The Geant4 concept

Luciano Pandola

INFN

GSSI, L'Aquila

http://geant4.org

What is Geant 4

- Toolkit for the **Monte Carlo simulation** of the interaction of **particles** with **matter**
  - physics processes (EM, hadronic, optical) cover a comprehensive set of particles, materials and over a wide energy range
  - it offers a complete set of support functionalities (tracking, geometry, hits)
- **Distributed** software production and management: developed by an **international Collaboration**
  - Established in 1998
  - Approximately 100 members, from Europe, America and Asia
- Written in **C++** language
  - Takes advantage from the Object Oriented software technology
- **Open source**

*S. Agostinelli et al., Nucl. Instr. Meth. A **506** (2003) 250*

*J. Allison et al., IEEE Trans. Nucl. Scie. **53** (2006) 270*

Geant 4

Download | User Forum | Gallery
Contact Us

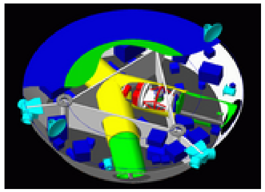
Search Geant4

Geant4 is a toolkit for the simulation of the passage of particles through matter. Its areas of application include high energy, nuclear and accelerator physics, as well as studies in medical and space science. The two main reference papers for Geant4 are published in *Nuclear Instruments and Methods in Physics Research* A 506 (2003) 250-303, and *IEEE Transactions on Nuclear Science* 53 No. 1 (2006) 270-278.

Applications

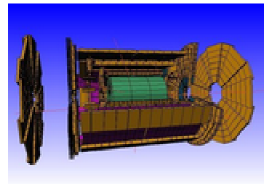
A sampling of applications, technology transfer and other uses of Geant4

User Support



Getting started, guides and information for users and developers

Publications



Validation of Geant4, results from experiments and publications

Collaboration

Who we are: collaborating institutions, members, organization and legal information

News

- *28 February 2014 -* **Patch-01** to **release 10.0** is available from the download area.
- *24 May 2013 -* **Patch-02** to **release 9.6** is available from the download area.

```
http://geant4.org
```

Events

- 2nd LPCC Detector Simulation Workshop, CERN, Geneva (Switzerland), **18-19 March 2014**.
- 37th Geant4 Technical Forum, CERN, Geneva (Switzerland), **20 March 2014**.
- 10th Geant4 Space Users Workshop, at NASA/MSFC, Huntsville, Alabama (USA), **27-29 May 2014**.
- International Workshop on Monte Carlo Techniques in Medical Physics, Quebec City (Canada), **17-20 June 2014**.
- 19th Geant4 Collaboration Meeting, Okinawa (Japan), **29 September - 4 October 2014**.
- Past events

- **Code** and **documentation** available in the main **web page**
- Regular **tutorial courses** held worldwide

Who/why is using Geant4?

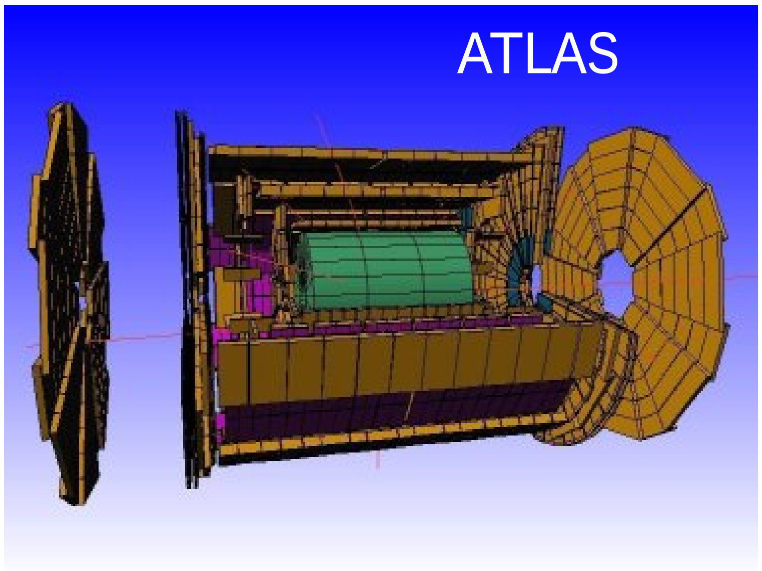
Experiments and MC

- In my knowledge, **all experiments** have a (more or less detailed) full-scale Monte Carlo simulation
- Design phase
  - Evaluation of background
  - Optimization of setup to maximize scientific yield
    - Minimize background, maximize signal efficiency
- Running/analysis phase
  - Support of data analysis (e.g. provide efficiency for signal, background, coincidences, tagging, ...).
    - often, Monte Carlo is the only way to convert *relative rates* (events/day) in *absolute yields*

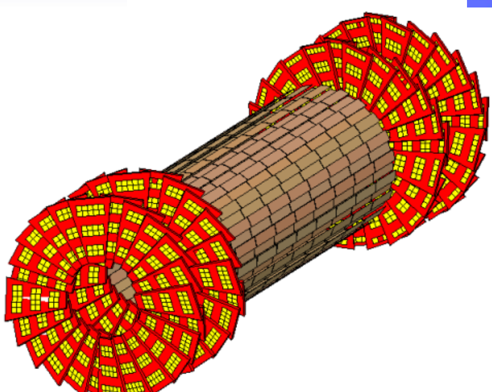
Why Geant4 is a common choice in the market

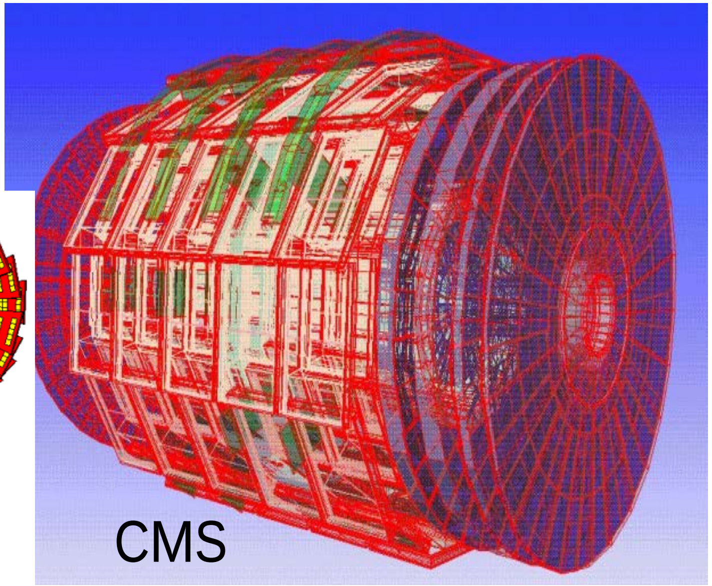
- Open source and object oriented/C++
  - No black box
  - Freely available on all platforms
  - Can be easily extended and customized by using the existing interfaces
    - New processes, new primary generators, interface to ROOT analysis, ...
- Can handle complex geometries
- Regular development, updates, bug fixes and validation
- Good physics, customizable per use-cases
- End-to-end simulation (all particles, including optical photons)

LHC @ CERN



- All four big **LHC experiments** have a Geant4 simulation
  - M of volumes
  - Physics at the TeV scale





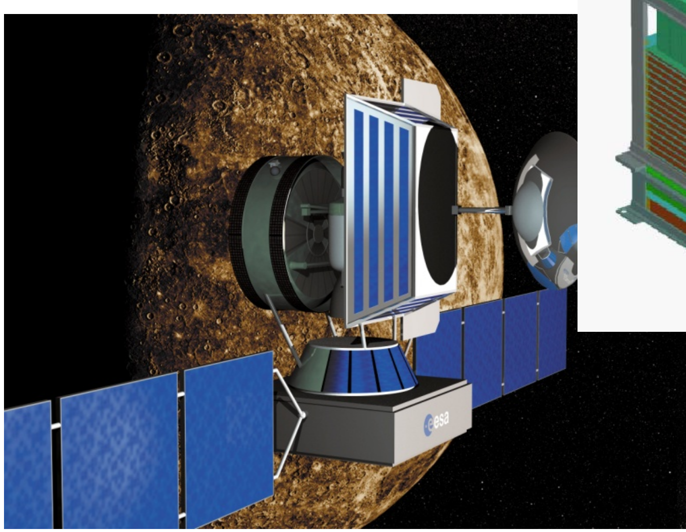
- Benchmark with test-beam data
- Key role for the Higgs searches

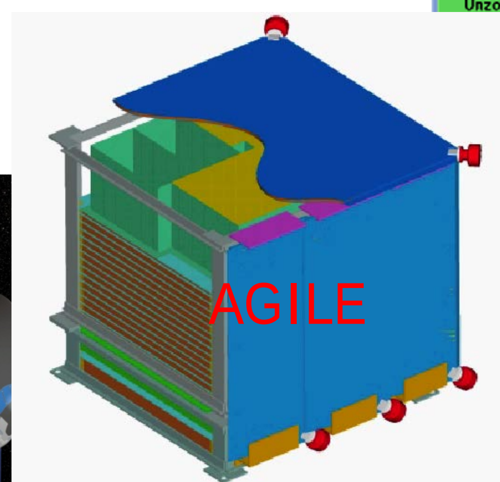
Space applications

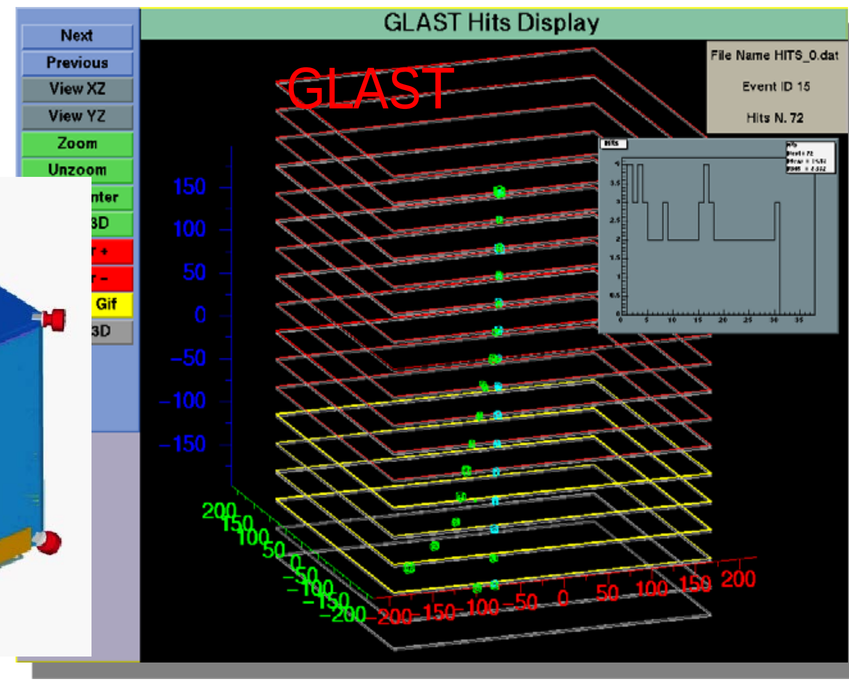
- Satellites (γ astrophysics, planetary sciences)
- Funding from ESA

<u>Typical telescope:</u>

Tracker
Calorimeter
Anticoincidence

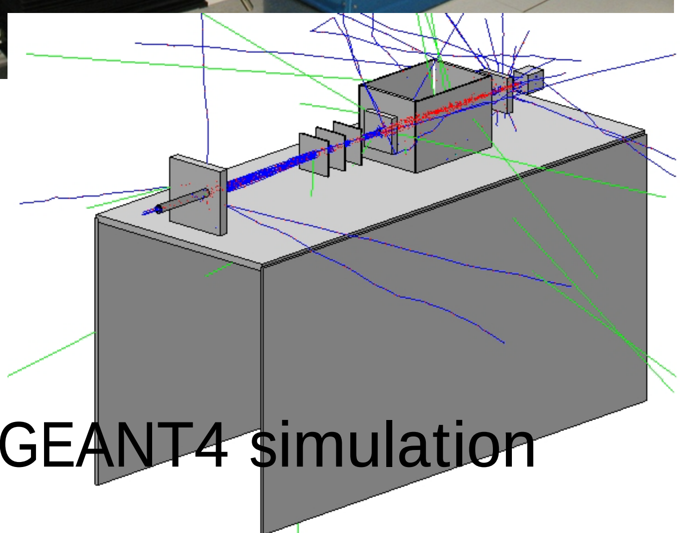






Medical applications

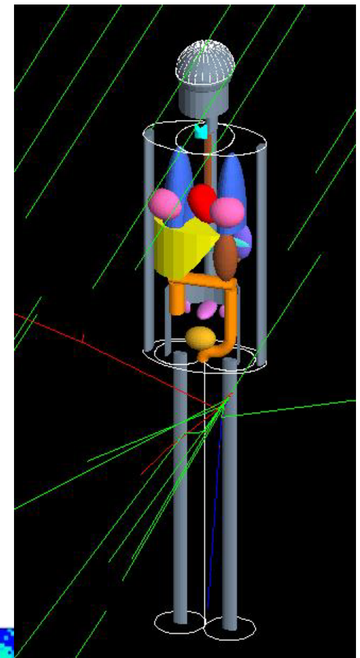
Proton-therapy beam line

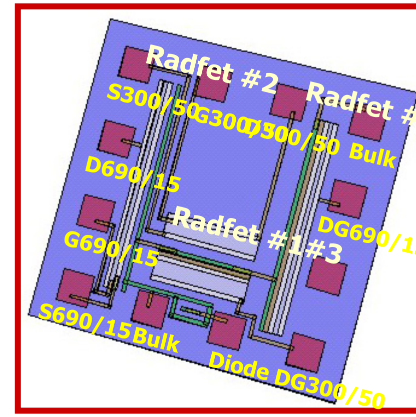


GEANT4 simulation

- **Treatment planning** for hadrontherapy and proton-therapy systems
  - Goal: deliver dose to the tumor while sparing the healthy tissues
  - Alternative to less-precise (and commercial) TP software
- Medical imaging
- Radiation fields from medical accelerators and devices
  - medical_linac
  - gamma-knife
  - brachytherapy

Dosimetry with Geant4

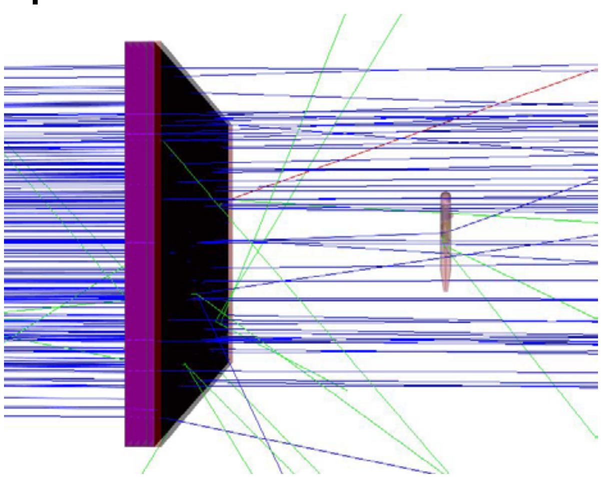


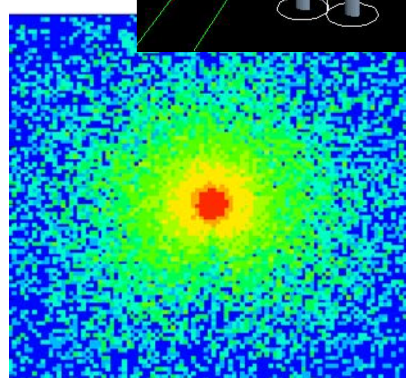


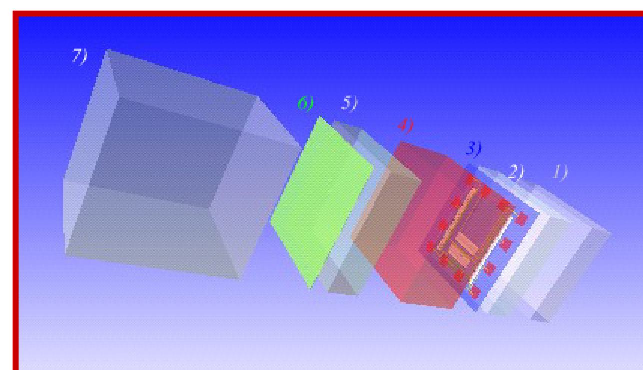
Space science

Radiotherapy

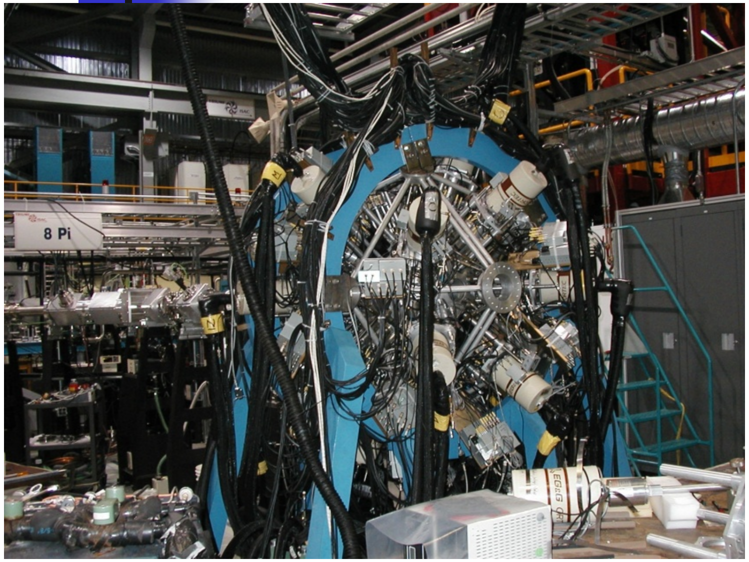
Effects on electronics components

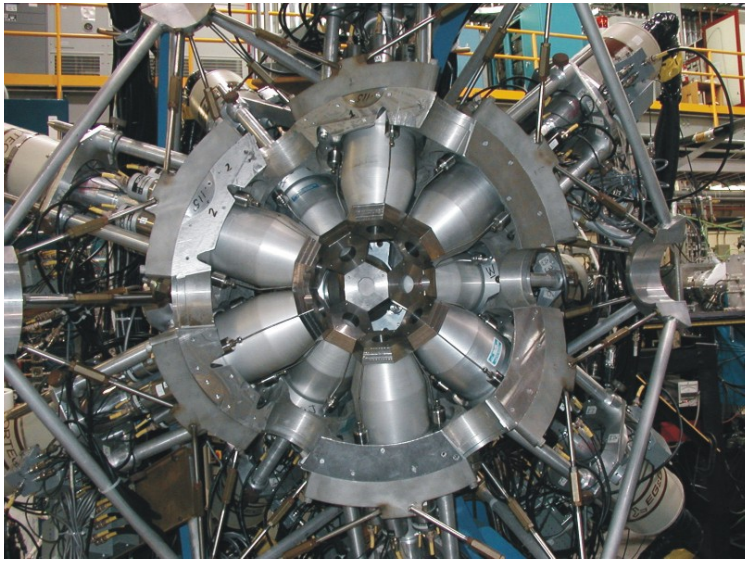


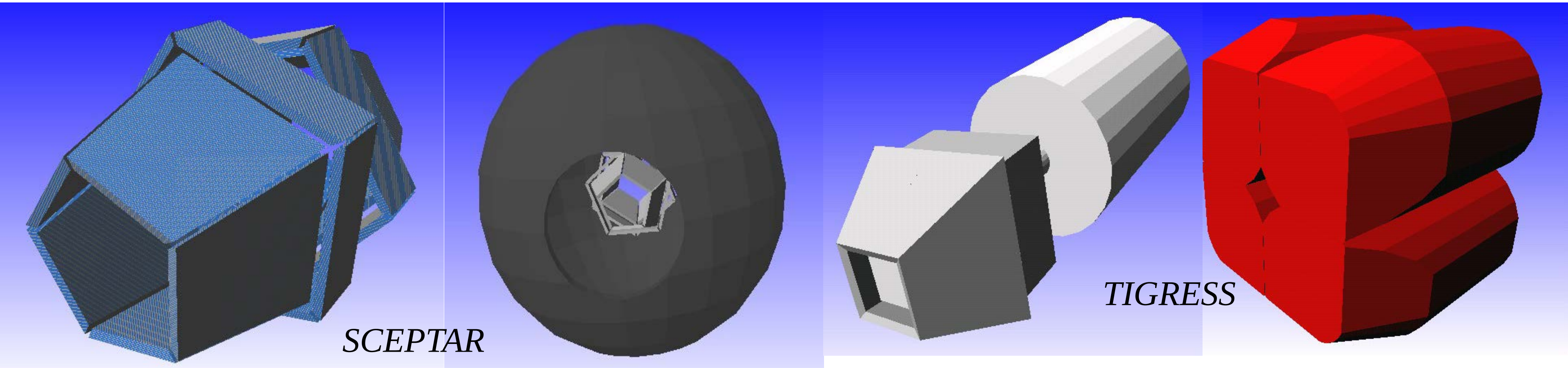




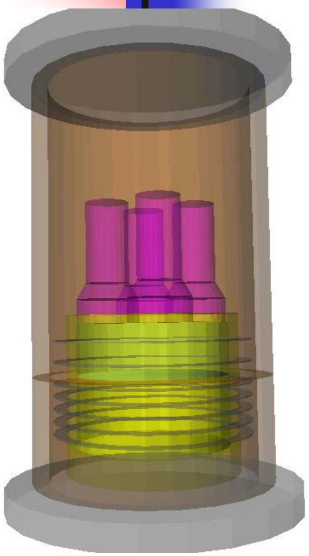
Nuclear spectroscopy





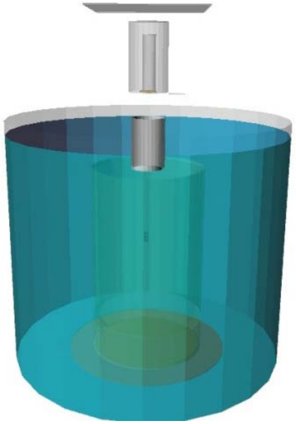


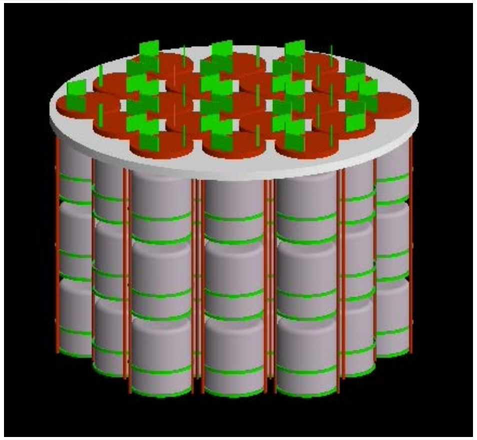
Low background experiments



Neutrinoless ββ decay:

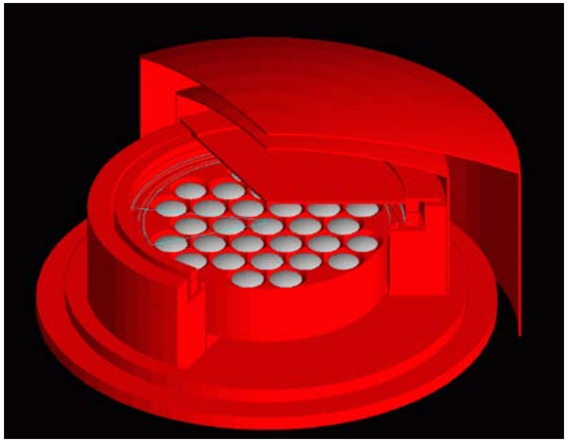
GERDA, Majorana
COBRA, CUORE, EXO

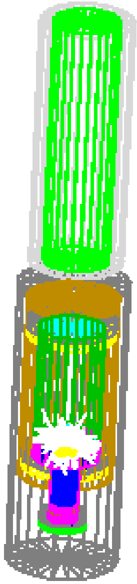




Dark matter detection:

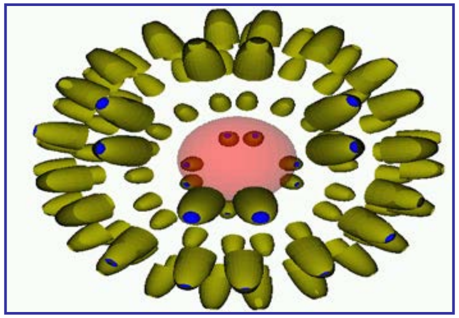
Zeplin-II/III, Drift, Edelweiss, ArDM, Xenon, CRESST, Lux, Elixir,



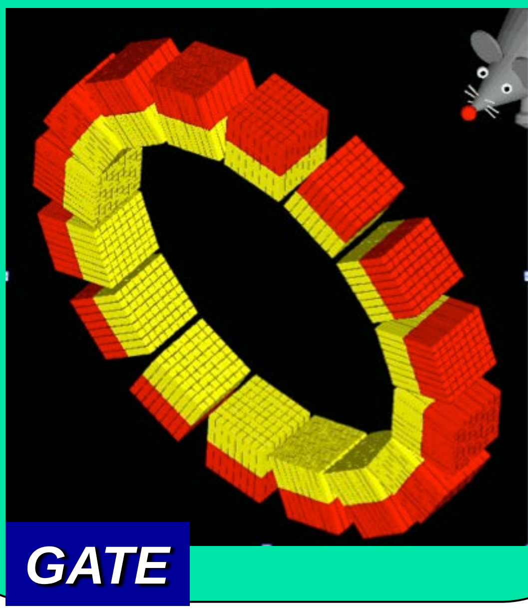


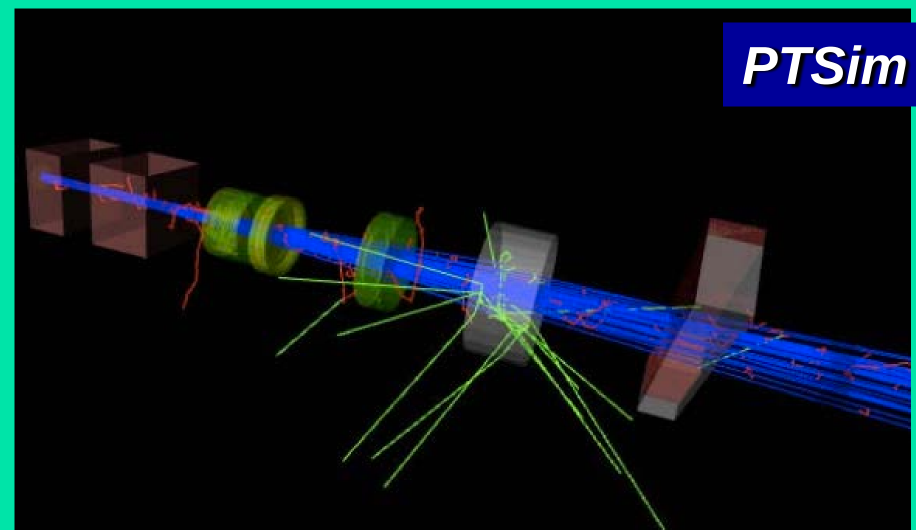
Solar neutrinos:

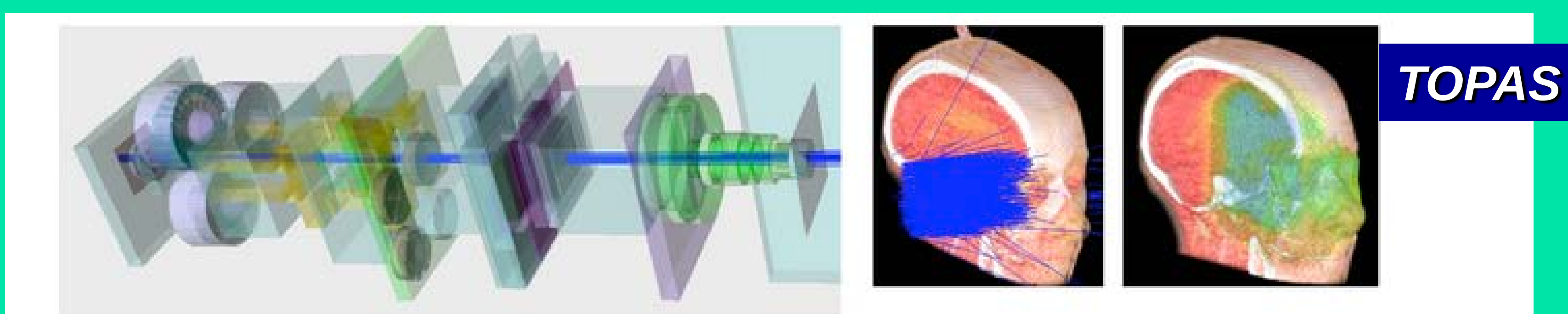
Borexino, ...



Geant4-based frameworks in the medical physics







Basic concept of Geant4

Toolkit and User Application

- Geant4 is a **toolkit** (= a collection of tools)
  - i.e. you cannot “run” it out of the box
  - You must **write an application**, which uses Geant4 tools

- Consequences:
  - There are no such concepts as “Geant4 defaults”
  - You must provide the necessary information to configure your simulation
  - You must deliberately choose which Geant4 tools to use

- Guidance: many **examples** are provided
  - **Basic Examples**: overview of Geant4 tools
  - **Advanced Examples**: Geant4 tools in real-life applications

Basic concepts

- What you **MUST** do:
  - Describe your **experimental set-up**
  - Provide the **primary particles** input to your simulation
  - Decide which **particles** and **physics models** you want to use out of those available in Geant4 and the precision of your simulation (cuts to produce and track secondary particles)

- You **may also want**
  - To interact with Geant4 kernel to **control** your simulation
  - To **visualise** your simulation configuration or results
  - To produce **histograms, tuples** etc. to be further analysed

Main Geant4 capabilities

- **Transportation of a particle 'step-by-step'** taking into account all possible interactions with materials and fields
- **The transport ends** if the particle
  - is slowed down to zero kinetic energy (and it doesn't have any interaction at rest)
  - disappears in some interaction
  - reaches the end of the simulation volume
- Geant4 allows the User to access the transportation process and retrieve the results (USER ACTIONS)
  - at the beginning and end of the transport
  - at the end of each step in transportation
  - if a particle reaches a sensitive detector
  - Others...

Multi-thread mode

- Geant4 10.0 (released Dec, 2013) supports multi-thread approach for multi-core machines
  - Simulation is automatically split on an event-by-event basis
    - different events are processed by different cores
  - Can fully profit of **all cores** available on modern machines → substantial **speed-up** of simulations
  - **Unique** copy (master) of geometry and physics
    - All cores have them as read-only (saves memory)
- Backwards compatible with the **sequential** mode
  - The MT programming requires some care: need to avoid conflicts between threads
  - Some modification and porting required

The (conceptual) recipe for a Geant4-based application

Interaction with the Geant4 kernel - 1

- Geant4 design provides **tools** for a user application
  - To tell the kernel about your simulation configuration
  - To interact with Geant4 kernel itself

- Geant4 tools for user interaction are **base classes**
  - You create **your own concrete class** derived from the base classes → interface to the Geant4 kernel
  - Geant4 kernel handles your own derived classes transparently through their base class interface (polymorphism)

Interaction with the Geant4 kernel - 2

Two types of Geant4 base classes:

- **Abstract base classes** for user interaction (classes starting with G4V)
  - User derived concrete classes are **mandatory**
  - User to implement the purely virtual methods

- **Concrete base classes** (with **virtual** dummy default methods) for user interaction
  - User derived classes are **optional**

User Classes (from 10.0)

Initialisation classes

Invoked at the initialization

- G4VUserDetectorConstruction
- G4VUserPhysicsList

Global: only one instance of them exists in memory, shared by all threads (**readonly**). Managed only by the master thread.

Action classes

Invoked during the execution loop

- G4VUserActionInitialization
  - G4VUserPrimaryGeneratorAction
  - G4UserRunAction (*)
  - G4UserEventAction
  - G4UserTrackingAction
  - G4UserStackingAction
  - G4UserSteppingAction

Local: an instance of each action class exists **for each thread**.

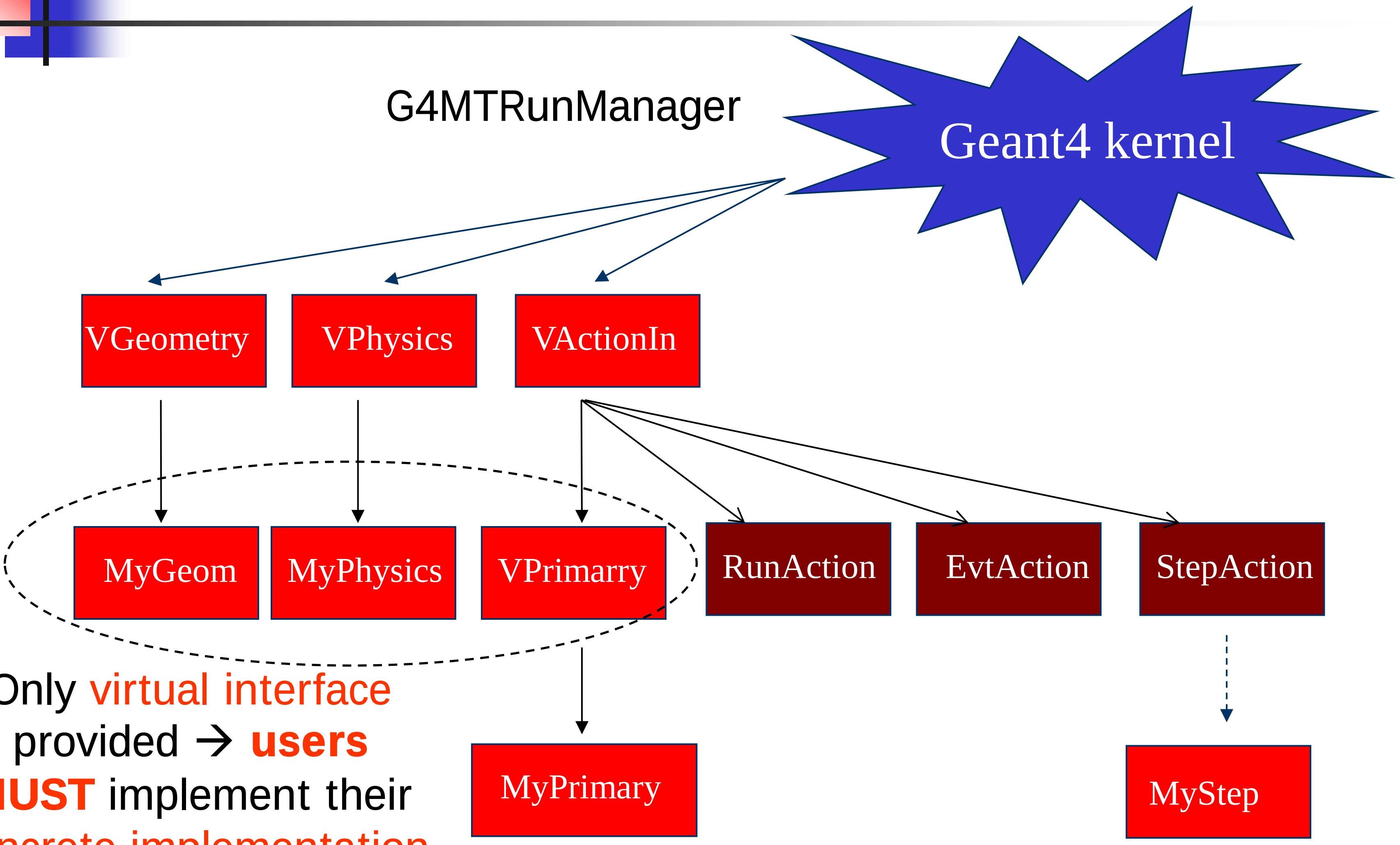
(*) Two RunAction's allowed: one for master and one for threads

User Classes - 2

Mandatory classes in ANY Geant4 User Application

- G4VUserDetectorConstruction
  describe the experimental set-up
- G4VUserPhysicsList
  select the physics you want to activate
- G4VUserActionInitialization
  takes care of the user initializations
  G4VUserPrimaryGeneratorAction

Geant4 concept (MT)



Only virtual interface provided → **users MUST** implement their concrete implementation

The mandatory user classes

The geometry

- User class which describes the geometry must inherit from **`G4VUserDetectorConstruction`** and registered in the Run Manager
- Virtual base class: the purely virtual method must be implemented
  - **`G4VPhysicalVolume* Construct() = 0;`**
    - Must return the pointer to the world volume: all other volumes are contained in it
- Optionally, implement the virtual method
  - **`void ConstructSDandField();`**
    - Defines sensitive volumes and EM fields

Select physics processes

- Geant4 doesn't have any default particles or processes
- Derive <u>your</u> own concrete class from the **`G4VUserPhysicsList`** abstract base class
  - define all necessary particles
  - define all necessary processes and assign them to proper particles
  - define γ/δ production thresholds (in terms of range)
- Pure virtual methods of **`G4VUserPhysicsList`**

ConstructParticles()
ConstructProcesses()
SetCuts()

→ **must** be implemented by the user in his/her concrete derived class

Physics Lists

- Geant4 doesn't have any default particles or processes
- *Partially true*: there is no default, but there are a set of **"ready-for-use" physics lists** released with Geant4, tailored to different use cases. Mix and match:
  - Different sets of hadronic models (depending on the energy scale and modeling of the interactions)
  - Different options for neutron tracking
    - Do we need (CPU-intensive) description of thermal neutrons, neutron capture, etc?
  - Different options for EM physics
    - Do you need (CPU-intensive) precise description at the low-energy scale (< 1 MeV)? E.g. fluorescence, Doppler effects in the Compton scattering, Auger emission, Rayleigh diffusion
    - Only a waste of CPU time for LHC, critical for many low-background experiments

Action Initialization

- New in Geant4 10.0 (supports multi-thread)
- User class must inherit from **`G4VUserActionInitialization`** and registered in the Run Manager
- Implement the purely virtual method
  - **`void Build() = 0;`**
  - Invoked in sequential mode and in MT mode by all workers
  - Must instantiate at least the primary generator
- Optional virtual method
  - **`void BuildForMaster();`**
  - Invoked by the master in MT mode. Applies **only** to Run Action (all other user actions are thread-local)

Primary generator

- User class must inherit from **`G4VUserPrimaryGeneratorAction`**
  - Registered to the Run Manager via the ActionInizialitation (MT mode)
  - Register directly to the RunManager in seq-mode
- Implement the purely virtual method
  - **`void GeneratePrimaries(G4Event*)=0;`**
  - Called by the RunManager during the event loop, to generate the primary vertices/particles
- Uses internally a concrete instance of **`G4VPrimaryGenerator`** (e.g. **`G4ParticleGun`**) to do the job

The optional user classes

Optional user classes - 1

- 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

e.g. actions to be done at the beginning and end of each event

- Each member function of the base classes has a dummy implementation (**not** purely virtual)
  - Empty implementation: does nothing

Optional user classes - 2

- The user may implement the member functions he desires in his/her derived classes
  - E.g. one may want to perform some action at each tracking step
- Objects of user action classes must be **registered** to the Run Manager via the **`G4VActionInizialization`**
  - Notice: in the old-style sequential mode, the user action classes can be registered directly to the Run Manager

Methods of user classes - 1

G4UserRunAction

- **BeginOfRunAction(const G4Run*)** // book histos
- **EndOfRunAction(const G4Run*)** //store histos

G4UserEventAction

- **BeginOfEventAction(const G4Event*)** //initialize event
- **EndOfEventAction (const G4Event*)** // analyze event

G4UserTrackingAction

- **PreUserTrackingAction(const G4Track*)**
  //decide to store/not store a given track
- **PostUserTrackingAction(const G4Track*)**

Methods of user classes - 2

G4UserSteppingAction

- **`UserSteppingAction(const G4Step*)`**

    //kill, suspend, pospone the track, draw the step, ...

G4UserStackingAction

- **`PrepareNewEvent()`** //reset priority control
- **`ClassifyNewTrack(const G4Track*)`**

    // Invoked when a new track is registered (e.g. kill, pospone)

- **`NewStage()`**

    // Invoked when the Urgent stack becomes empty (re-classify, abort event)

MyActionInitialization (MT mode)

- Register thread-local user actions

```
void MyActionInitialization::Build() const
{
  //Set mandatory classes
  SetUserAction(new MyPrimaryGeneratorAction());
  // Set optional user action classes
  SetUserAction(new MyEventAction());
  SetUserAction(newMyRunAction());
}
```

- Register RunAction for the master

```
void MyActionInitialization::BuildForMaster() const
{
  // Set optional user action classes
SetUserAction(newMyMasterRunAction());
}
```

The main() program

The main() program - 1

- Geant4 does **not** provide the **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`**
    - **`G4VUserActionInitialization`** (takes care of Primary)
  - In MT mode, use **`G4MTRunManager`**

The main() program - 2

- The user may define in his/her main()
  - optional user action classes
  - VisManager, (G)UI session
- The user also has to take care of retrieving and saving the relevant information from the simulation (Geant4 will not do that by default)
- Don’t forget to delete the G4RunManager at the end

Sequential vs. MT main()

- The MT vs. sequential mode can be chosen in the main() by picking the appropriate RunManager:
  - **G4RunManager** for sequential
  - **G4MTRunManager** for multi-thread

```
  // Construct the default run manager. Pick the proper run
  // manager depending if the multi-threading option is
  // active or not.
#ifdef G4MULTITHREADED
  G4MTRunManager* runManager = new G4MTRunManager;
#else
  G4RunManager* runManager = new G4RunManager;
#endif
```

An example of (MT) main()

```
{
  …
 // Construct the default run manager
  G4MTRunManager* runManager = new G4MTRunManager;

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

  // Set mandatory user action classes
  runManager->SetUserAction(new MyActionInitialization);
  …
}
```

Optional: select (G)UI

- In your **main()**, taking into account your computer environment, instantiate a **G4UIsession** concrete/derived class provided by Geant4 and invoke its **SessionStart()** method

```
mysession->SessionStart();
```

- It can be used to give commands at run-time (do not require the re-compilation of the application)
  - Select particle/energy, change settings, etc.
- Geant4 provides:
  - G4UIterminal
  - csh or tcsh like character terminal
  - Qt
  - batch job with macro file
  - …

Optional: select visualization

- In your **main()**, taking into account your computer environment, instantiate the **`G4VisExecutive`** and invoke its **`Initialize()`** method
- Geant4 provides interfaces to many graphics drivers:
  - DAWN *(Fukui renderer)*
  - WIRED
  - RayTracer *(ray tracing by Geant4 tracking)*
  - OpenGL
  - OpenInventor
  - Qt
  - VRML
  - X11-compliant

An example of (sequential) main()

```
{
  …
  // Construct the default run manager
  G4RunManager* runManager = new G4RunManager;
  // Set mandatory user initialization classes
  MyDetectorConstruction* detector = new MyDetectorConstruction;
  runManager->SetUserInitialization(detector);
  MyPhysicsList* physicsList = new MyPhysicsList;
  runManager->SetUserInitialization(myPhysicsList);
  // Set mandatory user action classes
  runManager->SetUserAction(new MyPrimaryGeneratorAction);
  // Set optional user action classes
  MyEventAction* eventAction = new MyEventAction();
   runManager->SetUserAction(eventAction);
  MyRunAction* runAction = new MyRunAction();
   runManager->SetUserAction(runAction);
  …
}
```

General recipe for novice users

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

- Design your application... requires some preliminar **thinking** (what is it supposed to do?)
- Create your derived **mandatory** user classes
  - My**DetectorConstruction**
  - My**PhysicsList**
  - My**ActionInitialization** (must register My**PrimaryGenerator**)
- Create optionally your derived user action classes
  - My**UserRunAction**, My**UserEventAction**, ...
- 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!

http://geant4.org

Documentation

- A few manuals available in the Geant4 webpage
  - Application developer manual
  - Physics manual
- Other tools available
  - LXR code repository
  - User forum
  - Bugzilla

User Support

1. Getting started
2. Training courses and materials
3. Source code
   a. Download page
   b. LXR code browser -or- draft doxygen documentation
4. Frequently Asked Questions (FAQ)
5. Bug reports and fixes
6. User requirements tracker
7. User Forum
8. Documentation
   a. Introduction to Geant4
   b. Installation Guide
   c. Application Developers Guide
   d. Toolkit Developers Guide
   e. Physics Reference Manual
9. Examples
10. Physics lists
    a. Electromagnetic
    b. Hadronic
11. User Aids
    a. Tips for improving CPU performance

Examples

- Ready-for-the-use Geant4 applications (examples) are distributed with Geant4
  - Very good starting point for new users
- Three suites of examples:
  - **"basic"**: oriented to novice users and covering the most typical use-cases of a Geant4 application with keeping simplicity and ease of use.
  - **"extended"**: covers many specific use cases for actual detector simulation.
  - **"advanced"**: where real-life complete applications for different simulation studies are provided
- The exercises of this course are based on the basic example **B3**

Examples

- A webpage with doxygen documentation is available for the basic/extended examples

Main Page | Related Pages | Modules | Namespaces | Classes | Files | Search

Basic Examples

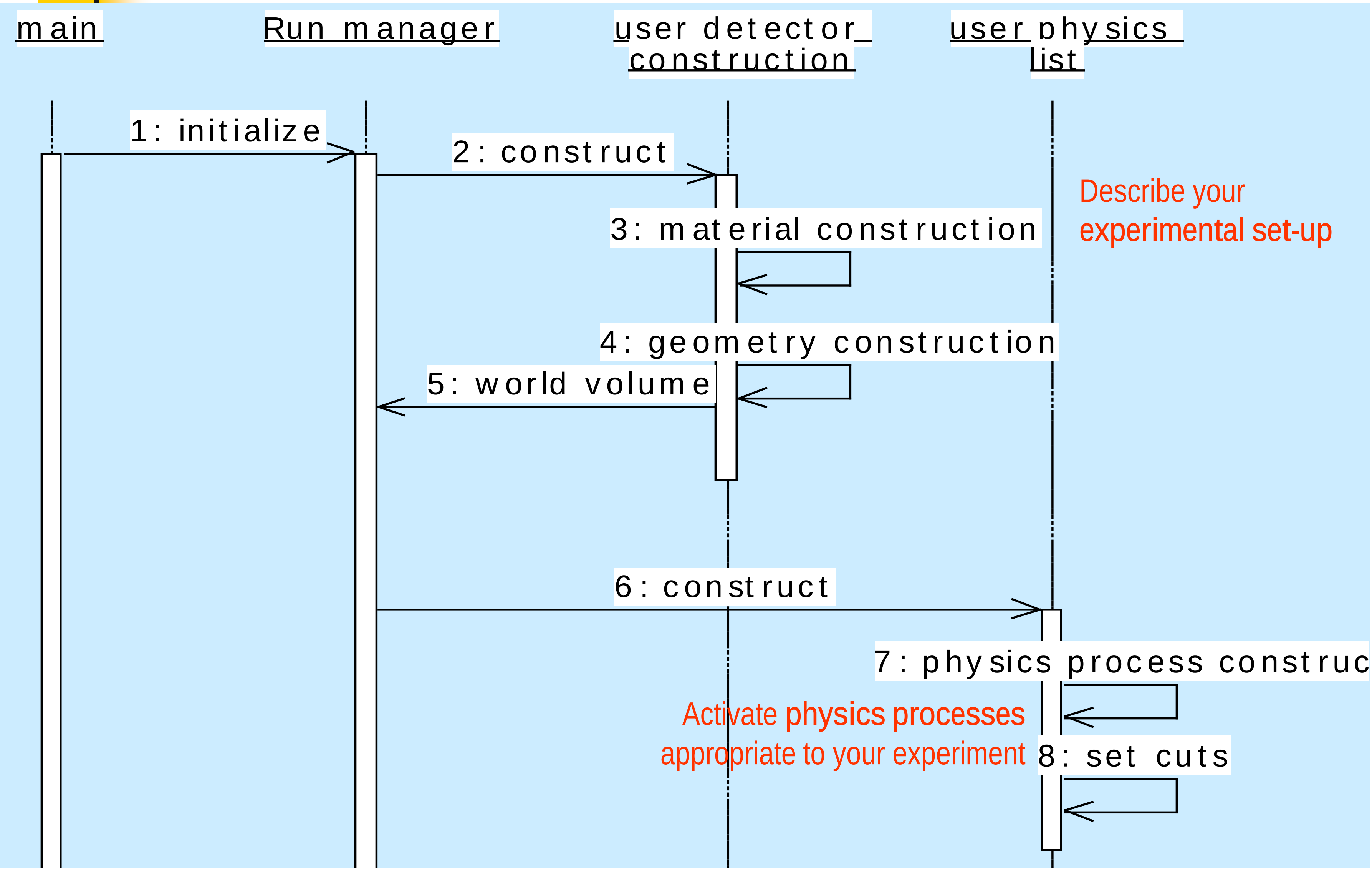
The set of basic examples is oriented to "novice" users and covering many basic general use-cases typical of an "application"-oriented kind of development.

- **Example B1**
  - Simple geometry with a few solids
  - Geometry with simple placements (G4PVPlacement)
  - Scoring total dose in a selected volume user action classes
  - Geant4 physics list (QBBC)
- **Example B2**
  - Simplified tracker geometry with global constant magnetic field
  - Geometry with simple placements (G4PVPlacement) and parameterisation (G4PVParameterisation)
  - Scoring within tracker via G4 sensitive detector and hits
  - Geant4 physics list (FTFP_BERT) with step limiter
  - Started from novice/N02 example
- **Example B3**
  - Schematic Positron Emitted Tomography system

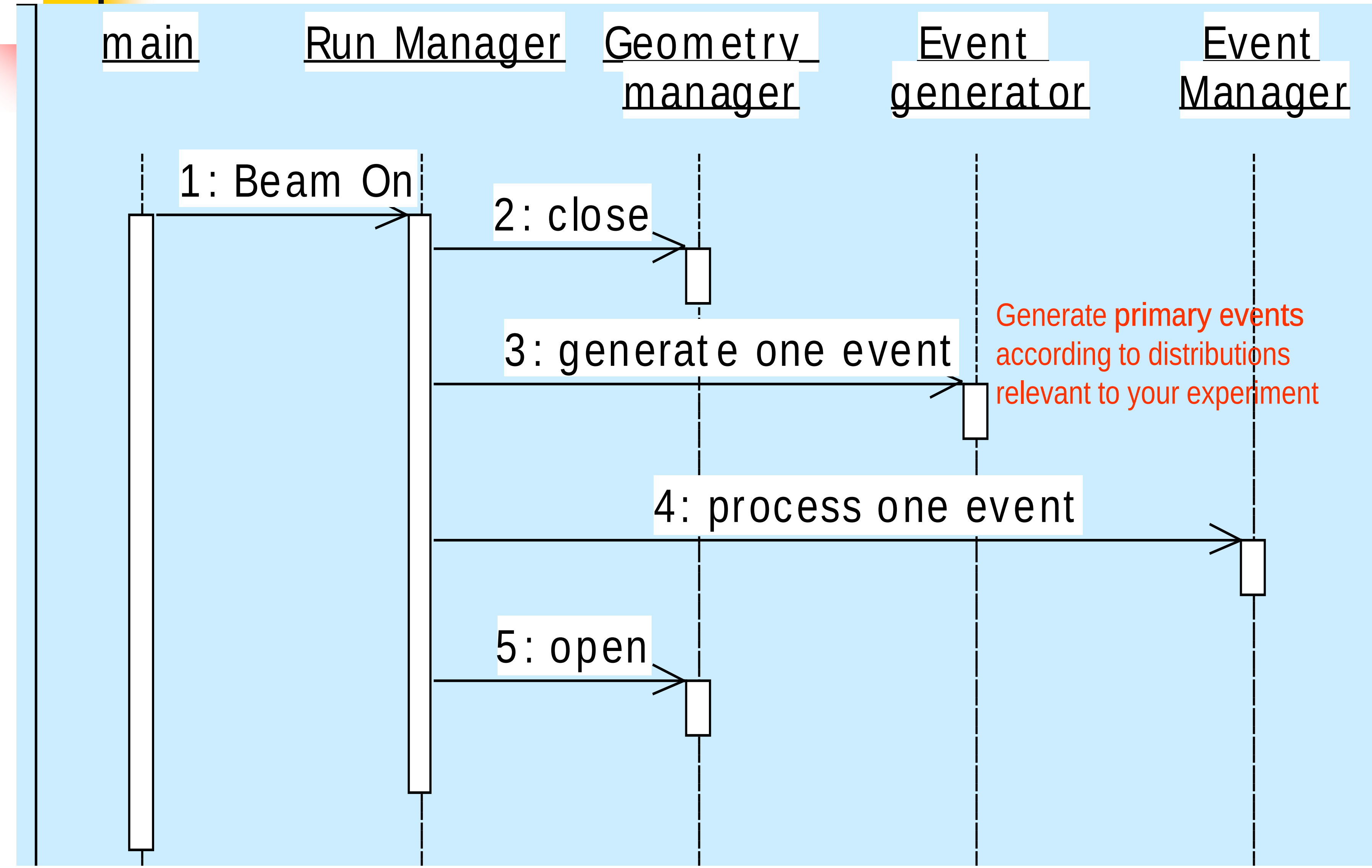
http://cern.ch/geant4/UserDocumentation/Doxygen/examples_doc/html

Backup

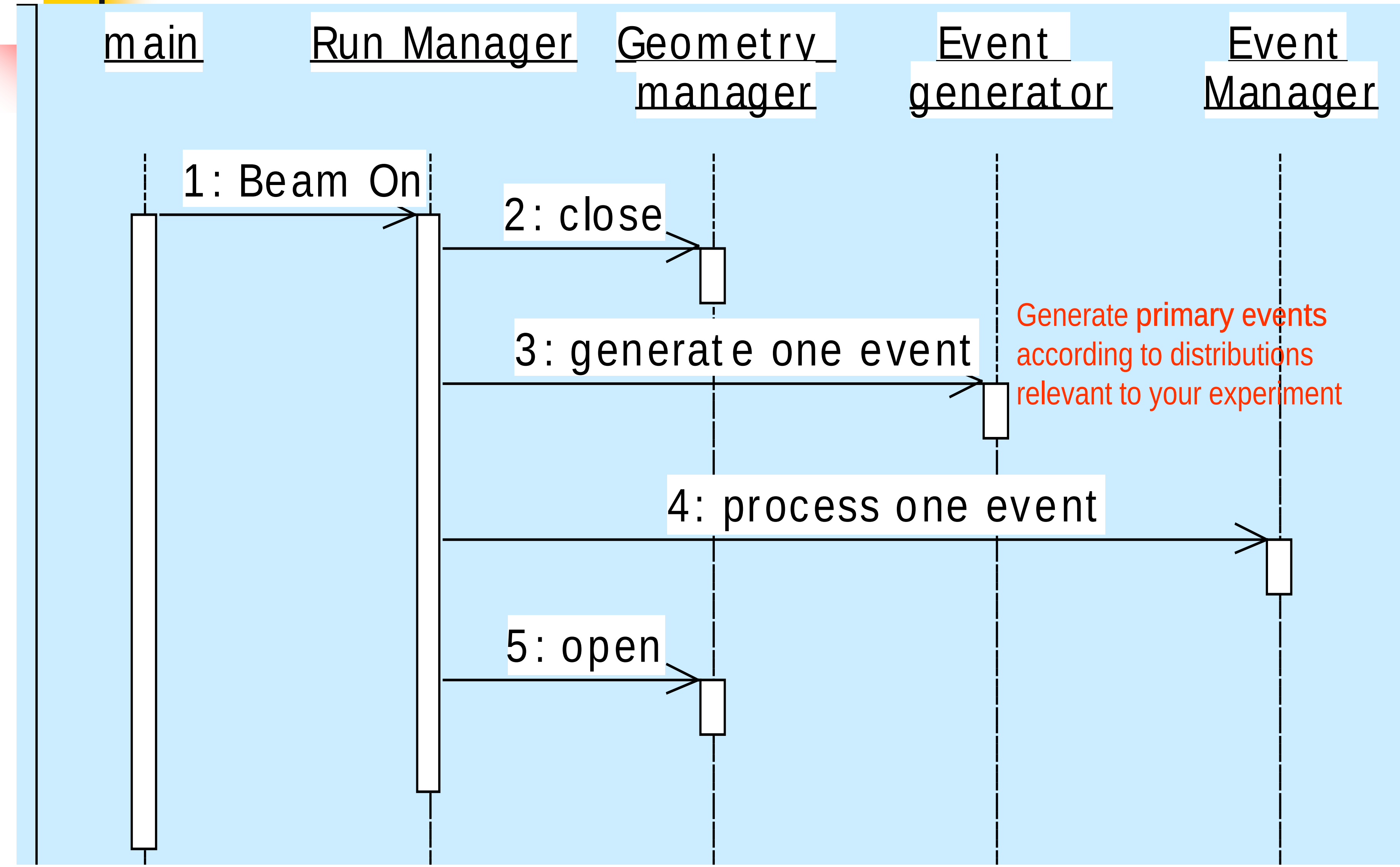
Initialization



Beam On


main
Run Manager
Geometry manager
Event generator
Event Manager
1: Beam On
2: close
3: generate one event
4: process one event
5: open
Generate primary events according to distributions relevant to your experiment

Event loop


main
Run Manager
Geometry_ manager
Event generator
Event Manager
1: Beam On
2: close
3: generate one event
4: process one event
5: open
Generate primary events according to distributions relevant to your experiment

User Classes (<= 9.6)

Initialisation classes

Invoked at the initialization

- G4VUserDetectorConstruction
- G4VUserPhysicsList

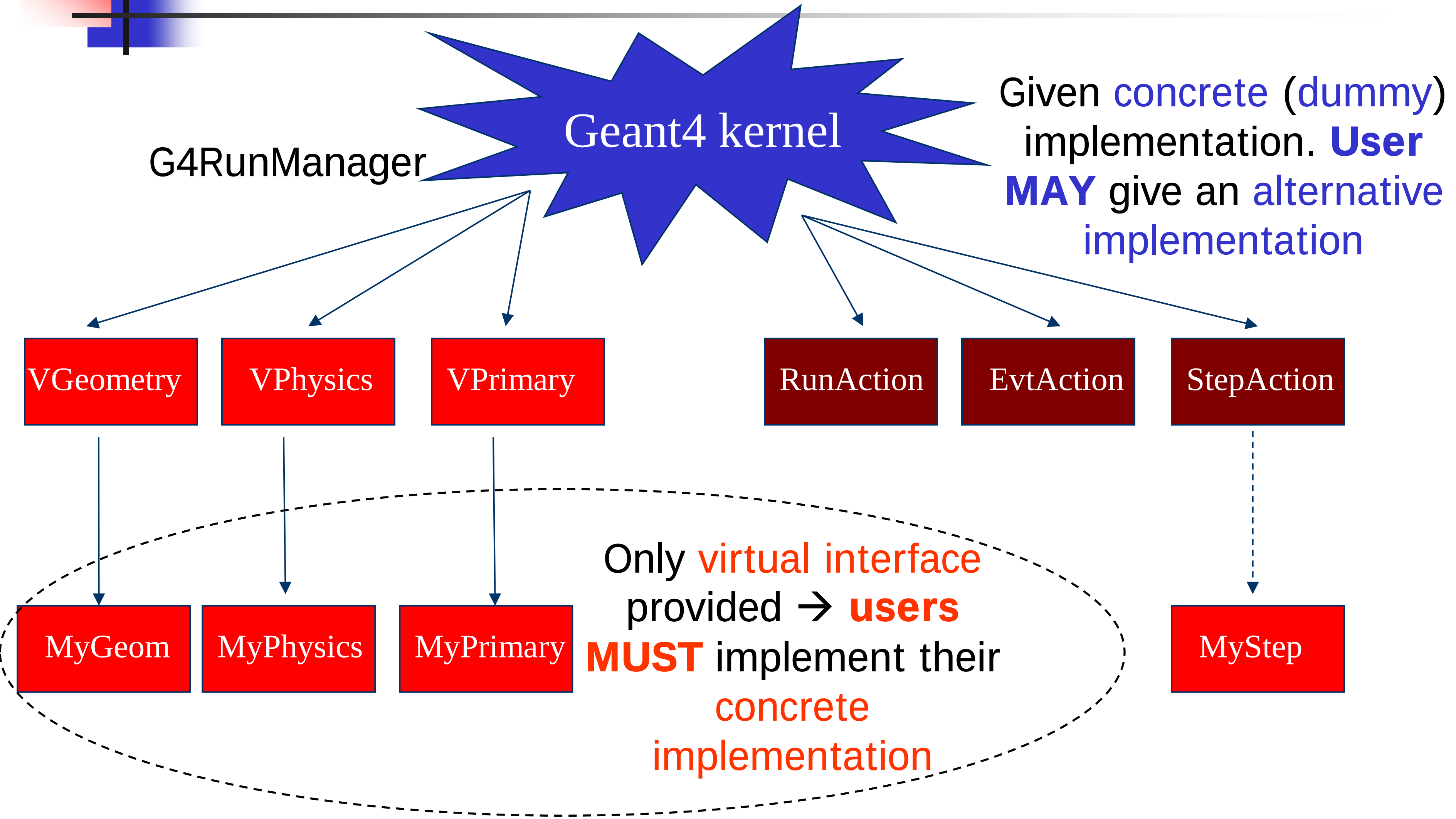
Classes having name starting with G4V are abstract classes (containing purely virtual methods)

Action classes

Invoked during the execution loop

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

Geant4 concept



Geant4 concept