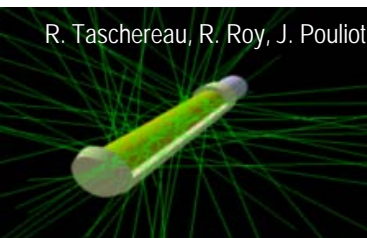
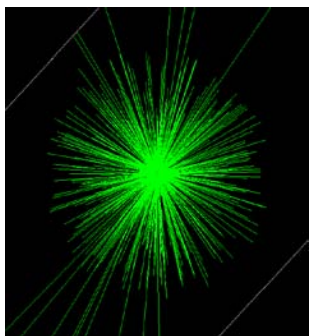


*Courtesy T. Ersmark, KTH Stockholm*



R. Taschereau, R. Roy, J. Pouliot



# Geant 4

## Using Geant4

Maria Grazia Pia

*INFN Genova, Italy*

**ANS Winter Meeting 2010**

Thursday, 11 November 2010

Las Vegas, NV

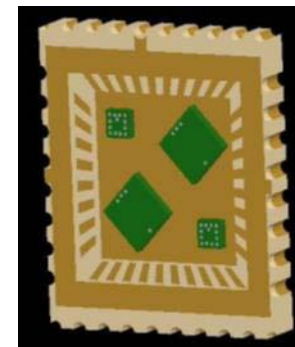
<http://cern.ch/geant4>



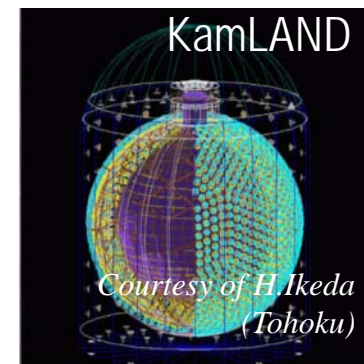
*GATE  
Collaboration*



*ATLAS  
Collaboration*

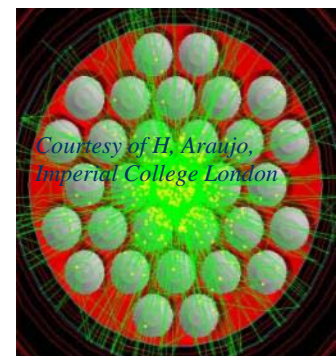


*RADMON and INFN Genova*



**KamLAND**

*Courtesy of H. Ikeda  
(Tohoku)*



*Courtesy of H. Araujo,  
Imperial College London*



Lina Quintieri (*INFN LNS*) and Mauro Augelli (*CNES*)

# 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 simulation
  - The user must deliberately **choose** which Geant4 tools to use
- Guidance: many **examples** are distributed with Geant4

# Basic actions

- What a user **must** do:
  - Describe the **experimental set-up**
  - Provide **primary particles** input to 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 and track secondary particles*)
- One may also **want**
  - To interact with Geant4 kernel to **control** the simulation
  - To **visualise** the simulation configuration or results
  - To produce **objects encoding simulation results** to be further analysed

# Interaction with Geant4 kernel

- Geant4 design provides **tools** for a user application
  - To tell the kernel about one's simulation configuration
  - To interact with Geant4 kernel itself
- Geant4 tools for user interaction are **base classes**
  - One creates **one's own concrete class** derived from the base classes
  - Geant4 kernel handles 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 methods) for user interaction
  - User derived classes are **optional**



# User classes

## Initialisation classes

*Invoked at initialization*


- *G4VUserDetectorConstruction*
- *G4VUserPhysicsList*

## Action classes

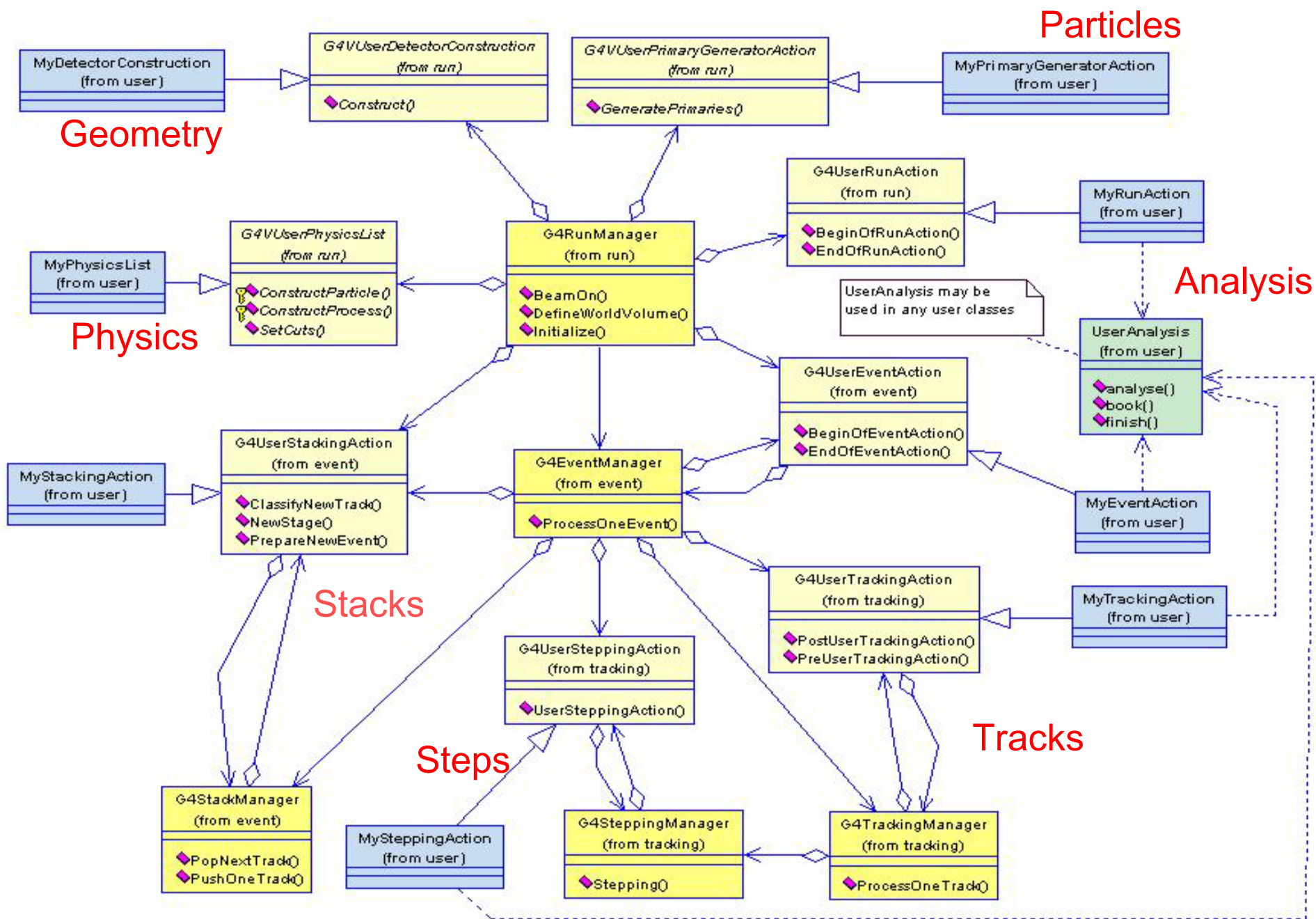
*Invoked during the execution*

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

## Mandatory classes:

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

## Overview of Geant4 advanced examples



# The main function

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



# main()

```
{  
  ...  
  // Instantiate the run manager  
  G4RunManager* runManager = new G4RunManager;  
  
  // Instantiate mandatory user initialization classes,  
  // notify runManager  
  MyDetectorConstruction* detector = new MyDetectorConstruction;  
  runManager->SetUserInitialization(detector);  
  MyPhysicsList* physicsList = new MyPhysicsList;  
  runManager->SetUserInitialization(myPhysicsList);  
  
  // Mandatory user action classes  
  runManager->SetUserAction(new MyPrimaryGeneratorAction);  
  
  // Optional user action classes  
  MyEventAction* eventAction = new MyEventAction();  
  runManager->SetUserAction(eventAction);  
  MyRunAction* runAction = new MyRunAction();  
  runManager->SetUserAction(runAction);  
}
```

# Describe the experimental set-up

- Derive one's own **concrete class** from ***G4VUserDetectorConstruction*** **abstract base class**
- Implement the **Construct()** method
  - construct all necessary **materials**
  - define **shapes/solids** required to describe the geometry
  - **construct** and **place volumes** of one's detector geometry
  - define **sensitive detectors** and identify detector volumes to associate them to
  - associate **magnetic field** to detector regions
  - define **visualisation** attributes for the detector elements

# How to define materials

Different kinds of materials can be defined

Isotopes  
Elements  
Molecules  
Compounds and mixtures

```
PVPhysicalVolume* MyDetectorConstruction::Construct()
```

```
{
```

```
...
```

```
a = 207.19*g/mole;
```

```
density = 11.35*g/cm3;
```

```
G4Material* lead = new G4Material(name="Pb", z=82., a, density);
```

```
density = 5.458*mg/cm3;
```

```
pressure = 1*atmosphere;
```

```
temperature = 293.15*kelvin;
```

```
G4Material* xenon = new G4Material(name="XenonGas", z=54.,  
a=131.29*g/mole, density,  
kStateGas, temperature, pressure);
```

```
...
```

```
}
```

# 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;
```

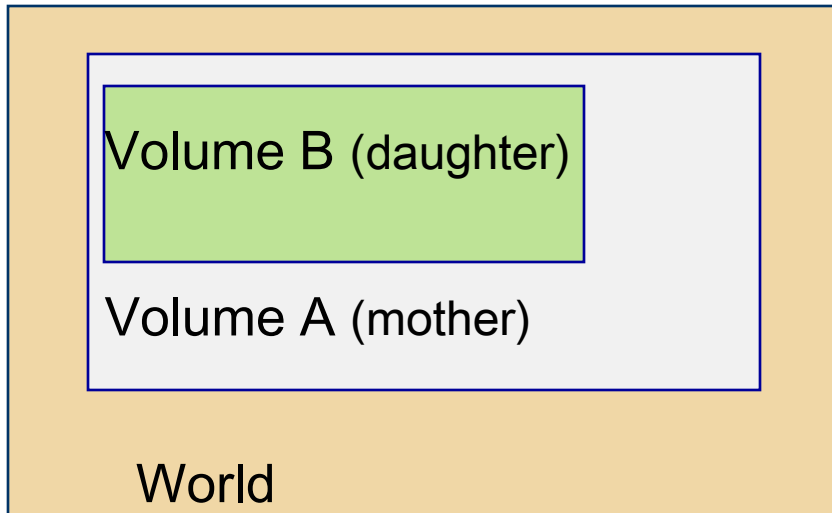
```
G4Material* scintillator = new G4Material(name = "Scintillator", density,  
numberOfComponents = 2);
```

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

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

# 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



# 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
```

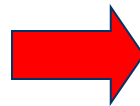
# How to build a volume inside the World

```
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
```

# Select physics processes

- Geant4 does not have any default particles or processes
- Derive a **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

# PhysicsList: particles and cuts

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

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

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

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

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

← Set the **production threshold**

# 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

# Physics List: processes

```
void MyPhysicsList :: ConstructProcesses()
```

Select physics processes to be activated for each particle type

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

```
    else if (particleName == "e-")  
    {
```

Geant4 *Standard* electromagnetic processes are selected in this example

```
        pManager->AddProcess(new G4MultipleScattering(), -1, 1,1);  
        pManager->AddProcess(new G4elonisation(), -1, 2,2);  
        pManager->AddProcess(new G4eBremsstrahlung(), -1,-1,3);  
    }
```

```
    else if (particleName == "e+")  
    {
```

```
        pManager->AddProcess(new G4MultipleScattering(), -1, 1,1);  
        pManager->AddProcess(new G4elonisation(), -1, 2,2);  
        pManager->AddProcess(new G4eBremsstrahlung(), -1,-1,3);  
        pManager->AddProcess(new G4eplusAnnihilation(), 0,-1,4);  
    }
```



# 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()**

# 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);  
}
```

# 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

# 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 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\*)

# 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)



# Select (G)UI and visualisation

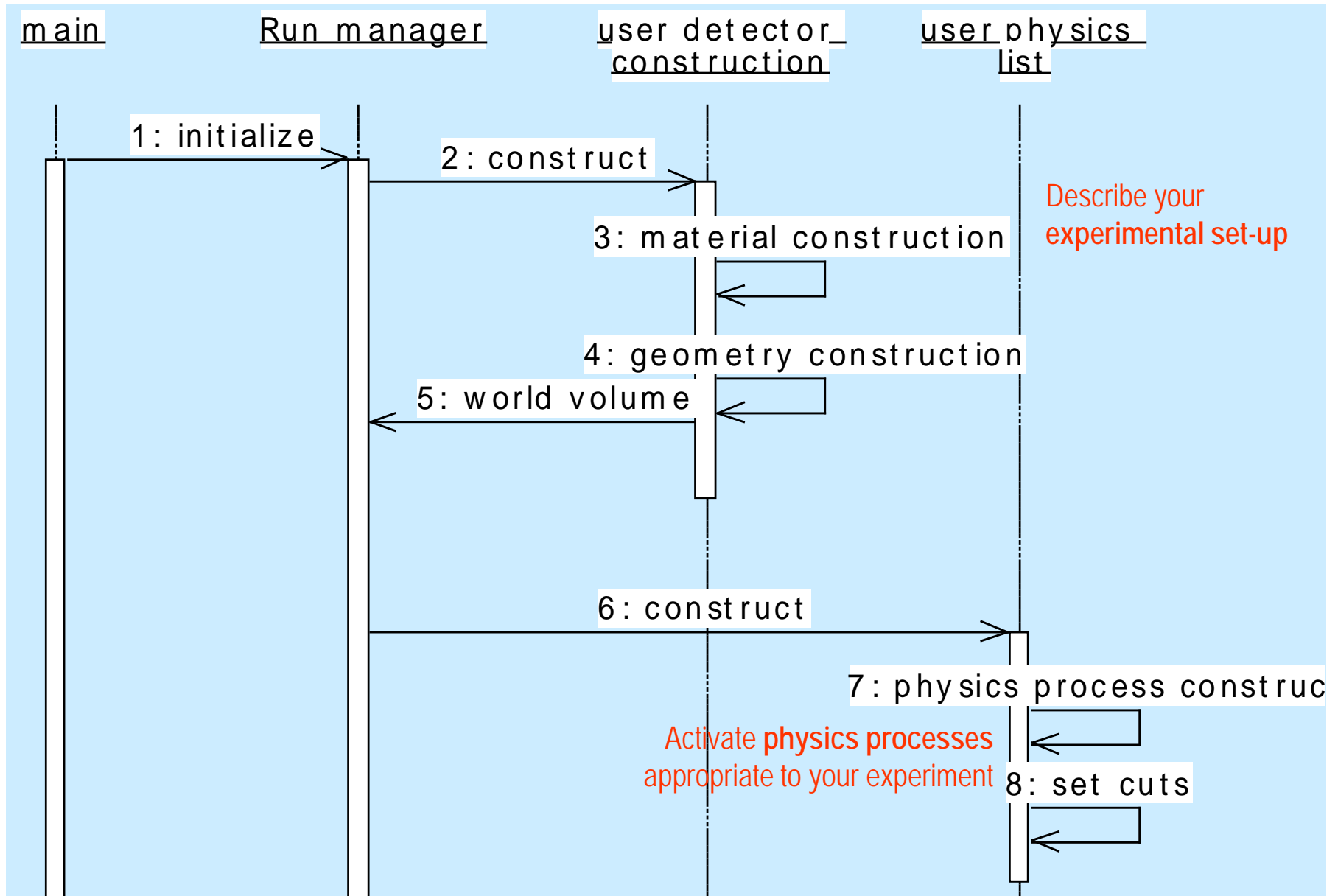
- In your **main()**, taking into account your computer environment, **instantiate a G4UIsession concrete class** provided by Geant4 and invoke its **sessionStart()** method
- Geant4 provides:
  - G4UITerminal
  - csh or tcsh like character terminal
  - G4GAG
  - tcl/tk or Java PVM based GUI
  - G4Wo
  - Opacs
  - G4UIBatch
  - batch job with macro file
  - ...
- 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
  - ...

# Recipe for novice users

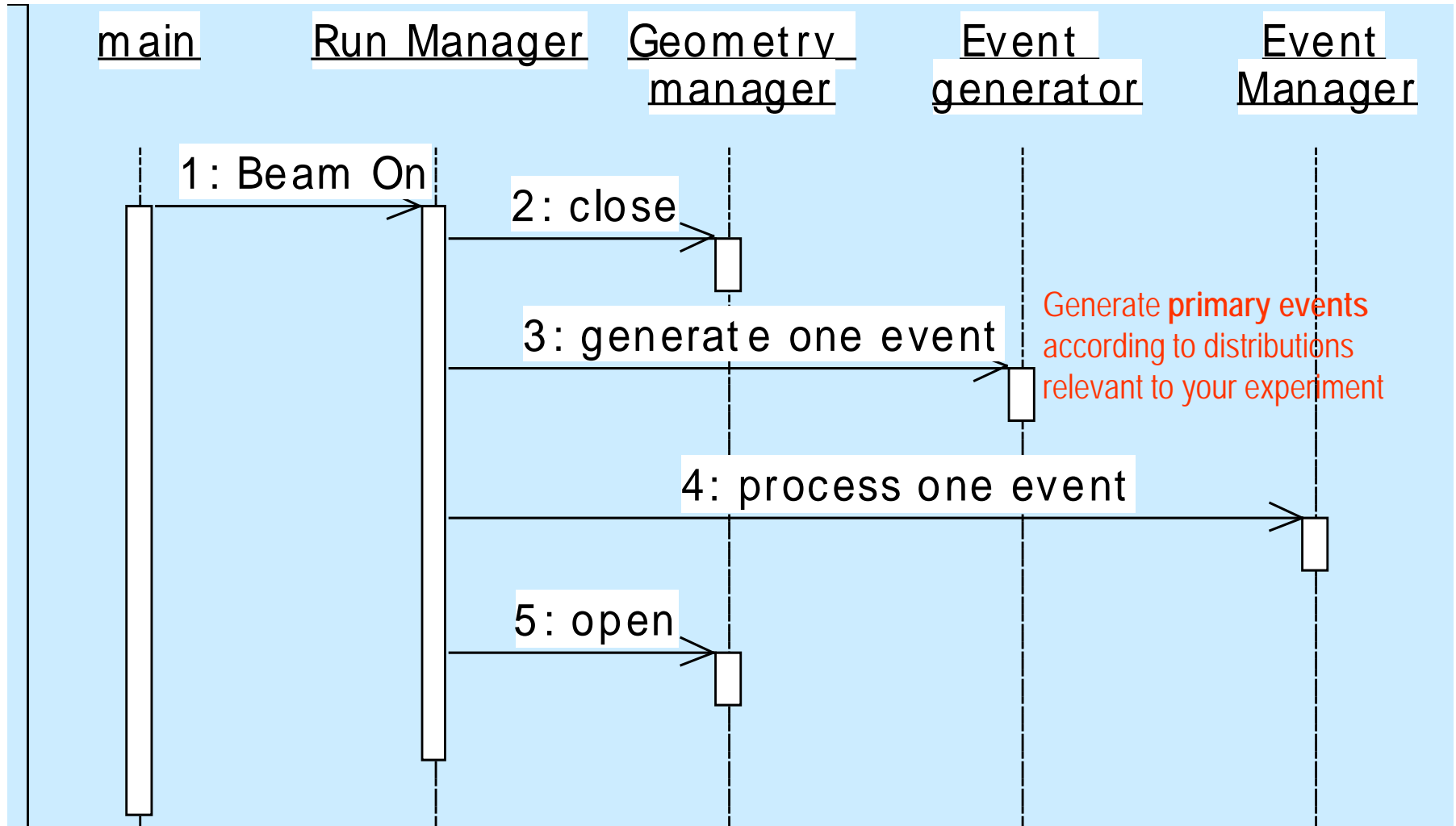
Experienced users may do much more, but the conceptual process is still the same...

- 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!**

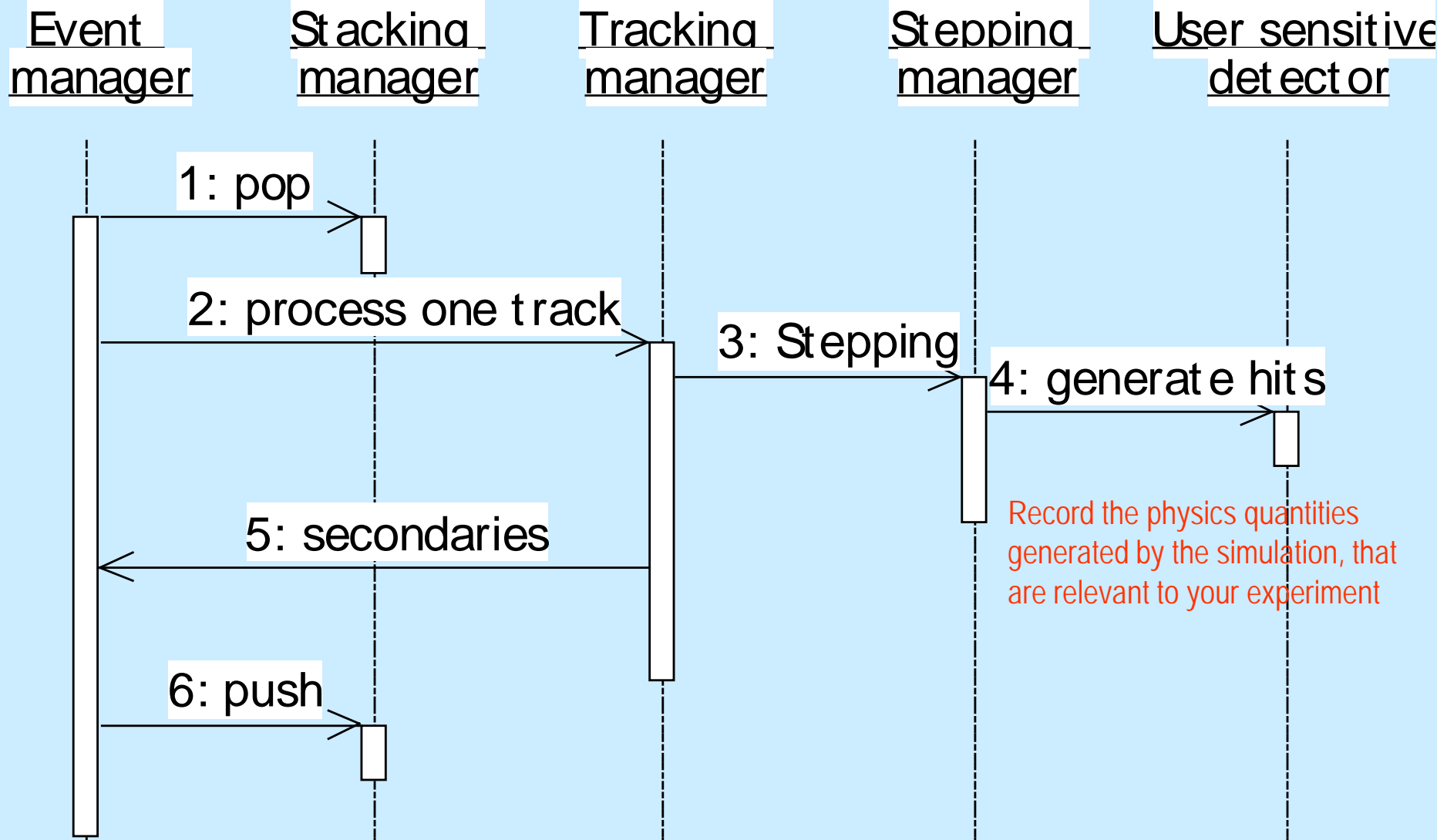
# Initialisation



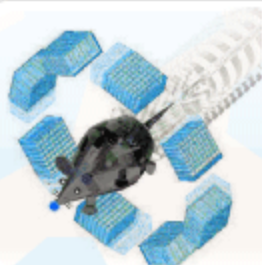
# Beam On



# Event processing







# GATE

SIMULATIONS OF  
PRECLINICAL AND  
CLINICAL SCANS IN  
EMISSION TOMOGRAPHY

[www.opengatecollaboration.org](http://www.opengatecollaboration.org)

You are here: [Forewords](#) >

[Forewords](#)  
[GATE-modelled systems](#)  
[Mailing List](#)  
[Publications](#)  
[Register](#)  
[OpenGATE collaboration](#)  
[Related projects](#)  
[Position announcements](#)  
[Training](#)  
[A brief history of GATE](#)

## Registered users

Already a registered user or  
a member of the  
collaboration?

Click here to **log in**

[Gate versions](#)

## Forewords

GATE is an advanced opensource software developed by the international OpenGATE collaboration and dedicated to the numerical simulations in medical imaging. It currently supports simulations of Emission Tomography (Positron Emission Tomography - PET and Single Photon Emission Computed Tomography - SPECT), and Computed Tomography (CT). Using an easy-to-learn macro mechanism to configure simple or highly sophisticated experimental settings, GATE now plays a key role in the design of new medical imaging devices, in the optimization of acquisition protocols and in the development and assessment of image reconstruction algorithms and correction techniques.

The OpenGATE collaboration has been awarded in 2009 the Physics in Medicine and Biology (PMB) Citations Prize for the research paper "GATE: a simulation toolkit for PET and SPECT". The annual prize is presented to the authors of the PMB research paper that received the most citations in the preceding five years ( according to ISI).



## Search



-> [advanced search](#)

## News

**GATE 6.0.0 is now released!**

Beginning of February:

GATE V6, supporting simulations of  
radiotherapy and hadrontherapy...

[\[more\]](#)

**GATE workshop at the IEEE MIC 2009  
in Orlando**

The OpenGATE collaboration has  
recently organized a GATE workshop  
during the IEEE MIC 2009...

[\[more\]](#)

Financé par



# GAMOS

Geant4-based Architecture for Medicine-Oriented Simulations

User Name:

Password:

Login

[New User](#)

## Menu

- [Main](#)
- [Talks and Publications](#)

## Documentation

- [User Guide](#)
- [Tutorials](#)

## User Support

- [Discussion Forum](#)
- [Bug report system](#)

## Registered Users

- [Download](#)
- [Software](#)
- [Reference Manual](#)
- [Users mailing-list](#)
- [Developers mailing-list](#)

## Welcome to the home page of the GAMOS Project

GAMOS is a GEANT4-based framework that is at the same time easy-to-use and flexible.

The comprehensive scripting language makes it easy to implement the most common requirements of a Medical Physics application, without any need of C++ coding.

The plug-in technology, together with a careful modular design, a detailed documentation and a set of examples and tutorials that explain in detail how to extend the framework in different directions allows to exploit the full flexibility of GEANT4, by creating new user code or by reusing any piece of GEANT4 code and mixing it seamlessly with the existing GAMOS components.

Thanks to its big flexibility, already a sensible fraction of GAMOS users work in other fields than medical physics. If this is your case we recommend you to have a look at the 'Histogram and Scorer tutorial'

**In summary, by using GAMOS you will be able to carry your GEANT4-based simulation in an easy way and at the same time you will have the flexibility of using any of the GEANT4 components and mix with or substitute the GAMOS components.**

## Related Links

- [Geant4](#)
- [CLHEP](#)
- [SEAL](#)
- [ROOT](#)
  
- [G4EMU](#)
- [G4NAMU](#)

## News

- **4 July 2010: New GAMOS release 2.2.0! (see [release notes](#))**



## MULTi-Layered Shielding SIMulation Software (MULASSIS)

MULASSIS is a M-C simulation based tool for dose and particle fluence analysis associated with the use of radiation shields. Users can define the shielding and detector geometry as planar or spherical layers, with the material in each layer defined by its density and elemental/isotopic composition. Incident particles can be any Geant4 particles, these include protons, neutrons, electrons, gammas, alphas and light ions. There is a wide choice for their initial energy and angular distribution. In addition, radiation spectra produced by SPENVIS can be inputted when the tool is used within this system.

Users can carry out fluence, total Ionising/Non-Ionising Energy Loss (NIEL) dose and Pulse Height Spectrum (PHS) analysis for any layer in the geometry. Fluence can be tallied into energy distribution histograms as a function of particle type and particle angular direction. NIEL analysis can be performed only for layers of silicon material only, as the required NIEL coefficients are not available for other material at the moment. The histograms are output in Comma Separated Values (CSV) format so they can be easily input into other analysis and plotting tools.

Users who have a local Geant4 installation can compile the MULASSIS code and use the tool interactively on their local system. The user can make full use of the Geant4 visualisation facilities for the shielding geometry and the particle interaction tracks.

MULASSIS has been integrated into the ESA SPENVIS system, thus making it one of the tools SPENVIS users can choose for their radiation analysis. MULASSIS and the Geant4 toolkit operate on a separate PC/Linux server which is http linked to the SPENVIS server. The full MULASSIS functionality is available via the SPENVIS user interface. In addition the radiation environment as evaluated by other SPENVIS models can be used as inputs for the MULASSIS simulation, thus making it very easy to obtain the modified radiation spectra behind a specific shield defined by the user.

The main reference paper for MULASSIS is published in IEEE Transactions on Nuclear Science [Vol 49 No 6 \(2002\) P2788-2793](#). Further information on MULASSIS, as well as the source code to be used with the Geant4 Toolkit, can be found in the documents and gzip file listed below.

### News:

11-February-2010: Version 1.21 released.

30-January-2009: Version 1.19 released. Fluence tally for primary and ions added.






5-December-2007: Version 1.17 released. Details in the [History file](#).

25-August-2006: [Microsoft Window version of MULASSIS](#) is available now!

23-August-2006: Version 1.14 released. Details in the [History file](#).

27-October-2005: Version 1.10 released. Details in the [History file](#).

### Download Documents

User Requirements Document	 PDF version
Software Specification Document	 PDF version
Interface Control Document	 PDF version
Software User's Manual	 PDF version
Need Adobe Acrobat Reader? Get it <a href="#">here</a> .	

### Download Source Code

*Note that correct version of [Geant4](#) must have already been installed on your system before installing the MULASSIS.*

[Version 1.21](#): Tested with Geant4 9.3

[Version 1.19](#): Tested with Geant4 9.2

[Version 1.17](#): Tested with Geant4 9.0

[Version 1.15](#): Tested with Geant4 8.1

[Version 1.10](#): Tested with Geant4 7.1

[Version 1.9](#): Tested with Geant4 7.1

[Version 1.5](#): Tested with Geant4 7.0

[Version 1.3](#): Tested with Geant4 6.2 (tag geant4-V06-02-ref-01).

[Version 1.2](#): Tested with Geant4 6.1 (tag geant4-V06-01-ref-02).



# Overview: MRED (Monte Carlo Radiative Energy Deposition)

Robert A. Weller, Marcus H. Mendenhall,  
Brian D. Sierawski, & Robert A. Reed  
Institute for Space & Defense Electronics  
Vanderbilt University

Sponsoring Agencies: NASA, DTRA, AFOSR



UNIVERSITY OF  
FLORIDA



NC STATE UNIVERSITY

THE STATE UNIVERSITY OF NEW JERSEY  
RUTGERS



[robert.a.weller@vanderbilt.edu](mailto:robert.a.weller@vanderbilt.edu)

1

VANDERBILT UNIVERSITY

Electrical Engineering & Computer Science

Slides available at  
<http://www.ge.infn.it/geant4/training>

Collection of physics references  
<http://www.ge.infn.it/geant4/papers>

General information: <http://cern.ch/geant4>

Acknowledgment: **Geant4 developers and users**