GAMOS

a user-friendly and flexible framework for GEANT4 medical applications

Pedro Arce Dubois
Pedro Rato Mendes
CIEMAT, Madrid

11th GEANT4 Workshop
LIP – Lisbon, 10-14 October 2006

Index

- Introduction
  - GAMOS objectives
  - plug-in's
- GAMOS components
  - Geometry
  - Generator
  - Physics
  - User actions
  - Sensitive detector and hits
  - Histograms
  - Visualization
- Utilities
  - Parameter management
  - Verbosity management
  - Input file management
- Examples
  - PET
  - Histograms
- Summary

GAMOS

(Geant4-based Architecture for Medicine-Oriented Simulations)

GAMOS is a framework designed to allow the user to

- ✓ Simulate a project with a **minimal knowledge of GEANT4** and **no need of C++**
- ✓ **Easily add new functionality** and **combine it with the existing functionality** in GAMOS
  - ➢ We cannot pretend to cover all the functionality, so we should let the advanced user to write C++ code
    - ➢ Load it dynamically
    - ➢ Easily transform it into a user command

⇨ **plug-in's**

⟹ It must be **complete, flexible, easy to extend** and **easy to use**

COMPLETE:

- Provide **all the functionality** for someone who wants to simulate a **medical physics project**
  - It is indeed impossible to cover all what all users may need
    - ⇒ It must be extendible
  - It will keep growing with time…

FLEXIBLE:

- Users should be able to control everything through **user commands (= no recompiling)**
- **Avoid hardcoding**
  - Do not force users to call its detector "CRYSTAL", or to have three levels of ancestors, or…
  - Do not force users to use your SD class, or your histogram format, or…
- Different modules can be combined at users will
  - Change geometry but not the histograms
  - Change sensitive detector type but do not toch digitization
- ***MODULAR:*** Each class, each module makes **one and only one thing**, clearly defined, but **as general as possible**
  - Keeping an eye on performance

EASY TO EXTEND:

- **Easy to add** any new **functionality**
- Mix seamlessy **existing functionality** together **with new one**
  - Add new modules **without affecting others**

- Based on “**plug-in’s**”
  - Convert **new C++** into **user commands**

GAMOS

(Geant4-based Architecture for Medicine-Oriented Simulations)

EASY TO USE:

- Almost everything can be done through **user commands**
- A **good design**, applying **software engineering techniques**
- **Well documented**

GAMOS plug-in´s

- **The main GAMOS program has no predefined components**
  - **At run-time user selects which components to load through user commands**

- User has **full freedom** in choosing components
- User can define a **component not foreseen** by GAMOS
  - **Write C++** and use it through an **user command**
  - Mix it with any other component

For the plug-in's implementation in GAMOS it has been chosen the CERN library: SEAL

Geometry

Three different ways to define:

C++ code:

- The usual GEANT4 way
- Add one line to **transform your class in a plug-in**

DEFINE_GAMOSGEOMETRY (MyGeometry);

so that you can select it in your input macro

/gamos/geometry MyGeometry

Define it in ASCII files

- The easiest way to define a geometry
- Based on simple tags
- Same order of parameters as corresponding GEANT4 classes

Using one of the GAMOS examples

- Simple PET can be defined through an 8-parameters file (n_crystals, crystal_x/y/z, radius, …)
- ...

Geometry from ASCII files

- Based on simple tags, with same order of parameters as corresponding GEANT4 classes

  :ELEM Hydrogen H 1. 1.

  :VOLU yoke :TUBS Iron 3 62.*cm 820. 1.27*m

  :PLACE yoke 1 expHall R0 0.0 0.0 370.*cm

MATERIALS:
- Isotopes
- Elements
- Simple materials
- Material mixtures by weight, volume or number of atoms

SOLIDS:
- All "CSG" and "specific" solids
- Boolean solids

ROTATION MATRICES:
- 3 rotation angles around X,Y,Z
- 6 theta and phi angles of X,Y,Z axis
- 9 matrix values

Geometry from ASCII files

PLACEMENTS:

- Simple placements
- Divisions
- Replicas
- Parameterisations
  - Linear or circular
  - For complicated parameterisations example of how to mix the C++ parameterisation with the ASCII geometry file

- Colour
- Visualisation ON/OFF

PARAMETERS:

- Can be defined to use them later

:P InnerR 12.

:VOLU yoke :TUBS Iron   3 **$InnerR** 820. 1270.

- Arithmetic expressions

:VOLU yoke :TUBS Iron   3 sin($ANGX)*2+4 820. 1270.

Geometry from ASCII files

UNITS:
- Default units for each value
- Each value can be overridden by user

- Include other files

#include mygeom2.txt.

- User can extend it: add new tags and process it without touching base code

- Install and use it as another GEANT4 library

```
G4VPhysicalVolume* MyDetectorConstruction::Construct(){
  G4tgbVolumeMgr* volmgr = G4tgbVolumeMgr::GetInstance();
  volmgr->AddTextFile(filename);      // SEVERAL FILES CAN BE ADDED
  return  = volmgr->ReadAndConstructDetector();
```

- GEANT4 in memory geometry -> ASCII files

HISTORY:
- In use to build GEANT4 geometries since 9 years ago
  - An evolving code…
- Built CMS and HARP experiments

Some geometry utilities

Utilities that can be used through a command or from any part of the user code

- Material factory
  - GAMOS reads material list from a text file
  - A G4Material can be built at user request

```
G4Material* bgo = GmMaterialMgr::GetInstance()
    ->GetG4Material("BGO");
```

- Printing list of
  - Materials
  - Sólids
  - Logical volumes
  - Physical volumes
  - **Touchables**
- Find a volume by name (LV, PV or touchable)
- Delete a volume (and daugthers) by name

Generator

C++ code

- The usual GEANT4 way
- Add one line to **transform your class in a plug-in**

DEFINE_GAMOSGENERATOR(MyGenerator);

so that you can select it in your input macro

/gamos/generator MyGenerator

GAMOS generator

- Combine any number of **single particles** or **isotopes decaying to e^+, e^-, γ**
- **For each particle or isotope** user may select by user commands a combination of **time, energy, position and direction** distributions
  - ✓ Or create its own and select it by a user command (transforming it into a plug-in)

Physics

C++ code

- The usual GEANT4 way
- Add one line to **transform your class in a plug-in**

DEFINE_GAMOSPHYSICSLIST (MyPhysicsList);

so that you can select it in your input macro

/gamos/physicsList MyPhysicsList

- **GAMOS physics list**
  - **Based on hadrotherapy advanced example**
    - **User can combine different physics lists for photons, electrons, positrons, muons, protons and ions**
  - **Dummy one for visualisation**

User actions

✓ User can have as many user actions of any type as he/she wants

✓ User can activate a user action by a user command

- GAMOS user actions or her/his own
- Just adding a line after the user action to transform it into a plug-in

```
DEFINE_GAMOSUSERACTION(MyUserAction);
/gasos/userAction MyUserAction
```

Sensitive Detectors

- To produce hits in GEANT4 a user has to:
  - Define a class inheriting from G4VSensitiveDetector
  - Associate it to a G4LogicalVolume
  - Create hits in the ProcessHits method
  - Clean the hits at EndOfEvent

- **In GAMOS you can do all this with a user command**

```
/gamos/assocSD2LogVol SD_CLASS SD_TYPE LOGVOL_NAME
```

  - SD_CLASS: Two classes of SD currently in GAMOS
    - Simple: each volume corresponds to an SD ⇒ a hit
    - VirtuallySegmented: a volume is segmented and each subvolume builds a different hit
  - SD_TYPE: an identifier string, so that different SD/hits can have different treatment

- User can create his/her own SD class

Hits

- **A GAMOS hit has the following information**
  - `G4int theDetUnitID;` ID of the sensitive volume copy
  - `G4int theEventID;`
  - `G4double theEnergy;`
  - `G4double theTimeMin;` time of the first E deposit
  - `G4double theTimeMax;` time of the last E deposit
  - `G4ThreeVector thePosition;`
  - `std::set$<$G4int$>$ theTrackIDs;` list of all tracks that contributed
  - `std::set$<$G4int$>$ thePrimaryTrackIDs;` list of all ‘primary´ tracks that contributed
  - `std::vector$<$GamosEDepo*$>$ theEDepos;` list of all deposited energies
  - `G4String theSDType;`

- User can create his/her own hit class

Digitizer

Digitization is very detector specific ⇨ it is not possible to provide a general solution

- GAMOS just provide a simple digitizer
  - 1 hit ⇨ 1 digit
  - Merge hits close enough
    - **Same set of sensitive volumes**
    - **Closer than a given distance**
- … and a basic structure
  - Hits compatible in time (spanning various events)
  - Trigger
  - Pulse simulation
  - Sampling
  - Noise

Some detector effects

Measuring time

- A detector is not able to separate signals from different evetns if they come close in time

Dead time

- When a detector is triggered, this detector (or even the whole group it belongs to) is not able to take data during some time

- Both can be set by the user in the input macro
  - A different time for each SD_TYPE

```
/gamos/setParam SD:Hits:MeasuringTime:Calor 10. ns
```

Histograms

Same code to create and fill histograms independent of the format

- GAMOS takes care of writing the file in the chosen format at the end of job

- Originally based on CERN package PI
  - ☹ But PI is not supported any more
  - Currently own format, output in ROOT

GmAnalysisMgr keeps a list of **histograms** so that they **can be accessed from any part of the code, by number or name**

```
GmHitsEventMgr::GetInstance("pet")->GetHisto1(1234)->Fill(ener);
GmHitsEventMgr::GetInstance("pet")->GetHisto1("CalorSD: hits energy")->Fill(ener);
```

There can be several files, each one with its own histograms

- **When creating an histogram, user chooses file name**

Parameter management

<u>**GmParameterMgr**</u> helps the user to define and use a parameter

- A parameter is defined in the **input macro**

```
/gamos/setParam SD:Hits:EnergyResolution 0.1
```

- User can **get** its value in **any part of the code**

```
float enerResol = GmParameterMgr::GetInstance()
->GetNumericValue("SD:Hits:EnergyResolution",0.);
```

Parameters can be number or strings

Verbosity management

- User can control the verbosity of the different GAMOS components independently

```
/gamos/verbosity GamosGenerVerb 3
/gamos/verbosity GamosSDVerb 2
```

- ✓ Can be used in new code trivially

```
G4cout << AnaVerb(3) << “creating my histograms” << G4endl;
```

- ✓ User can easily define its own verbosity type controlled by a user command

- 5 + 1 levels of verbosity
  - SilentVerb = -1
  - ErrorVerb = 0 (default)
  - WarningVerb = 1
  - InfoVerb = 2
  - DebugVerb = 3
  - TestVerb = 4

Verbosity management (II)

TrackingVerbose by event and track number:

- It can be selected for which events and track numbers the "/tracking/verbose" command becomes active

```
/gamos/userAction TrackingVerboseUA
/gamos/setParam TrackingVerbose:EventMin 1000
/gamos/setParam TrackingVerbose:EventMax 1010
/gamos/setParam TrackingVerbose:TrackMin 10
/gamos/setParam TrackingVerbose:TrackMax 20
```

Event counting:

- Prints the number of simulated events with the number of tracks in the last event and accumulated (useful when you are waiting for long times without nothing happening…)

```
/gamos/userAction TrackCountUA
/gamos/setParam TrackCount:EachNEvent 1000
```

Input file management

Some algorithms need to read in a data file

In GAMOS **the file does not have to be on the current directory**

- Easier to use the same file in several applications

❖ **GAMOS_SEARCH_PATH** variable contains the list of directories where the file is looked for

- User can add more directories

Applications and examples

<u>Medical physics applications</u>:

- ❑ PET
- ❑ Radiotherapy on progress

<u>Histogram examples</u>:

- As general as possible so that they can be reused

<u>Documentation examples</u>:

- A dummy one and a more complicated one

```
/gamos/geometry GmGeomtryFromText
/gamos/physicsList GmEMLowEnPhysics
/gamos/generator GmGenerator
/gamos/generator/addIsotopeSource F18_1 F18 1.E3 becquerel
/run/initialize
/run/beamOn 10
```

- Explained in detail

Summary

- **GAMOS is a plug-in based, and user-friendly GEANT4-based framework**
  - ✓ allows the user to do GEANT4 simulation through **user commands**
  - ✓ **plug-in's** allow to **extend functionality** by writing **C++** classes that can then be used through **user commands**

- We have tried in its design to make a framework
  - ✓ **Easy to use, flexible, extendible and complete**

- **GAMOS core is application independent**
  - ✓ Several **medical applications** are being built on top of GAMOS core