GAMOS

(GEANT4-based Architecture for Medical-Oriented Simulations)

Status and plans

Pedro Arce Dubois
Pedro Rato Mendes
CIEMAT, Madrid

12th GEANT4 Workshop
Hebden Bridge (UK), 13-19 September 2007

Index

- Introduction
  - GAMOS objectives
  - plug-in's
- GAMOS components
  - Geometry
  - Movements
  - Generator
  - Physics
  - User actions
  - Sensitive detector and hits
  - Reconstructed hits
  - Scoring
  - Histograms
  - Visualization
- Utilities
  - Parameter management
  - Verbosity management
  - Input file management
- Examples
  - PET
  - CONRAD (dosimetry)
  - Radiotherapy
- Use of GAMOS
- Summary

Objectives

Easy to use:

- ✓ Simulate a medical project with a **minimal knowledge of GEANT4** and **no need of C++**
  - ✓ Base it on a **scripting langauge**

Complete:

- ✓ Provide **all the functionality** for someone who wants to simulate a **medical physics project**
  - ✓ At least as complete as possible…

Objectives

Flexible:

- ✓ **Users always need to do something or to extract some info in a way you have not foreseen**
  - ❖ **Main users are research groups**
- ✓ **GEANT4 has a lot of possibilities, ever increasing**
  - ❖ **If your framework is too rigid, you will end hiding some of these possibilities to the users**
- ➢ **C++ cannot be avoided if you want to be flexible, but you have to make it easy:**
  - ✓ **Base it on the plug-in mechanism**
  - ✓ **Provide templates and easy instructions for each plug-in type**

GAMOS plug-in´s

- **The main GAMOS program has no predefined components**
  - **At run-time user selects which components to load through user commands**
  - **They are selected with a user command and loaded through the plug-in mechanism**

- 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 (only two libraries and 'make' compiling)

Geometry

C++ code:

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

```
DEFINE_GAMOS_GEOMETRY (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

Using one of the GAMOS examples

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

Reading DICOM files

- DCMTK toolkit → ASCII (EGS format) → GAMOS
- Uses G4RegularNavigation for fast navigation (see Thursday talk)

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, circular or square
  - For complicated parameterisations example of how to mix the C++ parameterisation with the ASCII geometry file

COLOURS

VISUALISATION ON/OFF

Geometry from ASCII files

PARAMETERS:

- Can be defined to use them later

:P InnerR 12.

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

ARITHMETIC EXPRESSIONS:

- Elaborated expressions can be used

:VOLU yoke :TUBS Iron 3 sin($ANGX)*2+4*cos(exp(1.)) 820. 1270.

UNITS:

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

:P ConePhi 1.*rad

Geometry from ASCII files

- Include other files

#include mygeom2.txt.

- User can extend it: add new tags and process it without touching base code
- Can mix a C++ geometry with an ASCII geometry
- GEANT4 in memory geometry -> ASCII files
- 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();
```

HISTORY:

- In use to build GEANT4 geometries since 10 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

Movements

You can move a volume by using commands

- Every N events or every interval of time
- N times or forever
- You can define an offset
- Displace or rotate it
- You can describe **any kind of movement** by using an ASCII file where you define the position and rotation of the volume at each step

- Several movements of different or the same volume can be set

Generator

C++ code

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

DEFINE_GAMOS_GENERATOR(MyGenerator);

so that you can select it in your input macro

/gamos/generator MyGenerator

GAMOS generator

- Combine any number of **single particles** or **decaying isotopes**
- **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)

    DEFINE_GAMOS_DIST_POSITION(MyPosOnPiramidWall);

    /gamos/generator/positionDist MYSOURCE1 MyPosOnPiramidWall 1.*cm
  - ✓ Also your class may derive from several of these (for introducing correlations)

Physics

C++ code

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

```
DEFINE_GAMOS_PHYSICS_LIST (MyPhysicsList);
```

so that you can select it in your input macro

```
/gamos/physicsList MyPhysicsList
```

- Any official GEANT4 physics list can be easily used through a user command

GAMOS physics list

- **Based on GEANT4 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_GAMOS_USER_ACTION(MyUserAction);

    so that you can select it in your input macro

    /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: the G4VSensitiveDetector class
  - SD_TYPE: an identifier string, so that different SD/hits can have different treatment

- You can create your own SD class and make it a plugin

DEFINE_GAMOS_SENSDET(MySD);

so that you can select it on your input macro

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

Reconstructed hits

The digital signal build by a digitizer can be converted to reconstructed hits (position, time, energy) through a RecHitBuilder (a plugin)

You can also use it acting directly on hits

e.g. for merging hits

Detector effects

Measuring time

- A detector is not able to separate signals from different events 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
```

Scoring

GAMOS scoring is based on GEANT4 scoring

- You can use it through user commands

- GEANT4 primitive scorers are plug-in´s
  - ✓ You can easily add your own one

- GEANT4 scorer filters are plug-in´s
  - ✓ You can easily add your own one

- A third type of plug-in's: GmScorerFormatter
  - ❑ To define the format of your scoring (by energy bins, by volume ID, ...)
  - ❑ To define your output format

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** or **CSV** (Comma Separated Value text files)

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

GmParameterMgr 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, list of numbers or list of 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
```

Also your own verbosity can be switched on/off by event

Fundamental for debugging strange events!

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: PET

- A full PET application is available
  - Module for building a simple pet wit a few parameters
  - Several control histograms
  - Event classification

- CTI ECAT Exact (922) & GE Discovery ST (PET/CT) detectors have been simulated
  - ✓ Results agree with data

- ClearPET detector is being simulated with GAMOS

- BrainPET research project is being simulated with GAMOS

Applications: CONRAD WP4

CONRAD (A Coordinated Network for Radiation Dosimetry) is a EU sponsored project to coordinate research into measurements and calculations for radiation protection at workplaces

http://www.eurados.org/conrad/conrad_overview.htm

- **Two problems of Work Package 4 (computational dosimetry) have been simulated with GAMOS**
  - ✓ **P.3: SIGMA simulated workplace neutron field**
  - ✓ **Intercomparison on Monte Carlo modelling for in vivo measurements of Americium in a knee phantom**

✓ **Results will be presented orally at the October 2007 CONRAD WP4 Workshop**

Applications: Radiotherapy

MIRAS (Medical Imaging Radiotherapy and Simulation)

Project aimed to facilitate all kind of Radiotherapy studies to a non-expert user

- ✓ GUI for image manipulation
- ✓ Simple script langauge to define input and output
- ✓ Tools to send jobs in PBS or grid
- ✓ User can choose to use BEAM(EGS) or GAMOS(GEANT4)
- ✓ DICOM reader → EGS text format → GAMOS
  - ➢ Based on DCMTK tooolkit (any DICOM can be read)
- ➢ BEAM (EGS) to GAMOS translator
  - ✓ Translates BEAM geometry, generator and all input parameters to GAMOS
  - ❖ On progress…

Hadrotherapy example being built

Use of GAMOS

- A set of scripts to download it and install it automatically
  - Included compilation of GEANT4, CLHEP, SEAL & ROOT
- Tested on Scientific Linux 3 & 4, Fedora Core 4, 5, 6 & 7 and Debian

- Base GEANT4 software at the medical physics and electronics departments of CIEMAT-Madrid
- GAMOS course at La Habana, Cuba, July 2006
- GAMOS course at Lima, Perou, October 2006
- Beta-testers in Brussels, Belgium and Tennessee, USA (Townsend´s group) universities
- Course at Madrid for Spain medical simulation users community in November 2007

Summary

➢ **GAMOS is a user-friendly and flexible GEANT4-based framework for medical simulations**

- ✓ allows the user to do GEANT4 simulation through **user commands**
- ✓ **plug-in's** allow to **easily extend its functionality** by writing **C++** classes that can then be used through **user commands**

❑ Software engineering techniques have been applied with the aim of building a framework

- ✓ **Easy to use, complete and flexible**

➢ **GAMOS core is application independent**

- ✓ Several **medical applications** are being built on top of GAMOS core (PET, Dosimetry, Radiotherapy, Hadrotherapy)