{

//...

private:

Gui* myGui;

//...

};

- In header files, only include what you must

- If only pointers to a class are used, use forward declarations

Geant 4

89

Header file and implementation

File Segment.hh

```
#ifndef SEGMENT_HEADER
#define SEGMENT_HEADER
```

```
class Point;
```

```
class Segment
```

```
{
```

```
public:
```

```
Segment();
```

```
virtual ~Segment();
```

```
double length();
```

```
private:
```

```
Point* p0,
```

```
Point* p1;
```

```
};
```

```
#endif // SEGMENT_HEADER
```

Geant 4

File Segment.cc

```
#include "Segment.hh"
```

```
#include "Point.hh"
```

```
Segment::Segment() // constructor
```

```
{
```

```
    p0 = new Point(0.,0.);
```

```
    p1 = new Point(1.,1.);
```

```
}
```

```
Segment::~~Segment() // destructor
```

```
{
```

```
    delete p0;
```

```
    delete p1;
```

```
}
```

```
double Segment::length()
```

```
{
```

```
    function implementation ...
```

```
}
```

90

“Segmentation fault (core dumped)”

Typical causes:

```
int intArray[10];  
intArray[10] = 6837;
```

Access outside of
array bounds

```
Image* image;  
image->SetAllPixels(colour);
```

Attempt to access
a NULL or previously
deleted pointer

These errors are often very difficult to catch and can cause
erratic, unpredictable behaviour

Geant 4

91

UML

Geant 4

Unified Modelling Language

- The UML is a graphical language for
 - specifying
 - visualizing
 - constructing
 - documentingthe artifacts of software systems
- Define an easy-to-learn, but semantically rich visual modeling language
- Added to the list of OMG adopted technologies in November 1997 as UML 1.1
- Version evolution

Geant 4

93

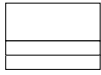
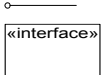
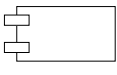
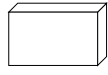
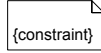
Building Blocks

- The basic building blocks of UML are:
 - model elements
 - classes, interfaces, components, use cases etc.
 - relationships
 - associations, generalization, dependencies etc.
 - diagrams
 - class diagrams, use case diagrams, interaction diagrams etc.
- Simple building blocks are used to create large, complex structures

Geant 4

94






Structural Modeling: Core Elements

Construct	Description	Syntax
class	a description of a set of objects that share the same attributes, operations, methods, relationships and semantics	
interface	a named set of operations that characterize the behavior of an element	
component	a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces.	
node	a run-time physical object that represents a computational resource	
constraint	a semantic condition or restriction	

Gea

95

Structural Modeling: Core Relationships

Construct	Description	Syntax
association	a relationship between two or more classifiers that involves connections among their instances	
aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part	
generalization	a taxonomic relationship between a more general and a more specific element	
dependency	a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element)	
realization	a relationship between a specification and its implementation	

Ge

96

Class

- Classes can have 4 parts
 - **Name**
 - **Attributes**
 - **Operations**
 - Responsibilities
- Classes can show **visibility** and **types**
- All parts but the Name are optional

MyClassName

+Some**Public**Attribute : SomeType

-Some**Private**Attribute : SomeType

#Some**Protected**Attribute : SomeType

+ClassMethodOne()

+ClassMethodTwo()

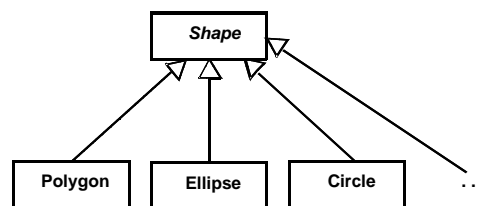
Responsibilities

-- can optionally be described here.

Geant 4

97

Generalization

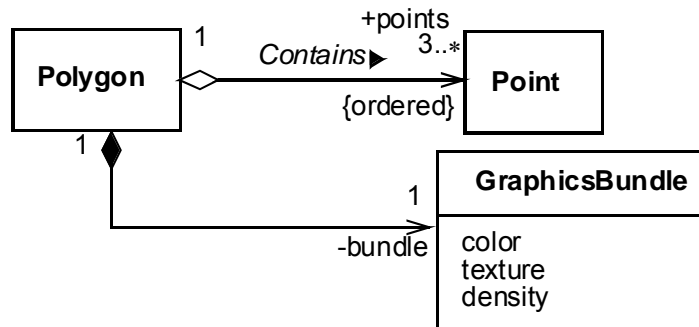


Models inheritance

Geant 4

98

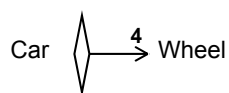
Associations



Geant 4

99

Aggregation or Composition?



Aggregation is a relationship in which one object is a part of another

A aggregates B

=

B is part of A, but their lifetimes may be different

Composition is a relationship in which one object is an integral part of another

A contains B

=

B is part of A, and their lifetimes are the same

Geant 4

100

Main UML Diagrams

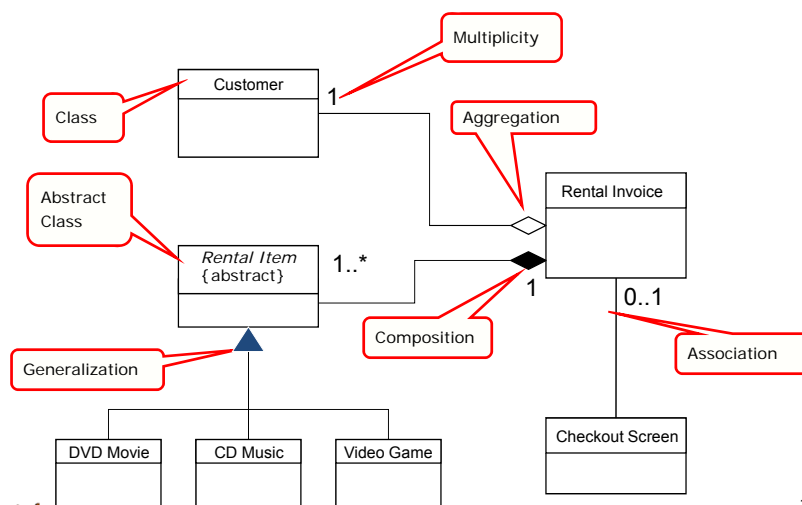
- **Class** Diagrams
- Use Case Diagrams
- **Collaboration** Diagrams
- **Sequence** Diagrams
- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Activity Diagrams
- State Diagrams

Geant 4

101

UML Class Diagram

Describe the **classes** in the system
and the **static** relationships between classes

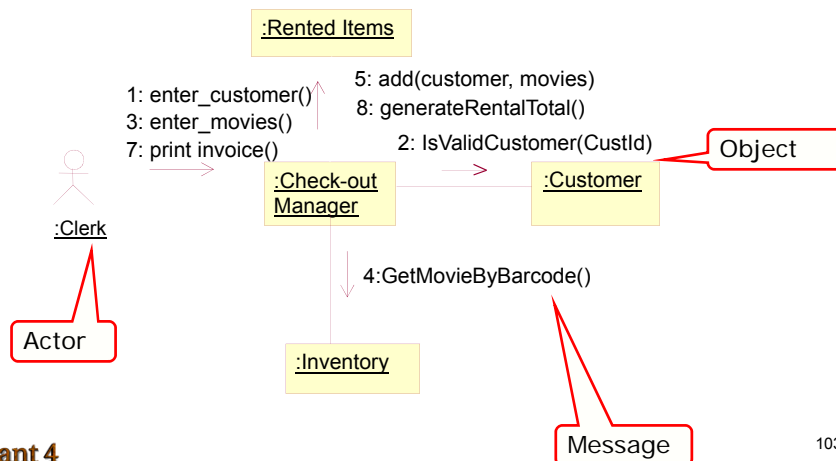


Geant 4

102

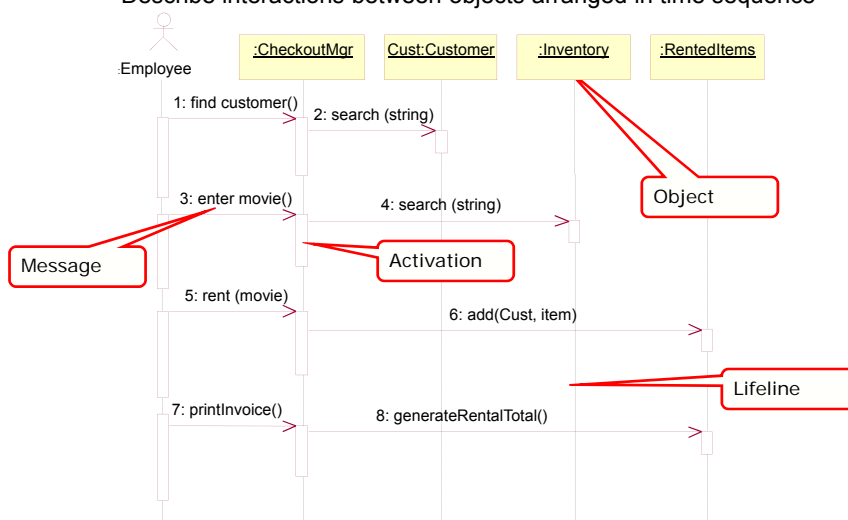
Collaboration Diagram - Rent Movie

Describe object **interactions** organized around the objects and their links to each other



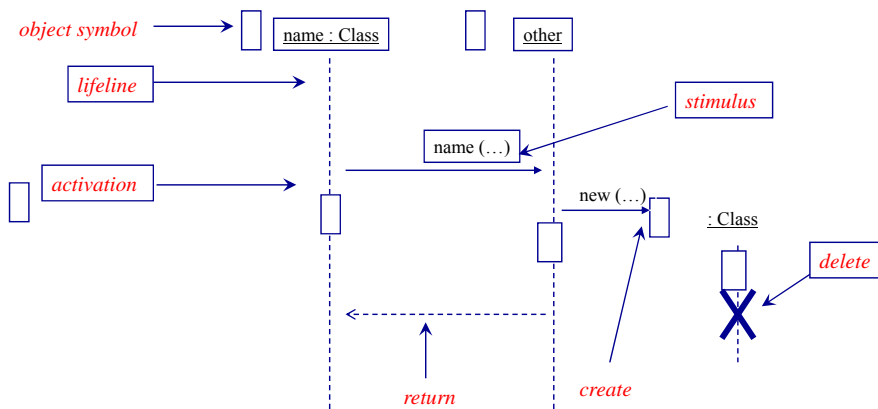
Sequence Diagram - Rent Movie

Describe interactions between objects arranged in time sequence



Sequence and collaboration diagrams can be cloned from each other

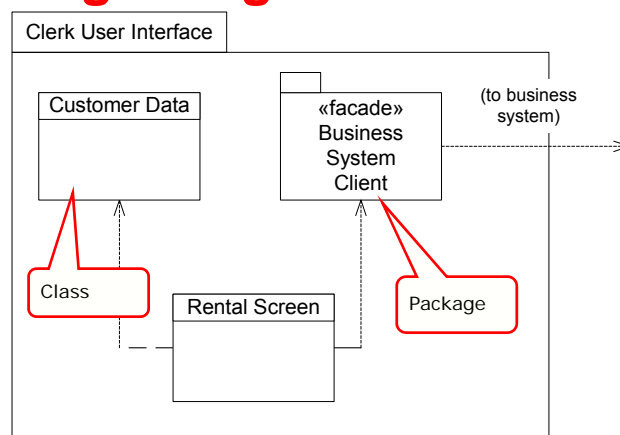
Sequence Diagram



Geant 4

105

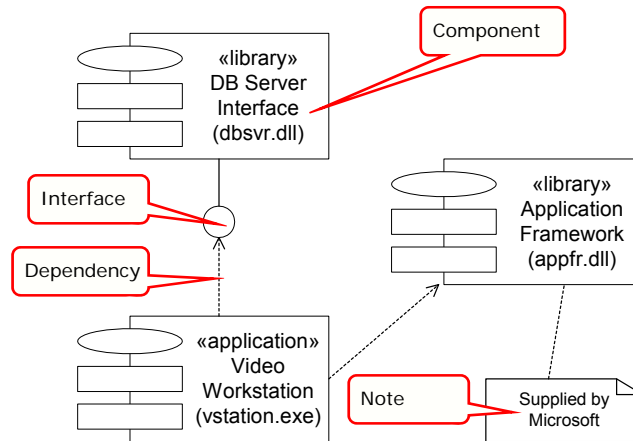
Package Diagram



Geant 4

106

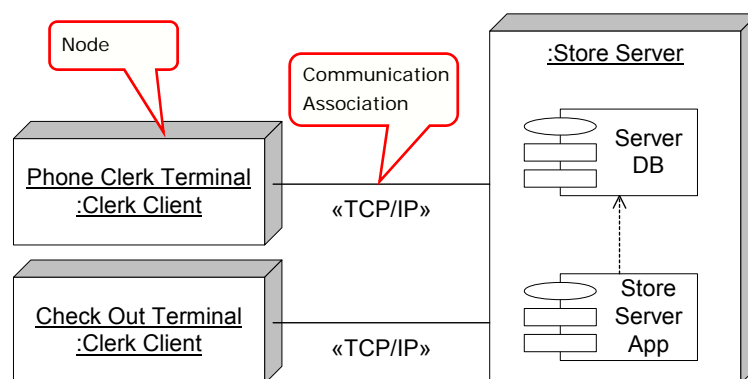
Component Diagram



Geant 4

107

Deployment Diagram



Geant 4

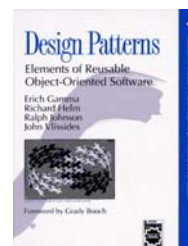
108

Introduction to Design Patterns

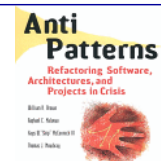
Geant 4

Design Patterns

Gamma, Helm, Johnson and Vlissides
Design Patterns,
Addison-Wesley 1995, ISBN 0-201-63361-2
(Gang-of-Four)



- Each design pattern names, explains and evaluates an important and recurring design in object-oriented systems
- A design pattern makes it easier to reuse successful designs and architectures
- Three categories of patterns
 - Creational
 - Structural
 - Behavioral



...also worthwhile reading!

Geant 4

List of design patterns

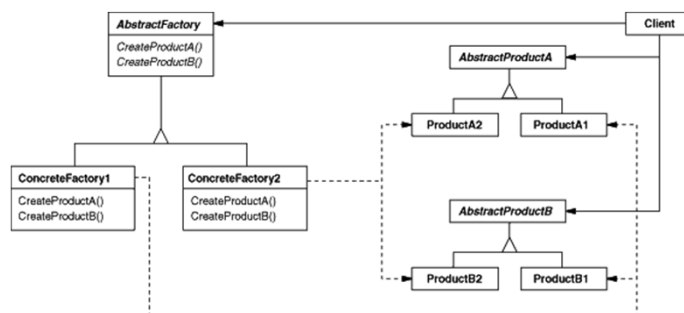
- Abstract Factory
- Adapter
- Bridge
- Builder
- Chain of Responsibility
- Command
- Composite
- Decorator
- Facade
- Factory Method
- Flyweight
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- Prototype
- Proxy
- Singleton
- State
- Strategy
- Template Method
- Visitor

Geant 4

111

Abstract Factory

Provide an interface for creating families of related or dependent objects without specifying their concrete classes

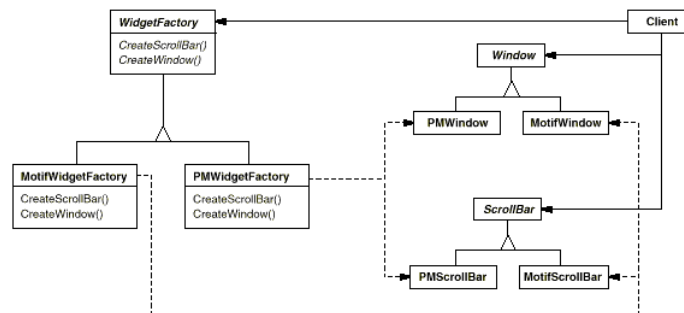


- AbstractFactory
 - declares an interface for operations that create product objects
- ConcreteFactory
 - implements the operations to create concrete product objects

Geant 4

112

Abstract Factory example

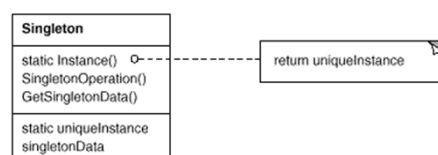


Geant 4

113

Singleton

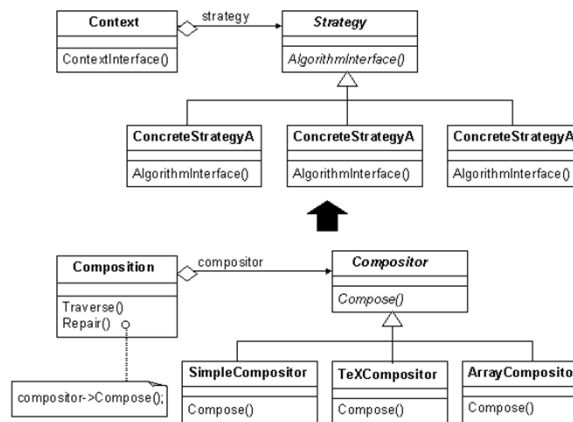
- Ensure a class only has one instance, and provide a global point of access to it
 - Many times need only one instance of an object
 - one file system
 - one print spooler
 - ...
- Singleton
 - defines a class-scoped `instance()` operation that lets clients access its unique instance
 - may be responsible for creating its own unique instance



Geant 4

114

Strategy



- Allow any one of a family of related algorithms to be easily substituted in a system
- Define a family of algorithms, encapsulate each one, and make them interchangeable
- Strategy lets the algorithm vary independently from clients that use it

Geant 4

115

Further reading

Geant 4

Books

+ Get a good mentor!

- There are many good C++, OOP, UML, OOAD books on the market

– *The following is just a personal selection*

Introductory C++ book

- S. B. Lippman, J. Lajoie, C++ primer, Addison-Wesley

Reference STL book

- N. Josuttis, The C++ Standard Library, A Tutorial and Reference, Addison-Wesley

More advanced C++ books

- S. Meyers, Effective C++, Addison-Wesley
- S. Meyers, More effective C++, Addison-Wesley
- S. Meyers, Effective STL, Addison-Wesley

UML books

- M. Fowler, UML distilled, Addison-Wesley
- G. Booch et al., The Unified Modeling Language, User Guide, Addison-Wesley

Basic OOAD books

- G. Booch, OO analysis and design, Addison-Wesley
- R. Martin, Designing OO C++ applications using the Booch method, Prentice Hall

Advanced design books

- E. Gamma et al., Design Patterns, Addison-Wesley
- John Lakos, Large-Scale C++ Software Design, Addison-Wesley

Hardcore design book

- A. Alexandrescu, Modern C++ design, Addison-Wesley

Geant 4

117

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

This course encompasses training material developed by several Geant4 members:

thanks to all of them!

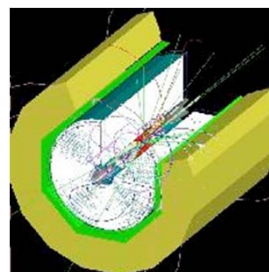
Geant 4

Geant4 User Application

Basic concepts to develop
a Geant4-based simulation application

Geant 4

From...



...to

Geant 4

120

Toolkit + User application



- You run your own simulation application
- Your application uses Geant4 tools
 - **You decide which Geant4 tools you want to use**
 - Your choice of tools should be appropriate to your experimental scenario
- Geant4 provides tools for particle transport
- Geant4 provides tools to model experimental environments
- Geant4 software design encompasses **tools** for user applications
 - To tell Geant4 kernel about your simulation configuration
 - To interact with Geant4 kernel itself

Geant 4

121

Interacting with Geant4

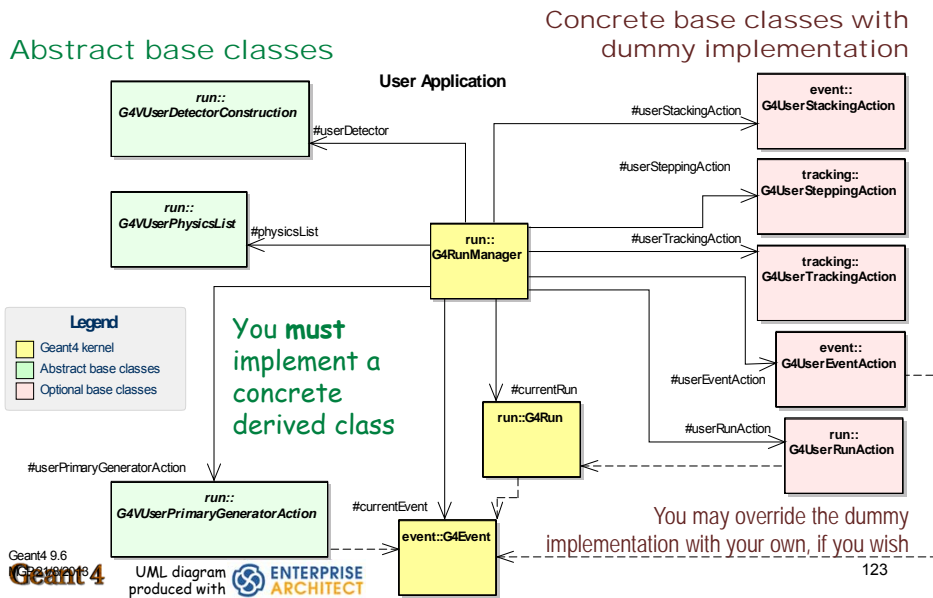
- You must tell Geant4 what **you only** know
 - Your **experimental scenario**
 - Geometry, materials, sensitive and passive elements
 - Primary particles, radiation environment
 - What **you want to happen** in the transport
 - Which particles you want to track
 - Which physics processes you are interested in (*and which options for modelling them you prefer*)
 - How precise your simulation you want to be
- You may want to **take some actions** during the simulation
 - e.g. at the beginning and the end of it (*initialize/store results*)
 - in the course of the transport (*e.g. inspect results, digitize signals*)

Geant 4

122

Interaction between user and Geant4

through **base classes** present in Geant4 kernel



Interaction with Geant4 kernel

- Geant4 tools for user interaction are **base classes**
 - You create **your own concrete class** derived from the base classes
 - Geant4 kernel handles your own derived classes transparently through their base class interface (**polymorphism**)
- Abstract base classes** for user interaction
 - User derived concrete classes are **mandatory**
- Concrete base classes** (with virtual dummy member functions) for user interaction
 - User derived classes are **optional**

User classes

Initialisation classes

Invoked during the execution loop

- *G4VUserDetectorConstruction*
- *G4VUserPhysicsList*

Action classes

Invoked during the execution loop

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

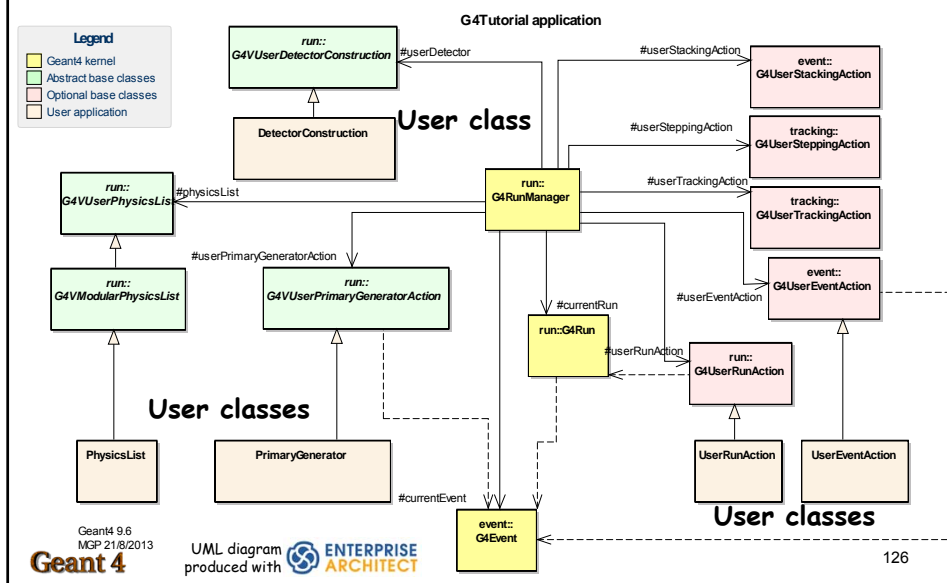
Mandatory classes:

- *G4VUserDetectorConstruction*
describe the experimental set-up
- *G4VUserPhysicsList*
select the physics you want to activate
- *G4VUserPrimaryGeneratorAction*
generate primary events

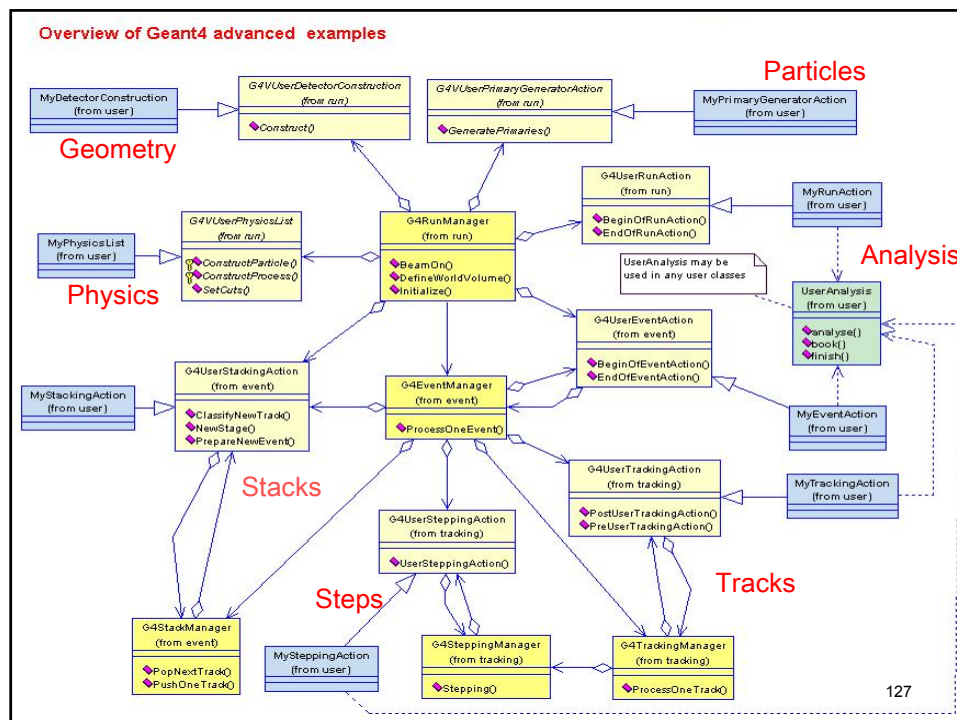
Geant 4

125

G4Tutorial application



126



Development of a Geant4 application

- The following slides provide an overview of the basic concepts of a Geant4 user application
- 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 in this course*
 - *(but we could in another course, if you are interested)*

The main function

- Geant4 does not provide the **main()** function
 - Geant4 is a toolkit!
 - The main() is part of the user application
- In his/her main(), the user **must**
 - instantiate **G4RunManager** (*or his/her own derived class*)
 - notify the G4RunManager mandatory user classes derived from
 - **G4VUserDetectorConstruction**
 - **G4VUserPhysicsList**
 - **G4VUserPrimaryGeneratorAction**
- The user **may** instantiate in his/her main() function
 - optional user action classes
 - visualisation manager, UI session

Geant 4

129

main()

```
{
...
// Instantiate the default run manager
G4RunManager* runManager = new G4RunManager;

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

// Instantiate mandatory user action classes and notify runManager
runManager->SetUserAction(new MyPrimaryGeneratorAction);

// Instantiate optional user action classes and notify runManager
MyEventAction* eventAction = new MyEventAction();
runManager->SetUserAction(eventAction);
MyRunAction* runAction = new MyRunAction();
runManager->SetUserAction(runAction);
...
}
```



Geant 4

130

- Derive your own **concrete class** from the ***G4VUserDetectorConstruction*** **abstract base class**

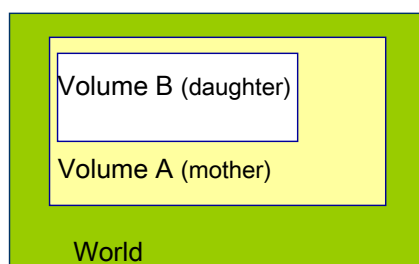
-

131



Define detector geometry

- Three conceptual layers
 - **G4VSolid** shape, size
 - **G4LogicalVolume** material, sensitivity, magnetic field, etc.
 - **G4VPhysicalVolume** position, rotation
- A unique physical volume (the **world** volume), which represents the experimental area, must exist and fully contain all other components



e.g.: Volume A is **mother** of Volume B

The mother must contain the daughter volume entirely

Geant 4

133

How to build the 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
```

How to build a volume inside the 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
```



Geant 4


134

How to define materials

Different kinds of materials can be defined

Isotopes
Elements
Molecules
Compounds and mixtures

```
PVPhysicalVolume* MyDetectorConstruction::Construct()
{
    ...
    Lead {
        a = 207.19*g/mole;
        density = 11.35*g/cm3;
        G4Material* lead = new G4Material(name="Pb", z=82., a, density);
    }
    Xenon gas {
        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);
    }
    ...
Geant 4 }
```



135

How to define a compound material

For example, a **scintillator** consisting of Hydrogen and Carbon:

```
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);
```

Geant 4

136

Select physics processes

- Geant4 does not have any default particles or processes
- Derive your own **concrete class** from the **G4VUserPhysicsList** **abstract base class**
 - define all necessary particles
 - define all necessary processes and assign them to proper particles
 - define production thresholds (in terms of range)
- Pure virtual methods of G4VUserPhysicsList

```
ConstructParticles()  
ConstructProcesses()  
SetCuts()
```



to be implemented by the user in
his/her concrete derived class

Geant 4

137

PhysicsList: particles and cuts

```
MyPhysicsList :: MyPhysicsList(): G4VUserPhysicsList()  
{  
    defaultCutValue = 1.0*cm;  
}  
  
void MyPhysicsList :: ConstructParticles()  
{  
    G4Electron::ElectronDefinition();  
    G4Positron::PositronDefinition();  
    G4Gamma::GammaDefinition();  
}  
  
void MyPhysicsList :: SetCuts()  
{  
    SetCutsWithDefault();  
}
```

← Define **production thresholds**
(the same for all particles)

← Define the **particles**
involved in the simulation

← Set the **production threshold**

Geant 4

138

PhysicsList: more about cuts

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



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

The user can define
different cuts for
different particles
or
different regions

Geant 4

139

Physics List: processes

```
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 G4eMultipleScattering(), -1, 1,1);
        pManager->AddProcess(new G4eIonisation(), -1, 2,2);
        pManager->AddProcess(new G4eBremsstrahlung(), -1,-1,3);
    }
    else if (particleName == "e+")
    {
        pManager->AddProcess(new G4eMultipleScattering(), -1, 1,1);
        pManager->AddProcess(new G4eIonisation(), -1, 2,2);
        pManager->AddProcess(new G4eBremsstrahlung(), -1,-1,3);
        pManager->AddProcess(new G4eplusAnnihilation(), 0,-1,4);
    }
}
```

Select physics processes to be
activated for each particle type

Geant 4

140

Physics List: processes

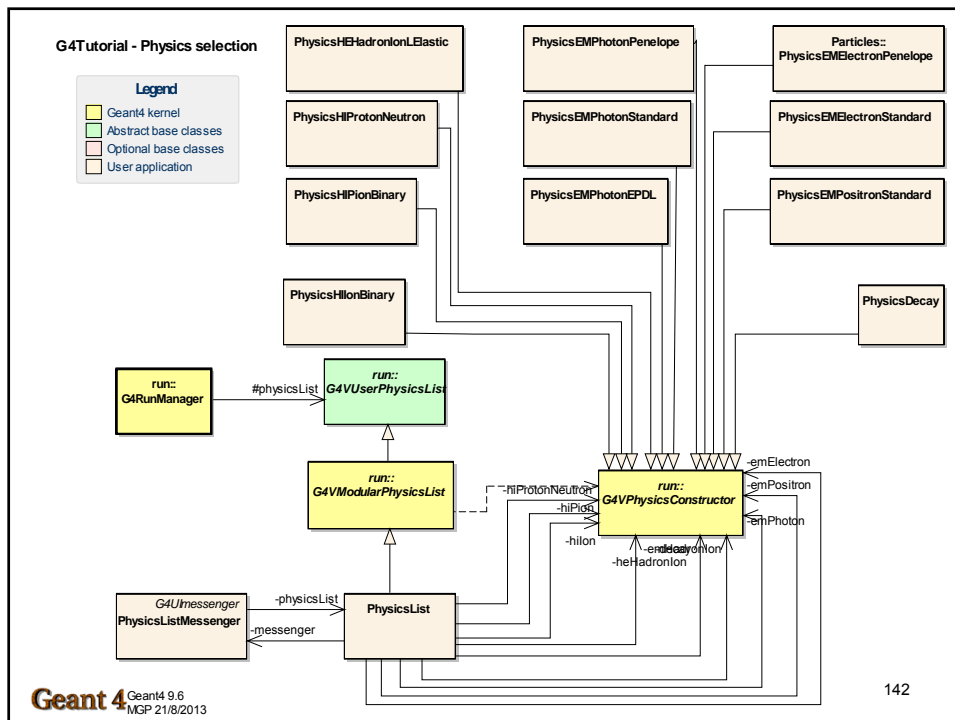
As in
examples/advanced/amsEcal

```
void MyPhysicsList :: ConstructParticles()
{
    if (particleName == "gamma") {
        // photon
        pManager->AddDiscreteProcess(new G4PhotoElectricEffect);
        pManager->AddDiscreteProcess(new G4ComptonScattering);
        pManager->AddDiscreteProcess(new G4GammaConversion);
    } else if (particleName == "e-") {
        // electron
        pManager->AddProcess(new G4eMultipleScattering, -1, 1, 1);
        pManager->AddProcess(new G4eIonisation, -1, 2, 2);
        pManager->AddProcess(new G4eBremsstrahlung(), -1, 3, 3);
    } else if (particleName == "e+") {
        // positron
        pManager->AddProcess(new G4eMultipleScattering, -1, 1, 1);
        pManager->AddProcess(new G4eIonisation, -1, 2, 2);
        pManager->AddProcess(new G4eBremsstrahlung(), -1, 3, 3);
        pManager->AddProcess(new G4eplusAnnihilation, 0, -1, 4);
    }
}
```

Select physics processes to be activated for each particle type

Geant 4

141



142

Pre-packaged PhysicsLists

- QGSP_INCLXX
- G4VHadronPhysics
- LHEP_EMV
- G4PhysListFactory
- Shielding
- G4GenericPhysicsList
- HadronPhysicsQGS_BIC
- HadronPhysicsQGSP_BERT_CHIPS
- HadronPhysicsQGSP_BIC_HP
- HadronPhysicsQGSP_FTFP_BERT_95
- QGSC_CHIPS
- QGSP_BERT_NOLEP
- FTFP_BERT_EMX
- HadronPhysicsQGSP_BIC
- HadronPhysicsLHEP
- QGSP_BIC
- LHEP
- FTF_BIC
- HadronPhysicsQGSP_INCLXX
- QGSP_BERT_EMX
- QGSP_BIC_EMY
- HadronPhysicsQGSP_BERT_95
- LBE
- QGSP_BERT_95
- HadronPhysicsQGSP_FTFP_BERT
- HadronPhysicsFTFP_BERT_TRV
- HadronPhysicsCHIPS
- QBBC
- HadronPhysicsQGSP_BERT_TRV
- QGSP_FTFP_BERT_95
- HadronPhysicsQGSP_BERT_NOLEP
- HadronPhysicsFTF_BIC
- HadronPhysicsCHIPS_HP
- QGSP_FTFP_BERT_95XS
- CHIPS_HP
- HadronPhysicsFTFP_BERT_HP
- FTFP_BERT_HP
- CHIPS
- FTFP_BERT_TRV
- HadronPhysicsQGSC_BERT
- G4PhysListUtil
- HadronPhysicsQGSP
- QGSP
- HadronPhysicsFTFP_BERT
- QGSP_BERT_TRV
- FTFP_BERT_EMV
- HadronPhysicsQGSC_CHIPS
- HadronPhysicsQGSP_BERT
- G4HadronInelasticQBBC
- QGSP_BERT_EMV
- HadronPhysicsShielding
- QGSP_FTFP_BERT
- QGS_BIC
- QGSC_BERT
- FTFP_BERT
- QGSP_QEL
- HadronPhysicsQGSP_BERT_HP
- QGSP_BERT_CHIPS
- QGSP_BERT_HP
- QGSP_BERT_95XS
- QGSP_BIC_HP

Geant 4

143

Pre-packaged PhysicsLists

The Geant4 toolkit encompasses a number of pre-assembled PhysicsLists in `geant4/source/physics_lists/`

...don't expect to be exempt from understanding Geant4 physics in depth!

- QGSP_INCLXX
- LHEP_EMV
- G4PhysListFactory
- Shielding
- G4GenericPhysicsList
- HadronPhysicsQGS_BIC
- HadronPhysicsQGSP_BERT_CHIPS
- HadronPhysicsQGSP_BIC_HP
- HadronPhysicsQGSP_FTFP_BERT_95
- QGSC_CHIPS
- QGSP_BERT_NOLEP
- FTFP_BERT_EMX
- HadronPhysicsQGSP_BIC
- HadronPhysicsLHEP
- QGSP_BIC
- LHEP
- FTF_BIC
- HadronPhysicsQGSP_INCLXX
- QGSP_BERT_EMX
- QGSP_BIC_EMY144
- HadronPhysicsQGSP_BERT_95
- LBE
- FTFP_BERT_TRV
- HadronPhysicsQGSC_BERT
- G4PhysListUtil
- HadronPhysicsQGSP
- HadronPhysicsFTFP_BERT
- QGSP_BERT_95
- HadronPhysicsQGSP_FTFP_BERT
- HadronPhysicsFTFP_BERT_TRV
- HadronPhysicsCHIPS
- QBBC
- HadronPhysicsQGSP_BERT_TRV
- QGSP_FTFP_BERT_95
- HadronPhysicsQGSP_BERT_NOLEP
- HadronPhysicsFTF_BIC
- HadronPhysicsCHIPS_HP
- QGSP_FTFP_BERT_95XS
- CHIPS_HP
- HadronPhysicsFTFP_BERT_HP
- FTFP_BERT_HP
- CHIPS
- QGSP_BERT_TRV
- FTFP_BERT_EMV
- HadronPhysicsQGSC_CHIPS
- HadronPhysicsQGSP_BERT
- G4HadronInelasticQBBC
- QGSP_BERT_EMV
- HadronPhysicsShielding
- QGSP_FTFP_BERT
- QGS_BIC
- QGSC_BERT
- FTFP_BERT
- QGSP_QEL
- HadronPhysicsQGSP_BERT_HP
- QGSP_BERT_CHIPS
- QGSP_BERT_HP
- QGSP_BERT_95XS
- QGSP_BIC_HP

144

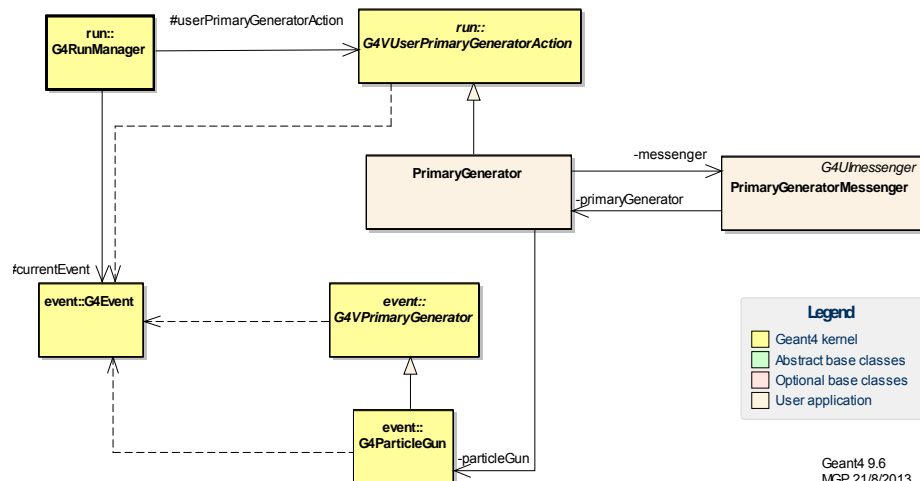
Primary events

- Derive your own **concrete class** from the ***G4VUserPrimaryGeneratorAction*** **abstract base class**
- Define primary particles providing:
 - Particle type
 - Initial position
 - Initial direction
 - Initial energy
- Implement the virtual member function **GeneratePrimaries()**

Geant 4

145

G4Tutorial - Primary generator



Geant 4

146

Generate primary particles

```
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);
}
}
```

Geant 4

147

Optional User Action classes

- Five **concrete base classes** whose **virtual member functions** the user may override to gain control of the simulation at various stages
 - G4User**Run**Action
 - G4User**Event**Action
 - G4User**Tracking**Action
 - G4User**Stacking**Action
 - G4User**Stepping**Action
- Each member function of the base classes has a dummy implementation
 - Empty implementation: does nothing
- The user may implement the member functions he desires in his/her derived classes
- Objects of user action classes must be registered with G4RunManager

Geant 4

148

Optional User Action classes

G4UserRunAction

- **BeginOfRunAction**(const G4Run*)
 - For example: book histograms
- **EndOfRunAction**(const G4Run*)
 - For example: store histograms

G4UserEventAction

- **BeginOfEventAction**(const G4Event*)
 - For example: perform and event selection
- **EndOfEventAction**(const G4Event*)
 - For example: analyse the event

G4UserTrackingAction

- **PreUserTrackingAction**(const G4Track*)
 - For example: decide whether a trajectory should be stored or not
- **PostUserTrackingAction**(const G4Track*)

Geant 4

149

Optional User Action classes

G4UserSteppingAction

- **UserSteppingAction**(const G4Step*)
 - For example: kill, suspend, postpone the track
 - For example: draw the step

G4UserStackingAction

- **PrepareNewEvent**()
 - For example: reset priority control
- **ClassifyNewTrack**(const G4Track*)
 - Invoked every time a new track is pushed
 - For example: classify a new track (priority control)
 - Urgent, Waiting, PostponeToNextEvent, Kill
- **NewStage**()
 - Invoked when the Urgent stack becomes empty
 - For example: change the classification criteria
 - For example: event filtering (event abortion)

Geant 4

150

Select (G)UI and visualisation

- In your **main()**, taking into account your computer environment, **instantiate a `G4UIsession` concrete class** provided by Geant4 and invoke its **`sessionStart()`** method
- In your **main()**, taking into account your computer environment, **instantiate a `G4VisExecutive`** and invoke its **`initialize()`** method
- Geant4 provides interfaces to various graphics drivers:
 - DAWN (*Fukui renderer*)
 - WIRED
 - RayTracer (*ray tracing by Geant4 tracking*)
 - OPACS
 - OpenGL
 - OpenInventor
 - VRML
 - ...
- Geant4 provides:
 - G4UITerminal
 - csh or tcsh like character terminal
 - G4GAG
 - tcl/tk or Java PVM based GUI
 - G4Wo
 - Opacs
 - G4UIBatch
 - batch job with macro file
 - ...

Geant 4

151

Recipe for novice users

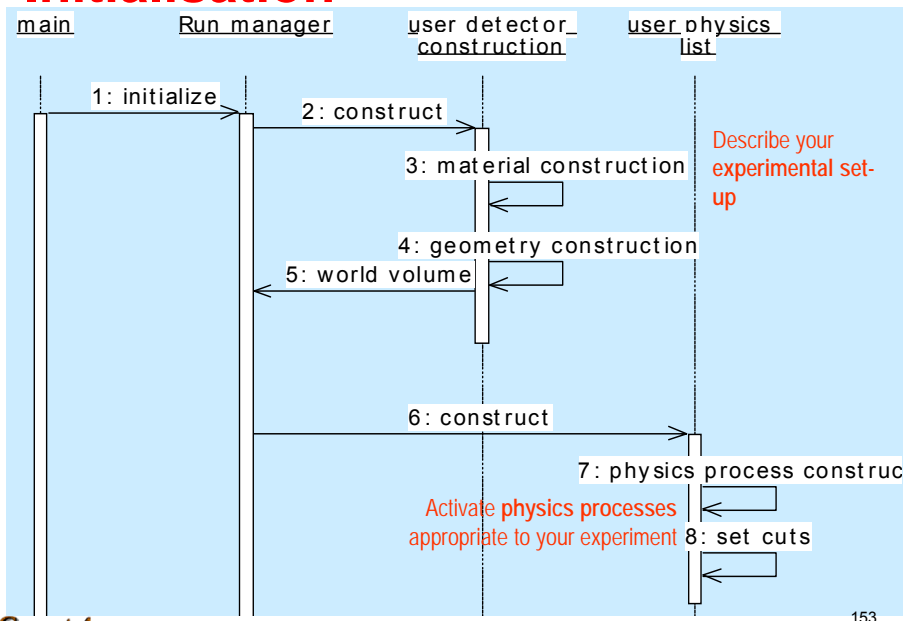
Experienced users may do much more, but the conceptual process is still the same...

- Design diagram as in generic Geant4 Advanced Example
- Create your derived mandatory user classes
 - `MyDetectorConstruction`
 - `MyPhysicsList`
 - `MyPrimaryGeneratorAction`
- Optionally create your derived user action classes
 - `MyUserRunAction`
 - `MyUserEventAction`
 - `MyUserTrackingAction`
 - `MyUserStackingAction`
 - `MyUserSteppingAction`
- Create your **`main()`**
 - Instantiate `G4RunManager` or your own derived `MyRunManager`
 - Notify the `RunManager` of your mandatory and optional user classes
 - Optionally initialize your favourite User Interface and Visualization
- **That's all!**

Geant 4

152

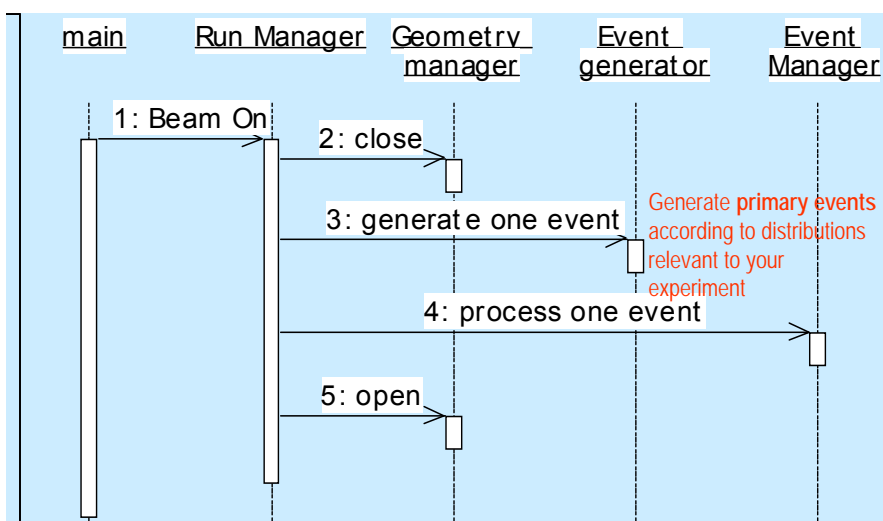
Initialisation



Geant 4

153

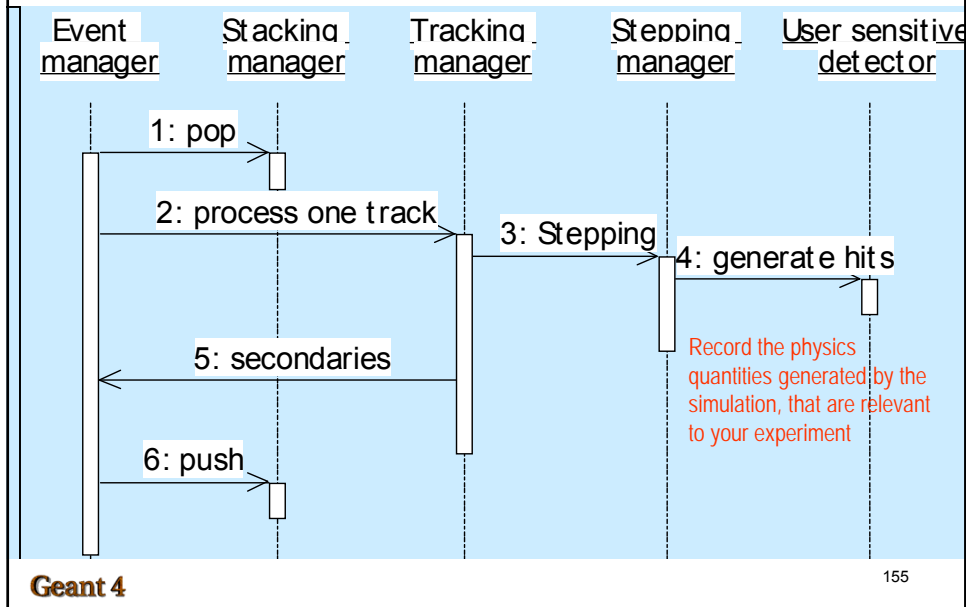
Beam On



Geant 4

154

Event processing



Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

This course encompasses training material developed by several Geant4 members:

thanks to all of them!

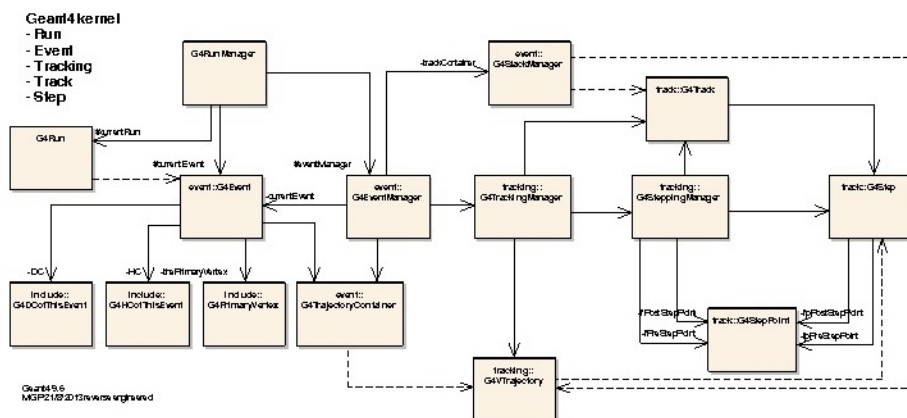
Geant 4

Geant4 Kernel

Geant 4

Some of the terminologies

- Run, Event, Track, Step, StepPoint
- Track \leftrightarrow trajectory, step \leftrightarrow trajectory point
- Process, Hit,



Geant4 Kernel

Kernel I - M.Asai (SLAC)

158

Run in Geant4

- As an analogy of the real experiment, a run of Geant4 starts with “Beam On”.
- Within a run, the user cannot change
 - detector setup
 - settings of physics processes
- Conceptually, a run is a collection of events which share the same detector and physics conditions.
 - A run consists of one event loop.
- At the beginning of a run, geometry is optimized for navigation and cross-section tables are calculated according to materials appear in the geometry and the cut-off values defined.
- **G4RunManager** class manages processing a run, a run is represented by **G4Run** class or a user-defined class derived from G4Run.
 - A run class may have a summary results of the run.
- **G4UserRunAction** is the optional user hook.

Geant4 Kernel

159

Event in Geant4

- An event is the basic unit of simulation in Geant4.
- At beginning of processing, primary tracks are generated. These primary tracks are pushed into a stack.
- A track is popped up from the stack one by one and “**tracked**”. Resulting secondary tracks are pushed into the stack.
 - This “tracking” lasts as long as the stack has a track.
- When the stack becomes empty, processing of one event is over.
- **G4Event** class represents an event. It has following objects at the end of its (successful) processing.
 - List of primary vertices and particles (as input)
 - Hits and Trajectory collections (as output)
- **G4EventManager** class manages processing an event.
- **G4UserEventAction** is the optional user hook.

Geant4 Kernel

160

Track in Geant4

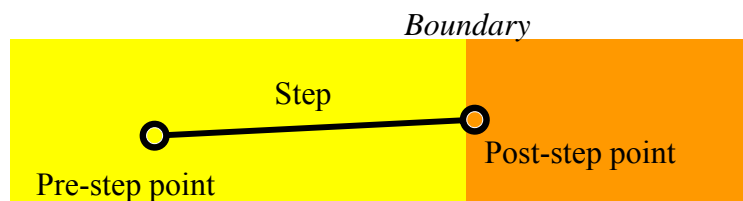
- Track is a **snapshot** of a particle.
 - It has physical quantities of **current instance** only. It does not record previous quantities.
 - Step is a “delta” information to a track. Track is not a collection of steps. Instead, a track is being updated by steps.
- Track object is deleted when
 - it goes out of the world volume,
 - it disappears (by e.g. decay, inelastic scattering),
 - it goes down to zero kinetic energy and no “AtRest” additional process is required, or
 - the user decides to kill it artificially.
- **No track object persists at the end of event.**
 - For the record of tracks, use trajectory class objects.
- **G4TrackingManager** manages processing a track, a track is represented by **G4Track** class.
- **G4UserTrackingAction** is the optional user hook.

Geant4

161

Step in Geant4

- Step has two points and also “delta” information of a particle (energy loss on the step, time-of-flight spent by the step, etc.).
- Each point knows the volume (and material). In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it **logically belongs to the next volume**.
 - Because one step knows materials of two volumes, boundary processes such as transition radiation or refraction could be simulated.
- **G4SteppingManager** class manages processing a step, a step is represented by **G4Step** class.
- **G4UserSteppingAction** is the optional user hook.



Geant4

162

Trajectory and Trajectory Point

- Track does not keep its trace. No track object persists at the end of event.
- **G4Trajectory** is the class which copies some of G4Track information.
G4TrajectoryPoint is the class which copies some of G4Step information.
 - G4Trajectory has a vector of G4TrajectoryPoint.
 - At the end of event processing, G4Event has a collection of G4Trajectory objects.
 - /tracking/storeTrajectory must be set to 1.
- Keep in mind the distinction:
 - $G4Track \leftrightarrow G4Trajectory$, $G4Step \leftrightarrow G4TrajectoryPoint$
- Given G4Trajectory and G4TrajectoryPoint objects persist till the end of an event, one should be careful not to store too many trajectories:
 - e.g. avoid for high energy EM shower tracks.
- G4Trajectory and G4TrajectoryPoint store only the minimum information
 - One can create one's own trajectory / trajectory point classes to store the required information. G4VTrajectory and G4VTrajectoryPoint are the base classes.

Particle in Geant4

- A particle in Geant4 is represented by three layers of classes.
- **G4Track**
 - Position, geometrical information, etc.
 - This is a class representing a particle to be tracked.
- **G4DynamicParticle**
 - "Dynamic" physical properties of a particle, such as momentum, energy, spin, etc.
 - Each G4Track object has its own and unique G4DynamicParticle object.
 - This is a class representing an individual particle.
- **G4ParticleDefinition**
 - "Static" properties of a particle, such as charge, mass, life time, decay channels, etc.
 - G4ProcessManager which describes processes involving to the particle
 - All G4DynamicParticle objects of same kind of particle share the same G4ParticleDefinition.

Tracking and Process

- Geant4 tracking is general.
 - It is independent of
 - the particle type
 - the physics processes involving to a particle
 - It gives the chance to all processes
 - to contribute to determining the step length
 - to contribute any possible changes in physical quantities of the track
 - to generate secondary particles
 - to suggest changes in the state of the track
 - e.g. to suspend, postpone or kill it.

Process in Geant4

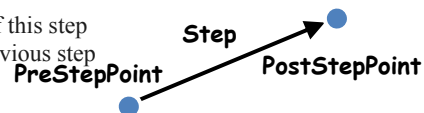
- In Geant4, particle transportation is a process as well, by which a particle interacts with geometrical volume boundaries and field of any kind.
 - Because of this, shower parameterization process can take over from the ordinary transportation without modifying the transportation process.
- Each particle has its own list of applicable processes. At each step, all processes listed are invoked to get proposed physical interaction lengths.
- The process which requires the shortest interaction length (in space-time) limits the step.
- Each process has one or combination of the following natures.
 - AtRest
 - e.g. muon decay at rest
 - AlongStep (a.k.a. continuous process)
 - e.g. Cerenkov process
 - PostStep (a.k.a. discrete process)
 - e.g. decay on flight

Track Status

- At the end of each step, according to the processes involved, the state of a track may be changed.
 - The user can also change the status in **UserSteppingAction**
 - Status shown in **brown** are artificial, i.e. Geant4 kernel won't set them, but the user can set
 - fAlive**
 - continue the tracking
 - fStopButAlive**
 - the track has come to zero kinetic energy, but still AtRest process to occur
 - fStopAndKill**
 - the track has lost its identity because it has decayed, interacted or gone beyond the world boundary
 - secondaries will be pushed to the stack
 - fKillTrackAndSecondaries**
 - Kill the current track and also associated secondaries.
 - fSuspend**
 - suspend processing of the current track and push it and its secondaries to the stack
 - fPostponeToNextEvent**
 - postpone processing of the current track to the next event
 - secondaries are still being processed within the current event.

Step Status

- Step status is attached to G4StepPoint to indicate why that particular step was determined
 - Use "**PostStepPoint**" to get the status of this step
 - "**PreStepPoint**" has the status of the previous step
 - fWorldBoundary**
 - step reached the world boundary
 - fGeomBoundary**
 - step is limited by a volume boundary except the world
 - fAtRestDoItProc, fAlongStepDoItProc, fPostStepDoItProc**
 - step is limited by a AtRest, AlongStep or PostStep process
 - fUserDefinedLimit**
 - step is limited by the user Step limit
 - fExclusivelyForcedProc**
 - step is limited by an exclusively forced (e.g. shower parameterization) process
 - fUndefined**
 - Step not defined yet
- If the **first step in a volume** is to be identified, pick **fGeomBoundary** status in **PreStepPoint**
- If a **step getting out of a volume** is to be identified, pick **fGeomBoundary** status in **PostStepPoint**

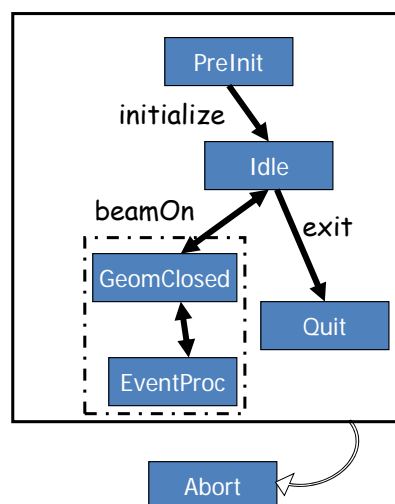


Extraction of useful information

- Given geometry, physics and primary track generation, Geant4 does proper physics simulation “silently”
 - the user has to add a bit of code to **extract useful information**
- There are two ways:
 - Use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)
 - the user has an access to almost all information
 - straight-forward, but do-it-yourself
 - Use Geant4 scoring functionality
 - assign **G4VSensitiveDetector** to a volume
 - Hits collection** is automatically stored in G4Event object, and automatically accumulated if **user-defined Run** object is used
 - use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary

Geant4 as a State Machine

- Geant4 has six application states:
 - G4State_PreInit
 - Material, Geometry, Particle and/or Physics Process need to be initialized/defined
 - G4State_Idle
 - ready to start a run
 - G4State_GeomClosed
 - geometry is optimized and ready to process an event
 - G4State_EventProc
 - an event is being processed
 - G4State_Quit
 - (Normal) termination
 - G4State_Abort
 - a fatal exception occurred and the program is aborting



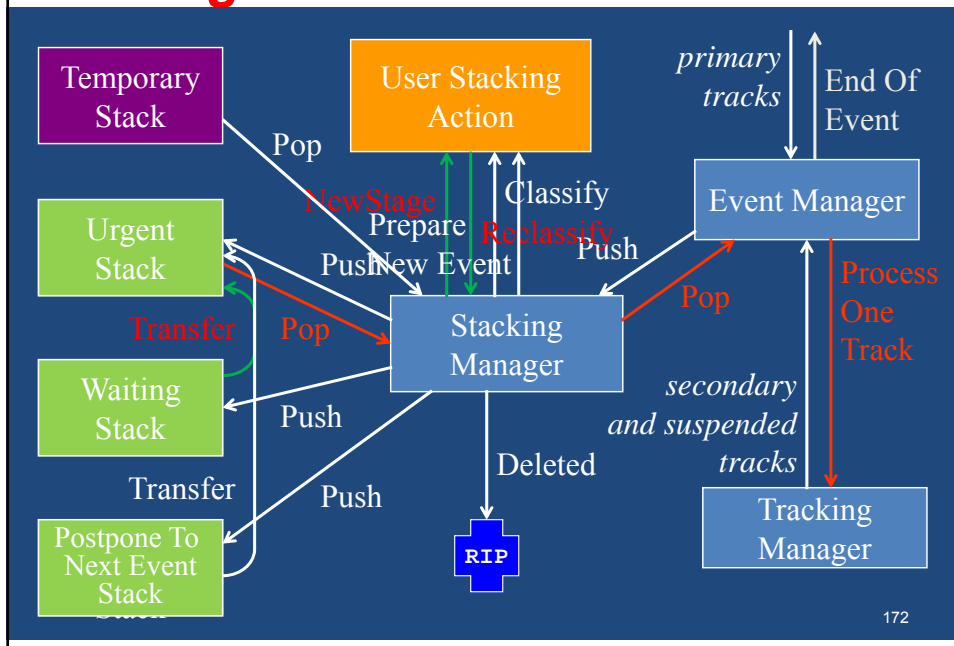
Track Stacks in Geant4

- By default, Geant4 has three track stacks:
 - "Urgent", "Waiting" and "PostponeToNextEvent"
 - Each stack is a simple "last-in-first-out" stack
 - User can arbitrary increase the number of stacks
- ClassifyNewTrack()** method of UserStackingAction decides which stack each newly storing track to be stacked (or to be killed)
 - By default, all tracks go to Urgent stack
- A Track is popped up **only from Urgent stack**
- Once Urgent stack becomes empty, all tracks in Waiting stack are transferred to Urgent stack
 - And **NewStage()** method of UserStackingAction is invoked
- Utilizing more than one stacks, user can control the priorities of processing tracks without paying the overhead of "scanning the highest priority track"
 - Proper selection/abortion of tracks/events with well designed stack management provides significant efficiency increase of the entire simulation

Geant4

171

Stacking Mechanism



172

Tips of stacking manipulations

- Classify all secondaries as **fWaiting** until **Reclassify()** method is invoked
 - One can simulate all primaries before any secondaries
- Classify secondary tracks below a certain energy as **fWaiting** until **Reclassify()** method is invoked
 - One can roughly simulate the event before being bothered by low energy EM showers
- **Suspend** a track on its fly. Then this track and all of already generated secondaries are pushed to the stack
 - Given a stack is "**last-in-first-out**", secondaries are popped out prior to the original suspended track
 - Quite effective for Cerenkov radiation
- **Suspend** all tracks that are **leaving from a region**, and classify these suspended tracks as **fWaiting** until **Reclassify()** method is invoked
 - One can simulate all tracks in this region prior to other regions
 - Note that some back splash tracks may come back into this region later

Geant4

173

Attaching user information

- Abstract classes:
 - The user can use his/her own class derived from the provided base class
 - **G4Run**, **G4VHit**, **G4VDigit**, **G4VTrajectory**, **G4VTrajectoryPoint**
- Concrete classes:
 - The user can attach a user information class object
 - G4Event - **G4VUserEventInformation**
 - G4Track - **G4VUserTrackInformation**
 - G4PrimaryVertex - **G4VUserPrimaryVertexInformation**
 - G4PrimaryParticle - **G4VUserPrimaryParticleInformation**
 - G4Region - **G4VUserRegionInformation**
 - User information class object is deleted when associated Geant4 class object is deleted

Geant4

174

Bookkeeping Issues

- Connection from G4PrimaryParticle to G4Track

`G4int G4PrimaryParticle::GetTrackID()`

- Returns the track ID if this primary particle had been converted into G4Track, otherwise -1
 - Both for primaries and pre-assigned decay products

- Connection from G4Track to G4PrimaryParticle

`G4PrimaryParticle* G4DynamicParticle::GetPrimaryParticle()`

- Returns the pointer of G4PrimaryParticle object if this track was defined as a primary or a pre-assigned decay product, otherwise null

- G4VUserPrimaryVertexInformation, G4VUserPrimaryParticleInformation and G4VUserTrackInformation may be used for storing additional information

- Information in UserTrackInformation should be then copied to user-defined trajectory class, so that such information is kept until the end of the event

Geant4

175

Generating Primary Particles

- Each Geant4 Event starts with generation of one or multiple primary particles
- It is up to the user to define primary particle properties
 - Particle type, e.g. electron, gamma, ion
 - Initial kinetics, e.g. energy, momentum, origin and direction
 - Additional properties, e.g. polarization
- These properties can be divided into a **primary vertex**: starting point in space and time
- **Primary particle**: initial momentum, polarization, PDG code, list of daughters for decay chains
- A primary particle can be a particle which can not usually be tracked by Geant4

Geant4

176

The PrimaryGenerator

- A primary generator is a class derived from **G4VPrimaryGenerator** which implements a **GeneratePrimaryVertex()** method
 - In this method the primary vertex and the primary particle are added to a Geant4 Event

- Often it is practical to use an existing generator:

- G4HEPEvtInterface
- G4HEPMCInterface
- G4GeneralParticleSource
- G4ParticleGun

Examples of experiment-specific generators. Control via text files

More general purpose. For volume and surface sources. Also for beams.

Can be used to produce a beam of particles

Geant 4

177

PrimaryGeneratorAction

- Mandatory user action which **controls** the generation of primary particles
- **It should not generate primaries itself.** The primary generator does this.
- Implement your particle “shot”, “rail”, or machine gun here. It can also be a particle bomb if you like.
 - By using e.g. the G4ParticleGun
 - Repeatedly for a single event
 - Sampling particle type and direction randomly
 - Or using one of the other event generators

Geant 4

178

PrimaryGeneratorAction

- Inherits from G4VUserPrimaryGeneratorAction
- User should override GeneratePrimaries for particle generation

```
PrimaryGeneratorAction::PrimaryGeneratorAction(const G4String &
parName, G4double energy, G4ThreeVector pos, G4ThreeVector
momDirection){

    const G4int nParticles = 1;
    fParticleGun = new G4ParticleGun(nParticles);
    G4ParticleTable* parTable = G4ParticleTable::GetParticleTable();
    G4ParticleDefinition* parDefinition = parTable-
>FindParticle(parName);
    fParticleGun->SetParticleDefinition(parDefinition);
    fParticleGun->SetParticleEnergy(energy);
    fParticleGun->SetParticlePosition(pos);
    fParticleGun->SetParticleMomentumDirection(momDirection);
}
```

The primary generator

Geant 4

179

Class PrimaryGeneratorAction

- Inherits from G4VUserPrimaryGeneratorAction
- User should override GeneratePrimaries for particle generation

```
PrimaryGeneratorAction::GeneratePrimaries(G4Event* evt){

    //some additional random sampling here

    fParticleGun->GeneratePrimaryVertex(evt);

}
```

Geant 4

180

Alternative Method: GPS

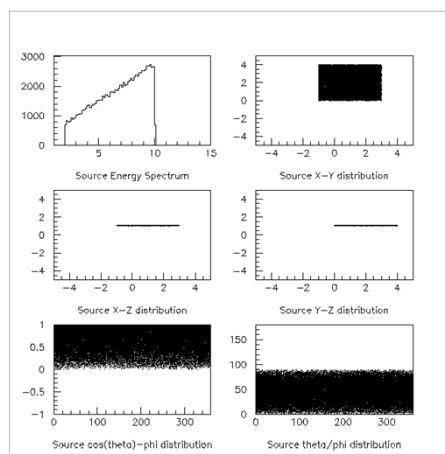
- The General Particle Source (GPS)¹ provides a high-level interface to G4ParticleGun, mainly using macros
 - Define source types: point, beam, plane, surface, volume
 - Define angular distribution: isotropic, cosine-law, planar, 1d/2d beams, user defined
 - Define energy distribution: mono-energetic, linear, power-law, exponential, gaussian, Bremsstrahlung-spectrum, black body spectrum, cosmic diffuse gamma ray, user defined
 - Angular and energy distributions can be interpolated from histogrammed distributions
- To use simply replace G4ParticleGun in PrimaryGeneratorAction with G4GeneralParticleSource

Geant 4

181

Alternative Method: GPS

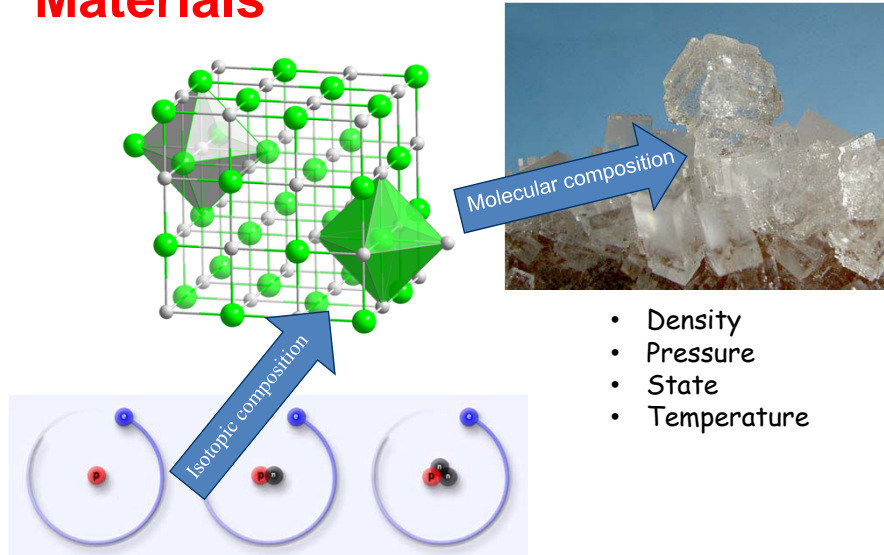
- For full documentation:
http://reat.space.qinetiq.com/gps/new_gps_sum_files/gps_sum.htm



Geant 4

182

Materials



- Density
- Pressure
- State
- Temperature

Geant 4

Images: www.wikipedia.org

183

Defining Materials

- Materials define how a particle interacts with the different components of an experimental setup/geometry.
- Material composition, density, temperature, pressure and state influence cross-section for physics processes.
- Geant4 offers multiple methods of defining a G4Material:
 - From existing materials in database, e.g. G4NISTMaterial
 - By molecular composition
 - By element composition
 - By element and isotope composition

Geant 4

184

System of Units

- The physical properties which define a material have units attached like g/cm³, K, g/mol...
- Geant4 has no default unit and does not deduce the unit for you – **therefore if you give a number you should also provide a unit by multiplying it to the number.**

```
G4double length = 1.0*m;  
G4double density = 100*g/cm3  
G4double magnetic_field = 2.*Tesla;
```

- If no unit is given Geant4 uses an internal unit. This is strongly discouraged.
- Almost all commonly used units are available
- You can define your own units

Geant 4

185

System of Units

- To retrieve a value with a specific unit you divide by this unit

```
G4double length = 1.0*m;  
G4cout<<"Length: "<<length/km<<" km"<<G4endl; // 0.001  
G4cout<<"Length: "<<length/cm<<" cm"<<G4endl; // 100  
G4cout<<"Length: "<<length/nm<<" nm"<<G4endl; // 1e9
```

Quite a few analysis' have shown mysterious results because internal units were output

- You can also let Geant4 choose the most appropriate unit to output in by giving it a unit category:

```
G4double dE= 100*keV;  
G4cout<<G4BestUnit(dE, "Energy")<<G4endl;  
//will print dE in MeV, keV, eV depending on value
```

Geant 4

186

System of Units

- Units are defined in CLHEP and are based on these basic units:
 - Millimeters (mm), nano-seconds (ns), Mega-electronvolts (MeV), positron charge (eplus), Kelvin (K), amount of substance (mole), luminosity (candela), radian (radian) and steradian (steradian)
- Defining an own unit can use these basics ones or already existing derived units:

```
G4UnitDefinition(name, symbol, category, value)
G4UnitDefinition("grammspercm2", "g/cm2", "MassThickness", g/cm2);
```

- To list units available in the G4UnitsTable:

```
G4UnitsDefinition::PrintUnitsTable(); //from code
```

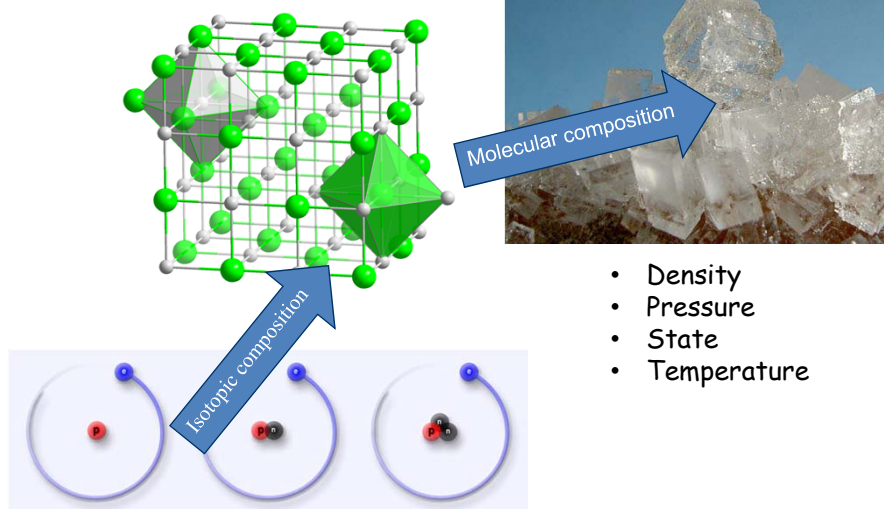
```
Idle> /units/list #from UI
```

While you certainly could define a "Foo" or a "Bar" unit, consider if you really need a new unit.

Geant 4

187

Defining Materials



- Density
- Pressure
- State
- Temperature

Geant 4

188

Images: www.wikipedia.org

Defining Materials using Database

- Geant4 includes a database of “common” materials, as defined by NIST

```
G4NistManager* matMan = G4NistManager::Instance();  
  
G4Material* H2O = matMan ->FindOrBuildMaterial("G4_WATER");  
G4Material* Air = matMan ->FindOrBuildMaterial("G4_AIR");
```

- To list available materials:

```
G4NistManager::Instance()->ListMaterials("all"); //from code
```

```
Idle> /material/nist/listMaterials all #from UI
```

Geant 4

189

Defining new Materials

- **In the “real” world materials consist of elements, molecules, or are mixtures of materials or elements.**
- Elements can be pure or have a specific isotopic composition
- Molecules are made from multiple elements
- **Elements define the microscopic properties** (cross-section of atoms, number of nucleons, shell energies) **of a material with macroscopic properties** (pressure, density, state, radiation length, absorption length)
- Geant4 reflects these concepts

Geant 4

190

Building an Element from Isotopes

- Most fine-grained material definition possible in Geant4

```
G4String name, symbol;
G4int numIsotopes, z, n;
G4double abundance;
```

```
G4Element* elU = new G4Element(name="Natural Uranium", symbol="U",
numIsotopes=3);
```

```
G4Isotope U238 = new G4Isotope(name="U238", z=92, n=146);
elU->AddIsotope(U238, abundance=0.99274);
```

```
G4Isotope U235= new G4Isotope(name="U235", z=92, n=143);
elU->AddIsotope(U235, abundance=0.0072);
```

```
G4Isotope U234= new G4Isotope(name="U234", z=92, n=142);
elU->AddIsotope(U234, abundance=0.00004);
```

Geant4

Geant4 will tell you if these fractions do not add up to one – but it will not crash

Elements with natural isotopic abundances

- No need to define isotope constituents

```
G4double z, a, density;
G4String name, symbol;
G4int ncomponents, natoms;
```

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

```
a = 16.00*g/mole;
G4Element* elO = new G4Element(name="Oxygen", symbol="O", z= 8.,
a);
```

Geant4

If you are not planning to use hadronic interactions such as spallation or fission, defining elements this way is usually sufficient!

192

Elements from Database

- Geant4 includes a database of predefined elements

```
#include "G4NistManager.hh"

G4NistManager* man = G4NistManager::Instance(); //get singleton instance

G4Element* elH = man->FindOrBuildElement("H")
```

- G4Element overrides << operator to provide useful print output

```
G4cout << elH << G4endl;

//Element: H (H)  Z =  1.0  N =  1.0  A =  1.01 g/mole
//    ---> Isotope:  H1  Z =  1  N =  1  A =  1.01 g/mole  abundance: 99.99 %
//    ---> Isotope:  H2  Z =  1  N =  2  A =  2.01 g/mole  abundance:  0.01 %
```

Geant 4

193

Defining Materials using Elements

- Material consisting of a single element can be quickly defined using

```
G4double density = 1.390 * g/cm3;
G4double a = 39.95 * g/mole;
G4double z = 18.;

G4Material liquidArgon = G4Material("liquid Argon", z, a, density);
```

Geant 4

194

Defining Materials using Elements

- Materials consisting of multiple elements are defined by their **molecular** or fractional element composition, and density or temperature and pressure.

```
G4double z, a, density;
G4String name, symbol;
G4int ncomponents, natoms;

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

a = 16.00*g/mole;
G4Element* elO = new G4Element(name="Oxygen" ,symbol="O" , z= 8., a);

density = 1.000*g/cm3;
G4Material* H2O = new G4Material(name="Water",density,ncomponents=2);
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

Geant 4

195

Defining Materials using Elements

- Materials consisting of multiple elements are defined by their **molecular** or **fractional** element composition, and density or temperature and pressure.

```
G4double z, a, density, fractionmass;
G4String name, symbol;
G4int ncomponents;

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

a = 16.00*g/mole;
G4Element* elO = new G4Element(name="Oxygen" ,symbol="O" , z= 8., a);

density = 1.000*g/cm3;
G4Material* H2O = new G4Material(name="Water",density,ncomponents=2);
Air->AddElement(elN, fractionmass=0.7);
Air->AddElement(elO, fractionmass=0.3);
```

Geant 4

196

Defining Mixtures

- Mixtures are Materials which consist multiple sub-materials and/or elements

```
G4Element* elC = ...; // elemental Carbon
G4Material* SiO2 = ...; // Silicon-DiOxide a.k.a quartz material
G4Material* H2O = ...; // water

G4double density = 0.2 * g/cm3;
G4Material* aeroGel = G4Material("Aerogel", density, nComponents = 3);
aeroGel->AddMaterial(SiO2, fractionMass = 0.625);
aeroGel->AddMaterial(H2O, fractionMass = 0.374);
aeroGel->AddElement(elC, fractionMass = 0.1);
```

Geant 4

197

Example: Defining a Gas

- dE/dx may be affected by pressure and temperature so these may need to be defined
- Materials with densities below 10mg/cm³ are automatically considered gases

```
G4double density = 27. * mg/cm3;
G4double temperature = 325. * Kelvin;
G4double pressure = 50. * atmosphere;

G4Material* CO2 = new G4Material("CO2-Gas", density,
nComponents=2, kStateGas, temperature, pressure);

CO2->AddElement(elC, natoms=1);
CO2->AddElement(elO, natoms=2);
```

Geant 4

198

Example: Defining Vacuum

- Vacuum is a gas with very low density. Absolute vacuum does not exist!
- Using cosmic vacuum as an example (consider that in your detector the vacuum might contain additional residual gases)

```
G4double density = 1e-25 * g/cm3;  
G4double temperature = 2.73 * Kelvin;  
G4double pressure = 3e-18 * pascal;  
  
G4Material* CO2 = new G4Material("cosmicVacuum", density,  
                                nComponents=1,          kStateGas, temperature, pressure);  
  
CO2->AddElement(e1H, natoms=1);
```

Geant 4

199

Materials Hints

- Don't go into unnecessary detail: if you are not interested in simulating hadronic processes which depend on isotopic composition, you are unlikely to need to define elements this way.
- Separate material definitions out into a separate class or a separate method (within the DetectorConstruction, see later). This way your material definitions are more easily reusable.
- Especially for mixtures it's a good idea to document where you found values for density etc. within the code.

Geant 4

200

Geant4 Geometry – building your virtual experiment

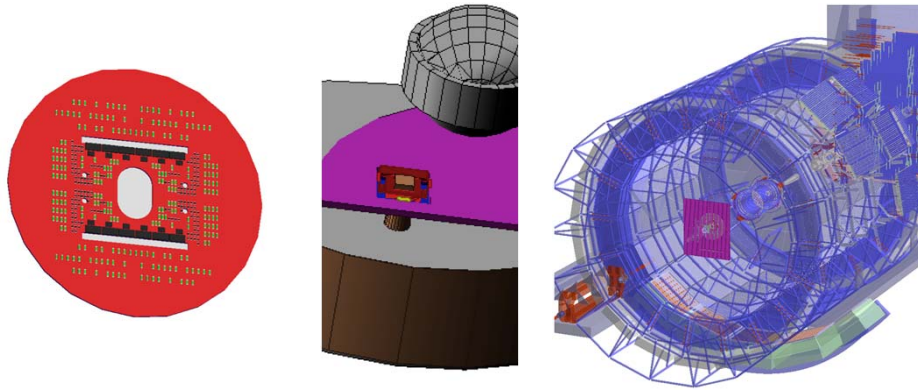


Image source: ALICE experiment

Geant 4

201

Geant4 Geometry – Building an Experiment

- The geometry defines the location and extents of materials in the simulation
- At each step in tracking it is checked in which object within the simulation geometry a particle is located.
- Simulated geometries can be very simple (e.g. a box) to very complex (e.g. an LHC)
- Simulated geometries can be very small (e.g. a SMD-component) to very large (e.g. Earth and its magnetosphere)
- Complex geometries lead to longer simulation times and more complex analysis – an important task **before** implementing a new simulation is defining a geometry which is as detailed as needed but as simple as possible.

Geant 4

202

Detector description

- Start with deriving your own concrete class from the **G4VUserDetectorConstruction** abstract base class

```
#include "G4VUserDetectorConstruction.hh"

class MyDetectorConstruction : public G4VUserDetectorConstruction {...}
```

- Overwrite the **Construct()** method of the base class with your geometry construction
 - Modularize your geometry according to detector sub-components
 - Define materials
 - Define solids required to describe the geometry
 - Place these solids within the *world volume*
 - Optionally, define sensitive detectors, visualization attributes, fields and detector regions

Geant 4

203

Example

G4VUserDetectorConstruction.hh

```
virtual G4VPhysicalVolume* Construct() = 0;
```

MyDetectorConstruction.hh

```
#include "G4VUserDetectorConstruction.hh"

class MyDetectorConstruction : public G4VUserDetectorConstruction {

public:
    G4VPhysicalVolume* Construct();

}
```

MyDetectorConstruction.cc

```
G4VPhysicalVolume* MyDetectorConstruction::Construct(){
    //construct detector geometry here
    ...
    return PhysWorld; //finally return world volume, more on this later
}
```

Dividing your detector geometry into sub-assemblies (e.g. calorimeter, tracker) eases code maintenance and allows for collaborative work

Geant 4

204

Example

MyApplication.cc

```
#include "G4RunManager.hh"
#include "MyDetectorConstruction.hh"

int main(char argc, char** argv){

    //Create a run manager
    G4RunManager* runManager = new G4RunManager();

    //Create geometry
    MyDetectorConstruction* detector = new MyDetectorConstruction();

    //Register the geometry with the run manager
    RunManager->SetUserInitialization(detector);

    // additional mandatory user classes (PhysicsList, PrimaryGenerator)

    delete detector; //not needed, performed by run manager
}
```

Of course you could implement a constructor which takes arguments, e.g. number of pixels, thickness. Parameterizing the geometry allows for altering it without recompiling

Geant 4

If you get segmentation faults at the end of each simulation, check for this!

Adding Parts to the Detector

- Start with describing its shape:
 - Box 5m x 2m x 10m
 - Sphere $r=10\text{m}$
 - Cylinder $r = 5\text{m}$, $h = 10\text{ m}$...
- Which properties does it have?
 - Material
 - Any B- or E-fields?
 - Is it sensitive, i.e. used for read-out?
- Finally, place it:
 - Single placement?
 - Or repeatedly as described by some function?

Solids



Logical Volumes



Physical Volumes

Geant 4

206

Adding Parts to the Detector

- Defines the shape of a volume
 - Multiple ways of doing so, from geometric primitives to complex meshes
- Defines the size of a volume
 - Don't forget to give units!

Solids



Logical Volumes



Physical Volumes

Geant 4

207

Adding Parts to the Detector

- Attach a material to a solid
- Attach E- and/or B-fields to a solid
- Make a solid sensitive
- Define visualization attributes for a solid
- Set user limits
- Add physical daughter volumes

Solids



Logical Volumes



Physical Volumes

Geant 4

208

Adding Parts to the Detector

- Set the location and the rotation of a volume
- Generate replicas for repeated volumes

Solids



Logical Volumes



Physical Volumes

Geant 4

209

Example: A Cubic Detector

- Solids define object shape, i.e. a box

```
G4Box* boxSolid = new G4Box("aBox", 1.0*m, 1.0*m, 1.0*m);
```

- Logical volumes tie solid to a material and define properties

```
G4LogicalVolume* boxLogic = new G4LogicalVolume(boxSolid, BoxMaterial, "Box1Logic")
```

- Volumes which have been placed in the Geometry are called physical volumes

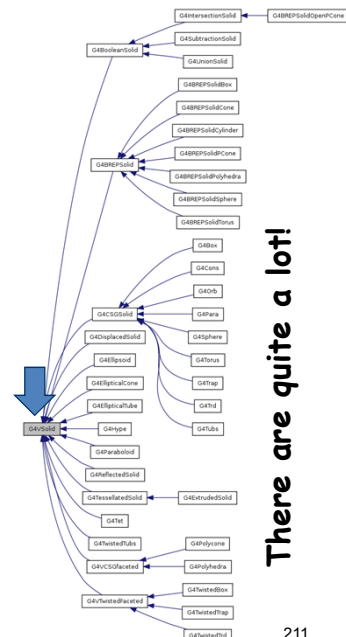
```
G4ThreeVector pos(1*m, 1*cm, 1*mm);  
G4RotationMatrix* rot = 0;  
G4VPhysicalVolume* boxPhys = new G4PVPlacement(rot, 0, boxLogic, "Box1Placed", World, pMany = 0, copyNo = 0, surfChk = true);
```

Geant 4

210

Solids

- All solids are derived from the abstract G4VSolid base class which provides interface for e.g.
 - bool Inside(G4ThreeVector)
 - G4double DistanceToIn(G4ThreeVector)
 - G4double DistanceToOut(G4ThreeVector)
 - G4ThreeVector SurfaceNormal(G4Three...
- Upon construction a solid is automatically registered in a solid store



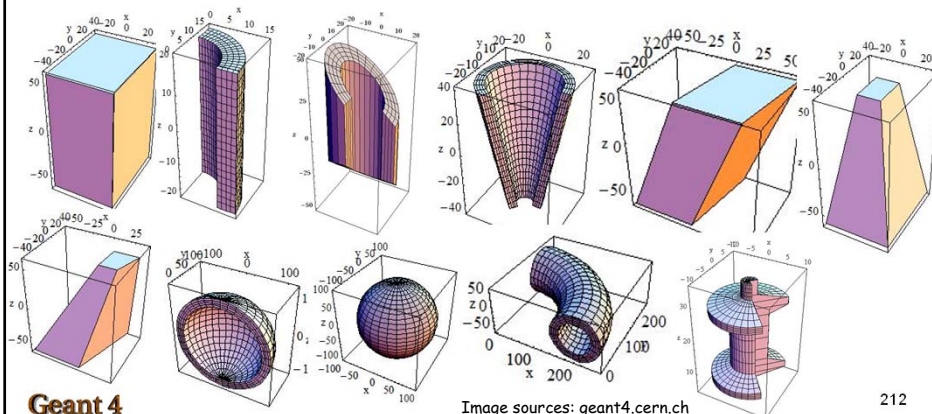
There are quite a lot!

Geant 4

211

Solids

- Constructed Solid Geometries (CSG)
- Boundary Represented Shapes (BREPs)
- Complex shapes by parameterization
- Boolean operations also possible



Geant 4

Image sources: geant4.cern.ch

212

Solids: Example CSGs

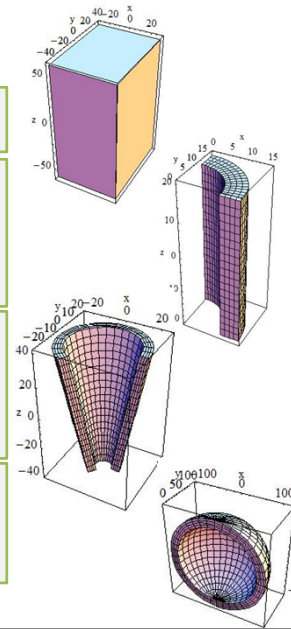
```
G4VSolid* boxSolid = new G4Box("aBox", 1.0*m,
1.0*m, 1.0*m);
```

```
G4VSolid* tubeSolid = new G4Tubs("aTube",
1.0*m, //inner radius (0 possible)
2.0*m, //outer radius
4.0*m // half - height
0.*deg, 360.*deg); //segment angles
```

```
G4VSolid* coneSolid = new G4Cons("aCone",
1.0*m, 1.5*m, //inner/outer radius 1
2.0*m, 2.5*m, //inner/outer radius 2
4.0*m // half - height
0.*deg, 360.*deg); //segment angles
```

```
G4VSolid* sphereSolid = new G4Sphere("aSphere",
1.0*m, 1.5*m //inner/outer radius
0.*deg, 360.*deg, // phi
0.*deg, 180.*deg); // theta
```

Geant 4



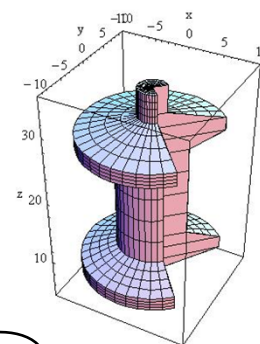
Solids: not-trivial CSG

```
G4int numRZ = 10;
```

```
G4double r[] = {0.*cm, 1.*cm, 1.*cm, 5.*cm, 5.*cm,
3.*cm, 3.*cm, 5.*cm, 5.*cm, 1.*cm};
```

```
G4double z[] = {0.*cm, 0.*cm, 1.*cm, 1.2*cm,
2.*cm, 2.2*cm, 4.8*cm, 5.*cm, 6.*cm, 6.*cm};
```

```
G4VSolid* polyConeSolid = new
G4PolyCone("aPolycone",
0.*deg, //start angle
360.*deg, //total angle
numRZ, // number of cross-sections
r, // r-coordinate of sections
z); // z-coordinates
```



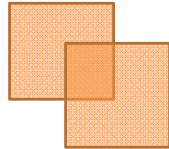
It is often a good idea to make a sketch of such volumes before starting to write any code.

Geant 4

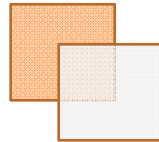
214

Solids: Boolean Operations

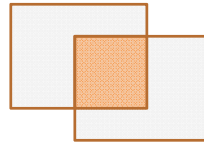
G4UnionSolid



G4SubtractionSolid



G4IntersectionSolid



- Boolean operations allow combinations of **two** solids
- 2nd solid is positioned (and optionally transformed) w.r.t. coordinate system of first solid
- Result is a new solid, which again can again participate in boolean operations
- **Boolean operations allow for “surprisingly” complex geometries with usually quite little effort.**

Geant 4

215

Example: Boolean Solid

```
G4VSolid* boxSolid1 = new G4Box("aBox1", 1.0*m, 1.0*m, 1.0*m);
G4VSolid* boxSolid2 = new G4Box("aBox1", 0.5*m, 0.5*m, 0.5*m);

G4VSolid* BoxBoxSolid = new G4UnionSolid("aBoxBox",
    boxSolid1, boxSolid2, //two solids to operate on
    0, G4ThreeVector(0., 0., 95.*cm)); // translation

G4VSolid* hollowBoxSolid = new G4SubtractionSolid("aHollowBox",
    boxSolid1, boxSolid2, //two solids to operate on
    0, G4ThreeVector(0., 0., 5.*cm)); // translation

G4VSolid* BoxBoxSolid = new G4IntersectionSolid("aIntersectedBox",
    boxSolid1, boxSolid2, //two solids to operate on
    0, G4ThreeVector(0., 0.95*cm, 95.*cm)); // translation
```

- The origin and coordinates of the resulting solid are those of the **first** solid.

Geant 4

216

Geant4 Geometry – CAD Import

- Import from existing CAD models is not trivial
- Geant4 can import GDML files – G4GDMLParser
 - `parser.read("myCoolGeometry.gdml")`
 - `G4VPhysicalVolume world = parser.GetWorldVolume();`
- Problem: most CAD tools do not export GDML
- Solution: convert to STEP, open e.g. in FastRad or ST-Developer, convert to GDML
 - **Material information can be lost**
 - **Conversion errors may occur**
- **Question to ask:** is geometry sufficiently complex that I need to import from CAD or can it be quickly reconstructed from Geant4 provided geometric objects? The latter is usually preferable if possible.

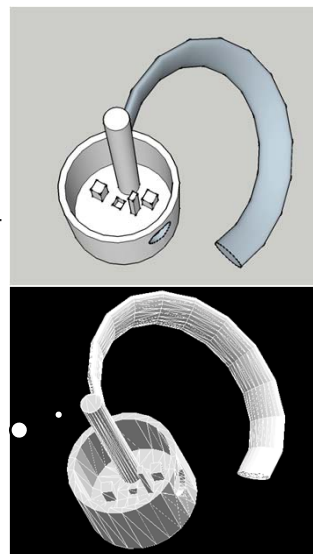
Geant 4

217

Geant4 Geometry – CAD Import

- CADMesh allows to import CAD files (STL, PLY, COLLADA) into Geant4 directly
- Three backends are selectable
- Problems may occur with some complex models (personally had success with Sketchup model, but problems with Solid-Edge models)
- Material information is not imported, i.e. components with different materials need to be imported separately
- Very active developers
- code.google.com/p/cadmesh

Note that this could have easily been constructed using CSGs and boolean operations as well!.



Geant 4

Logical Volumes

- The solids just discussed define the shape and size of a volume
- Now let's add properties to it:
 - Materials
 - E-/B- fields
 - Visualization attributes
 - Sensitivity
 - Position of daughter volumes
 - Regions
- A logical volume does not define the position and rotation of a volume (physical volumes define these)
- Logical volumes can be shared amongst physical volumes of the same type

Geant 4

219

Logical Volumes

```
G4LogicalVolume(G4VSolid* pSolid,          //pointer to solid, must to be 0
  G4Material* pMat,          //pointer to material, must not be 0
  const G4String& name,      //name of this logical volume
  G4FieldManager* pFieldMgr = 0, //pointer to field manager
  G4VSensitiveDetector* pSensDet = 0, //sensitive detector
  G4UserLimits* pUserLimits = 0, // user limits
  G4bool optimise = true);      // optimization on
```

- Volumes are automatically registered in the logical volume store
- Logical volumes should be instantiated, not derived from – they are not meant as base classes

Geant 4

220

Physical Volumes

- So far we have created a description of our volume:
 - Defined shape and size of volume (Solid)
 - Defined properties such as material, sensitivity (Logical volume)
- Now we make it “physical” by actually placing it:
 - Define position and rotation with respect to other (logical) volumes
- => Physical volumes are placed instances of logical volumes

Geant 4

221

Physical Volumes

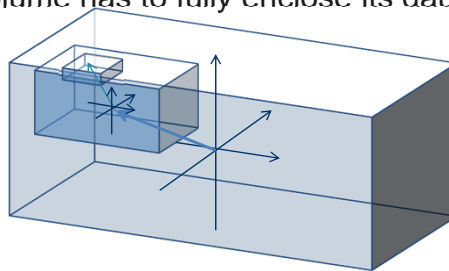
- Multiple options exist for placing volumes
 - Single placement of a volume
 - Repeated placement as replicas or parameterized volumes (think of a detector which has many identical subcomponents)
 - A logical volume may be placed more than once (be careful that you distinguish it properly e.g. when using it as a sensitive detector)
- Volumes in Geant4 constitute a hierarchical geometry
 - At the top: the root volume, often referred to as the world volume
 - All other volumes have a mother volume
 - A mother volume may have any number of daughter volumes
- Physical volumes derive from **G4VPhysicalVolume**

Geant 4

222

Hierarchical Placement

- Coordinate systems in Geant4 are defined by the respective mother volume, i.e. all daughter volumes are placed relative to their mother volume's local coordinate system
- The origin of the mother volume's local coordinate system is the center of the volume
- A mother volume has to fully enclose its daughter volumes



Geant 4

223

Hierarchical Placement

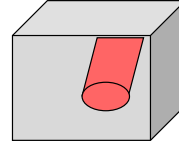
- A **logical** mother volume knows about the physical volumes it contains, it is the unique mother volume to these.
- If a logical mother volume is placed multiple times, all daughter volumes (and accordingly also daughters of daughters) appear in all physical instances of this mother.
- **The hierarchy requires a root volume, known as the world volume**
 - Defines the global coordinate system with origin at its center
 - The position of a particle track is given relative to the world coordinate system
 - The simplest world volume is a box

Geant 4

224

Single placement

- One physical volume represents one “real” volume.



```
G4PVPlacement(G4RotationMatrix* pRot, // rotation w.r.t. to mother
volume
    const G4ThreeVector& trans, // translation w.r.t. mother
    G4LogicalVolume* pLog, // solid logical volume
    const G4String& name,
    G4LogicalVolume* pMLog, // mother logical volume
    G4bool pMany, // not used
    G4int copyNo, // set to 0 for first volume of type
    G4bool surfChk); // check for overlaps
```

- The placement occurs in the mother volumes coordinate system and reference frame.

Geant 4

225

Example: Single placement

```
G4VSolid* boxSolid= new G4Box("aBox", 1.0*m, 1.0*m, 1.0*m);

G4LogicalVolume* boxLogic = new G4LogicalVolume(boxSolid, boxMat,
"logicBox");

G4RotationMatrix* xRot = new G4RotationMatrix();
xRot->rotateX(M_PI/4*rad);

G4ThreeVector yTrans(0., 1.*m, 0.);

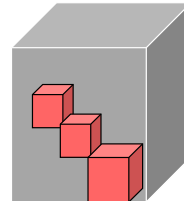
//constructor 1
G4VPhysicalVolume* boxPhys = new G4PVPlacement(xRot, yTrans, boxLogic,
"physicBox", motherLog, 0, copyNo, true);

//constructor 2
G4VPhysicalVolume* boxPhys = new G4PVPlacement(G4Transform3D(xRot,
yTrans), boxLogic, "physicBox", motherLog, 0, copyNo, true);

//constructor 3
G4VPhysicalVolume* boxPhys = new G4PVPlacement(xRot, yTrans, boxLogic,
"physicBox", motherPhys, 0, copyNo, true);
```

Parameterized Placement

- A logical volume can be placed multiple times with G4PVParameterized. Position and size are parameterized w.r.t. the copy number
- Only works for primitives
- User must provide concrete implementation of G4PVParameterisation



```
G4PVParameterized(const G4String& name,
  G4LogicalVolume* pLogical, //pointer to logical volume
  G4LogicalVolume* pMotherLog, //pointer to mother logical
  const EAxis pAxis, // axis along which to parameterize
  const G4int nReplicas, // number of replicas
  const G4PVParameterisation* pPara // param. definition
```

Geant 4

227

Parameterized Placement

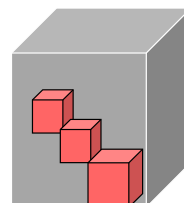
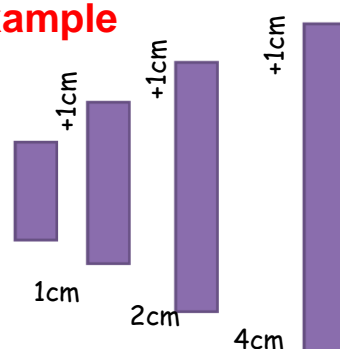
- User must provide concrete implementation of G4PVParameterisation
 - ComputeDimensions()
 - ComputeTransformations()

Optionally:

- ComputeMaterial()
- ComputeSolid()

- Limited to CSG solids

Example



Geant 4

228

Example: Param. Volume

```
G4VSolid* boxSolid= new G4Box("aBox", 1.0*cm, 10.0*m, 1.0*m);

G4LogicalVolume* boxLogic = new G4LogicalVolume(boxSolid, boxMat,
"logicBox");

G4PVParameterisation* boxParam = new UserParameterisation();

G4VPhysicalVolume* boxPhys = new G4PVParameterised("paramBox", boxLogic,
motherLogic, kXAxis, 4, boxParam);
```

Geant 4

229

Example: Param. Volume

```
#include "G4PVParameterisation.hh"

class UserParameterisation : public G4PVParameterisation {

public:
    UserParameterisation();
    ~UserParameterisation();

    void ComputeTransformation(const G4int copyNo,G4VPhysicalVolume*
pVol)      const;
    void ComputeDimension(G4Box* paramVol, const G4int
copyNo,G4VPhysicalVolume* pVol)      const;
}
```

Geant 4

230

Example: Param. Volume

```
#include "UserParameterisation.hh"

void UserParameterisation::ComputeTransformation(const G4int
copyNo,G4VPhysicalVolume* pVol){
    G4ThreeVector origin(pow(2, copyNo)*cm, 0., 0.);
    pVol->SetTranslation(origin);
    pVol->SetRotation(0);
}

void UserParameterisation::ComputeDimension(G4Box* paramVol, const G4int
copyNo,G4VPhysicalVolume* pVol) {

    G4double boxHeight = paramVol->GetYHalfLength();
    G4double newHeight = boxHeight+copyNo*1.*cm;
    paramVol->SetYHalfLength(newHeight);
}
```

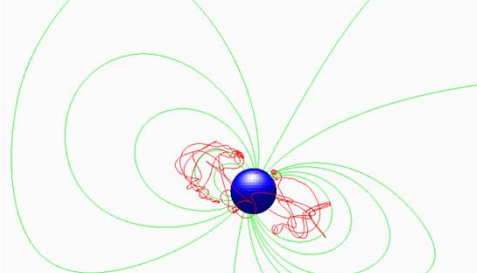
Geant 4

231

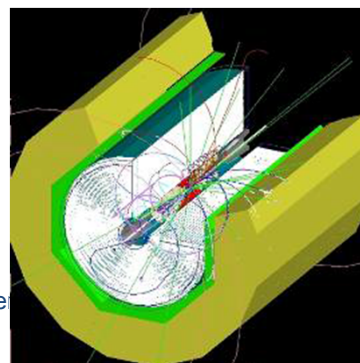
Geant4 Geometry – Fields

1 GeV proton in the Earth's geomagnetic field

MOKKA Linear Collider



Courtesy Laurent Desorgher, University of Be

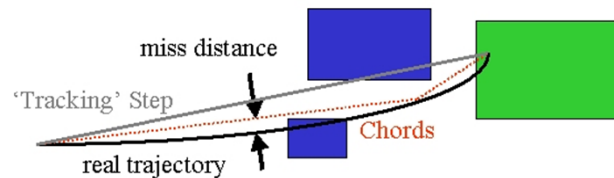


Geant 4

232

Geant4 Geometry – Fields

- Geant4 allows to define electro-magnetic and gravity fields as part of the geometry.
- Fields can be added using predefined uniform fields, by giving a field equation or a field map
- Particle motion within a field is calculated using Runge-Kutta integration of the particle equation of motion (if an analytics solution is known other solvers can also be used)
- The curved particle trajectory is divided into chords, the user is responsible for setting an appropriate precision for this divisions



Geant 4

233

Examples: Magnetic Fields

- For uniform fields use an instance of **G4UniformMagField**

```
G4MagneticField* magField
= new G4UniformMagField(G4ThreeVector(1.*Tesla, 0.,0.);
```

- Non-uniform fields can be created by providing a concrete class derived from G4MagneticField which implements **GetFieldValue**

```
Void MyField::GetFieldValue(const double point[4], // 0,1,2: position in global
                           //coordinate system, 3: time
                           double* field); // 0,1,2: field vector
```

Geant 4

234

Registering the Field

- Register the field with a field manager (here globally)

```
G4FieldManager* fManager =  
G4TransportationManager::GetTransportationManager()->GetFieldManager();  
//retrieve a pointer to the global field manager from the transportation  
manager singleton  
  
//register the field  
fManager->SetDetectorField(myMagField)  
  
//create a chord finder for the field  
fManager->CreateChordFinder(myMagField);
```

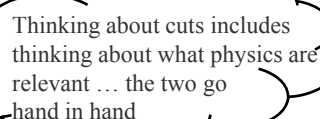
- Fields may also be registered to individual logical volumes, see e.g. examples N04 for further information

Geant 4

235

Regions & Cuts

- In Geant4 production thresholds (cuts) are expressed by lengths (i.e. minimum range of secondary) on a per-particle (or global) basis
- The user should define appropriate length scales for individual detector components, i.e. 5 micron for a vertex detector but 5 mm for a muon tracker
- **Simply using small cuts everywhere is not a good choice, it will result in performance penalties!**
- Cuts can be defined globally or for different detector sub regions.



Thinking about cuts includes thinking about what physics are relevant ... the two go hand in hand

Geant 4

236

Regions & Cuts

- The world volume is the default root volume with default cuts as defined in the physics list
- Assigning a logical volume to a region makes it a root volume and defines the cuts for all its daughter volumes (hierarchy applies, i.e. a daughter may be assigned to another region)
- A region cannot be shared amongst multiple logical volumes

Geant 4

237

Regions & Cuts

```
G4Region* muonTracker = new G4Region("Muon-Tracker");
muonTracker->AddRootLogicalVolume(muonTrackerLog);

G4ProductionCuts* cuts = new G4ProductionCuts();
cuts->SetProductionCut(2.*mm, G4ProductionCuts::GetIndex("e-"));
cuts->SetProductionCut(2.*mm, G4ProductionCuts::GetIndex("e+"));
cuts->SetProductionCut(1.*mm, G4ProductionCuts::GetIndex("gamma"));

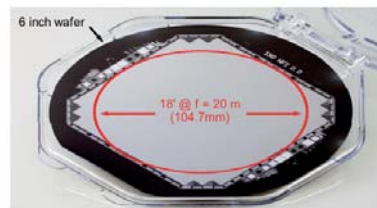
muonTracker->SetCuts(cuts);
```

Geant 4

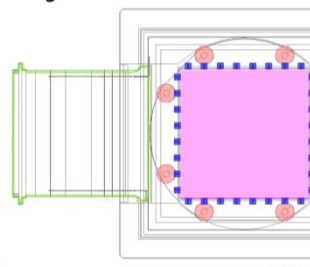
238

Read Out Geometries

- Logical assignment of particle position in an experiment output usually does not directly map to a single detector component
- Example: pixelized X-ray sensor: readout is per pixel, but a pixel is not a physical separate object, but part of a silicon bulk, with may also include passivation layers
- Readout geometries map this logical output scheme to underlying physical components.
- Defined similar to “normal” geometries
- Attaches to sensitive detectors



Original CAD



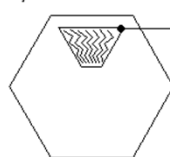
Geant 4

Read Out Geometries

```
class MyReadoutGeom : public G4VReadoutGeometry
{
public:
    void build(){
        //build geometry similar to DetectorConstruction here
    }

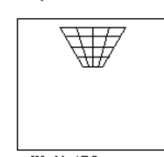
    //in DetectorConstruction
    MyReadoutGeom* ROGeom = new MyReadoutGeom("name");
    ROGeom -> BuildROGeometry();
    MySensitiveDet -> SetROGeometry(ROGeom);
}
```

The Tracking Geometry
(builds by G4VUserDetectorConstruction)



G4VSensitiveDetector
object

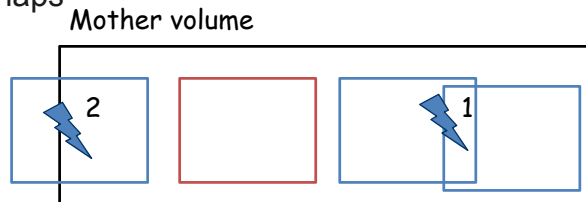
A Readout Geometry
(builds by a G4VReadoutGeometry)



Geant 4

Debugging Geometries

- In Geant4 volumes should not overlap on the same hierarchical level (1) and daughters should be fully enclosed in their mother volumes (2)
- Even if great care is taken, it is not improbable that one or both of these problems may occur when implementing a new geometry
- Geant4 and external tools help in finding such mishaps



Geant 4

241

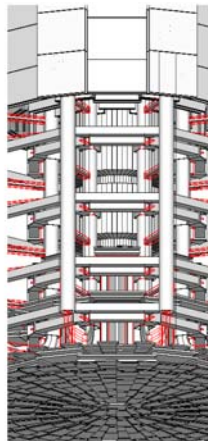
Debugging Geometries

```
G4PVPlacement(G4RotationMatrix* pRot,    // rotation w.r.t. to mother volume
               const G4ThreeVector& trans, // translation w.r.t. mother
               G4LogicalVolume* pLog,    // solid logical volume
               const G4String& name,
               G4LogicalVolume* pMLog,    // mother logical volume
               G4bool pMany,              // not used
               G4int copyNo,              // set to 0 for first volume of type
               G4bool surfChk);          // check for overlaps
```

Geant 4

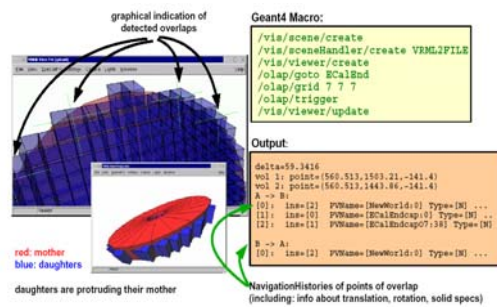
242

Debugging Geometries



DAVID

Tools to detect badly defined geometries



Geant 4

243

Debugging Geometries

- Additionally, UI commands are provided to run tests:

Idle> geometry/test/run

or

Idle> geometry/test/grid_test

Tests along a grid setup (can be customized)

Idle> geometry/test/line_test

Will test along a specified direction starting from a specified position

Geant 4

244

Geometry Hints

- Before you start coding:
 - Think about how to simplify the real world detector. **You usually do not need to model every nut and bolt!**
 - **Think about if your detector can be grouped into sub-components/assemblies.** Do they require different levels of detail? Do they require different cuts – define regions where appropriate.
 - **Identify which components are repetitive, and whether they can be parameterized.**
 - **If you have an existing CAD model, estimate how complex it will be after simplification.** Does it make sense to go through the troubles of importing it component-wise and re-assigning materials? Or will it be quicker to implement it in CSGs possibly using boolean operations?
 - **Identify mother volumes, i.e. such volumes which are suitable reference frames for positioning other volumes.**

Geant 4

245

Geometry Hints

- While you code:
 - If you can't visualize the shape of a volume (like boolean or complex CSG) by thought, often it is better to sketch it on paper than trying to debug in code.
 - **Visualize your geometry using Geant4 visualization options while you build your detector. This way you find mistakes early on and can be sure parts you have already tested are okay!** Set surfChk=true when placing physical volumes.
 - **Maintain a consistent and expressive style for naming your materials and volumes.** E.g. add Solid to all solids, Logic to all logical volumes and Phys to all physical volumes.
 - Define positions and sizes using variables with meaningful names (trackerLength = 5.*cm) and not just give numbers in the solid definition or the PVPlacement.

Geant 4

246

Geant4 Geometry – Sensitive Detectors

```
class MySensitiveDetector : public G4VSensitiveDetector

MySensitiveDetector::Initialize(G4HCoFThisEvent* hc) {
    //initialize for next event here
}

MySensitiveDetector::ProcessHits(G4Step* aStep, G4TouchableHistory*
ROHist){
    //stop information on particle as hit in collection
}

MySensitiveDetector::EndOfEvent(G4HCoFThisEvent* hc){
    //process hit collection at end of event
}
}
```

Geant 4

247

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

*This course encompasses training material developed by several Geant4
members:*

thanks to all of them!

Geant 4

Geant4 Detector Response

Geant 4

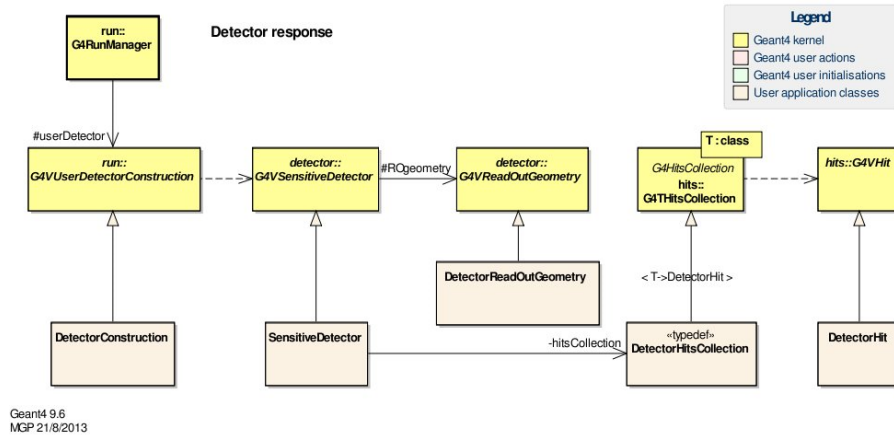
Extraction of useful information

- Given geometry, physics and primary track generation, Geant4 does proper physics simulation “silently”
 - The user needs to add a bit of code to **extract useful information**
- There are three ways:
 - Built-in scoring commands
 - Most commonly-used physics quantities are available
 - Use scorers in the tracking volume
 - Create scores for each event
 - Create own Run class to accumulate scores
 - Assign **G4VSensitiveDetector** to a volume to generate “hit”
 - Use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary
- The user may also use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)
 - The user has full access to almost all information

Geant4 Detector Response

250

Detector Response



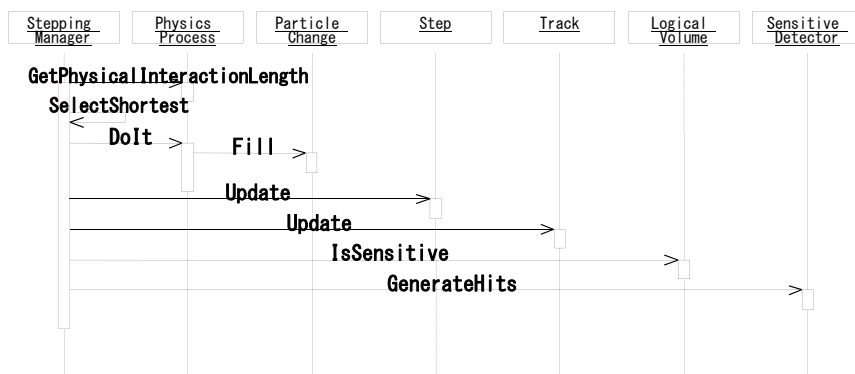
- Please refer to earlier lecture for `DetectorConstruction` and `ReadOutGeometry`

Geant4 Detector Response

251

Sensitive detector

- A `G4VSensitiveDetector` object can be assigned to a `G4LogicalVolume`
- In case a step takes place in a logical volume that has a `G4VSensitiveDetector` object, this `G4VSensitiveDetector` is



Geant4 Detector Response

252

Defining a sensitive detector

- The basic strategy

```
G4LogicalVolume* myLogCalor = .....;
G4VSensitiveDetector* pSensitivePart = new
    MyDetector( "/mydet" );
G4SDManager* SDMan = G4SDManager::GetSDMpointer();
SDMan->AddNewDetector( pSensitivePart );
myLogCalor->SetSensitiveDetector( pSensitivePart );
```

- Each detector **object** must have a unique name
 - Some logical volumes can share one detector object
 - More than one detector objects can be made from one detector class **with different detector name**
 - One logical volume cannot have more than one detector objects. But, one detector object can generate more than one kinds of hits
 - e.g. a double-sided silicon micro-strip detector can generate hits for each side separately

Geant4 Detector Response

253

Hits collection, hits map

- G4VHitsCollection is the common abstract base class of both **G4THitsCollection** and **G4THitsMap**
- **G4THitsCollection** is a **template vector class** to store pointers of objects of one concrete hit class type
 - A hit class (deliverable of G4VHit abstract base class) should have its own identifier (e.g. cell ID)
 - G4THitsCollection requires the user to implement own hit class
- **G4THitsMap** is a **template map class** so that it stores keys (typically cell ID, i.e. copy number of the volume) with pointers of objects of one type
 - Objects may not be those of hit class
 - All of currently provided scorer classes use G4THitsMap with simple double
 - Since G4THitsMap is a template, it can be used by the sensitive detector class to store hits

Geant4 Detector Response

254

Sensitive detector and Hit

- Each Logical Volume can have a pointer to a sensitive detector
 - Then this volume becomes **sensitive**
- Hit is a snapshot of the physical interaction of a track or an accumulation of interactions of tracks in the sensitive region of your detector
- A sensitive detector creates hit(s) using the information given in G4Step object. The user has to provide his/her own implementation of the detector response
- Hit objects, which are still the user's class objects, are collected in a G4Event object at the end of an event

Geant4 Detector Response

255

Hit Class

- Hit is a user-defined class derived from **G4VHit**
- The user can store various types information by implementing one's own concrete Hit class. For example:
 - Position and time of the step
 - Momentum and energy of the track
 - Energy deposition of the step
 - Geometrical information
 - or any combination of above
- Hit objects of a concrete hit class must be stored in a dedicated collection which is instantiated from **G4THitsCollection template class**
- The collection is associated to a G4Event object via **G4HCofThisEvent**
- Hits are accessible as collections:
 - through G4Event at the end of event
 - to be used for analyzing an event
 - through G4SDManager during processing an event
 - to be used for event filtering

Geant4 Detector Response

256

Implementation of Hit class

```
#include "G4VHit.hh"
class MyHit : public G4VHit
{
public:
    MyHit(some_arguments);
    virtual ~MyHit();
    virtual void Draw();
    virtual void Print();
private:
    // some data members
public:
    // some set/get methods
};

#include "G4THitsCollection.hh"
typedef G4THitsCollection<MyHit> MyHitsCollection;
```

Geant4 Detector Response

Scoring II - M.Asai (SLAC)

25257

Sensitive Detector class

- Sensitive detector is a user-defined class derived from G4VSensitiveDetector

```
#include "G4VSensitiveDetector.hh"
#include "MyHit.hh"
class G4Step;
class G4HCofThisEvent;
class MyDetector : public G4VSensitiveDetector
{
public:
    MyDetector(G4String name);
    virtual ~MyDetector();
    virtual void Initialize(G4HCofThisEvent*HCE);
    virtual G4bool ProcessHits(G4Step*aStep,
                               G4TouchableHistory*ROhist);
    virtual void EndOfEvent(G4HCofThisEvent*HCE);
private:
    MyHitsCollection * hitsCollection;
    G4int collectionID;
};
```

Geant4 Detector Response

258

Sensitive Detector

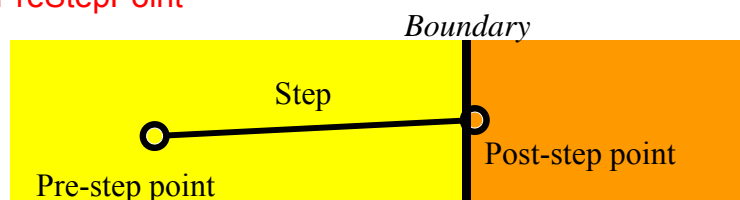
- A **tracker** detector typically generates a hit for every single step of every single (charged) track
 - A tracker hit typically contains
 - Position and time
 - Energy deposition of the step
 - Track ID
- A **calorimeter** detector typically generates a hit for every cell, and accumulates energy deposition in each cell for all steps of all tracks
 - A calorimeter hit typically contains
 - Sum of deposited energy
 - Cell ID
- The user can instantiate more than one objects for one sensitive detector class. Each object should have its unique detector name
 - For example, each of two sets of detectors can have their dedicated sensitive detector objects. But, the functionalities of them are exactly the same. Some use each other so that they can share the same class. See <https://cds.cern.ch/record/1401190/files/ATLAS-CONF-2017-026.pdf>

Geant4 Detector Response

259

Step

- Step has two points and also “delta” information of a particle (energy loss on the step, time-of-flight spent by the step, etc.)
- Each point knows the volume (and material). In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it **logically belongs to the next volume**
- Note that you must get the volume information from the “PreStepPoint”



Geant4 Detector Response

260

Implementation of Sensitive Detector - 1

```
MyDetector::MyDetector(G4String detector_name)
    :G4VSensitiveDetector(detector_name),
      collectionID(-1)
{
    collectionName.insert("collection_name");
}
```

- In the constructor, the name of the hits collection which is handled by this sensitive detector is to be defined
- In case the sensitive detector generates more than one kinds of hits (e.g. anode and cathode hits separately), all collection names need to be defined

Geant4 Detector Response

261

Implementation of Sensitive Detector - 2

```
void MyDetector::Initialize(G4HCofThisEvent*HCE)
{
    if(collectionID<0) collectionID = GetCollectionID(0);
    hitsCollection = new MyHitsCollection
        (SensitiveDetectorName,collectionName[0]);
    HCE->AddHitsCollection(collectionID,hitsCollection);
}
```

- Initialize() method is invoked **at the beginning of each event**.
- Get the unique ID number for this collection
 - GetCollectionID() is a heavy operation. It should not be used for every event
 - GetCollectionID() is available **after** this sensitive detector object is constructed and registered to G4SDManager. Thus, this method **cannot** be invoked in the constructor of this detector class
- The hits collection(s) are to be instantiated and then attached to the **G4HCofThisEvent** object given in the argument
- In case of calorimeter-type detector, hits for all calorimeter cells may be instantiated with zero energy depositions, and then inserted to the collection

Geant4 Detector Response

262

Implementation of Sensitive Detector - 3

```
G4bool MyDetector::ProcessHits
(G4Step*aStep,G4TouchableHistory*ROhist)
{
    MyHit* aHit = new MyHit();
    ...
    // some set methods
    ...
    hitsCollection->insert(aHit);
    return true;
}
```

- This ProcessHits() method is invoked for every steps in the volume(s) where this sensitive detector is assigned
- In this method, generate a hit corresponding to the current step (for tracking detector), or accumulate the energy deposition of the current step to the existing hit object where the current step belongs to (for calorimeter detector)
- geometry information is to collected (e.g. copy number) from "PreStepPoint"
- Currently, returning boolean value is not used.

Geant4
Sensitive Detector Response

263

Implementation of Sensitive Detector - 4

```
void MyDetector::EndOfEvent(G4HCofThisEvent*HCE) {}
```

- This method is invoked at the end of processing an event.
 - It is invoked even if the event is aborted.
 - It is invoked before UserEndOfEventAction.

Geant4
Sensitive Detector Response

264

Step point and touchable

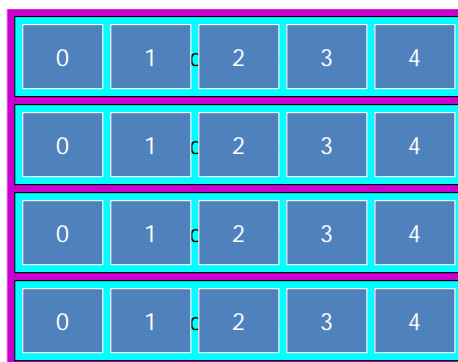
- As mentioned already, G4Step has two G4StepPoint objects as its starting and ending points. All the geometrical information of the particular step should be taken from "PreStepPoint"
 - Geometrical information associated with G4Track is identical to "PostStepPoint"
- Each G4StepPoint object has
 - Position in world coordinate system
 - Global and local time
 - Material
 - G4TouchableHistory for geometrical information
- G4TouchableHistory object is a vector of information for each geometrical hierarchy
 - copy number
 - transformation / rotation to its mother
- Since release 4.0, handles (or smart-pointers) to touchables are intrinsically used. Touchables are reference counted

Geant4 Detector Response

265

Copy number

- Suppose a calorimeter is made of 4x5 cells
 - and it is implemented by two levels of replica
- In reality, there is only one physical volume object for each level. Its position is parameterized by its copy number
- To get the copy number of each level, suppose what happens if a step belongs to two cells



- geometrical information in G4Track is identical to that in "PostStepPoint"
- User cannot get the correct copy number for "PreStepPoint" if one directly accesses to the physical volume

- touchable is to be used to get the proper copy number, transform matrix, etc.

Geant4 Detector Response

266

Touchable

- G4TouchableHistory has information of geometrical hierarchy of the point.

```
G4Step* aStep;
G4StepPoint* preStepPoint = aStep->GetPreStepPoint();
G4TouchableHistory* theTouchable =
    (G4TouchableHistory*)(preStepPoint->GetTouchable());
G4int copyNo = theTouchable->GetVolume()->GetCopyNo();
G4int motherCopyNo
    = theTouchable->GetVolume(1)->GetCopyNo();
G4int grandMotherCopyNo
    = theTouchable->GetVolume(2)->GetCopyNo();
G4ThreeVector worldPos = preStepPoint->GetPosition();
G4ThreeVector localPos = theTouchable->GetHistory()
    ->GetTopTransform().TransformPoint(worldPos);
```

G4HCofThisEvent

- A G4Event object has a G4HCofThisEvent object at the end of (successful) event processing. G4HCofThisEvent object stores all hits collections made within the event.
 - Pointer(s) to the collections may be NULL if collections are not created in the particular event
 - Hits collections are stored by pointers of G4VHitsCollection base class. Thus, one has to cast them to types of individual concrete classes
 - The index number of a Hits collection is unique and unchanged for a run. The index number can be obtained by

```
G4SDManager::GetCollectionID("detName/colName"
    );
```

- The index table is also stored in G4Run

Usage of G4HCofThisEvent

```
void MyEventAction::EndOfEventAction(const G4Event* evt) {
    static int CHCID = -1;
    if(CHCID<0) CHCID = G4SDManager::GetSDMpointer()
        ->GetCollectionID("myDet/collection1");
    G4HCofThisEvent* HCE = evt->GetHCofThisEvent();
    MyHitsCollection* CHC = 0;
    if (HCE) {
        CHC = (MyHitsCollection*)(HCE->GetHC(CHCID)); }
    if (CHC) {
        int n_hit = CHC->entries();
        G4cout<<"My detector has "<<n_hit<<" hits."<<G4endl;
        for (int i1=0;i1<n_hit;i1++) {
            MyHit* aHit = (*CHC)[i1];
            aHit->Print();
        }
    }
}
```

When to invoke GetCollectionID()?

- Which is the better place to invoke G4SDManager::GetCollectionID() in a user event action class, in its constructor or in the BeginOfEventAction()?
- It actually depends on the user's application
 - Note that construction of sensitive detectors (and thus registration of their hits collections to SDManager) takes place when the user issues RunManager::Initialize(), and thus the user's geometry is constructed.
- In case user's EventAction class should be instantiated before RunManager::Initialize() (or /run/initialize command), GetCollectionID() should **not** be in the constructor of EventAction.
- While, if the user has nothing to do to Geant4 before RunManager::Initialize(), this initialize method can be hard-coded in the main() before the instantiation of EventAction (e.g. exampleA01), so that GetCollectionID() could be in the constructor

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

*This course encompasses training material developed by several Geant4
members:*

thanks to all of them!

Geant 4

Particles and Processes

Basic concepts underlying Geant4 physics

Geant 4

Physics

From the Minutes of LCB (LHCC Computing Board) meeting on 21/10/1997:

“ It was noted that experiments have requirements for **independent, alternative physics models**. In Geant4 these models, differently from the concept of packages, allow the user to **understand** how the results are produced, and hence improve the **physics validation**. Geant4 is developed with a modular architecture and is the ideal framework where existing components are integrated and new models continue to be developed.”

Geant 4

273

Physics: general features

- Ample variety of physics functionality
- Uniform treatment of electromagnetic and hadronic processes
- Abstract interface to physics processes
 - Tracking independent from physics
- Distinction between processes and models
 - often multiple models for the same physics process (complementary/alternative)
- Open system
 - Users can easily create and use their own models

Geant 4

274

Physics: general features

- **Transparency**

- *supported by encapsulation and polymorphism*
- Calculation of cross-sections independent from the way they are accessed (data files, analytical formulae etc.)
- Distinction between the calculation of cross sections and their use
- Calculation of the final state independent from tracking
- Etc.

- Modular design, at a fine granularity, to expose the physics
- Explicit use of units throughout the code
- Public distribution of the code, from one reference repository worldwide

Geant 4

275

Data libraries

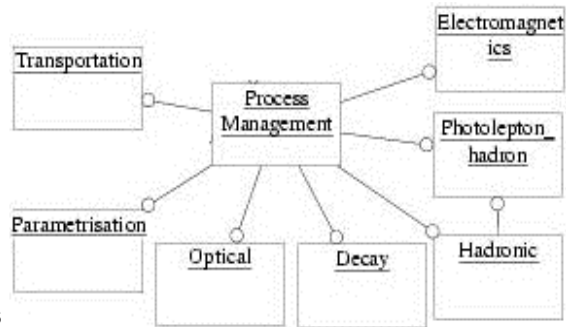
- Systematic collection and evaluation of experimental data from many sources worldwide
- Databases
 - ENDF/B, JENDL, FENDL, CENDL, ENSDF, JEF, BROND, EFF, MENDL, IRDF, SAID, EPDL, EEDL, EADL, SANDIA, ICRU etc.
- Collaborating distribution centres
 - NEA, LLNL, BNL, KEK, IAEA, IHEP, TRIUMF, FNAL, Helsinki, Durham, Japan etc.
- The use of evaluated data is important for the validation of physics results of the experiments

Geant 4

276

Processes

- Processes describe how particles interact with material or with a volume
- Three basic types
 - At rest process
(eg. decay at rest)
 - Continuous process
(eg. ionisation)
 - Discrete process
(eg. Compton scattering)
- Transportation is a process
 - interacting with volume boundary
- A process which requires the shortest interaction length limits the step



Geant 4

277

Outline

- What is tracked

G4ParticleDefinition
G4DynamicParticle
G4Track
- The process interface

G4VProcess
How processes interact with tracking
- The production cuts

Why production cuts are needed
The cuts scheme in Geant4
- Using Geant4 physics

G4VUserPhysicsList
Concrete physics lists

Geant 4

278

G4ParticleDefinition

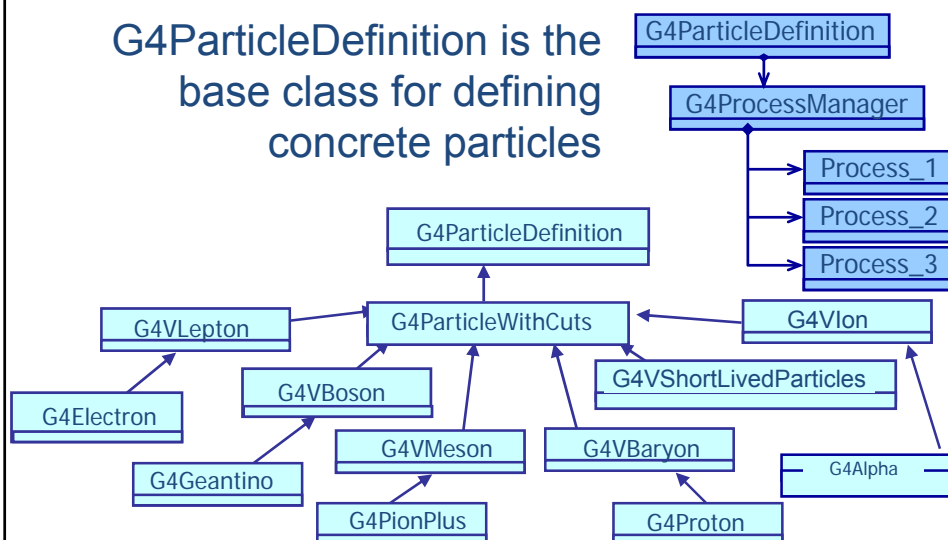
- Intrinsic particle properties
 - mass, width, spin, lifetime...
- Sensitivity to physics
- This is realized by **G4ProcessManager** attached to **G4ParticleDefinition**
- **G4ProcessManager** manages the list of processes the user wants the particle to be sensitive to
- **G4ParticleDefinition** does not know by itself its sensitivity to physics

Geant 4

279

Particles and Process design

G4ParticleDefinition is the base class for defining concrete particles



Geant 4

280

More about particle design

G4DynamicParticle

- Describes the purely dynamic part (*i.e. no position, nor geometrical information...*) of the particle state:
 - momentum, energy, polarization
- Holds a G4ParticleDefinition pointer
- Retains eventual pre-assigned decay information
 - decay products
 - lifetime

Geant 4

281

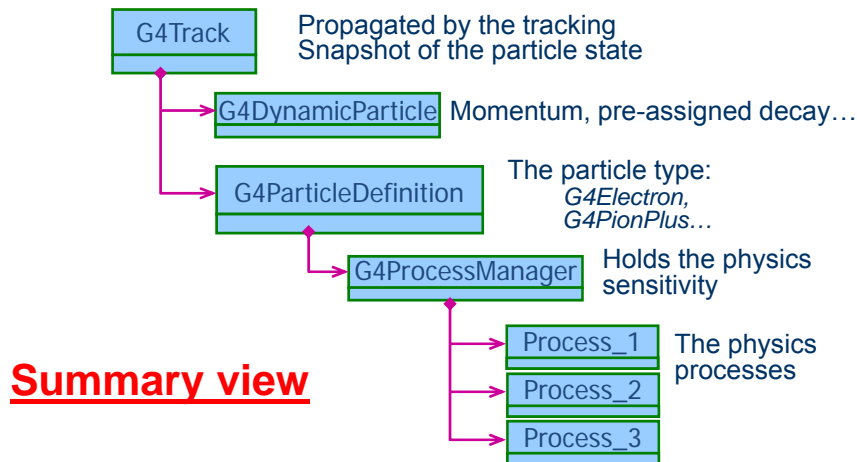
More about particle design

G4Track

- Defines the class of objects propagated by Geant4 tracking
- Represents a snapshot of the particle state
- Aggregates
 - a G4ParticleDefinition
 - a G4DynamicParticle
 - geometrical information:
 - *position, current volume ...*
 - track ID, parent ID;
 - process which created this G4Track
 - weight, used for event biasing

Geant 4

282



Geant 4

283

G4VProcess

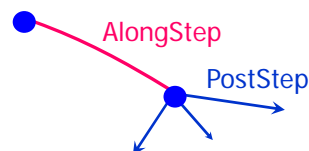
Abstract class defining the common interface of all processes in Geant4

Define three kinds of actions:

- **AtRest** actions: decay, annihilation ...
- **AlongStep** actions: continuous interactions occurring along the path, like ionisation
- **PostStep** actions: point-like interactions, like decay in flight, hard radiation...

A process can implement *any combination* of the three AtRest, AlongStep and PostStep actions:

- eg: decay = AtRest + PostStep

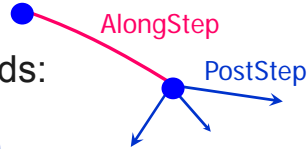


Geant 4

284

G4VProcess actions

- Each action defines two methods:
- **GetPhysicalInteractionLength()**
 - used to limit the step size
 - either because the process triggers an interaction or a decay
 - or in other cases, like fraction of energy loss, geometry boundary, user's limit...
- **DoIt()**
 - implements the actual action to be applied to the track
 - implements the related production of secondaries



Geant 4

285

Processes, ProcessManager and Stepping

- G4ProcessManager retains three vectors of actions:
 - one for the **AtRest** methods of the particle
 - one for the **AlongStep** ones
 - one for the **PostStep** actions
 - *these are the vectors which the user sets up in the PhysicsList and which are used by the tracking*
- The stepping treats processes generically
 - it does not know which process it is handling
- The stepping lets the processes
 - cooperate for **AlongStep** actions
 - compete for **PostStep** and **AtRest** actions
- Processes emit also signals to require particular treatment:
 - **notForced**: normal case
 - **forced**: PostStepDoIt action applied anyway;
 - **conditionallyForced**: PostStepDoIt applied if AlongStep has limited the step

Geant 4

Invocation sequence of processes: particle in flight

- At the beginning of the step, determine the **step length**
 - consider all processes attached to the current G4Track
 - define the step length as the smallest of the lengths among
 - all AlongStepGetPhysicalInteractionLength()
 - all PostStepGetPhysicalInteractionLength()
- Apply all **AlongStepDoIt()** actions **at once**
 - changes computed from particle state at the beginning of the step
 - accumulated in G4Step
 - then applied to G4Track, by G4Step
- Apply **PostStepDoIt()** action(s) **sequentially**, as long as the particle is alive
 - apply PostStepDoIt() of the process which proposed the smallest step length
 - apply *forced* and *conditionally forced* actions

Geant 4

287

Invocation sequence of processes: particle at rest

- If the particle is at rest, is stable and cannot annihilate, it is **killed** by tracking
 - more properly said: if a particle at rest has no *AtRest* actions defined, it is killed
- Otherwise determine the **lifetime**
 - Take the smallest time among all AtRestGetPhysicalInteractionLength()
 - Called *physical interaction length*, but it returns a time
- Apply the **AtRestDoIt()** action of the process which returned the smallest time

Geant 4

288

Processes ordering

- Ordering of some processes is critical:

- assuming n processes, the ordering of the `AlongGetPhysicalInteractionLength` of the last processes should be:

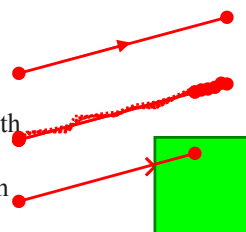
[$n-2$] ...

[$n-1$] multiple scattering

[n] transportation

- Why ?

- Processes return a *true path length*
- Multiple scattering virtually folds up this true path length into a shorter *geometrical path length*
- Based on this new length, the transportation process can geometrically limit the step



- Other processes ordering usually does not matter

Geant 4

289

Cuts in Geant4

- In Geant4 there are **no tracking cuts**

- particles are tracked down to a zero range/kinetic energy

- Only **production cuts** exist

- i.e. cuts allowing a particle to be born or not

Why are production cuts needed ?

- Some electromagnetic processes involve **infrared divergences**

- this leads to an infinity [huge number] of smaller and smaller energy photons/electrons (*such as in Bremsstrahlung, δ -ray production*)
- production cuts limit this production to particles above the threshold
- the remaining, divergent part is treated as a continuous effect (i.e. *AlongStep action*)

Geant 4

290

Range vs. energy production cuts

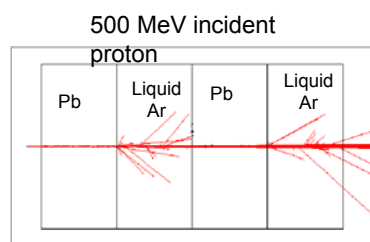
- The production of a secondary particle is relevant if it can generate visible effects in the detector
 - otherwise “local energy deposit”
- A **range cut** allows one to easily define such visibility
 - “I want to produce particles able to travel at least 1 mm”
 - criterion which can be applied uniformly across the detector (whole or “region”)
- The same **energy cut** leads to very different ranges
 - for the same particle type, depending on the material
 - for the same material, depending on particle type
- The user specifies a range cut in the PhysicsList
 - this range cut is converted into energy cuts
 - each particle (*G4ParticleWithCut*) converts the range cut into an energy cut, for each material
 - processes then compute the cross-sections based on the energy cut

Geant 4

291

Effect of production thresholds

Geant 4

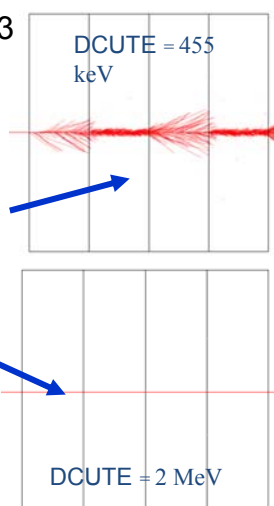


Threshold in range: 1.5 mm

455 keV electron energy in liquid Ar
2 MeV electron energy in Pb

In GEANT 3

one must set the cut for delta-rays (DCUTE) either to the Liquid Argon value, thus producing many small ~~under the Pb~~ ~~values in Pb,~~ killing the δ -rays production everywhere



Geant 4

292

Violations of the production threshold

- In some cases particles are produced even if they are *below the production threshold*
- This is intended to let the processes *do the best they can*
- It happens typically for
 - decays
 - positron production:
 - to simulate the photons resulting from annihilation
 - hadronic processes:
 - since no infrared divergence affects the cross-sections
- *Note these are not “hard-coded” exceptions, but a sophisticated, generic mechanism of the tracking*

Geant 4

293

G4VUserPhysicsList

- It is one of the mandatory user classes (*abstract class*)
- Pure virtual methods
 - *ConstructParticles()*
 - *ConstructProcesses()*
 - *SetCuts()*

to be implemented by the user in his/her concrete derived class

Geant 4

294

Electromagnetic physics

energy loss

- electrons and positrons
- γ , X-ray and optical photons
- muons
- charged hadrons
- ions

Comparable to Geant3 already in the α release (1997)

Further extensions (facilitated by the OO technology)

- **High energy extensions**
 - needed for LHC experiments, cosmic ray experiments...
- **Low energy extensions**
 - fundamental for space and medical applications, dark matter and ν experiments, antimatter spectroscopy etc.
- **Alternative models for the same process**

All obeying to the same abstract Process interface → transparent to tracking

Geant 4

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



Hadronic physics

- Completely different approach w.r.t. the past (GEANT 3)
 - native
 - transparent
 - no longer interface to external packages
 - clear separation between data and their use in algorithms
- Cross section data sets
 - transparent and interchangeable
- Final state calculation
 - models by particle, energy, material
- Ample variety of models
 - An extensive collection of hadronic interaction models
 - Alternative/complementary models
 - it is possible to mix-and-match, with fine granularity
 - data-driven, parameterised and theoretical models
- Consequences for the users
 - no more confined to the black box of one package
 - the user has control on the physics used in the simulation, which contributes to the validation of experiment's results

Geant 4

296

Summary

- **Transparency** and **modularity** are key characteristics of Geant4 physics
- Ample variety of processes and models
 - Open to extension and evolution thanks to the OO technology
- The PhysicsList exposes, **deliberately**, the user to the **choice** of physics (*particles + processes*) relevant to his/her application
 - This is a critical task, but guided by the framework
 - Examples can be used as starting point
- Physics processes and models are documented in Geant4 Physics Reference Manual and journal publications

Geant 4

297

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

This course encompasses training material developed by several Geant4 members:

thanks to all of them!

Geant 4

Electromagnetic Physics

Overview

Due to time constraints, this Short Course illustrates only basic concepts and highlights of functionality of Geant4 electromagnetic physics

Further details in Geant4 documentation and journal publications

Validation: *Geant4 Physics Validation*

Geant 4

Electromagnetic packages in Geant4

- Standard
- Low Energy
- Optical
- Muons

- Different modeling approach
- Specialized according to particle type, energy scope

Geant 4

300

Electromagnetic physics

- electrons and positrons
 - γ , X-ray and optical photons
 - muons
 - charged hadrons
 - ions
- High energy extensions
 - needed for LHC experiments, cosmic ray experiments...
 - Low energy extensions
 - fundamental for space and medical applications, dark matter and ν experiments, antimatter spectroscopy etc.
 - Alternative models for the same process
- Multiple scattering
 - Bremsstrahlung
 - Ionisation
 - Annihilation
 - Photoelectric effect
 - Compton scattering
 - Rayleigh scattering
 - γ conversion
 - e^+e^- pair production
 - Synchrotron radiation
 - Transition radiation
 - Cherenkov
 - Refraction
 - Reflection
 - Absorption
 - Scintillation
 - Fluorescence
 - Auger emission

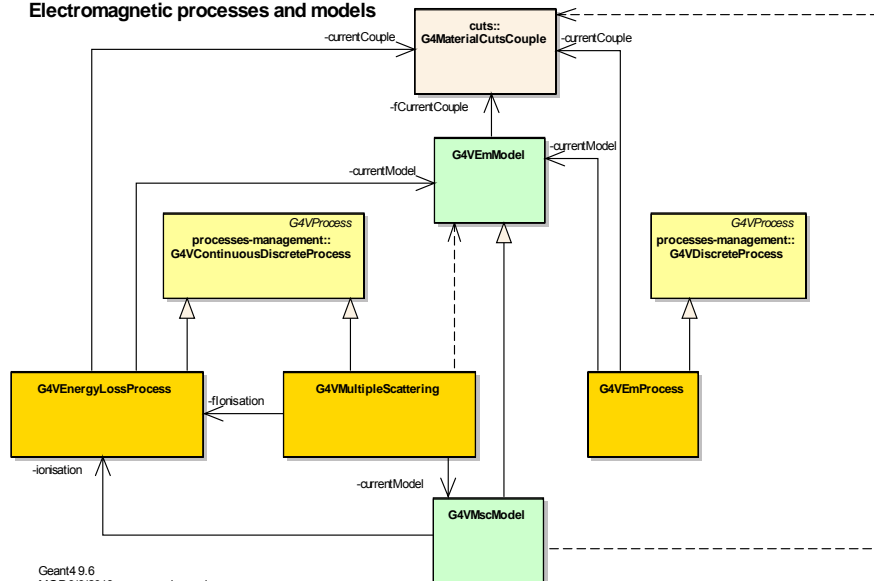
All obeying to the same abstract Process interface: transparent to tracking

Geant 4

301

Standard electromagnetic design

Electromagnetic processes and models



Geant4 9.6
MCP 9/9/2013 reverse engineered

Geant 4

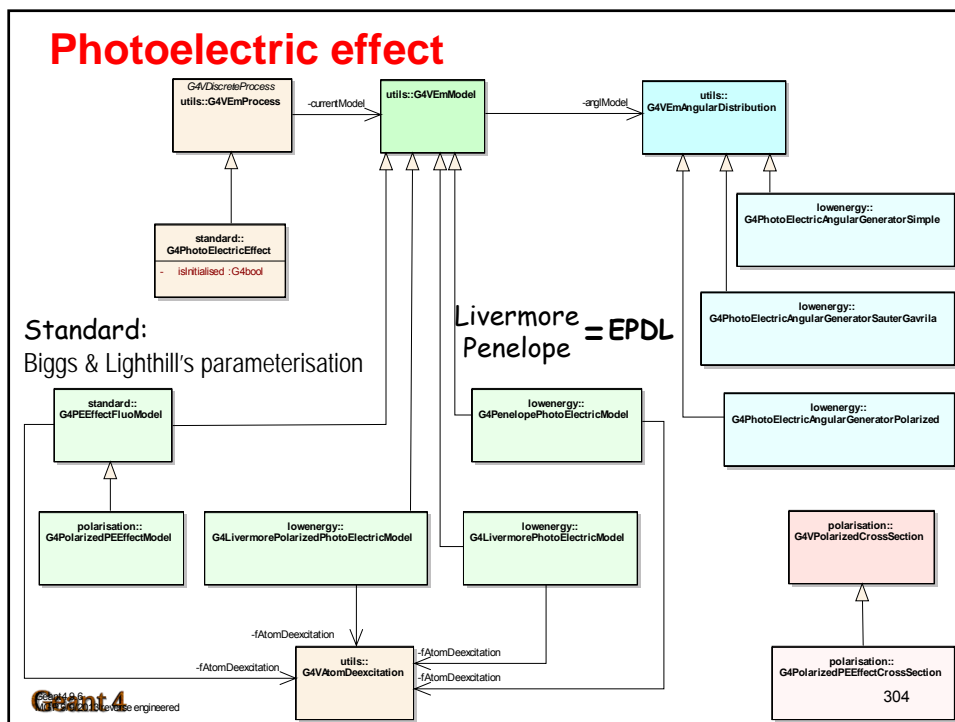
302

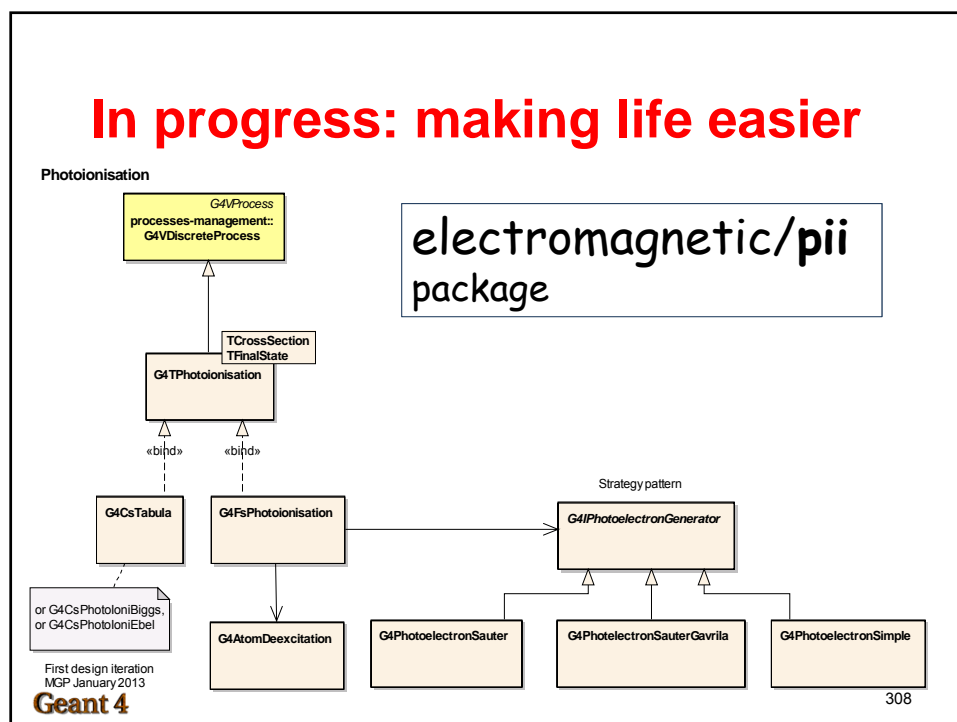
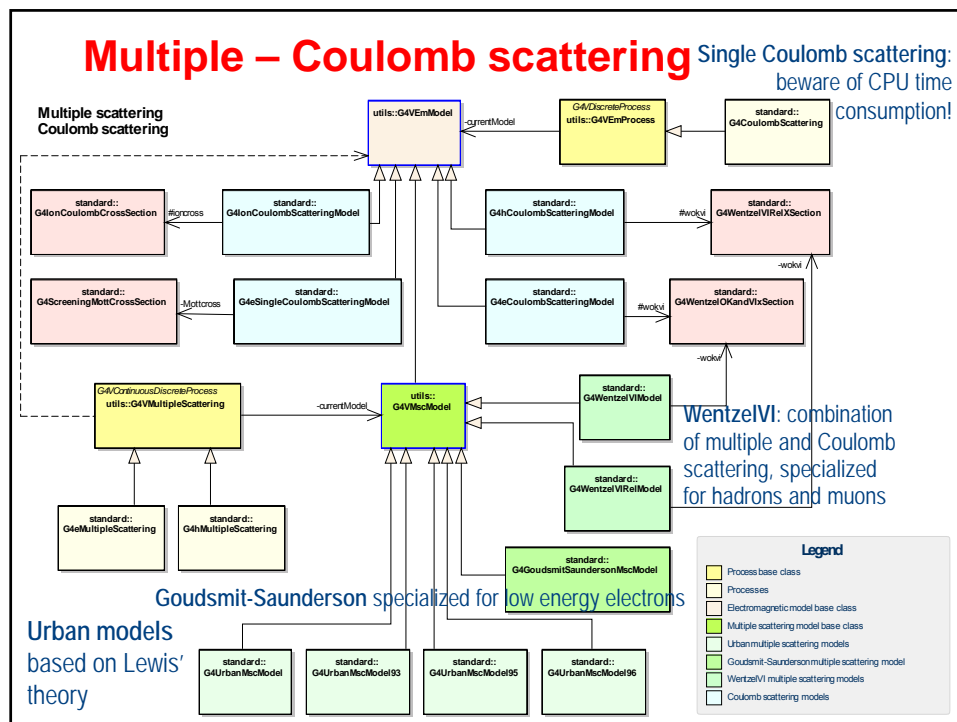
In detail



Geant 4

Photoelectric effect





Standard Electromagnetic Physics

Overview

Geant 4

Standard electromagnetic physics Packages

Package	Scope
Standard	Photons, electrons, positrons up to 100 TeV Hadrons, ions up to 100 TeV
Muons	Muons up to 1 PeV
X-rays	X-ray and optical photon production
Optical	Optical photon interactions
High-energy	Processes at high energy ($E > 10$ GeV) Physics for exotic particles
Polarisation	Simulation for polarized beams

Geant 4

310

Processes: electrons and photons

Particle	Interactions
Photons	γ conversion Compton scattering Photoelectric effect
Electrons, positrons	Ionisation Bremsstrahlung Coulomb scattering
Positrons	Annihilation

Geant 4

311

Processes: hadrons

Particle	Processes
Hadrons, Ions	Coulomb Scattering, Ionization

$$\frac{dE}{dx} = 2\pi r_e^2 m c^2 n_{el} \frac{z^2}{\beta^2} \left[\ln \left(\frac{2mc^2 \beta^2 \gamma^2 T_{up}}{I^2} \right) - \beta^2 \left(1 + \frac{T_{up}}{T_{max}} \right) - \delta - \frac{2C_e}{Z} + F \right]$$

- Ionization:
 - $E > 2$ MeV: Bethe-Bloch formula with corrections
 - $E < 2$ MeV: Parametrization (ICRU 49, NIST)

Ex: Proton Bragg peak

The graph shows energy deposition in arbitrary units (a.u.) on the y-axis (ranging from 0 to 0.12) against depth in water in millimeters (mm) on the x-axis (ranging from 0 to 35). A blue curve rises gradually from 0 at 0 mm to about 0.04 at 25 mm, then rises sharply to a peak of approximately 0.11 at 30 mm, before dropping to zero. The peak is labeled 'Proton Bragg peak'.

Geant 4

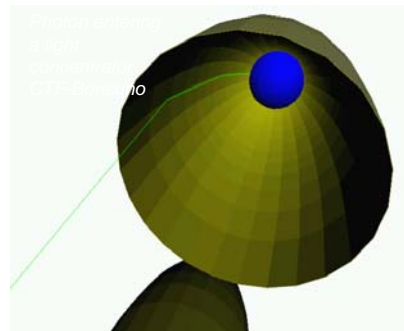
312

Optical photons

Production of optical photons in detectors is mainly due to Cherenkov effect and scintillation

Processes in Geant4:

- in-flight absorption
- Rayleigh scattering
- medium-boundary interactions (reflection, refraction)



Geant 4

313

Muons

1 keV up to 1000 PeV scale

- simulation of ultra-high energy and cosmic ray physics
- High energy extensions based on theoretical models

P. Arce et al, **Matter deviation of 45 GeV muons in GEANT3 and GEANT4 : a comparison with L3 data**
CMS-NOTE-2000-016

Geant 4

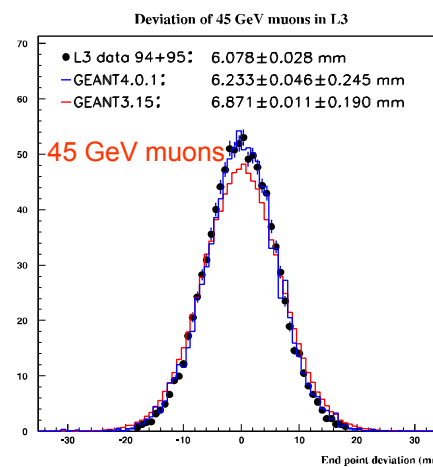
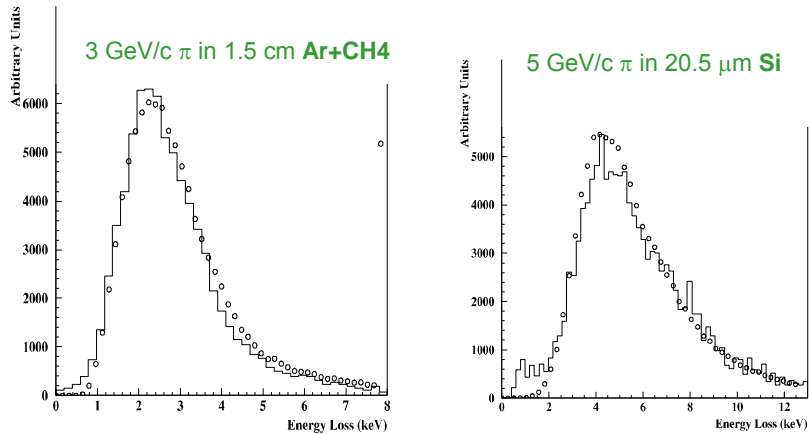


Photo Absorption Ionisation (PAI) Model

Ionisation energy loss produced by charged particles in thin layers of absorbers



Ionisation energy loss distribution produced by pions, PAI model

Geant 4

315

Multiple scattering

- Geant4 is (mostly) a **condensed** Monte Carlo code
 - Global effects due to collisions along a macroscopic step are simulated, using approximations
- Is in contrast to detailed simulations at a microscopic level, which are exact, but require an enormous amount of time for higher energies
- **Cumulative effects**
 - Charged particles, that travel through a finite material layer are subject to a large number of Coloumb interactions, where the particles are elastically scattered
 - Summing up individual effects gives a net deflection of the particles from their initial direction
- For sufficient collisions (>20) multiple scattering angular distribution is
 - a **Gaussian** distribution for small angles
 - like **Rutherford scattering** at large angles

Geant 4

316

Multiple scattering

- The Geant4 Multiple Scattering (MSC) model by L. Urban is applicable to all charged particles
- It is based on Lewis' theory
- Relies on transport equation of charged particles
- Uses phenomenological functions to sample angular and spatial distributions after the simulation step
 - The function parameters are chosen, in order that the moments of the distribution are the same as given by Lewis' theory
- See the *Physics Reference Manual* for details
- Other multiple scattering models specialized for hadrons/muons, low energy electrons etc.

Geant 4 Further details in *Geant4 Physics Validation Refresher*

High energy knock-on electrons

- Above a given threshold T_{cut} high energy knock-on electrons are explicitly taken into account
 - i.e. δ -rays are explicitly generated (particles excluded from continuous energy loss)
- Below T_{cut} the soft electrons are only considered as continuous energy loss of the incident particle

Geant 4

318

Energy loss fluctuations

- Urban model of fluctuations,
 - based on a simple modelling approach of particle-atom interactions:
- Atoms are assumed to have only two energy levels: E1 and E2
- The particle-atom interaction can be:
 - excitation of the atom with energy loss $E = E_1 - E_2$
 - ionization with energy loss distribution $g(E) \sim 1/E^2$
- PAI model uses photo absorption data
- All energy transfers are sampled with production of secondary electrons and photons
 - Slow model, should only be applied for sensitive region

Geant 4

319

Low Energy Electromagnetic Physics

Overview

Geant 4

What is

- A package in the Geant4 electromagnetic package
 - *geant4/source/processes/electromagnetic/lowenergy/*
- A set of processes extending the coverage of electromagnetic interactions in Geant4 down to “low” energy
 - 250 eV (*in principle even below this limit*) for electrons and photons
 - down to the approximately the ionisation potential of the interacting material for hadrons and ions
- A set of processes based on detailed models
 - shell structure of the atom
 - precise angular distributions
- Complementary to the “standard” electromagnetic package

Geant 4

321

Low energy electrons and photons

- Two “flavours” of models:
 - based on the **EEDL/EPDL/EADL** data libraries (AKA “Livermore Library”)
 - Compton scattering
 - Rayleigh scattering
 - Photoelectric effect
 - Pair production
 - **à la Penelope**
 - Bremsstrahlung
 - Ionisation
 - Nominally down
 - to 250 eV
 - based on the “Livermore” library
 - to a “few hundred eV”
 - Penelope-like
 - Polarised processes
 - + atomic relaxation
 - fluorescence
 - Auger electron emission following processes leaving a vacancy in an atom
- EADL (Evaluated Atomic Data Library)
 EEDL (Evaluated Electrons Data Library)
 EPDL97 (Evaluated Photons Data Library)
 especially formatted for Geant4 distribution
 (courtesy of D. Cullen, LLNL)

Geant 4

Calculation of cross sections

Interpolation from the data libraries:

$$\log(\sigma(E)) = \frac{\log(\sigma_1)\log(E_2/E) + \log(\sigma_2)\log(E/E_1)}{\log(E_2/E_1)}$$

E_1 and E_2 are the lower and higher energy for which data (σ_1 and σ_2) are available

Mean free path for a process, at energy E:

$$\lambda = \frac{1}{\sum_i \sigma_i(E) \cdot n_i}$$

n_i = atomic density of the i^{th} element contributing to the material composition

Geant 4

323

Compton scattering

Klein-Nishina cross section:
$$\frac{d\sigma}{d\Omega} = \frac{1}{4} r_0^2 \frac{h\nu^2}{h\nu_0^2} \left[\frac{h\nu_0}{h\nu} + \frac{h\nu}{h\nu_0} - 2 + 4\cos^2\Theta \right]$$

- Energy distribution of the scattered photon according to the *Klein-Nishina formula*, multiplied by **scattering function** $F(q)$ from EPDL97 data library
- The effect of scattering function becomes significant at low energies
 - suppresses forward scattering
- Angular distribution of the scattered photon and the recoil electron also based on EPDL97

Geant 4

324

Rayleigh scattering

- Angular distribution: $F(E,q)=[1+\cos^2(q)]\cdot F^2(q)$
 - where $F(q)$ is the energy-dependent **form factor** obtained from EPDL97
- This process is only available in the *lowenergy* package
 - Not available in the *standard* package

Geant 4

325

Photoelectric effect

- Cross section
 - Integrated cross section (over the shells) from EPDL + interpolation
 - Shell from which the electron is emitted selected according to the detailed cross sections of the EPDL library
- Final state generation
 - Various angular distribution generators (“naïve”, Sauter-Gavrila, Gavrila)
- Deexcitation via the atomic relaxation sub-process
 - Initial vacancy + following chain of vacancies created
- *Improved angular distribution in preparation*

Geant 4

326

γ conversion

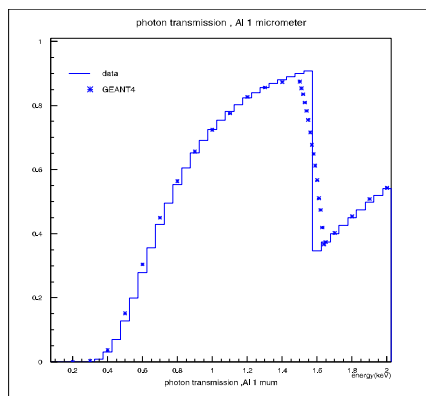
- The secondary e^- and e^+ energies are sampled using Bethe-Heitler cross sections with Coulomb correction
- e^- and e^+ assumed to have symmetric angular distribution
- Energy and polar angle sampled w.r.t. the incoming photon using Tsai differential cross section
- Azimuthal angle generated isotropically
- Choice of which particle in the pair is e^- or e^+ is made randomly

Geant 4

327

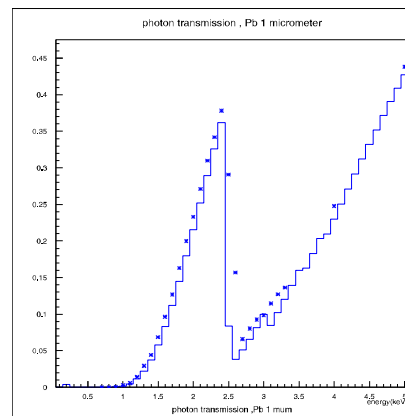
Photons, evidence of shell effects

Photon transmission, 1 μm Al



Geant 4

Photon transmission, 1 μm Pb



328

Polarisation

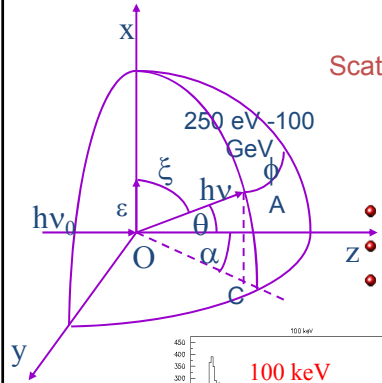
Cross
section:

$$\frac{d\sigma}{d\Omega} = \frac{1}{2} r_0^2 \frac{h\nu^2}{h\nu_0^2} \left[\frac{h\nu_0}{h\nu} + \frac{h\nu}{h\nu_0} - 2 \sin^2 \theta \cos^2 \phi \right]$$

$$\cos \xi = \sin \theta \cos \phi \Rightarrow \sin \xi = \sqrt{1 - \sin^2 \theta \cos^2 \phi} = N$$

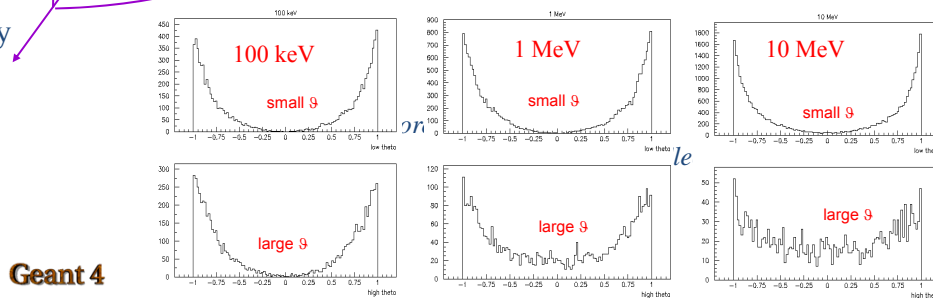
Scattered Photon Polarization $\vec{\epsilon}_\perp = \frac{1}{N} (\cos \theta \hat{j} - \sin \theta \sin \phi \hat{k}) \sin \beta$

$$\vec{\epsilon}_\parallel = \left(N \hat{i} - \frac{1}{N} \sin^2 \theta \sin \phi \cos \phi \hat{j} - \frac{1}{N} \sin \theta \cos \theta \cos \phi \hat{k} \right) \cos \beta$$



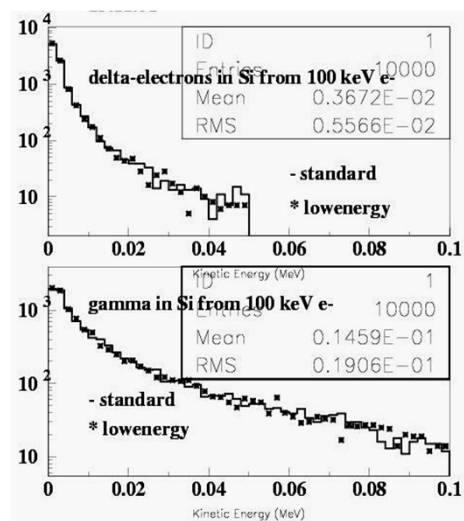
- θ Polar angle
- ϕ Azimuthal angle
- ϵ Polarization vector

Low Energy
Polarised Compton



Electron Bremsstrahlung

- Parameterisation of EEDL data
 - 16 parameters for each atom
 - At high energy the parameterisation reproduces the Bethe-Heitler formula
 - Precision is $\sim 1.5\%$



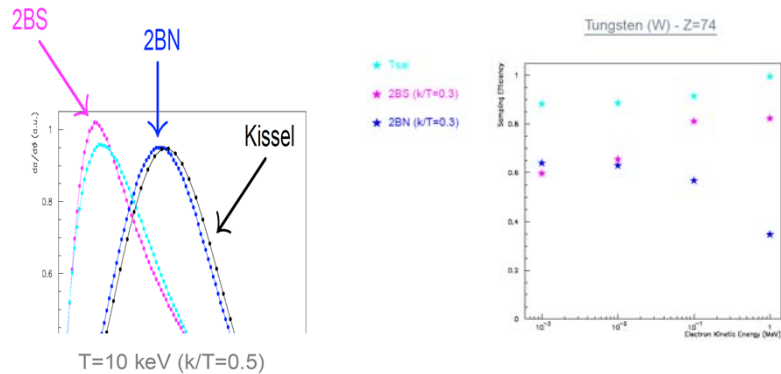
330

Bremsstrahlung Angular Distributions

Three LowE generators available in GEANT4 :

G4ModifiedTsai, G4Generator2BS and G4Generator2BN

G4Generator2BN allows a correct treatment at low energies (< 500 keV)

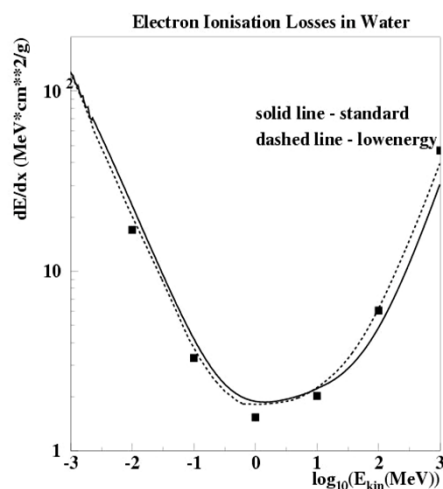


Geant 4

331

Electron ionisation

- Parameterisation based on 5 parameters for each shell
- Precision of parametrisation is better than 5% for 50 % of shells, less accurate for the remaining shells



Geant 4

332

Processes à la Penelope

- The whole physics content of the Penelope Monte Carlo code has been re-engineered into Geant4 (*except for multiple scattering*)
 - As in Penelope 2001: processes for photons since release 5.2, for electrons since release 6.0
 - Currently as in Penelope 2008
- Physics models by F. Salvat et al.
- Power of the OO technology:
 - extending the software system is easy
 - all processes obey to the same abstract interfaces
 - using new implementations in application code is simple
- Profit of Geant4 advanced geometry modeling, interactive facilities *etc.*
 - same physics as original Penelope

Differences and similarities between Penelope-like and other Geant4 modeling options discussed in Refresher

Geant 4

333

Hadrons and ions

- Variety of models, depending on
 - energy range
 - particle type
 - charge
- Composition of models across the energy range, with different approaches
 - analytical
 - based on data reviews + parameterisations
- Specialised models for fluctuations
- Open to extension and evolution

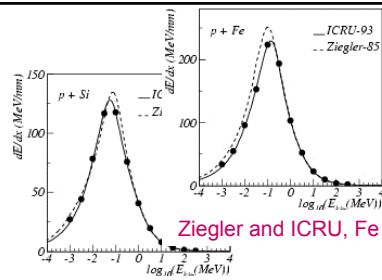
Geant 4

334

Positive charged hadrons

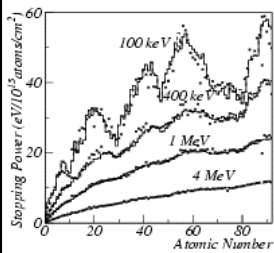
- Bethe-Bloch model of energy loss, $E > 2$ MeV
- 5 parameterisation models, $E < 2$ MeV
 - based on Ziegler and ICRU reviews
- 3 models of energy loss fluctuations

- Density correction for high energy
- Shell correction term for intermediate energy



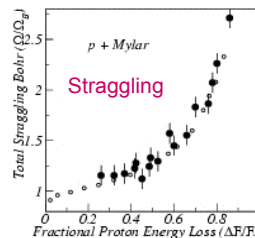
Ziegler and ICRU, Si

- Spin dependent term
- Barkas and Bloch terms

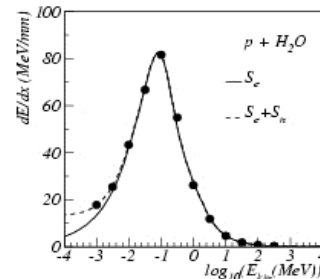


Stopping power
Z dependence for various energies
Ziegler and ICRU models

- Chemical effect for compounds
- Nuclear stopping power
- PIXE included



Straggling



Nuclear stopping power

Positive charged ions

Implementation of ICRU73-based model and comparison with experimental data (A. Lechner et al.)

- Scaling: $S_{ion}(T) = Z_{ion}^2 S_p(T_p)$, $T_p = T \frac{m_p}{m_{ion}}$
- $0.01 < \beta < 0.05$ parameterisations, Bragg peak
 - based on Ziegler and ICRU reviews
- $\beta < 0.01$: Free Electron Gas Model

- Effective charge model
- Nuclear stopping power

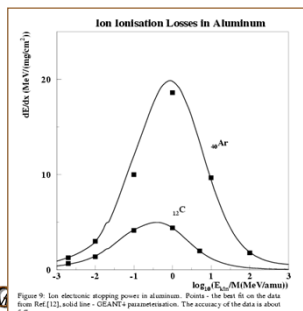
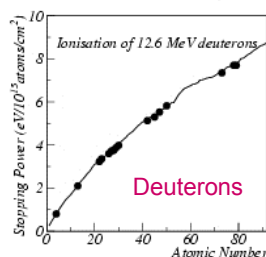
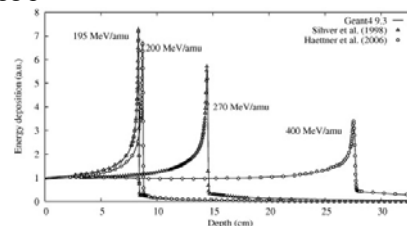


Figure 9: Ion electronic stopping power in aluminum. Points - the best fit on the data from Ref [12], solid line - GEANT4 parameterisation. The accuracy of the data is about 10%.



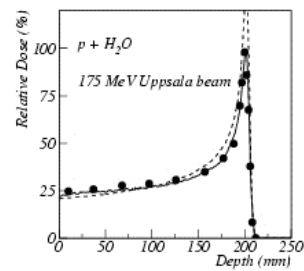
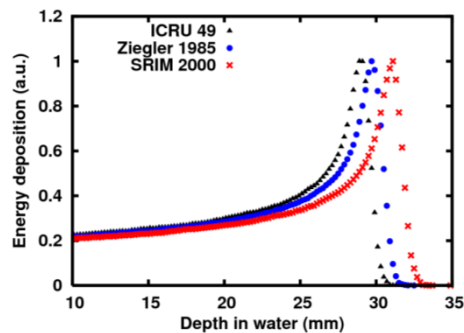
Deuterons



Comparison of simulated and measured ^{12}C depth-dose profiles in water (0.997 g/cm^3). Simulations were performed with Geant4 9.3, using revised ICRU 73 stopping power tables and the QMD nuclear reaction model. Experimental data derive from Silver et al. (triangles) and Haettner et al. (circles), where profiles of Haettner et al. were shifted to match more precise measurements of the peak position by D. Schardt et al. All experimental data by courtesy of D. Schardt.

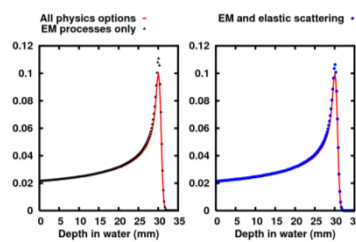
A. Lechner et al., NIM B 268-14 (2010) 2343-2354

336



Bragg peak

(with hadronic interactions)



More in Refresher Course

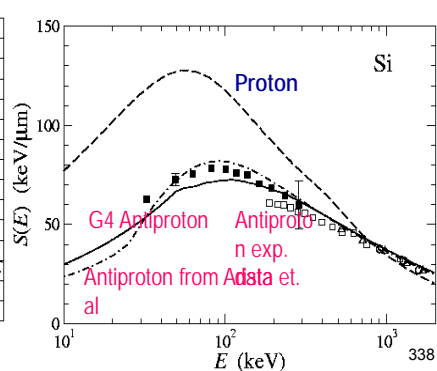
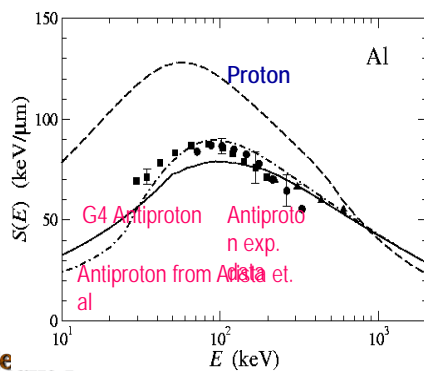
M. G. Pia et al.
**Physics-related epistemic
 uncertainties of proton depth dose
 simulation**
IEEE Trans. Nucl. Sci., vol. 57, no. 5, pp.
 2805-2830, 2010

Geant 4

337

Models for antiprotons


- $\beta > 0.5$ Bethe-Bloch formula
- $0.01 < \beta < 0.5$ Quantum harmonic oscillator model
- $\beta < 0.01$ Free electron gas mode



Ge

338

IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 54, NO. 3, JUNE 2007	585
<p align="center">Geant4 Atomic Relaxation 9 pages</p> <p align="center">Susanna Guatelli, Alfonso Mantero, Barbara Mascialino, Petteri Nieminen, and Maria Grazia Pia</p>	
594	IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 54, NO. 3, JUNE 2007
<p align="center">Validation of Geant4 Atomic Relaxation Against the NIST Physical Reference Data 10 pages</p> <p align="center">S. Guatelli, A. Mantero, B. Mascialino, M. G. Pia, and V. Zampichelli</p>	
3650	IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 56, NO. 6, DECEMBER 2009
<p align="center">Validation of K and L Shell Radiative Transition Probability Calculations 12 pages</p> <p align="center">Maria Grazia Pia, Paolo Saracco, and Manju Sudhakar</p>	
3614	IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 56, NO. 6, DECEMBER 2009
<p align="center">PIXE Simulation With Geant4 36 pages</p> <p align="center">Maria Grazia Pia, Georg Weidenspointner, Mauro Augelli, Lina Quintieri, Paolo Saracco, Manju Sudhakar, and Andreas Zoglauer</p>	
Geant 4	339

IEEE TRANSACTIONS ON NUCLEAR SCIENCE	IEEE Trans. Nucl. Sci., vol. 58, no. 6, December 2011
<p align="center">Validation of Proton Ionization Cross Section Generators for Monte Carlo Particle Transport</p> <p align="center">Matej Batič, Maria Grazia Pia, and Paolo Saracco</p>	
	<p align="center">Contents lists available at SciVerse ScienceDirect</p> <p align="center">Computer Physics Communications</p> <p align="center">www.elsevier.com/locate/cpc</p>
<p>ISICSoo: A class for the calculation of ionization cross sections from ECPSSR and PWBA theory [☆]</p> <p>Matej Batič ^{a,b,*}, Maria Grazia Pia ^a, Sam J. Cipolla ^c</p>	
IEEE TRANSACTIONS ON NUCLEAR SCIENCE	IEEE Trans. Nucl. Sci., vol. 58, no. 6, December 2011
<p align="center">Evaluation of atomic electron binding energies for Monte Carlo particle transport</p> <p align="center">Maria Grazia Pia, Hee Seo, Matej Batič, Marcia Begalli, Chan Hyeon Kim, Lina Quintieri and Paolo Saracco</p>	
Geant 4	340

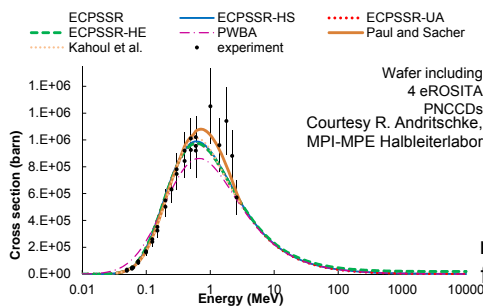
PIXE

3614 IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 56, NO. 6, DECEMBER 2009

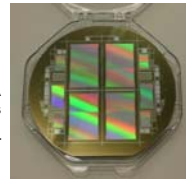
PIXE Simulation With Geant4

Maria Grazia Pia, Georg Weidenspointner, Mauro Augelli, Lina Quintieri, Paolo Saracco, Manju Sudhakar, and Andreas Zoglauer

- Critical evaluation of **conceptual challenges**
- Wide collection of ionisation **cross section** models
- **Validation** and **comparative evaluation** of theoretical and empirical cross sections



Geant 4



Software applied to a real-life problem: X-ray full-sky survey mission eROSITA

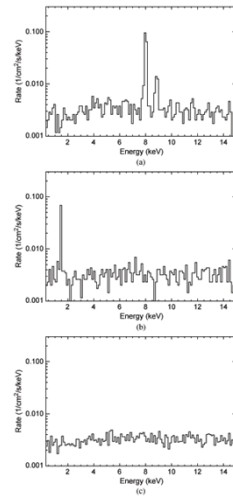


Fig. 12. A comparison of the fluorescence background due to ionization by cosmic-ray protons in an L2 unit for three different graded Z shield designs for the eROSITA X-ray detector. (a) Cu shield, (b) Cu-Al shield, (c) Cu-Al-B₄C shield.

Other implementation released in Geant4 9.2: several flaws



Geant4 comparison vs. NIST database

- All Geant4 physics models of electrons, photons, protons and α compared to NIST database
 - Photoelectric, Compton, Rayleigh, Pair Production cross-sections
 - Photon attenuation coefficients
 - Electron, proton, α stopping power and range
- Quantitative comparison
 - Statistical goodness-of-fit tests
- **NOT** validation

Further details in Geant4 Physics Validation

Geant 4

342

Very-low energy extensions

Physics of interactions in water down to the eV scale

- **Complex domain**

- **Physics:** beyond IPA and isolated atom calculations
- **Technology:** innovative design technique introduced in Geant4
(1st time in Monte Carlo), then dropped

- **Experimental complexity as well**

- Scarce experimental data

Geant 4

343

Very-low energy extensions

1st development cycle:

Physics of interactions in water down to the eV scale

IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 54, NO. 6, DECEMBER 2007

2619

Geant4 Physics Processes for Microdosimetry Simulation: Design Foundation and Implementation of the First Set of Models

S. Chauvie, Z. Francis, S. Guatelli, S. Incerti, B. Mascialino, P. Moretto, P. Nieminen, and M. G. Pia

Further developments

Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2010 (SNA + MC2010)
Hitotsubashi Memorial Hall, Tokyo, Japan, October 17-21, 2010

Modeling Radiation Chemistry and Biology in the Geant4 Toolkit

M. Karamitros¹, A. Mantero², S. Incerti^{1*}, G. Baldacchino³, P. Barberet¹, M. Bernal^{4,5}, R. Capra⁶, C. Champion⁷, Z. El Bitar⁸,
Z. Francis⁹, W. Friedland¹⁰, P. Guèye¹¹, A. Ivanchenko¹, V. Ivanchenko^{7,12}, H. Kurashige¹³, B. Mascialino¹⁴, P. Moretto¹,
P. Nieminen¹⁵, G. Santin¹⁵, H. Seznec¹, H. N. Tran¹, C. Villagrasa⁹ and C. Zacharatou¹⁶

Still consistent with transport assumptions?

Geant 4

344

Geant4-DNA physics processes

Specialised processes for low energy interactions with water

Models in liquid water

- More realistic than water vapour
- Theoretically more challenging
- Hardly any experimental data
- New measurements needed

Toolkit: offer a wide choice among available alternative models for each process

Particle	Processes
e ⁻	Elastic scattering Excitation Ionisation
p	Charge decrease Excitation Ionisation
H	Charge increase Ionisation
He ⁺⁺	Charge decrease Excitation Ionisation
He ⁺	Charge decrease Charge increase Excitation Ionisation
He	Charge increase Excitation Ionisation

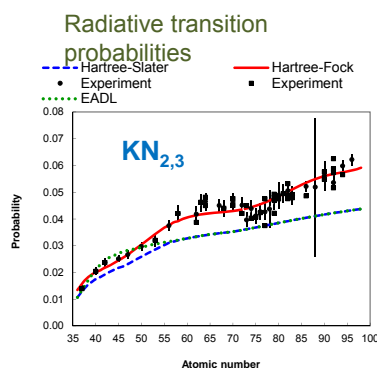
Geant 4

Atomic parameters

Geant4 Atomic Relaxation: X-ray fluorescence + Auger electron emission

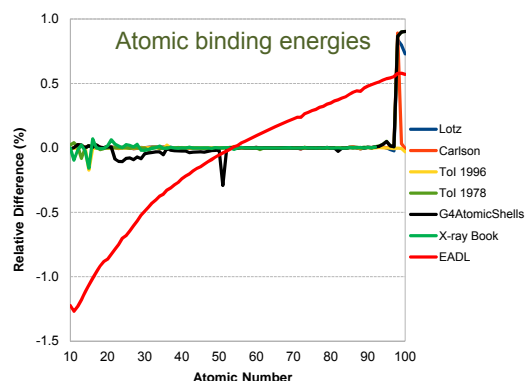
Data-driven → Based on **EADL** (Evaluated Atomic Data Library)

Geant4 X-ray fluorescence simulation is as good as EADL



Hartree-Slater and Hartree-Fock calculations compared to experiments

Geant 4



Difference w.r.t. DesLattes et al., experimental review

346

Novelties



pii

- ◆ **State-of-the-art photon physics**
 - Ample evaluation of established and not-yet explored modeling methods
 - Rigorous quantitative validation w.r.t. experimental data
 - Quantified computational performance
- ◆ **Electron impact ionisation down to the eV scale**
 - Validated against experiment
 - New models (BEB, DM, Bote)
- ◆ **Optimised atomic relaxation**
 - Physics and performance
- ◆ **Atomic parameters**
 - Consistent, validated basis for atomic interactions
- ◆ **Investigation of IPA and IA validity**

Geant 4

347

Ionisation models for nano-scale simulation

Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2010 (SNA + MC2010)
Hiroshima Memorial Hall, Tokyo, Japan, October 17-23, 2010

Design, development and validation of electron ionisation models for nano-scale simulation

Hee SEO¹, Maria Grazia Pia², Paolo SARACCO², Chan-Hyeon KIM³

¹ Hanyang University, 133-701 Seoul, Korea
² INFN Sezione di Genova, 16146 Genova, Italy

Best Student Award Monte Carlo 2010

IEEE TRANSACTIONS ON NUCLEAR SCIENCE

Ionization cross sections for low energy electron transport

Hee Seo, Maria Grazia Pia, Paolo Saracco and Chan Hyeon Kim

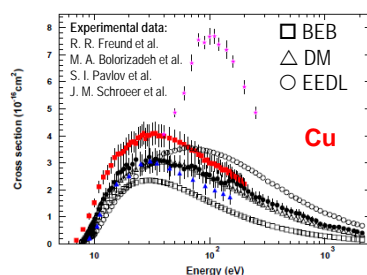
IEEE Trans. Nucl. Sci., vol. 58, no. 6, December 2011

Cross section models

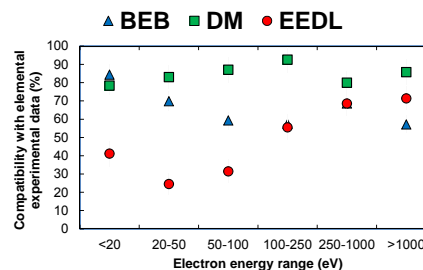
- Binary-Encounter-Bethe (BEB)
- Deutsch-Märk (DM)
- EEDL (Geant4 Low Energy)

Validation

- 57 elements
- 181 experimental data sets



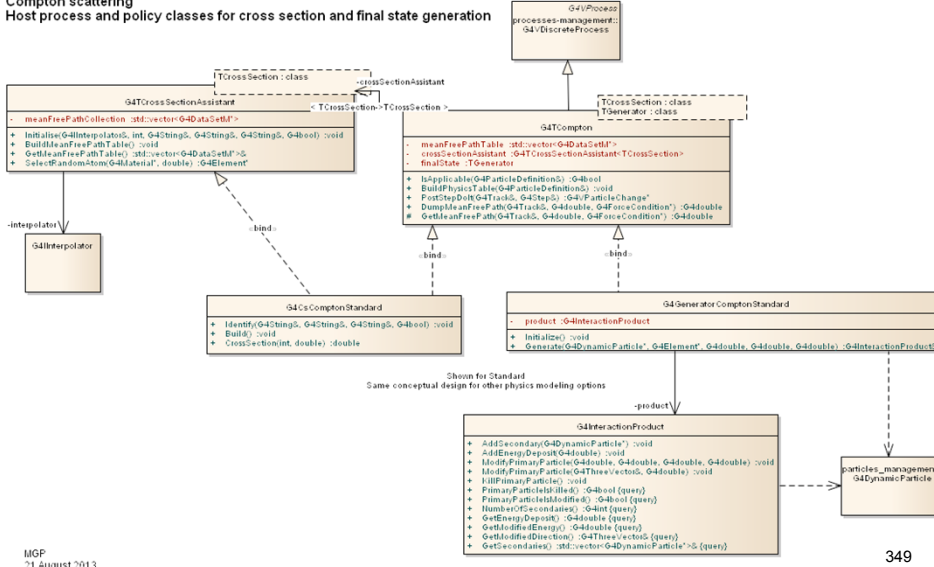
Geant 4



Percentage of elements for which a model is compatible with experimental data at 95% CL

Agility & simplicity

Compton scattering
Host process and policy classes for cross section and final state generation



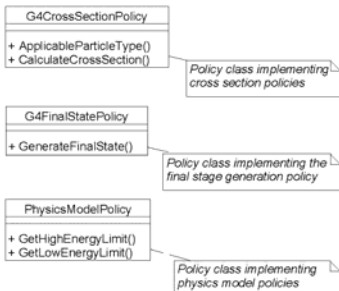
349

What is behind...

Policy-based class design

- A policy defines a class or class template interface
- Policy host classes are parameterised classes
 - classes that use other classes as a parameter
- Advantage w.r.t. a conventional strategy pattern
 - Policies are not required to inherit from a base class
 - The code is bound at compilation time
 - No need of virtual methods, resulting in faster execution

New
technique
1st time
introduced
in Monte
Carlo



Weak dependency of the policy and the policy based class on the policy interface

Highly **customizable** design
Open to extension

350

Summary

- OO technology provides the mechanism for a rich set of electromagnetic physics models in Geant4
 - further extensions and refinements are possible, without affecting Geant4 kernel or user code
 - Two main approaches in Geant4:
 - Standard package
 - Low Energy package
- both offering a variety of models for specialised applications
- Innovative and rigorous approach in **pii** package
 - Extensive validation activity and results
 - More on Physics Reference Manual and scholarly literature

Geant 4

351

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

This course encompasses training material developed by several Geant4 members:

thanks to all of them!

Geant 4

Geant4 Hadronic Models

Geant 4

Outline

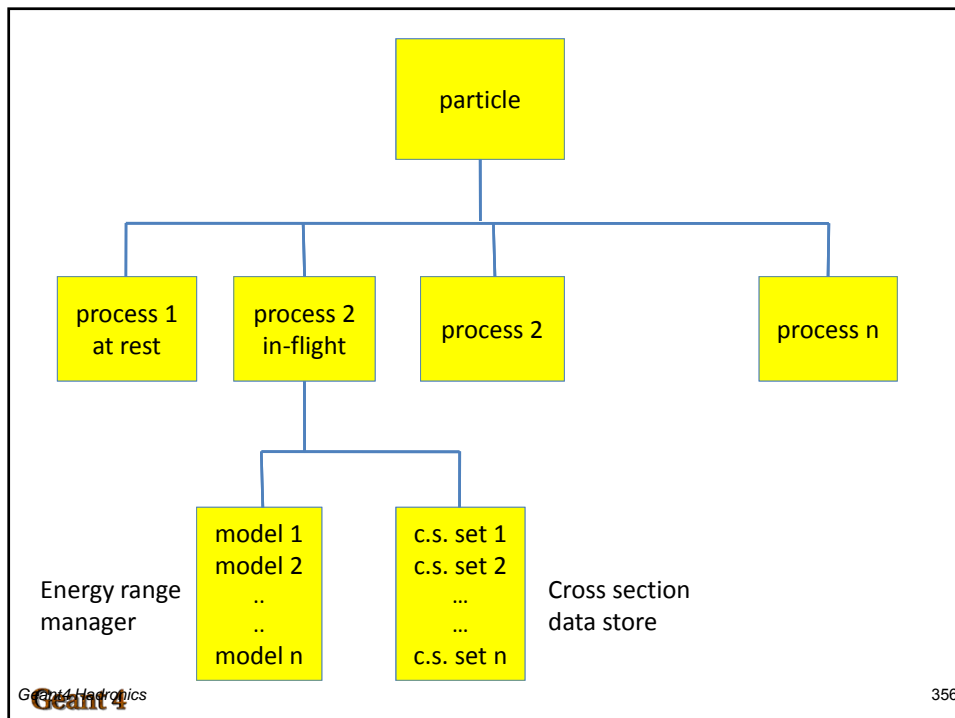
- Overview of hadronic physics
 - processes, cross sections, models, hadronic framework and organization
- Elastic scattering
- Precompound models
- Cascade models
 - Bertini-style, binary, INCL/ABLA
- Neutron physics
 - HP models, LEND models
- Ion-ion physics
 - Wilson abrasion/ablation, binary light ion cascade, INCL++, QMD
- Radioactive decay
- Gamma- and lepto-nuclear models
- QCD string models
 - Quark-gluon string (QGS) model, Fritiof (FTF) model
- Other Models
 - Capture, Fission

Geant 4

354

Hadronic Processes, Models and Cross Sections

- In Geant4 physics is assigned to a particle through **processes**
- Each process may be implemented
 - directly, as part of the process, or
 - in terms of a **model** class
- Geant4 often provides several models for a given process
 - user must choose
 - can, and sometimes must, have more than one per process
- A process must also have **cross sections** assigned
 - here too, there are options



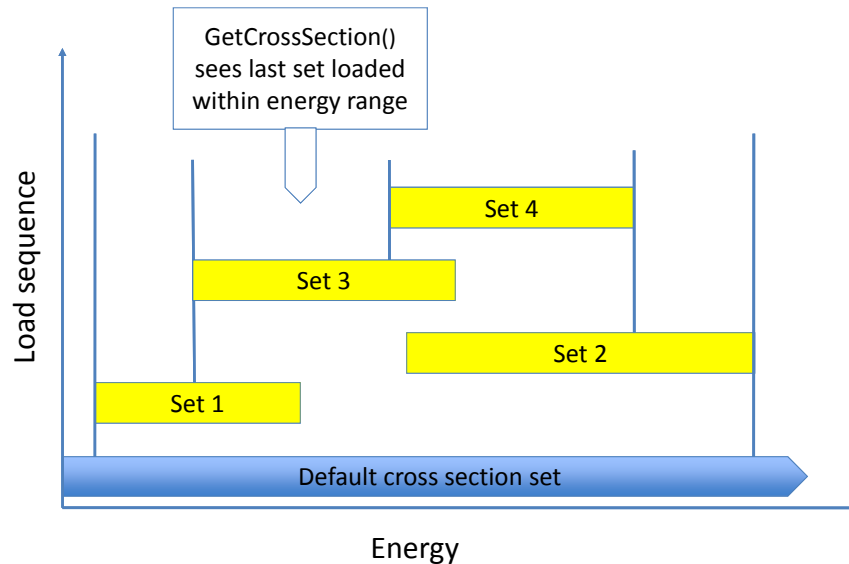
Cross Sections

- Default cross section sets are provided for each type of hadronic process
 - fission, capture, elastic, inelastic
 - can be overridden or completely replaced
- Different types of cross section sets
 - some contain only a few numbers to parameterize the cross section
 - some represent large databases
 - some are purely theoretical (equation-driven)

Alternative Cross Sections

- Low energy neutrons
 - G4NDL available as Geant4 distribution files
 - Livermore database (LEND) also available
 - available with or without thermal cross sections
- Medium energy neutron and proton reaction cross sections
 - $14 \text{ MeV} < E < 20 \text{ GeV}$
- Ion-nucleus reaction cross sections (several alternatives exist)
 - These are empirical and parameterized cross section formulae with some theoretical insight
 - good for $E/A < 10 \text{ GeV}$
- Pion reaction cross sections

Cross Section Management



Geant4 Hadronics

359

Data-driven Hadronic Models

- Characterized by lots of data
 - cross sections
 - angular distributions
 - multiplicities, etc.
- To get interaction length and final state, models depend on interpolation of data
 - cross sections, Legendre coefficients
- Examples
 - neutrons ($E < 20$ MeV)
 - coherent elastic scattering (pp, np, nn)
 - radioactive decay

Geant4 Hadronics

360

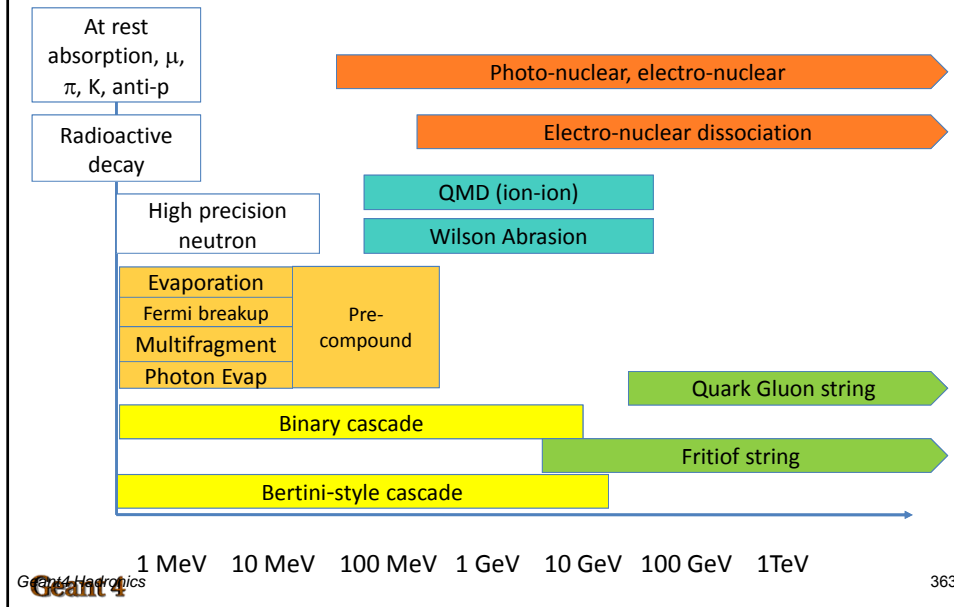
Theory-driven Hadronic Models

- Dominated by theoretical arguments (QCD, Glauber theory, exciton theory...)
- Final states (number and type of particles and their energy and angular distributions) determined by sampling theoretically calculated distributions
- This type of model is preferred, as it is the most predictive
- Examples
 - quark-gluon string (projectiles with $E > 20$ GeV)
 - intra-nuclear cascade (intermediate energies)
 - nuclear de-excitation and break-up

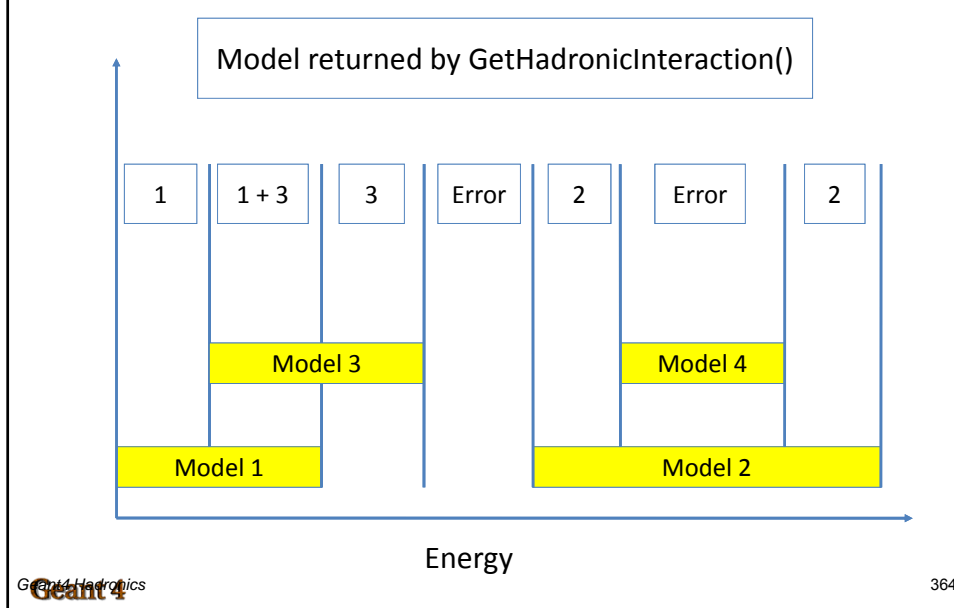
Parametrized Hadronic Models

- Depend mostly on fits to data with some theoretical guidance
- Two such models available:
 - Low Energy Parameterized (LEP) for $E < 20$ GeV
 - High Energy Parameterized (HEP) for $E > 20$ GeV
 - each type refers to a collection of models (one for each hadron type)
- Both LEP and HEP are re-engineered versions of the Fortran Gheisha code used in Geant3
- Code is fast and applies to all particle types, but is not particularly detailed
 - will be phased out by Geant4 10.0

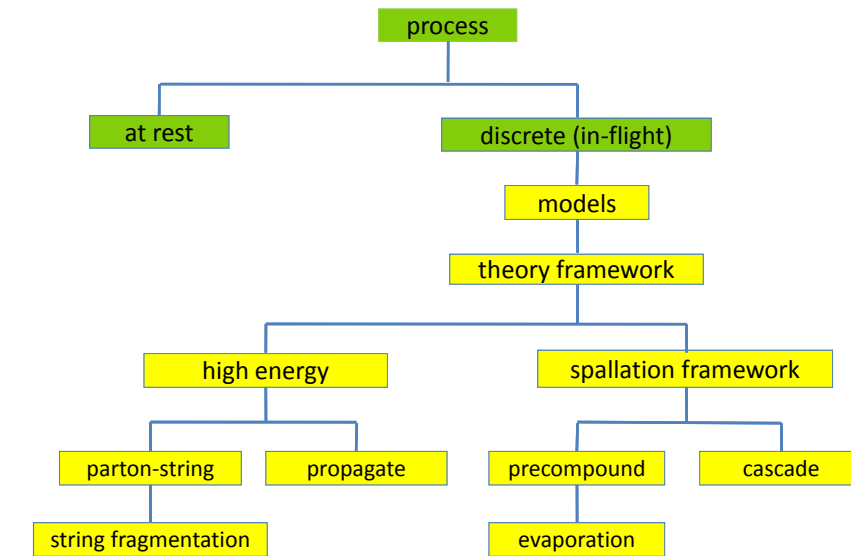
Partial Hadronic Model Inventory



Model Management



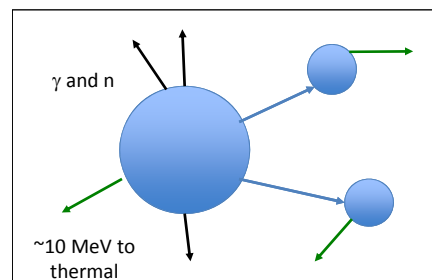
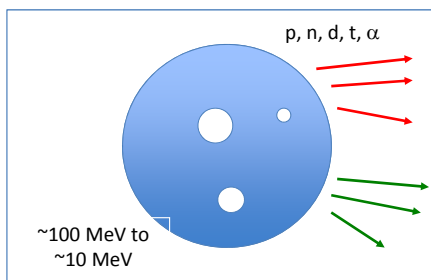
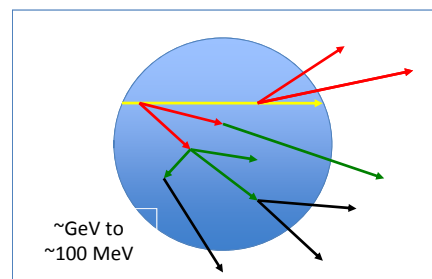
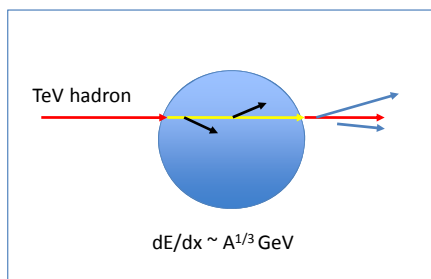
Hadronic Model Organization



Geant4

365

Hadronic Interactions from TeV to meV



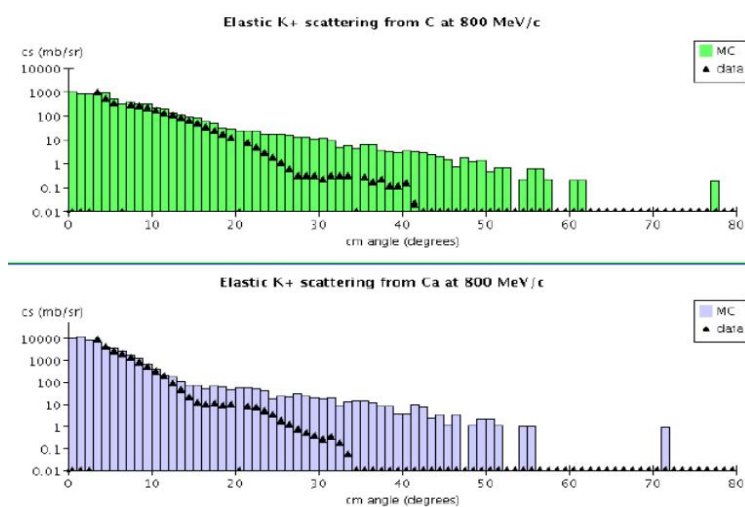
Geant4

366

Hadron Elastic Scattering

- ◆ G4WHadronElasticProcess: general elastic scattering
 - valid for all energies
 - valid for p, n, π , K, hyperons, anti-nucleons, anti-hyperons
 - based in part on old Gheisha code, but with relativistic corrections
- ◆ Coherent elastic
 - G4LEpp for (p,p), (n,n) : taken from SAID phase shift analysis, good up to 1.2 GeV
 - G4LEnp for (n,p) : same as above
 - G4HadronElastic for general hadron-nucleus scattering
- ◆ Neutron elastic
 - high precision (HP) model uses data from ENDF (E < 20 MeV)

Elastic Scattering Validation (G4HadronElastic)



Precompound Models

- G4PrecompoundModel is used for nucleon-nucleus interactions at low energy and as a nuclear de-excitation model within higher-energy codes
 - valid for incident p, n from 0 to 170 MeV
 - takes a nucleus from a highly excited set of particle-hole states down to equilibrium energy by emitting p, n, d, t, ^3He and α
 - once equilibrium is reached, four other models are called to take care of nuclear evaporation and break-up
 - these 4 models not currently callable by users
- The parameterized models and two cascade models have their own version of nuclear de-excitation models embedded in them

Precompound Models

- Invocation of Precompound model:


```
G4ExcitationHandler* handler = new G4ExcitationHandler;
G4PrecompoundModel* preco = new
  G4PrecompoundModel(handler);
// Create de-excitation models and assign them to precompound
  model

G4NeutronInelasticProcess* nproc = new
  G4NeutronInelasticProcess;
nproc->RegisterMe(preco);
neutronManager->AddDiscreteProcess(nproc);
// Register model to process, process to particle
```
- Here the model is invoked in isolation, but usually it is used in combination with high energy or cascade models
 - a standard interface exists for this

Bertini-style Cascade Model

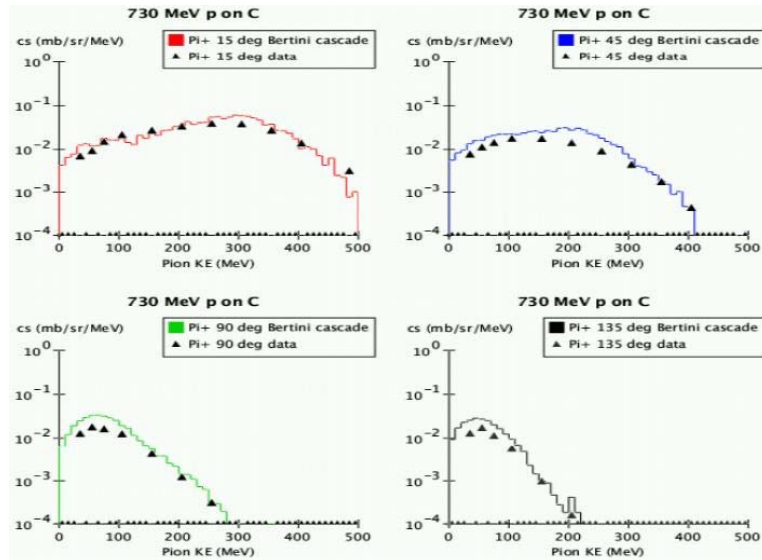
- A classical (non-quantum mechanical) cascade
 - average solution of a particle traveling through a medium (Boltzmann equation)
 - no scattering matrix calculated
 - can be traced back to some of the earliest codes (1960s)
- Core code:
 - elementary particle collisions with individual protons and neutrons: free space cross sections used to generate secondaries
 - cascade in nuclear medium
 - pre-equilibrium and equilibrium decay of residual nucleus
 - target nucleus built of three concentric shells

Using the Bertini Cascade

- In Geant4 the Bertini cascade is used for $p, n, \pi^+, \pi^-, K^+, K^-, K_L^0, K_S^0, \Lambda, \Sigma^0, \Sigma^+, \Sigma^-, \Xi^0, \Xi^-, \Omega^-$
 - valid for incident energies of 0 – 10 GeV
 - can also be used for gammas
- Invocation sequence


```
G4CascadeInterface* bert = new G4CascadeInterface;
G4ProtonInelasticProcess* pproc = new G4ProtonInelasticProcess;
pproc->RegisterMe(bert);
protonManager->AddDiscreteProcess(pproc);
// same sequence for all other hadrons and gamma
```

Validation of Bertini Cascade



Geant4

373

Binary Cascade Model

- Modeling sequence similar to Bertini, except
 - it's a time-dependent model
 - hadron-nucleon collisions handled by forming resonances which then decay according to their quantum numbers
 - particles follow curved trajectories in smooth nuclear potential
- Binary cascade is currently used for incident p, n and π
 - valid for incident p, n from 0 to 10 GeV
 - valid for incident π^+ , π^- from 0 to 1.3 GeV
- A variant of the model, G4BinaryLightIonReaction, is valid for incident ions up to $A = 12$ (or higher if target has $A < 12$)

Geant4

374

Using the Binary Cascade

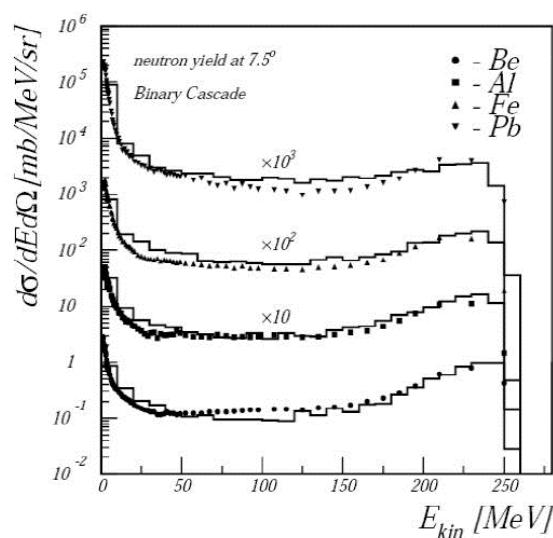
- Invocation sequence:

```
G4BinaryCascade* binary = new G4BinaryCascade();
G4PionPlusInelasticProcess* piproc =
    new G4PionPlusInelasticProcess();
piproc->RegisterMe(binary);
piplus_Manager->AddDiscreteProcess(piproc);
```

- Invoking BinaryLightIonReaction

```
G4BinaryLightIonReaction* ionBinary =
    new G4BinaryLightIonReaction();
G4IonInelasticProcess* ionProc = new G4IonInelasticProcess();
ionProc->RegisterMe(ionBinary);
genericIonManager->AddDiscreteProcess(ionProc);
```

Validation of Binary Cascade 256 MeV protons

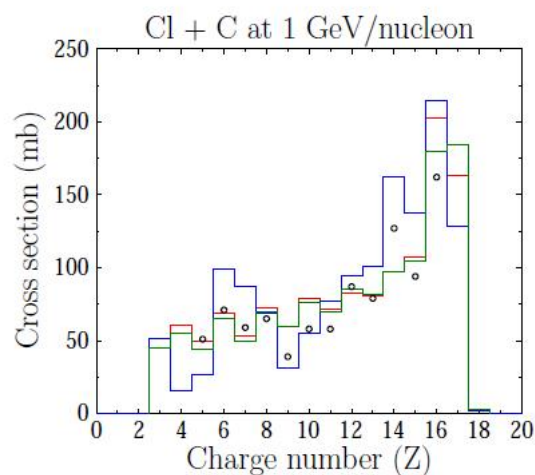


INCL++ Cascade Model

- ◆ Model elements:
 - time-dependent model
 - smooth Woods-Saxon or harmonic oscillator potential
 - particles travel in straight lines through potential
 - delta resonance formation and decay (like Binary cascade)
- ◆ Valid for incident p, n and π , d, t, ^3He , α from 150 MeV to 3 GeV
 - also works for projectiles up to $A = 12$
 - targets must be $11 < A < 239$
 - ablation model (ABLA) can be used to de-excite nucleus
- ◆ Used successfully in spallation studies
 - also expected to be good in medical applications

Validation of INCL Model

Green: INCL4.3 Red: INCL4.2 Blue: Binary cascade



Low Energy Neutron Physics

- Below 20 MeV incident energy, Geant4 provides several models for treating neutron interactions in detail
- The high precision models (NeutronHP) are data-driven and depend on a large database of cross sections, etc.
 - the G4NDL database is available for download from the Geant4 web site
 - elastic, inelastic, capture and fission models all use this isotope-dependent data
- There are also models to handle thermal scattering from chemically bound atoms

Geant4 Neutron Data Library (G4NDL)

- Contains the data files for the high precision neutron models
 - includes both cross sections and final states
- From Geant4 9.5 onward, G4NDL is based solely on the ENDF/B-VII database
 - G4NDL data is now taken only from ENDF/B-VII, but still has G4NDL format
 - use G4NDL 4.0 or later
- Prior to G4 9.5 G4NDL selected data from 9 different databases, each with its own format
 - Brond-2.1, CENDL2.2, EFF-3, ENDF/B-VI, FENDL/E2.0, JEF2.2, JENDL-FF, JENDL-3 and MENDL-2
 - G4NDL also had its own (undocumented) format

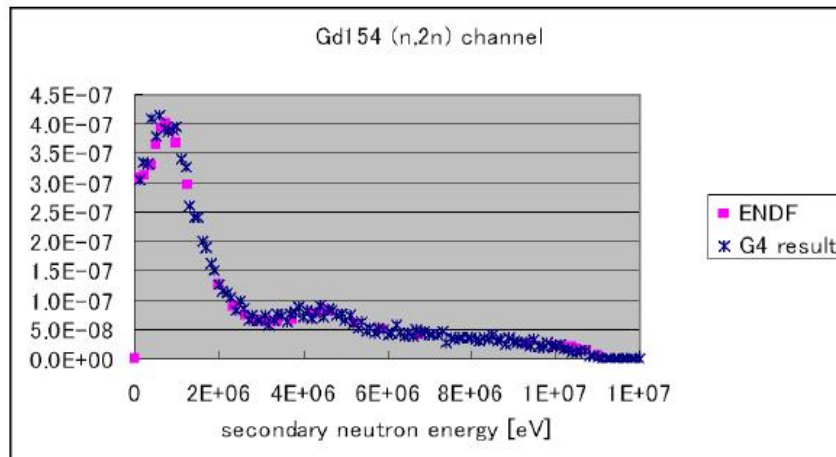
G4NeutronHPElastic

- Handles elastic scattering of neutrons by sampling differential cross section data
 - interpolates between points in the cross section tables as a function of energy
 - also interpolates between Legendre polynomial coefficients to get the angular distribution as a function of energy
 - scattered neutron and recoil nucleus generated as final state
- Note that because look-up tables are based on binned data, there will always be a small energy non-conservation
 - true for inelastic, capture and fission processes as well

G4NeutronHPInelastic

- Currently supports 34 inelastic final states + n gamma (discrete and continuum)
 - $n(A,Z) \rightarrow (A-1, Z-1) n p$
 - $n(A,Z) \rightarrow (A-3, Z) n n n n$
 - $n(A,Z) \rightarrow (A-4, Z-2) d t$
 -
- Secondary distribution probabilities
 - isotropic emission
 - discrete two-body kinematics
 - N-body phase space
 - continuum energy-angle distributions (in lab and CM)

Neutron Inelastic: ^{154}Gd (n,2n) Comparison to Data



Geant4 Hadronics

383

G4NeutronHPCapture

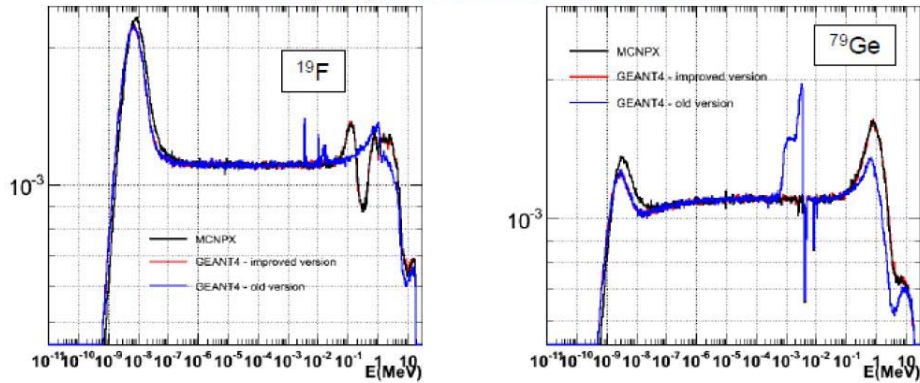
- Neutron capture on a nucleus produces photons described by either
 - the number of photons (multiplicity), or
 - photon production cross sections
- Photon spectra are either
 - discrete emission, using either cross sections or multiplicities from data libraries, or
 - continuous, with the spectrum calculated according to tabulated parameters
 - $f(E \rightarrow E_g) = \sum_i \pi_i(E) g_i(E \rightarrow E_g)$
where π_i and g_i come from the libraries

Geant4 Hadronics

384

High Precision Neutrons Comparing Geant4 and MCNPX

Comparing G4 HP old & new with MCNPX



Geant4-Hadronics

385

G4NeutronHPFission

- Currently only uranium fission data are available in Geant4
- First chance, second chance, third chance and fourth chance fission taken into account
- Resulting neutron energy distributions are implemented in different ways
 - as a function of incoming and outgoing neutron energy
 - as a Maxwell spectrum
 - evaporation spectrum
 - energy-dependent Watt spectrum
 - Madland-Nix spectrum

Geant4-Hadronics

386

Including HP Neutrons in the Physics List

- Elastic scattering

```
G4HadronElasticProcess* theNEP = new G4HadronElasticProcess;
```

```
// the cross sections
```

```
G4NeutronHPElasticData* theNEDData = new
```

```
G4NeutronHPElasticData;
```

```
theNEP->AddDataSet(theNEDData);
```

```
// the model
```

```
G4NeutronHPElastic* theNEM = new G4NeutronHPElastic;
```

```
theNEP->RegisterMe(theNEM);
```

```
neutManager->AddDiscreteProcess(theNEP);
```

Geant4 Hadronics

387

G4NeutronHPorLE Models

- The high precision neutron models do not cover all elements/isotopes
 - often no data: latest G4NDL has 395 isotopes, but there are thousands
 - data may exist but not yet be evaluated
- A Geant4 application must have a model under all circumstances, otherwise a fatal error occurs
- G4NeutronHPorLE models were developed to solve this problem
 - HPorLE models call the HP models if data exists
 - if no data, the Low Energy Parameterized (GHEISHA-style) model is called
 - elastic, inelastic, capture and fission provided

Geant4 Hadronics

388

Thermal Neutron Scattering from Chemically Bound Atoms

- At thermal energies, atomic motion, vibration, rotation of bound atoms affect neutron scattering cross sections and the angular distribution of secondary neutrons
- The energy loss (or gain) of such scattered neutrons may be different from those from interactions with unbound atoms
- Original HP models included only individual Maxwellian motion of target nucleus (free gas model)
- New behavior handled by model and cross section classes
 - G4HPThermalScatteringData, and
 - G4HPThermalScattering

Geant4

389

LEND – the new Livermore Neutron Models

- An alternative to the HP models
 - better code design
 - faster
 - Livermore database not yet as extensive G4NDL
- Corresponding model for each model in HP
 - elastic, inelastic, capture, fission
- Invocation in physics list:
 - use model names G4LENDElastic, G4LENDInelastic, G4LENDCapture, G4LENDFission, and cross sections G4LENDElasticCrossSection, G4LENDInelasticCrossSection, G4LENDCaptureCrossSection, G4LENDFissionCrossSection
- Database to use: go to <ftp://gdo-nuclear.ucllnl.org/pub/> and select G4LEND, then ENDF-B-VII.0.targ.gz

Geant4

390

Ion-Ion Inelastic Scattering

- Up to now we've considered only hadron-nucleus interactions, but Geant4 has five different nucleus-nucleus collision models
 - [G4BinaryLightIon](#)
 - [G4WilsonAbrasion/G4WilsonAblation](#)
 - [G4EMDissociationModel](#)
 - [G4QMD](#)
 - [G4Incl](#)
- Also provided are several ion-ion cross section data sets
- Currently no ion-ion elastic scattering models provided

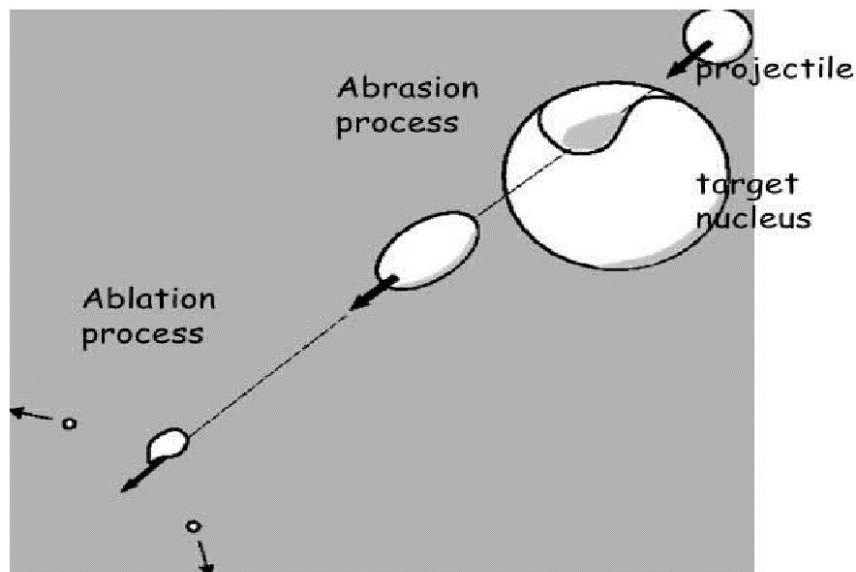
G4BinaryLightIonReaction

- This model is an extension of the G4BinaryCascade model (to be discussed later)
- The hadron-nuclear interaction part is identical, but the nucleus-nucleus part involves:
 - preparation of two 3D nuclei with Woods-Saxon or harmonic oscillator potentials
 - lighter nucleus is always assumed to be the projectile
 - nucleons in the projectile are entered with their positions and momenta into the initial collision state
 - nucleons are interacted one-by-one with the target nucleus, using the original Binary cascade model

G4WilsonAbrasion and G4WilsonAblation

- A simplified macroscopic model of nucleus-nucleus collisions
 - based largely on geometric arguments
 - faster than Binary cascade or QMD models, but less detailed
- The two models are used together
 - G4WilsonAbrasion handles the initial collision in which a chunk of the target nucleus is gouged out by the projectile nucleus
 - G4WilsonAblation handles the de-excitation of the resulting fragments
- Based on the NUCFRG2 model (NASA TP 3533)
- Can be used up to 10 GeV/n

Wilson Abrasion/Ablation

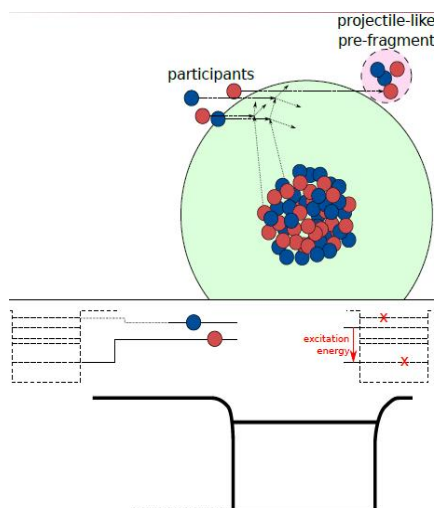


G4EMDissociation Model

- Electromagnetic dissociation is the liberation of nucleons or nuclear fragments as a result of strong EM fields
 - as when two high-Z nuclei approach
 - exchange of virtual photons instead of nuclear force
- Useful for relativistic nucleus-nucleus collisions where the Z of the nucleus is large
- Model and cross sections are an implementation of the NUCFRG2 model (NASA TP 3533)
- Can be used up to 100 TeV

INCL Nucleus-Nucleus

- INCL hadron-nucleus model used to interact projectile nucleons with target
- True potential is not used for projectile nucleus, but binding energy is taken into account
- True potential is used for target
- Projectile nucleons can pass through to form fragment or interact with nucleus

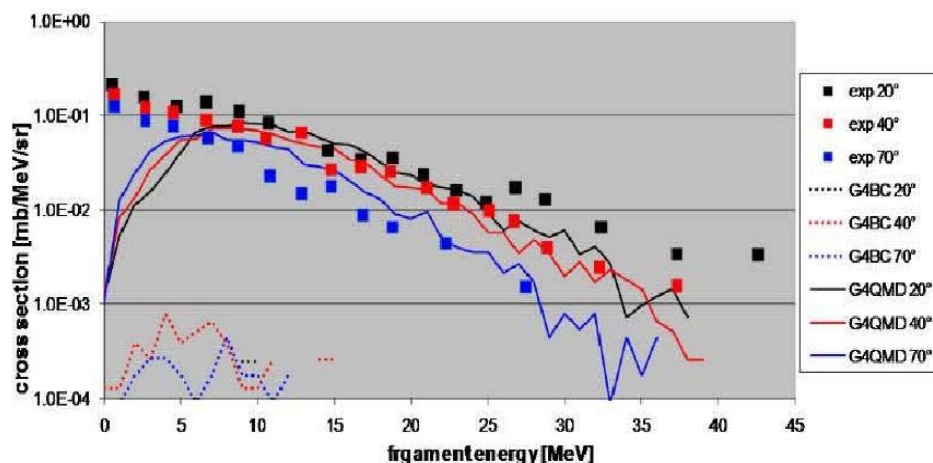


G4QMD Model

- BinaryLightIonReaction has some limitations
 - neglects participant-participant scattering
 - uses simple time-independent nuclear potential
 - imposes small A limitation for target or projectile
 - Binary cascade base model can only go to 5-10 GeV
- Solution is QMD (quantum molecular dynamics) model
 - an extension of the classical molecular dynamics model
 - treats each nucleon as a gaussian wave packet
 - propagation with scattering which takes Pauli principal into account
 - can be used for high energy, high Z collisions

QMD Validation

180MeV Proton on Al
Fragment A=7



Including QMD in the Physics List

```
• G4HadronInelasticProcess* ionInel =  
    new G4HadronInelasticProcess("ionInelastic",  
    G4GenericIon::G4GenericIon());  
    // the cross sections  
    G4TripathiCrossSection* tripCS = new G4TripathiCrossSection;  
    G4IonsShenCrossSection* shenCS = new  
    G4IonsShenCrossSection;  
    ionInel->AddDataSet(shenCS);  
    ionInel->AddDataSet(tripCS);  
    // assign model to process  
    G4QMDReaction* theQMD = new G4QMDReaction;  
    ionInel->RegisterMe(theQMD);  
    G4ProcessManager* pman = G4GenericIon::G4GenericIon()->  
        GetProcessManager();  
    pman->AddDiscreteProcess(ionInel);
```

Geant4 Hadronics

399

Radioactive Decay

- Process to simulate radioactive decay of nuclei
 - in flight
 - at rest
- α , β^+ , β^- decay, electron capture (EC) implemented
- Empirical and data-driven
 - data files taken from Evaluated Nuclear Structure Data Files (ENSDF)
 - half lives, nuclear level structure for parent and daughter nuclides, decay branching ratios, energy of decay process
- If daughter of nuclear decay is an isomer, prompt de-excitation is done by using G4PhotonEvaporation
- Analog (non-biased) sampling is the default
- Biased sampling also implemented

Geant4 Hadronics

400

Using Radioactive Decay

- Can be accessed with messengers (biasing options, etc.)
- To put in your physics list:

```
G4RadioactiveDecay* theDecay = new G4RadioactiveDecay;  
G4ProcessManager* pmanager = G4ParticleTable::GetParticleTable()->  
FindParticle("GenericIon")->GetProcessManager();
```

```
pmanager->AddProcess(theDecay);  
pmanager->SetProcessOrdering(theDecay, idxPostStep);  
pmanager->SetProcessOrdering(theDecay, idxAtRest);
```

Gamma- and Lepto-nuclear Processes

- These models are neither exclusively electromagnetic nor hadronic
 - gamma-nuclear
 - electro-nuclear
 - muon-nuclear
- Geant4 processes available:
 - G4PhotoNuclearProcess (implemented by two models)
 - G4ElectronNuclearProcess (implemented by one model)
 - G4PositronNuclearProcess (implemented by one model)
 - G4MuonNuclearProcess (implemented by two models)
- Gammas interact directly with the nucleus
 - at low energies they are absorbed and excite the nucleus as a whole
 - at high energies they act like hadrons (pion, rho, etc.) and form resonances with protons and neutrons
- Electrons and muons cannot interact hadronically, except through virtual photons
 - electron or muon passes by a nucleus and exchanges virtual photon
 - virtual photon then interacts directly with nucleus (or nucleons within nucleus)

Neutrino Scattering

- Not yet implemented in Geant4
 - but neutral current and charged current scattering could be added
- The nuclear part of the interaction is essentially done
 - can be handled by Bertini, FTF
 - see following slides
- Weak part of interaction will take some work
 - implementation of existing formulae for sampling ν , Q^2

QCD String Models

- Fritiof (FTF) valid for
 - p, n, π , K, Λ , Σ , Ω from 3 GeV to \sim TeV
 - anti-proton, anti-neutron, anti-hyperons at all energies
 - anti-d, anti-t, anti- ^3He , anti- α with momenta between 150 MeV/nucleon and 2 GeV/nucleon
- Quark-Gluon String (QGS) valid for
 - p, n, π , K from 15 GeV to \sim TeV
- Both models handle:
 - building 3-D model of nucleus from individual nucleons
 - splitting nucleons into quarks and di-quarks
 - formation and excitation of QCD strings
 - string fragmentation and hadronization

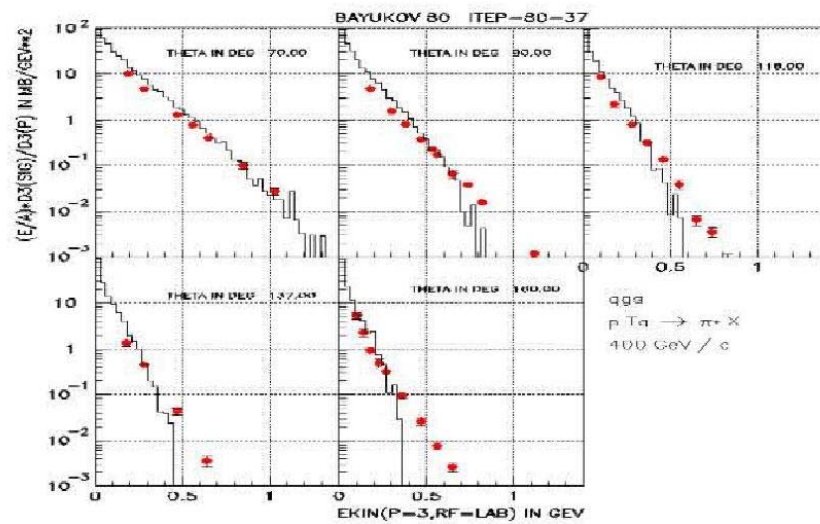
How the QCD String Model Works (FTF)

- Lorentz contraction turns nucleus into pancake
- All nucleons within 1 fm of path of incident hadron are possible targets
- Excited nucleons along path collide with neighbors
 - $n + n \rightarrow n\Delta, NN, \Delta\Delta, N\Delta, ..$
 - essentially a quark-level cascade in vicinity of path \rightarrow Reggeon cascade
- All hadrons treated as QCD strings
 - projectile is quark-antiquark pair or quark-diquark pair
 - target nucleons are quark-diquark pairs

How the QCD String Model Works (FTF)

- Hadron excitation is represented by stretched string
 - string is set of QCD color lines connecting the quarks
- When string is stretched beyond a certain point it breaks
 - replaced by two shorter strings with newly created quarks, anti-quarks on each side of the break
- High energy strings then decay into hadrons according to fragmentation functions
 - fragmentation functions are theoretical distributions fitted to experiment
- Resulting hadrons can then interact with nucleus in a traditional cascade

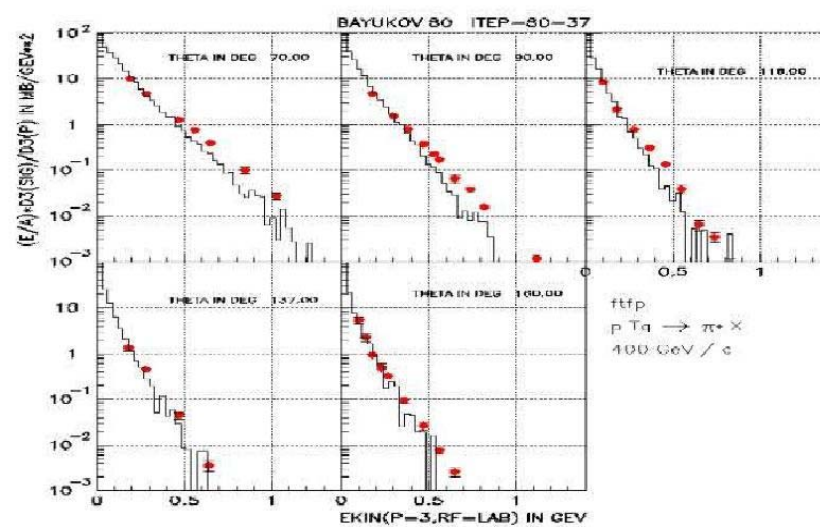
QGS Validation



Geant4

407

FTF Validation



Geant4

408

Adding FTF Model to the Physics List (1)

```
• G4TheoFSGenerator* heModel = new G4TheoFSGenerator("FTFP");  
  // model class that contains the sub-models  
  
  // Build the high energy string part of the interaction  
  G4FTFModel* ftf = new G4FTFModel;           // string interaction code  
  G4ExcitedStringDecay* esdk =                  // string decay code  
    new G4ExcitedStringDecay(new G4LundFragmentation);  
  
  ftf->SetFragmentationModel(esdk);  
  // assign decay code to model  
  
  heModel->SetHighEnergyGenerator(ftf);  
  // assign string sub-model to high energy model
```

Adding FTF Model to the Physics List (2)

```
  // Now set the de-excitation models to handle the  
  nuclear after the  
  // high energy interaction  
  G4GeneratorPrecompoundInterface* intfce =  
    new  
  G4GeneratorPrecompoundInterface;  
  G4PrecompoundModel* preco =  
    new G4PreCompoundModel(new G4ExcitationHandler);  
  // precompound model handles medium energy de-excitation  
  // G4ExcitationHandler does low energy de-excitation  
  
  intfce->SetDeExcitation(preco); // assign de-excitation models  
  
  heModel->SetTransport(intfce); // assign to high energy model
```

Capture Processes and Models

- G4PionMinusAbsorptionAtRest
 - process with direct implementation (no model class)
- G4PionMinusAbsorptionBertini **
 - at rest process implemented with Bertini cascade model
- G4KaonMinusAbsorption
 - at rest process with direct implementation
- G4AntiProtonAnnihilationAtRest
 - process with direct implementation
- G4FTFCaptureAtRest **
 - process implemented for anti-protons by FTF model

** recommended

Capture Processes and Models

- G4MuonMinusCaptureAtRest
 - process with direct implementation (no model class)
- G4AntiNeutronAnnihilationAtRest
 - process with direct implementation
- G4HadronCaptureProcess
 - in-flight capture for neutrons
 - model implementations:
 - G4NeutronHPCapture (below 20 MeV)

Fission Processes and Models

- G4HadronFissionProcess can use two models
 - G4NeutronHPFission (specifically for neutrons below 20 MeV)
- A third model handles spontaneous fission as an inelastic process (rather than fission)
 - G4FissLib: Livermore Spontaneous Fission

Summary (I)

- Geant4 hadronic physics allows user to choose how a physics process should be implemented
 - cross sections
 - models
- Many processes, models and cross sections to choose from
 - hadronic framework makes it easier for users to add more
- General hadron elastic scattering handled by G4WHadronElasticProcess
- Precompound models are available for low energy nucleon projectiles and nuclear de-excitation
- Three intra-nuclear cascade models available to cover medium energies (up to 10 GeV)
 - Bertini-style, Binary cascade, INCL++
- There are specialized high precision neutron models
 - HP models which use G4NDL, now based entirely on ENDF/B-VII
 - alternative LEND (Livermore) models are faster but currently less extensive – use the ENDF.B-VII library

Summary (II)

- Several models for ion-ion collisions
 - Wilson models fast, but not so detailed
 - Binary light ion cascade more detailed but slower
 - INCL++ ion cascade
 - QMD model very detailed but not so fast
- Radioactive decay
 - handles decay of isotopes at rest and in flight
 - ENSDF database files required
- Gamma-nuclear and lepto-nuclear processes are available for nuclear reactions initiated by non-hadrons
- Two QCD string models are available for implementing high energy interactions
 - Fritiof (FTF) : the more versatile, covers many particle types, larger energy range
 - Quark-Gluon String (QGS)
- Several stopping processes and models available for μ , π , K, anti-p, anti-n
- Capture and fission (mostly for neutrons)

Geant4

415

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

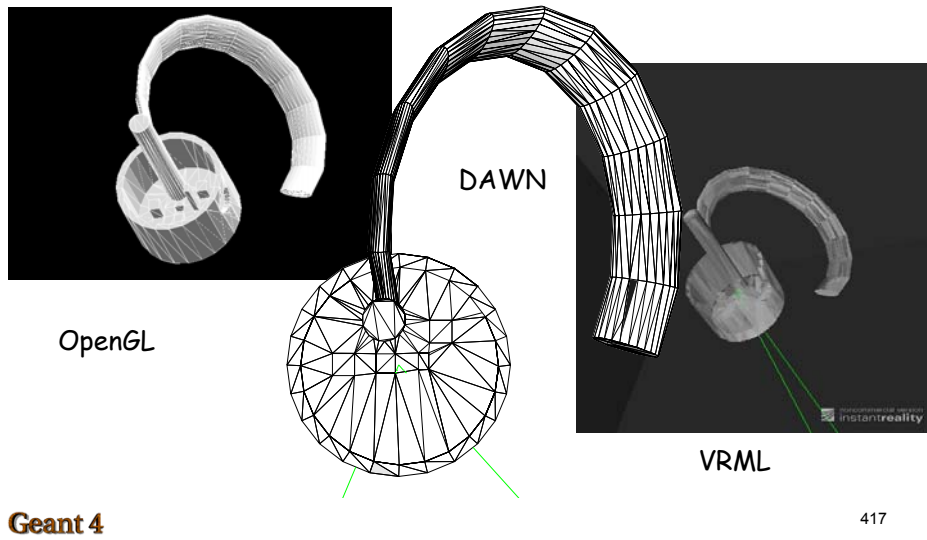
<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

This course encompasses training material developed by several Geant4 members:

thanks to all of them!

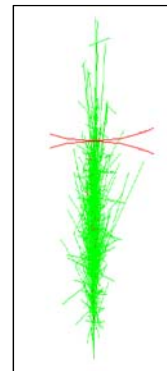
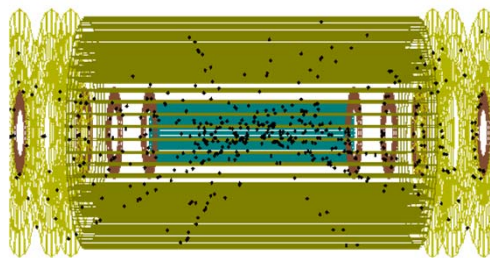
Geant 4

Geant4 Visualization



Visualization

- Seeing what one does usually helps
- One can visualize geometry, hits, trajectories
- One can assign color, transparency to volumes, tracks, hits displays
- Important for analysis and debugging



Visualization

- Additionally, users may wish to add markers, text or lines to highlight/mark specific detector components or tracks
- To visually distinguish detector components and tracks it is helpful to give them
 - Different colors
 - Transparency
 - Force wire-frame style display
- This is done by setting visualization attributes using **G4VisAttributes**

Geant 4

419

Controlling Visualization from Code

- Setting the visualization attributes for a given logical volume

```
G4VisAttributes* visAtt = new G4VisAttributes();

//set visibility (display or not)
visAtt->SetVisibility(true) //now you see me
visAtt->SetVisibility(false) //now you don't

//set color
G4Color red(r = 1.0, g = 0., b = 0.); //in RGB-color space;
visAtt->SetColor(red);

//set attributes for logical volume
redCubeLog->SetVisAttributes(visAtt);

//vis attributes can be reused
redCubeLog2->SetVisAttributes(visAtt);
```

Consider if every component really needs a different color – usually the shape and position is already quite helpful

Geant 4

420

Controlling Visualization from Code

- Adding a polyline (e.g. for axis, tracks) , marker (e.g. for hits) or text to a 3D scene

```
G4PolyLine xAxis;  
xAxis.append(G4Point3d(0., 0., 0.)); //polylines are defined by  
segments between points  
xAxis.append(G4Point3d(5.0*m, 0., 0.));  
G4Color blue(0., 0., 1.);  
G4VisAttributes visAtt(blue);  
xAxis.SetVisAttributes(visAtt);
```

```
G4Point3d position(5.0*m, 0., 0.);  
G4Circle mark(position); //G4Circle, G4Square, G4Text  
  
Mark.SetScreenDiameter(1.0);  
Mark.SetVisAttributes(visAtt);
```

/vis/scene/add/axis
will draw axis at a given
position for you!
This is just as an
example

Geant 4

421

Interactive Visualization using OpenGL

- Useful for debugging, quick checks but also analysis because the detector can easily be viewed “from all sides”
- Directly interact with the simulation
- Easiest to use with Qt-based UI interface

```
//in application main  
#ifdef G4VIS_USE  
G4VisManager* visMan = new G4VisExecutive();  
visMan->initialize();  
#endif  
...  
#ifdef G4VIS_USE  
delete visMan; // don't forget to delete the vis manager  
#endif
```

Geant 4

422

Interactive Visualization using OpenGL

- In UI:

```
Idle> /vis/open OGLIQt #open Qt OGL viewer
```

```
Idle> /vis/drawVolume #draw the detector volume
```

```
Idle> /vis/viewer/set/style wireframe
```

```
Idle> /vis/viewer/viewpointThetaPhi 20 70
```

Using the mouse is usually more intuitive for selecting viewpoint. Also try right-clicking into the viewer to get menu access to quite a few visualization options

Geant 4

423

Publication-quality Visualization using DAWN

- First install the DAWN viewer from http://geant4.kek.jp/~tanaka/DAWN/About_DAWN.html

```
Idle> /vis/open DAWNFILE #open dawn file
```

```
Idle> /vis/drawVolume #draw the detector volume
```

```
Idle> /vis/viewer/flush #create a .prim file for DAWN to read
```

You can then open the .prim file in the DAWN viewer, set visualization style, camera positions etc. The output will be a high quality .eps file

Geant 4

424

Adding trajectories and hits

- Up to now you have visualized your detector. Now its time to visualize some physics

```
Idle> /vis/scene/add/trajectories
```

```
Idle> /vis/scene/add/hits
```

```
Idle> /run/beamOn 1
```

Geant4 provides a rich set of commands to customize track visualization, e.g. drawing particle tracks in different colors depending on charge, id, type etc..

Geant 4

425

Overview of visualization drivers

- Multiple visualization drivers are available in Geant4 (some require external libraries)
 - OpenGL, as X11, Motif and Qt flavors (latter two support mouse interaction)
 - RayTracer
 - DAWN file, can be opened in DAWN viewer
 - VRML
 - OpenInventor
 - HepRep
 - PostScript

Geant 4

426

Geant4 Visualization

- For interactive visualization OpenGL is usually best-suited, also allows for display tracks, hits etc.
- DAWN and HepRep are well suited for event display and high quality plotting
- Raytracer can produce high quality renderings with transparency, but does not support the display of particle tracks
- Interactive OpenGL visualization is a helpful tool for inspecting geometries: are volumes misplaced, were transformation chosen correctly
- Visualization can be used to detect overlaps

Geant 4

427

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

This course encompasses training material developed by several Geant4 members:

thanks to all of them!

Geant 4

Geant4 User Interface (UI)

• Geant4 UI

- Geant4 supports various built-in UI commands.
- UI command consists of
command directory, command, and parameter.
Ex) /run/beamOn 10
- Geant4 has a two types of UI processing (batch mode, interactive mode)
- We can check the list of built-in commands in this web site.
http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/AllResources/Control/UIcommands/_html

Geant 4

429

Geant4 User Interface (UI)

• ‘Hard-coded’ batch mode

- Geant4 UI Command can be run in Geant4 source code.
(called ‘Hard-coded’ batch mode)
- Example:

```
int main(int argc, char** argv)
{ ...
    G4UImanager* UImanager =
    G4UImanager::GetUIpointer();
    UImanager->ApplyCommand("/run/verbose 1");
    UImanager->ApplyCommand("/event/verbose 1");
    UImanager->ApplyCommand("/tracking/verbose 1");
    ...
}
```

Geant 4

430

Geant4 User Interface (UI)

◆ Batch mode with macro file

- Geant4 simulation will run in batch mode with macro file.
- Example:

```
int main(int argc, char** argv)
{ ...
  G4UIManager* UImanager =
  G4UIManager::GetUIpointer();
  if(argc==1) {
    // Define UI session for interactive mode
  } else {
    // Define batch mode using UImanager class
    G4String command = "/control/execute " + argv[1];
    UImanager->ApplyCommand(command);
  }
  ...
}
```

Geant

431

Geant4 User Interface (UI)

◆ Batch mode with macro file

- The example will be executed with below command:
 > example run.mac
- Example - run.mac file:

```
# Macro file: run.mac
# set verbose level for this run
/run/verbose 2
/event/verbose 0
/tracking/verbose 1

/gun/particle e-
/gun/energy 1 GeV
/run/beamOn 100
```

※ First '#' is used for comment line in macro file.

Geant 4

432

Geant4 User Interface (UI)

• Interactive mode - Character UI (CUI)

- Geant4 provide character user interface (CUI) by using G4UIterminal class (or G4UIWin32 in windows OS) as interactive mode.
- Example:

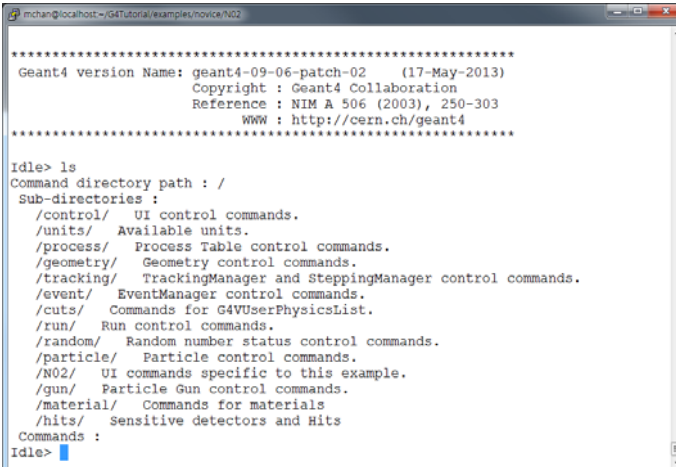
```
int main(int argc, char** argv)
{ ...
  if(argc==1) {
    // Define UI session for interactive mode
    G4UISession* session = new G4UIterminal(new G4UItcsh);
    session->SessionStart();
    delete session;
  }
  ...
}
```

Geant4

433

Geant4 User Interface (UI)

• Interactive mode - Character UI (CUI)



```
mchan@localhost:~/G4Tutorial/examples/novice/N02
*****
Geant4 Version Name: geant4-09-06-patch-02   (17-May-2013)
Copyright : Geant4 Collaboration
Reference : NIM A 506 (2003), 250-303
WWW : http://cern.ch/geant4
*****

Idle> ls
Command directory path : /
Sub-directories :
/control/   UI control commands.
/units/    Available units.
/process/   Process Table control commands.
/geometry/  Geometry control commands.
/tracking/  TrackingManager and SteppingManager control commands.
/event/     EventManager control commands.
/cuts/      Commands for G4VUserPhysicsList.
/run/       Run control commands.
/random/    Random number status control commands.
/particle/  Particle control commands.
/N02/       UI commands specific to this example.
/gun/       Particle Gun control commands.
/material/  Commands for materials
/hits/      Sensitive detectors and Hits
Commands :
Idle>
```

Geant4

434

Geant4 User Interface (UI)

- **Interactive mode - Graphical UI (GUI)**

- Geant4 supports various graphical user interface (GUI) by using G4UIExecutive class.
- Example:

```
int main(int argc, char** argv)
{ ...
    if(argc==1) {
        // Define UI session for interactive mode
        G4UIExecutive* ui = new G4UIExecutive(argc, argv, "XX");
        session->SessionStart();
        delete session;
    }
    ...
}
```

Geant 4

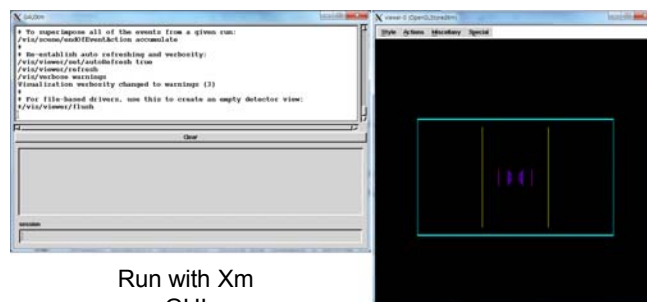
435

Geant4 User Interface (UI)

- **Interactive mode - Graphical UI (GUI)**

- User can choose a specific interface initial parameter of G4UIExecutive class.

```
ex) G4UIExecutive* ui = new G4UIExecutive(argc, argv, "xx")
    (where xx = qt, xm, win32, gag, tcsh, ...)
```



Run with Xm
GUI

Geant 4

436

Geant4 User Interface (UI)

♦ Interactive mode - Qt UI

- The one of Geant4 GUI Application
- Qt (UI) with OpenGL (Graphic viewer)
- If GEANT4_USE_QT=1 when geant4 was installed, Qt is default UI in Geant4.
- Homepage - <http://qt.digia.com/>

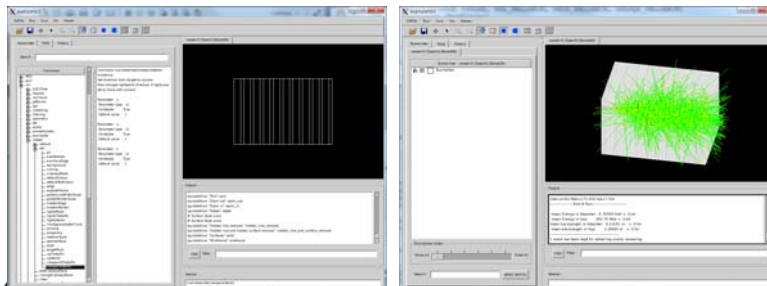
Geant 4

437

Geant4 User Interface (UI)

♦ Interactive mode - Qt UI

- Enable/Disable specific geometry using “Scene tree”
- Command (or Command parameter) check
- Geometry viewer
- Run specific macro, command alias, etc...

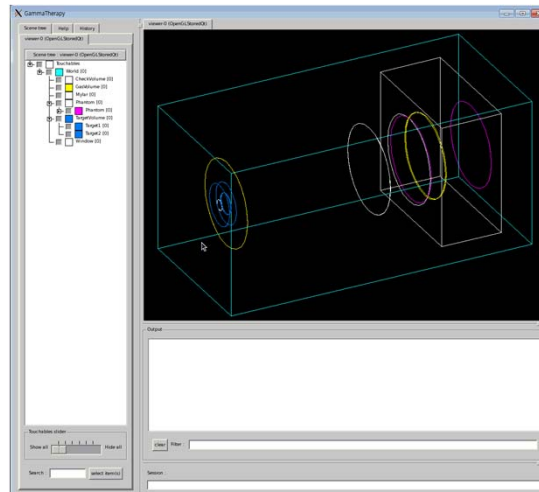


Geant 4

38

Geant4 User Interface (UI)

• Interactive mode - Qt Demo



Geant 4

439

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

This course encompasses training material developed by several Geant4 members:

thanks to all of them!

Geant 4

Preliminary slides

- The printed handouts are a preliminary version of the course slides in response to the conference organizers' request to deliver slides for printing more than one month before the course
- The actual slides of the course will be available at the course web site
<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html> one week before the course
- Colour-coded information contained in UML diagrams is available online only in PDF files
- We support environmentally-conscious attitudes

Event biasing

Basic concepts of simulation with Geant4 as
a general purpose Monte Carlo toolkit

Event biasing

- What is analogue simulation ?
 - Sample using natural probability distribution, $N(x)$
 - Predicts mean with correct fluctuations
 - Can be inefficient for certain applications
- What is non-analogue/event biased simulation ?
 - Cheat - apply artificial biasing probability distribution, $B(x)$ in place of natural one, $N(x)$
 - $B(x)$ enhances production of whatever it is that is interesting
 - To get meaningful results, must apply a weight correction
 - Predicts same analogue mean with smaller variance
 - Increases efficiency of the Monte Carlo
 - Does not predict correct fluctuations
 - Should be used with care

Geant 4

443

Event biasing

- Geant4 provides built-in general use biasing techniques
- The effect consists in producing a small number of secondaries, which are artificially recognized as a huge number of particles by their statistical weights → reduce CPU time
- Event biasing can be used, for instance, for the transportation of particles through a thick shielding
- An utility class **G4WrapperProcess** support user-defined biasing

Geant 4

444

Event biasing techniques

- **Production cuts / threshold**

- This is a biasing technique – most popular for many applications: set **high cuts** to reduce secondary production

- **Geometry based biasing**

- Importance **weighting** for volume/region
- Duplication or sudden death of tracks

- **Primary event biasing**

- Biasing **primary events** and/or primary particles in terms of type of event, momentum distribution → generate *only primaries* that can produce *events that are interesting for you*

Geant 4

445

Event biasing techniques

- **Forced interaction**

- **Force** a particular interaction, e.g. within a volume

- **Enhanced process or channel and physics-based biasing**

- Increasing **cross section** for a given process (e.g. Bremsstrahlung)
- Biasing **secondary production** in terms of particle type, momentum distribution, cross-section, etc.

- **Leading particle biasing**

- Take into account only the **most energetic** (or most important) **secondary**
- Currently **NOT supported** in Geant4

Geant 4

446

Variance Reduction

- Use variance reduction techniques to **reduce computing time** taken to calculate a result with a **given variance** (= statistic error)
- Want to **increase efficiency** of the Monte Carlo

Geant 4

447

Geometric Biasing

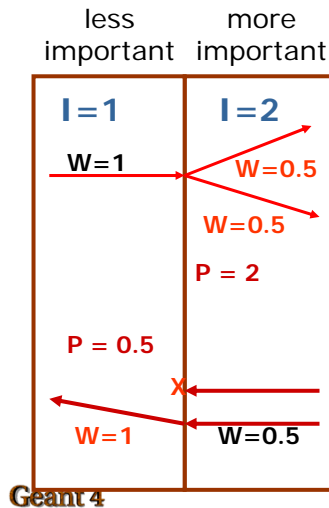
The purpose of **geometry-based event biasing** is to save computing time by sampling less often the particle histories entering “less important” geometry regions, and **more often in more “important” regions**

- * Importance sampling technique
- * Weight window technique

Geant 4

448

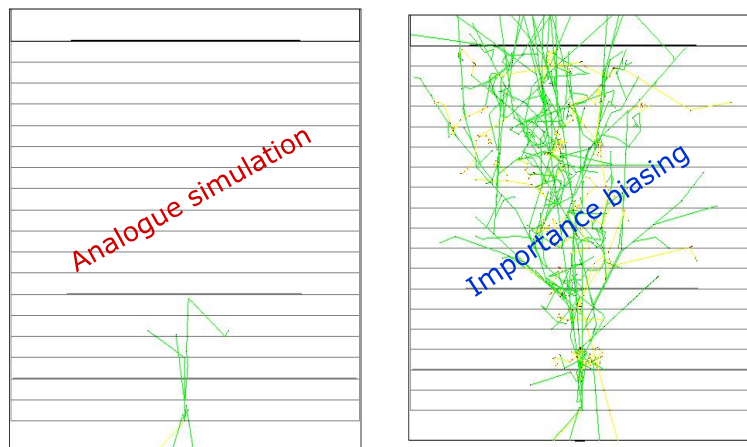
Importance sampling technique



- Importance sampling acts on particles **crossing boundaries** between “importance cells”
- The action taken depends on the **importance value (I)** assigned to the cell
- In general, a track is played either **split** or **Russian roulette** at the geometrical boundary depending on the importance value assigned to the cell

449

Importance biasing



10 MeV neutron in thick concrete cylinder

Geant 4

450

Physics biasing

- Built-in **cross section biasing** for PhotoNuclear, ElectronNuclear and PositronNuclear processes

```
G4ElectroNuclearReaction * theReaction = new G4ElectroNuclearReaction;  
G4ElectronNuclearProcess theElectronNuclearProcess;  
theElectronNuclearProcess.RegisterMe(theReaction);  
theElectronNuclearProcess.BiasCrossSectionByFactor(100);
```

- Similar tool for **rare EM processes** (e^+e^- annihilation to $\mu^+\mu^-$ or hadrons, γ conversion to $\mu^+\mu^-$)

```
G4AnnihiToMuPair* theProcess = new G4AnnihiToMuPair();  
theProcess->SetCrossSecFactor(100);
```

- It is possible to introduce these factors for **all EM processes**, with a definition of customized processes that inherit from the “normal” ones (\rightarrow extended example)

- **Artificially enhance/reduce cross section** of a process (useful for thin layer interactions or thick layer shielding)

General implementation under development

Geant 4

Geant 4

*IEEE Nuclear Science Symposium and Medical Imaging
Conference
Short Course*

Simulation Techniques Using Geant4

Maria Grazia Pia (*INFN Genova, Italy*)
MariaGrazia.Pia@ge.infn.it

Dresden, 18 October 2008

<http://www.ge.infn.it/geant4/events/nss2008/geant4course.html>

This course exploits training material developed by several Geant4
Collaboration members: thanks to all of them!

Geant 4

Fast Simulation

Basic concepts of simulation with Geant4 as
a general purpose Monte Carlo toolkit

Geant 4

Generalities

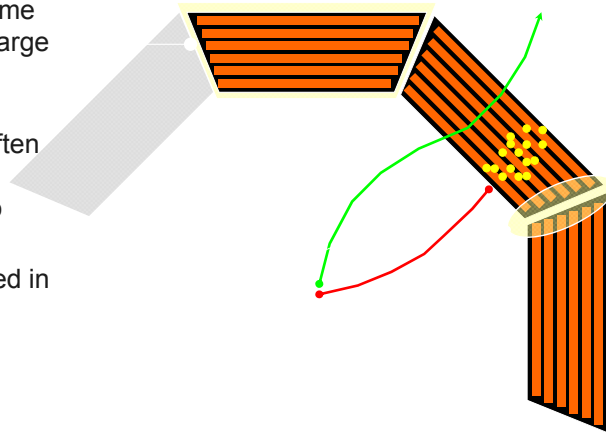
- Fast Simulation, also called parameterisation, is a shortcut to the tracking
- Fast Simulation allows you to take over the tracking to implement your own fast physics and detector response
- The classical use case of fast simulation is the shower parameterisation
 - the typical several thousand steps per GeV computed by the tracking are replaced by a few ten of deposits per GeV
- Parameterisations are generally experiment dependent

Geant 4

454

Parameterisation features

- Parameterisations take place in an envelope. This is typically the mother volume of a sub-system or of a large module of such a sub-system
- Parameterisations are often particle type dependent and/or may apply only to some.
- They are often not applied in complicated regions.



Geant 4

455

G4VFastSimulationModel

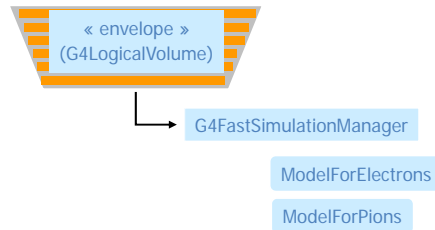
- This is the base class allowing to implement concrete parameterisation models
- It has three pure virtual methods to be overridden :
 - G4bool IsApplicable(const G4ParticleDefinition *)
 - Which specify for which particles the model is valid
 - G4bool ModelTrigger(const G4FastTrack &)
 - Which allow to decide or not to trigger the model at the current point, in order to avoid to trigger in a « complicated region ».
 - void DoIt(const G4FastTrack &, G4FastStep &)
 - Which is the parameterisation properly said, invoked when the model has triggered.
- G4FastTrack provides input informations to the model (G4Track, envelope informations, ...)

Geant 4

456

Binding concrete models to an envelope

- Concrete models are bound to the envelope through a G4FastSimulationManager object
- This allows several models to be bound to a same envelope
- The « envelope » is simply a G4LogicalVolume which has received a G4FastSimulationManager
- All its [grand[...]]daughters will be sensitive to the parameterisations



Geant 4

457

G4FastSimulationManagerProcess

- The G4FastSimulationManagerProcess is a process providing the interface between the tracking and the fast simulation
- It has to be set to the particles to be parameterised:
 - The process ordering is the following:
 - [n-3] ...
 - [n-2] Multiple Scattering
 - [n-1] G4FastSimulationManagerProcess
 - [n] G4Transportation
- It can be set as a discrete process or it must be set as a continuous & discrete process if using ghost volumes

Geant 4

458

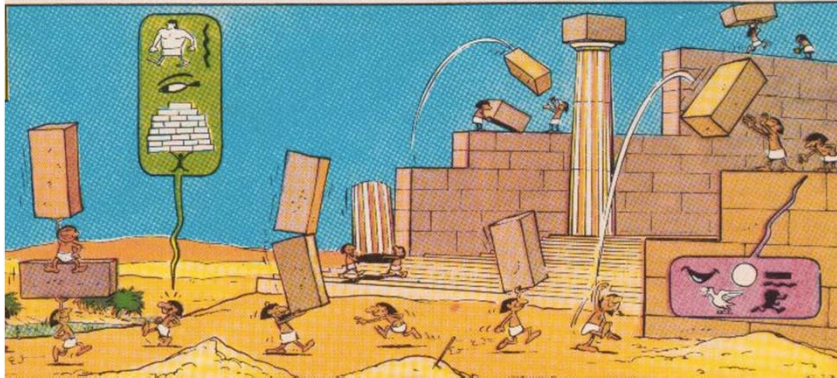
- The Fast Simulation components are indicated in blue.



- Ghost volumes allow to define envelopes independently of the volumes of the tracking geometry
- This allows to group together the electromagnetic and hadronic calorimeters for pion parameterisation for example or to define envelopes for geometries coming out of a CAD system which don't have a hierarchical structure
- In addition Ghost volumes are sensitive to the particle flavor, allowing to define in a completely independent way envelopes for electrons, envelopes for pion etc...

Parallelization

- Geant4 simulation is an embarrassingly parallel computational problem – each event can possibly be treated independently



Geant 4

461

Why parallelize

- **Monte-Carlo simulation is a computationally demanding problem.**
 - For good statistics need large number of primary events
 - Each primary may generate large numbers of secondaries
 - Complex and large detector setups – from the LHC to Earth or space
- Simply increasing the GHz count on CPUs has reached its limits, more practical (thermally and cost-wise) are multicore, multi-processor and cluster-architectures
- Just another form of **divide and conquer**: if the problem is too large to solve it as a whole, sub-divide it.

Geant 4

462

How to parallelize?

- Simplest solution: run the same application on multiple cores/computers with different initial parameters in parallel.
 - Few to no adaptations of application necessary
 - Overhead because resource sharing is not used
 - Need additional tools to collect results
- Intermediate solution: integrate task scheduling into application. It is still run as a complete process but is aware that it may share resource and need to collect results.
 - Modification of application necessary
 - Processes still generally run in their own memory space
 - Result collection implemented in application

Geant 4

463

How to parallelize?

- Multi-Threaded
 - Less overhead because resources are shared (only one process)
 - Need to take great care to not have race conditions (e.g. on thread writes into memory another thread reads)
 - Bound to local machine
- Multi-Threaded and Distributed
 - Requires extensive adaption of application
 - Message passing e.g. with ZeroMQ needed
 - Data is passed as message, which may not be replied to
 - Message are passed transparently over network or memory
 - Very powerful and scalable, nowadays often runs into I/O-bound limits.

Geant 4

464

Application-level parallelization

- Currently available concurrency solutions in Geant4:
- Parallelize on a run level, i.e. run multiple full simulations with different random seeds, then combine results afterwards
 - Not really concurrent Geant4, but concurrently run Geant4
 - User needs to take care of result composition afterwards
 - Overhead from full Geant4 application instantiated for each simulation (process vs. thread)
 - Nevertheless: straight-forward, minimal adaption of existing simulation, useful for great variety of parallel architectures

```
//in application main  
G4long seed = time(NULL); // get a "unique" seed  
CLHEP::HepRandom::setTheSeed(seed); //set this seed for the RNG
```

Geant 4

465

Process-level Parallelization

- Parallelize on an per-event level using ParGeant4
 - Replaces run manager with a parallel version which uses TOP-C based messaging to launch slave processes
 - Marshaling of hit data has to be done by user using specific comments, i.e. which data can be copied, how should it be copied.
 - ParGeant4 takes care of aggregating results of parallel processing into sequential containers
 - Can parallelize on multiple cores and multiple machines
 - Each slave is essentially a complete Geant4 application in itself
 - No thread level parallelism

Geant 4

466

Parallelization Hints

- If you do not want to have to think about sharing memory, marshaling data or where to modify your code: run parallel application.
 - A simple bash script can start up multiple simulations
 - Depending on how you persist your results they can often easily be combined by pure concatenation.
- Using ParGeant4 is more elaborate but also requires more code modification, i.e. there are more banana skins along the way.

Geant 4

467

Parallelization

- Thread-parallel Geant4 MT
 - Parallelizes Geant4 kernel by using multiple threads
 - Currently bound to Unix by using gcc low-level thread API
 - Developers are discussing to use `std::thread` of C++11, this would make it more portable
 - Read-only data such as geometry and element definitions are shared between threads
 - Read-Write data is either protected by mutexes or thread-private were applicable
 - User must check that event-level user actions are thread-safe
 - Only useful for multi-core machines, not clusters
 - Beta-code: use at own risk

Geant 4

468

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

*This course encompasses training material developed by several Geant4
members:*

thanks to all of them!

Geant 4

Geant4 MultiThreading

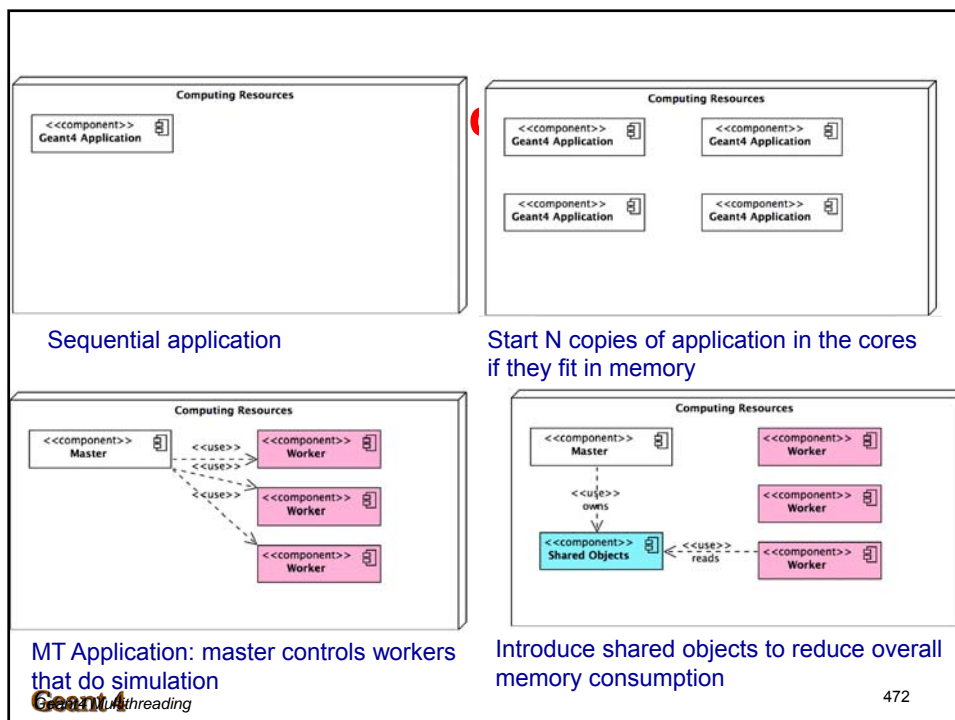
Geant 4

Evolution of Computer Technology

- Increase in CPU power used to be done by increasing frequency. This has reached a plateau because of power needs. So expectation in future trend in computer technology:
 - No major increase in single core performance
 - Many cores in a processor (double/2 year?)
 - Less memory per core
- Need new software models to take these into account → increase parallelism
- In Geant4 events are supposed to be independent.
 - Each event can be simulated separately
 - To keep backward compatibility with user code, multi-threading at event level is a natural choice for parallelism

Geant4 Multithreading

471

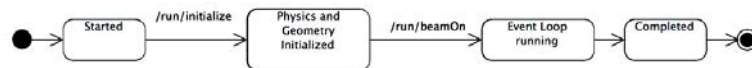


Geant4 Multithreading

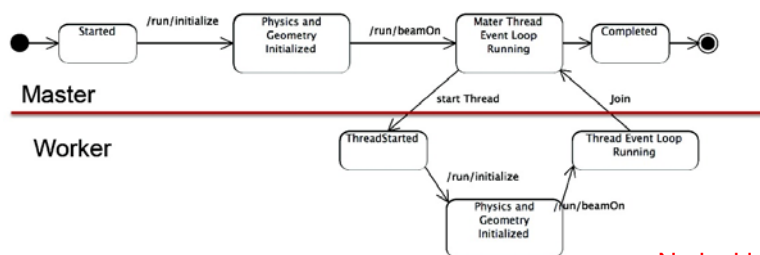
472

Geant4 MT Model

- Steps to be taken
 - Isolate static data structure from the dynamic one
 - Protect dynamic part of memory which can be shared among the working threads
- A Geant4 (with MT) application can be seen as simple finite state machine



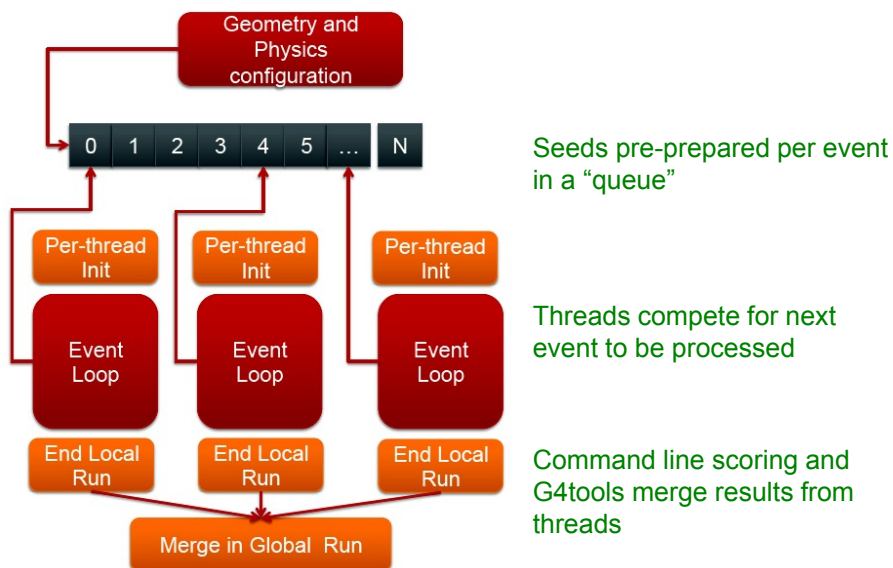
- Threads do not exist before /run/beamOn. Master writes shared memory. When threads start they cannot change the shared memory



Geant4 Multithreading

No locking⁴⁷³

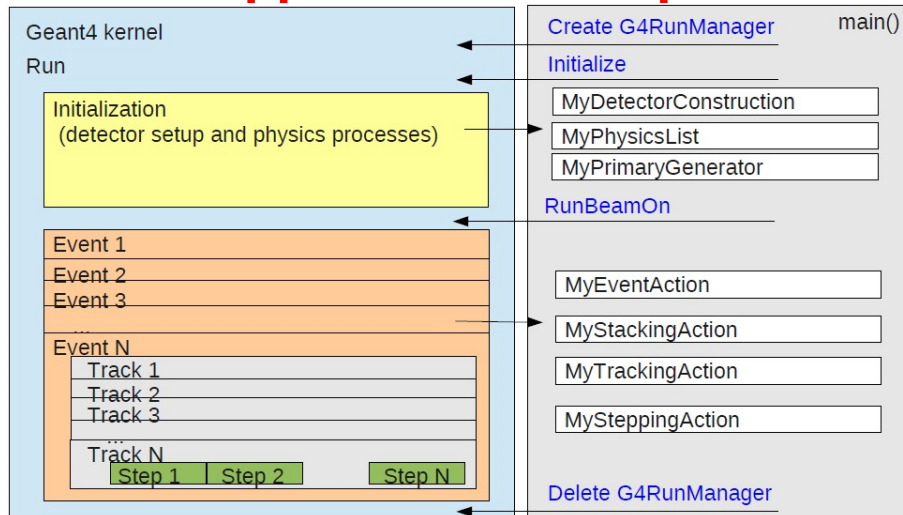
Geant4 MT Model



Geant4 Multithreading

474

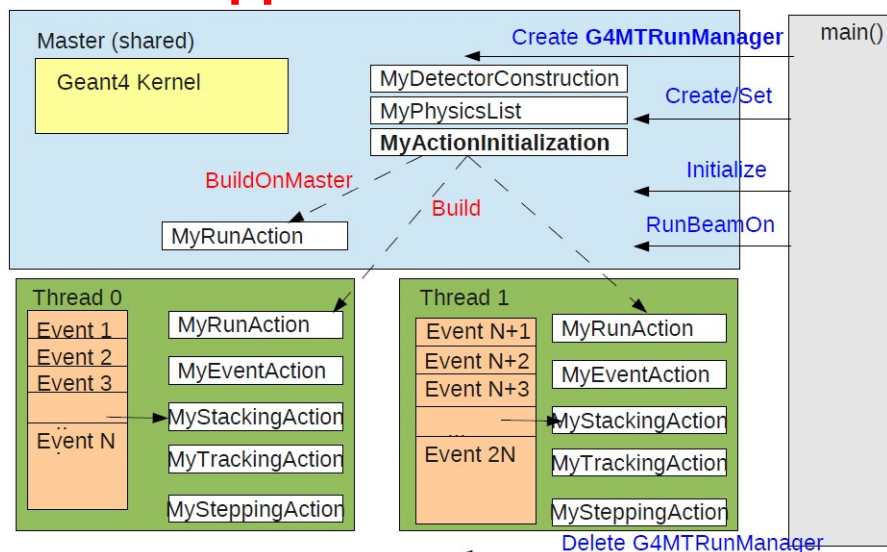
User Application in Sequential



Geant4 threading

475

User Application in MT Mode



Geant4 threading

476

Migration to MT (1)

- Move user action class from `main()` to a new `ActionInitialization` class

exampleB1.cc 9.6.p02

```
// Detector construction
runManager->SetUserInitialization(new B1DetectorConstruction());

// Physics list
G4VModularPhysicsList* physicsList = new QBBC;
runManager->SetUserInitialization(physicsList);
```

Keep in main

```
// Primary generator action
runManager->SetUserAction(new B1PrimaryGeneratorAction());

// Stepping action
runManager->SetUserAction(new B1SteppingAction());

// Event action
runManager->SetUserAction(new B1EventAction());

// Run action
runManager->SetUserAction(new B1RunAction());
```

Move to new class

Geant4 Multithreading

477

Migration to MT (2)

B1ActionInitialization.hh - 10.00

```
#include "G4VUserActionInitialization.hh"

class B1ActionInitialization :
public G4VUserActionInitialization
{
public:
    B1ActionInitialization();
    virtual ~B1ActionInitialization();

    virtual void BuildForMaster() const;
    virtual void Build() const;
};
```

In MT: both functions are called
In S: only Build() is called

B1ActionInitialization.cc - 10.00

```
void B1ActionInitialization::BuildForMaster() const
{
    SetUserAction(new B1RunAction);
}

void B1ActionInitialization::Build() const
{
    SetUserAction(new B1PrimaryGeneratorAction);
    SetUserAction(new B1RunAction);
    SetUserAction(new B1EventAction);
    SetUserAction(new B1SteppingAction);
}
```

Geant4 Multithreading

478

Migration to MT (3)

- Instantiate `G4MTRunManager` and replace the code moved in `ActionInitialization` with use of this class
- If sensitive detectors are present separate their creation in a new function `CreateSDandField()`
 - A new utility function can be used:
`G4VUserDetectorConstruction::SetSensitiveDetector`
- If magnetic field is present, move it to `CreateSDandField()`

`CreateSDandField()`

9.6.p02

```
B2bDetectorConstruction::B2bDetectorConstruction() {
// ...
fMessenger = new B2bDetectorMessenger(this);
fMagField = new B2MagneticField();
}
```

ref-07

```
G4VPhysicalVolume* B2bDetectorConstruction::ConstructSDandField() {
// ...
fMagField = new B2MagneticField();
// ...
}

B2MagneticField::B2MagneticField() {
// ...
fMagField = new B2FieldMessenger();
// ...
}
```

New magnetic field messenger
class separated from
B2bDetectorMessenger

Geant4

479

Migration to MT (4)

- Typically `G4Allocator` is used in `Hit`, `Trajectory` and `TrajectoryPoint` classes
 - Change `static` declarations to `static G4ThreadLocal`

B2TrackerHit.hh 9.6.p02

```
extern G4Allocator<B2TrackerHit>* B2TrackerHitAllocator;
```

B2TrackerHit.cc 9.6.p02

```
G4Allocator<B2TrackerHit>* B2TrackerHitAllocator=0;
```

B2TrackerHit.hh 10.00

```
extern G4ThreadLocal G4Allocator<B2TrackerHit> B2TrackerHitAllocator;
```

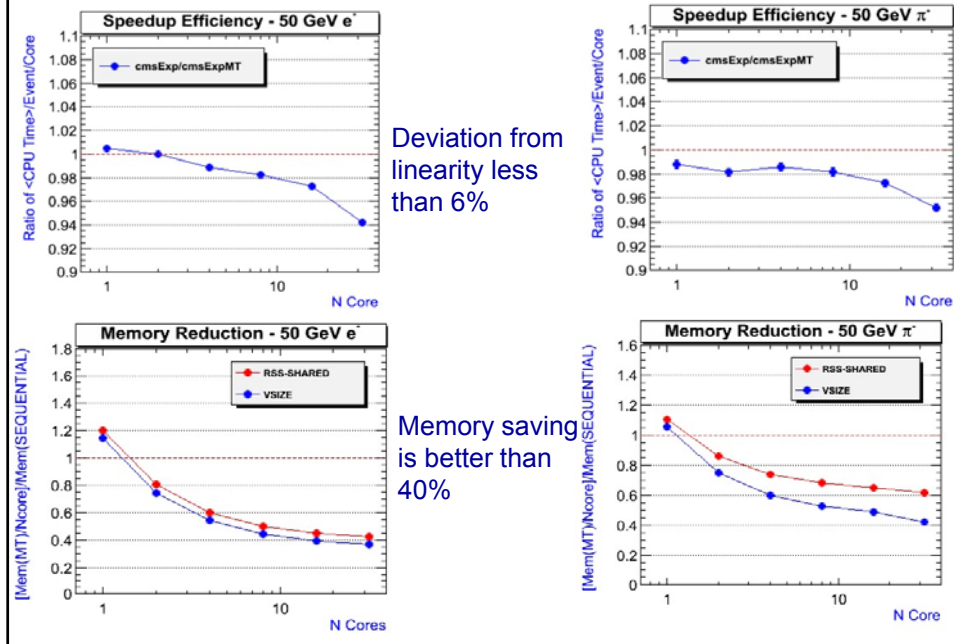
B2TrackerHit.cc 10.00

```
G4ThreadLocal G4Allocator<B2TrackerHit>* B2TrackerHitAllocator=0;
```

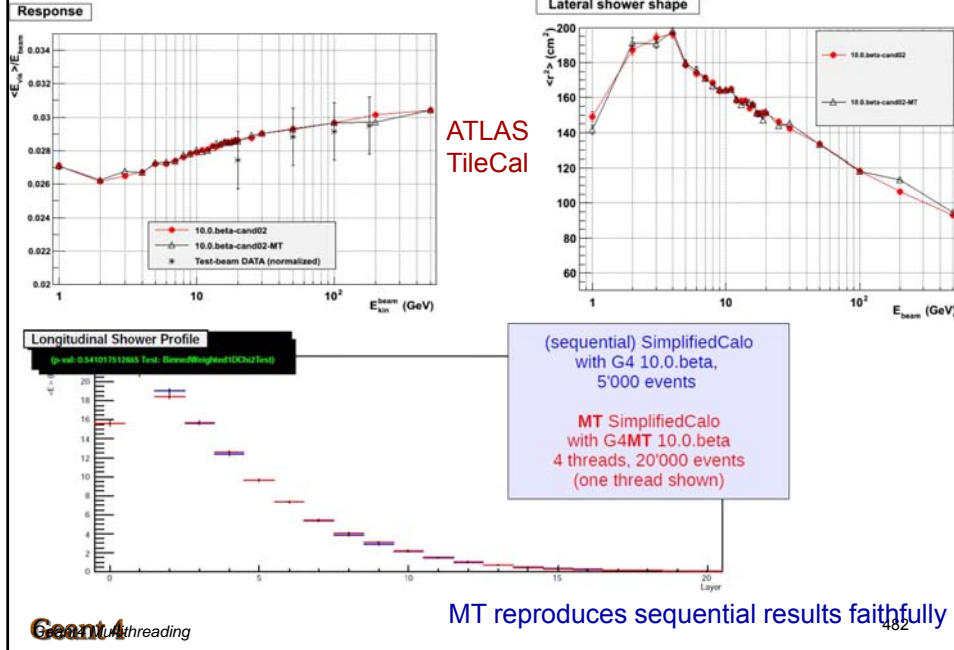
Geant4

480

Computing Performance



Physics Performance



Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

*This course encompasses training material developed by several Geant4
members:*

thanks to all of them!

Geant 4

Analysis

In a Geant4-based simulation application

Geant 4

See N45-8 talk

- Geant4 is a particle transport system
- Data analysis is out of its scope
- A user is responsible for his/her own functionality for data analysis in his/her simulation application
 - Many software tools exist, which may be used for analysis of simulation results
- Basic strategy**
 - Store simulation output in an appropriate format for further analysis
 - Process the data after the simulation is over using analysis tools
- Several systems for data analysis are available
 - AIDA-compliant tools, ROOT, ...
 - Gnuplot, Matlab, Octave, ...
 - R
- Choose an analysis tool according to your needs**

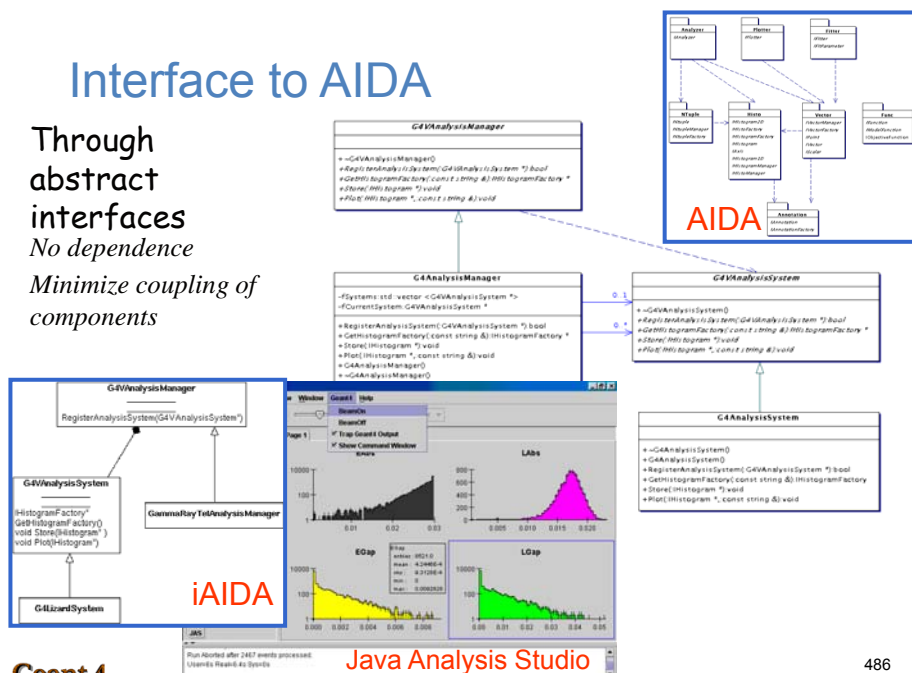
Demo application in this course: AIDA + iAIDA + R

Geant 4

485

Interface to AIDA

Through
abstract
interfaces
No dependence
Minimize coupling of
components



486

Interfacing to Geant4

- Requires setting the environmental variable `G4ANALYSIS_USE`
- AIDA (Abstract Interfaces for Data Analysis) can be used in Geant4
 - Requires AIDA headers installed in the system
 - Requires an AIDA compliant tool for analysis
- Tools for analysis compliant with AIDA interfaces currently are:
 - iAIDA (C++)
 - JAS (Java Analysis Studio)
 - Open Scientist Lab (C++)
 - PAIDA (Python)

Geant 4

487

Geant4 Installation

- Supported platforms:
 - Scientific Linux 5 + 6 with gcc 4.1.2/4.6
 - Mac Os X 10.7 and 10.8 with gcc 4.21
 - Windows7 with Visual Studio 10
- Other Linux distributions will usually work as well but paths to external libraries may have to be given manually, e.g. often Qt when using Ubuntu
- Geant4 itself requires only aforementioned compilers + CMake
- Additional capabilities (Visualization, GDML...) require external libraries (OGL, Mesa, Qt, LibXerces...)

Geant 4

488

Geant4 Installation

- Installation is CMake based
- Download source and untar/zip into (temporary) directory “/path/to/”, create geant4.x.x dir
- Create build directory “/path/to/geant4-build”
- In build dir:
`cmake -DCMAKE_INSTALL_PREFIX /install_dir/
[additional options] ../path/to/geant4.x.x`
- `make [-jN]` , with N cores to compile on
- `make install`

Geant 4

489

Geant4 Installation – Some Useful Options

- **-DGEANT4_INSTALL_DATA=ON**
Downloads and installs data files during make
- **-DGEANT4_USE_GDML=ON**
Build GDML support
- **-DGEANT4_USE_QT=ON**
Build Qt user interface and visualization driver
- **-DGEANT4_USE_OPENGL_X11=ON**
Build OpenGL X11 visualization driver
- Full list of options available in Geant4 installation documentation

Geant 4

490

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

This course encompasses training material developed by several Geant4 members:

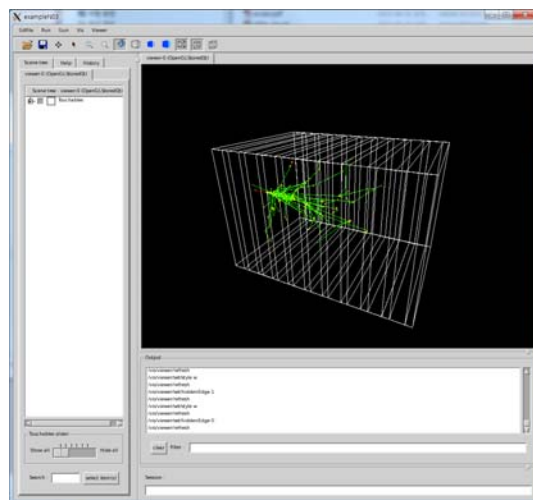
thanks to all of them!

Geant 4

Demonstration of Geant4

♦ examples /novice/N03

- A Layer consists of an absorber (Pb) and liquid detection gap (Ar)
- All ElectroMagnetic (EM) + decay process
- Energy deposit & track length
- Visualization



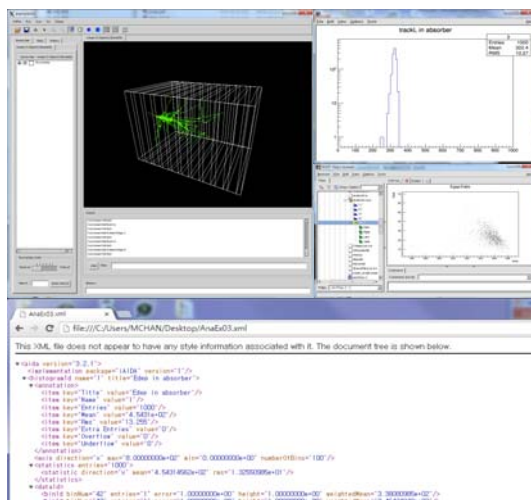
Geant 4

492

Demonstration of Geant4

examples /extended/analysis

- N03 + Histograms
- AnaEx01 → using
G4Analysis class
(*HBOOK, *.CSV, ...)
- AnaEx02 → using
ROOT library
(*ROOT)
- AnaEx03 → using
AIDA library
(*AIDA, *.XML, ...)

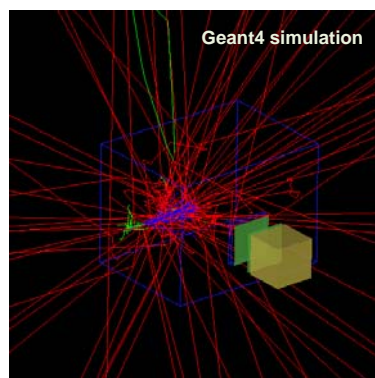
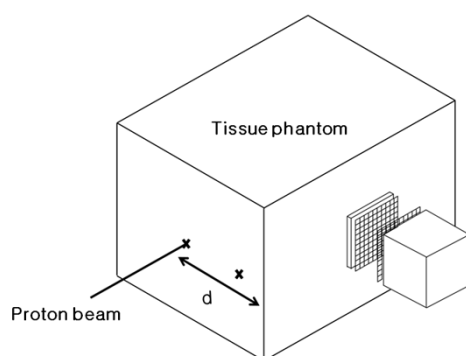


Geant 4

493

Demonstration of Geant4

• Preliminary study for hardware development

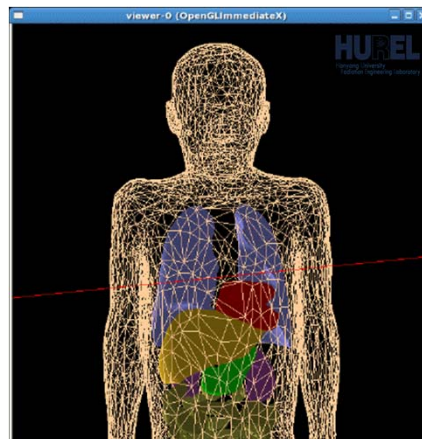


Geant 4

494

Demonstration of Geant4

- Time dependent (4D) simulation

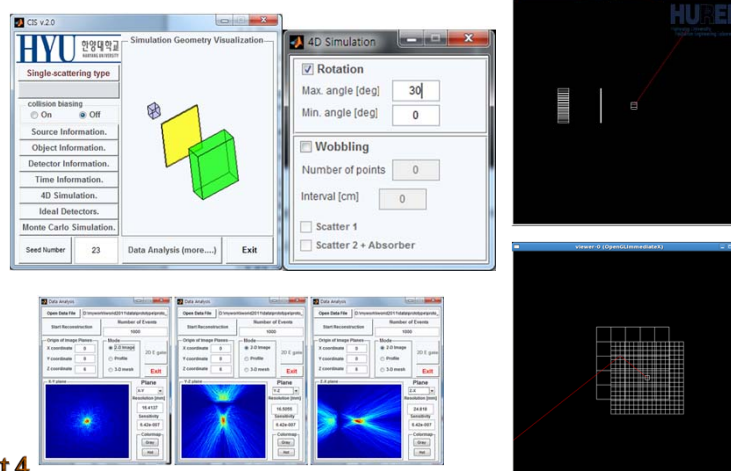


Geant 4

495

Demonstration of Geant4

- Stand-alone simulation code using Geant4



Geant 4

496

Geant 4

IEEE Nuclear Science Symposium and Medical Imaging Conference
Short Course

The Geant4 Simulation Toolkit

Sunanda Banerjee (*Saha Inst. Nucl. Phys., Kolkata, India*)

Min Cheol Han (*Hanyang Univ., Seoul, Korea*)

Steffen Hauf (*XFEL, Hamburg, Germany*)

Maria Grazia Pia (*INFN Genova, Italy*)

MariaGrazia.Pia@ge.infn.it

Seoul, 27 October 2013

<http://www.ge.infn.it/geant4/events/nss2013/geant4course.html>

*This course encompasses training material developed by several Geant4
members:*

thanks to all of them!

Geant 4

How to learn more

Documentation

User Support

Geant 4

Geant4 Web Home Page

- The main source of information

- Address

- <http://cern.ch/geant4>

- You can find

- Documentation
 - User Forum
 - Support
 - Download



- Results & Publications
- News (*sorry, incomplete...*)
- Organization

Geant 4

499

Main User Manuals

- Installation Guide

- How to install Geant4 in the user's computing environment
 - Detail instructions for supported platforms

- User's Guide: For Application Developers

- Most important document both for novice and advanced users
 - Step-by-step tutorial for a novice user
 - How to use the toolkit with a lot of example code
 - You should read this first if you are new to Geant4

- User's Guide: For Toolkit Developers

- For a user who wants to extend the functionality
 - For example, to add a new physics process, a new volume shape, etc.
 - Description of the object-oriented analysis and design
 - Guidance on how to extend specific aspects of the functionality of each package

Geant 4

500

Publications in scholarly journals

- S. Agostinelli et al.
Geant4: a simulation toolkit
NIM A, vol. 506, pp. 250-303, 2003
- J. Allison et al.
Geant4 Developments and Applications
IEEE Trans. Nucl. Sci., vol. 53, no. 1, pp. 270-278, 2006
- Other collections of publications authored by Geant4 collaboration members at
 - <http://geant4.web.cern.ch/geant4/results/publications.shtml>
 - <http://www.ge.infn.it/geant4/papers/index.html>
- Beware of the difference between regular articles published in refereed journals and conference papers
(which sometimes are published in the same journals!)

Geant 4

501

User Support: HyperNews User Forum

- Exchange of questions and experience among users and developers
- Based on the HyperNews system
 - Anyone can read
 - Anyone can add messages
 - Anyone can join as a member
- Many subjects
 - Control of runs, events, tracks, particles
 - Experimental Setup
 - General matters
 - Interfaces
 - Physics etc...



Geant 4

502

Conclusion

Feedback and outlook

Geant 4

What comes next

- Once cannot learn Geant4 in one day...
 - ...but hopefully the vision you acquired today will guide you in your next steps
- Feel free to contact the instructors after the course
 - Your feedback is welcome!
- Various Geant4-related papers by former Geant4 Course students published in IEEE TNS
 - a pleasure for the Associate Editor!

IEEE TRANSACTIONS ON	
NUCLEAR SCIENCE	
A PUBLICATION OF THE IEEE NUCLEAR AND PLASMA SCIENCES SOCIETY	
JUNE 2007	VOLUME 54 NUMBER 3 IETNAE (ISSN 0018-9489)
PART 2 OF TWO PARTS	
REGULAR PAPERS	
ANALOG AND DIGITAL CIRCUITS	
Resolution Limits in 130 nm and 90 nm CMOS Technologies for Analog Front-End Applications.....	512
A 60-mW High-Linearity CMOS Peak-Detectifier/Transimpedance Amplifier.....	518
ASIC for Small-Angle Neutron Scattering Experiments at the SNS.....	541
40-Channel 10-ps Time-Resolvable Counter Array for Long-Term Continuous Event Counting.....	549
Noise Optimization of Charge Amplifiers With MOS Input Transistors Operating in Moderate Inversion Region for Short-Pulsing Times.....	555
ACCELERATORS AND SPACE INSTRUMENTATION	
Beam Performance of PHENIX: A Parameter for High-Energy X-rays.....	561
Characterization of the Zero-Degree Calorimeter for the ALICE Experiment.....	567
Performance of the Zero-Degree Calorimeter for the ALICE Experiment.....	574
COMPUTING AND SOFTWARE	
Geant Model for the Stopping Power of Low-Energy Negatively Charged Hadrons.....	578
Geant4: A Software Framework for Accelerator and High-Energy Physics.....	585

IEEE

504

Geant 4

Want to become an expert?

- Individual training projects
- Very successful experience!



Contact us: maria.grazia.pia@cern.ch
mariagrazia.pia@ge.infn.it

Geant 4

505