

# ***GAMOS***

***an easy and flexible framework for  
GEANT4 simulations***

***Pedro Arce  
Juan Ignacio Lagares  
Daniel Pérez Astudillo  
Pedro Rato Mendes  
Mario Cañadas  
CIEMAT, Madrid***

***Geant4 Users workshop  
Catania, 15-17 October 2009***

# Outline

- *GAMOS objectives*
  - Easy
  - Flexible
  - Debugging capabilities
  - Optimised
- *GAMOS functionality*
  - Geometry
  - Movements
  - Generator
  - Physics
  - Sensitive detector and hits
  - User actions
  - Scoring
  - Histograms
  - Parameter management
- *Managing output*
  - Obtaining data
  - Filters
  - Classifiers
  - Example
- *Applications*
  - PET
  - Radiotherapy
    - Accelerator
    - Dose in phantom
    - Analysis of results
    - Optimisation
- *Installation*
- *Documentation*

# *GAMOS*

## *objectives*

# ***GAMOS objectives***

## **1. Easy**

- Easy to install
- Provide the functionality required by the user with simple user commands (Geant4-like commands)
  - ❑ No C++
- Detailed documentation
- Step-by-step tutorials

## 2. Flexible

Framework developers will do their best to provide all the required functionality, but most user are researchers:

- ❖ They want to try new things, that the framework authors have not thought about

For example:

- Use a detector they have already built with C++...
- Use a peculiar position distribution for the primary generator...
- Make an histogram of a new variable under certain conditions...
- ..
- Something you have never imagined...

Best solution for biggest flexibility: plug-in's

- Any code written for Geant4 can be easily converted into a plug-in
  - ✓ Conversion just need one line of code (almost always)
  - ⇒ selectable with a user command
  - ⇒ mix it seamlessly with existing framework code

# GAMOS objectives

**For example:** instead of GAMOS generator use the one from Geant4 example  
underground\_physics

Add one line::

**DEFINE\_GAMOS\_GENERATOR(DMXPrimaryGeneratorAction);**

and then you can select it in your script, instead of the GAMOS primary generator  
**/gamos/generator DMXPrimaryGeneratorAction**

- If the functionality required does not exist, it is enough to write a class providing it
  - ☺ No need to know how to the framework instantiates it
  - ☺ No need to modify the framework

**GAMOS provides a lot of functionality, but has no predefined way to do things**

**At run-time user selects which components to load with *user commands*:**

**GAMOS ones, Geant4 ones or own ones**

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



## **3. Debugging capabilities**

- Most users are researchers:

- They always want to have a deep understanding of what happens in the simulation

For example:

- Length travelled by electrons produced by compton interactions in a crystal...
  - Angle of bremsstrahlung photons when original electron  $> 1$  MeV...
  - Time spent in each volume...
  - ...
  - Something you have never imagined...

## ⇒ Not only easy and flexible input, also **easy and flexible output**

- Provide a flexible framework to get the desired data under the user required conditions
  - Tables, histograms and text/binary files with many variables, selectable by user commands + filters & classifiers

## **4. Optimised**

- Easy way to set cuts and user limits
- Automatic optimisation of production cuts and user limits
- Studies of best electromagnetic parameters for each application
- Provide optimisation tools specific for each application domain



# *GAMOS*

## *functionality*

# Geometry

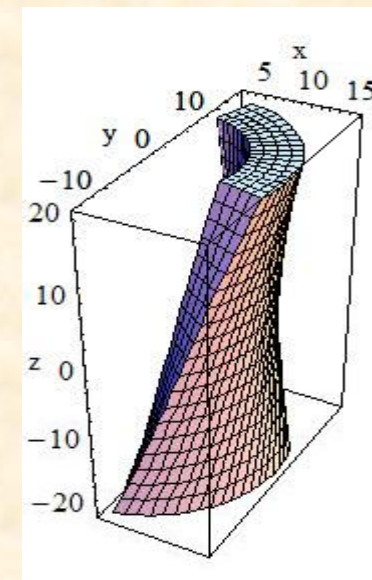
## Define it in ASCII (text) files ([geant4/examples/extended/persistency/P03](#))

- The easiest way to define a geometry, based on simple tags
- Added tags for region and cuts definition
- Added tags for simplifying RT accelerator components
- Added management of parallel worlds (to be included in Geant4)



## Using one of the GAMOS examples for simple cases

- Simple PET can be defined through a file with a few parameters
- Simple phantom with a few user commands



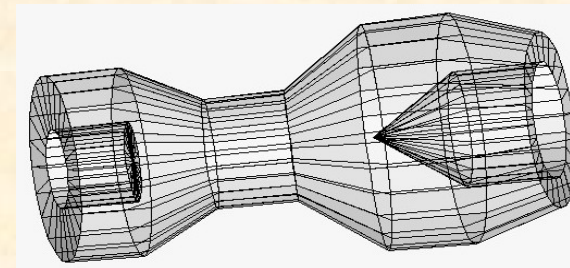
## Or use your own C++ code:

- ❖ Any class inheriting from `G4VUserDetectorConstruction`
- ❖ 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`

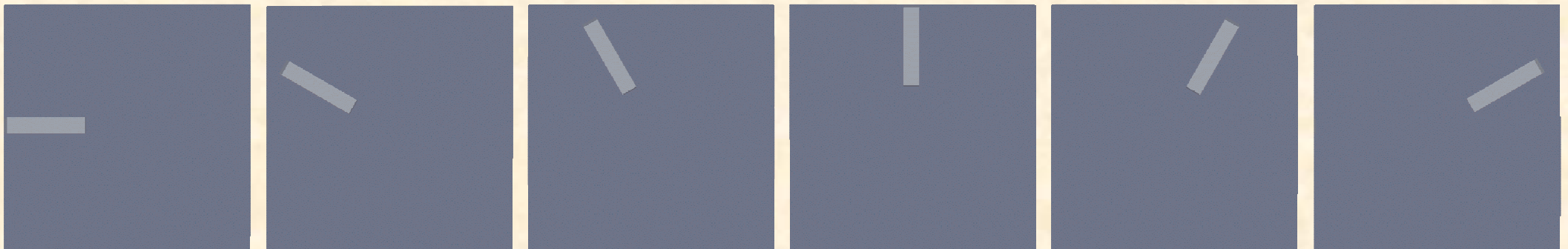


# Movements

## User can move any volume with user commands

- Several movements of the same volume or different ones can be set at the same time
- Displacements, rotations or text file that allows to define any movement
- Every N events or every interval of time
- N times or forever
- Offset can be defined

### Sinusoidal movement simulated with GAMOS



# Generator

## GAMOS generator

- Combine any number of **single particles** or **isotopes**
- **For each particle or isotope** user may select by user commands any combination of **time, energy, position and direction** distributions
- **Geant4 radioactive decay** can be selected with user commands
  - Alternatively user can define the branching ratios and use energy from LNBL
- **Common medical physics distributions** are available
  - Specially useful: position inside one or several G4Volumes or user-defined volumes and position in one or several G4Volumes or user-defined volume surfaces
- ✓ User can create its own distribution and select it with a user command (plug-in)

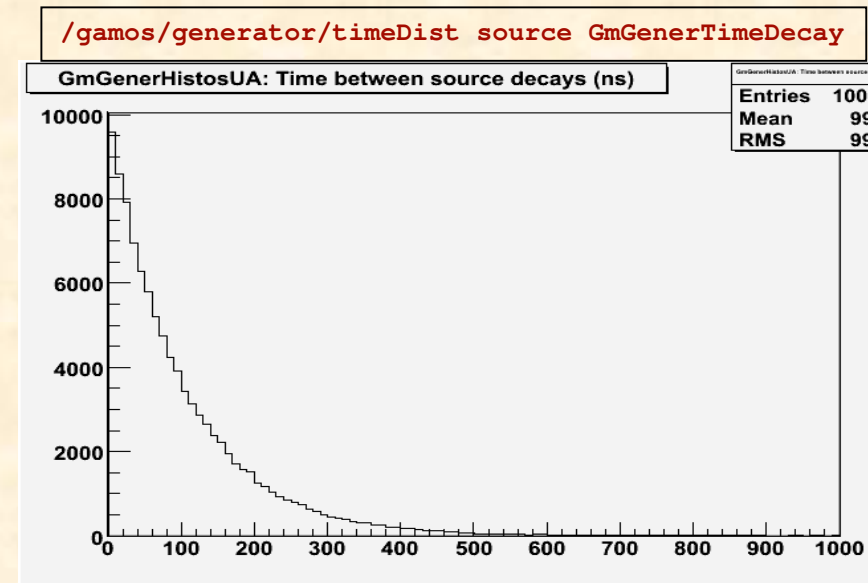
## Or use your own C++ code

- ❖ Any class inheriting from G4VUserPrimaryGeneratorAction
- ❖ 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 physics list

- Based on hadrontherapy advanced example
  - User can combine different physics lists for photons, electrons, positrons, muons, protons and ions
- A dummy one for visualisation

## Or use your own C++ code

- ❖ Any class inheriting from *G4VUserPhysicsList*
- ❖ Add one line to **transform your class in a plug-in**

`DEFINE_GAMOS_PHYSICS(ExN02PhysicsList);`

so that you can select it in your input macro

`/gamos/physicsList ExN02PhysicsList`

## **CUTS:**

- Production cuts can be set per region with user commands
- User limits can be set per volume with user commands

# Sensitive Detectors

- ❖ To produce hits in GEANT4 a user has to (with C++):
  - 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

## Or use your own C++ code

- ❖ Any class inheriting from *G4VSensitiveDetector*
- ❖ Add one line to **transform your class in a plug-in**

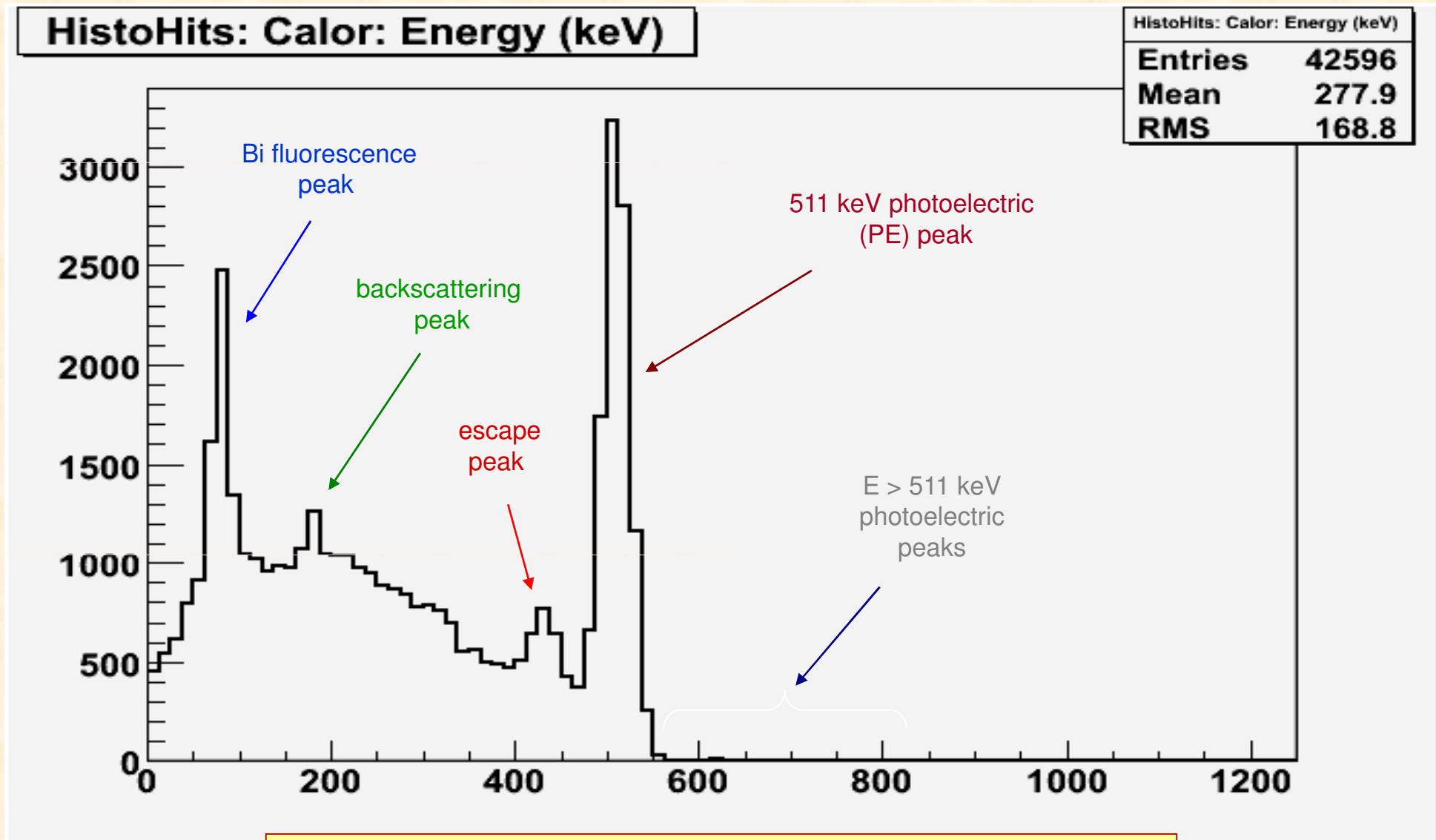
```
DEFINE_GAMOS_SENSDET(MySD);
```

so that you can select it in your input macro

```
/gamos/assocSD2LogVol Calor MySD crystal
```



/gamos/userAction GmHitsHistosUA



**Hits energy spectrum in BGO crystals**

# User actions

They are the main way for a user to extract information of what is happening and modify the running conditions: **a flexible framework is needed:**

- **As many user actions of any type** as a user wants
  - ❖ Only 1 of each type in GEANT4
- One user action can be of **several types** (run/event/stacking/tracking/stepping)
  - ❖ Only 1 class - 1 type in GEANT4
- Activated **by a user command**
- **Add filters and classifiers** in the user command line

## Or use your own C++ code

- ❖ Any class inheriting from *G4VUserXXXAction*
- ❖ Change inheritance to *GmVUserXXXAction*
- ❖ Add one line to **transform your class in a plug-in**

`DEFINE_GAMOS_USER_ACTION(ExN07RunAction)`

so that you can select it in your input macro

`/gamos/userAction ExN07RunAction`

# Scoring

Scoring is an important part of your simulation ⇒ **powerful and flexible framework developed:**

➤ **Many possible quantities** can be scored in one or several volumes (based on Geant4 scorers)

- |                            |                          |  |
|----------------------------|--------------------------|--|
| • Dose                     | • Deposited energy       | • Flux (in/out/passage)                |
| • Current (in/out/passage) | • Charge                 | • Step length                          |
| • Number of particles      | • Number of interactions | • Number of 2 <sup>ary</sup> particles |
| • Number of steps          | • Minimum kinetic energy | • <u>Kerma</u>                         |

➤ For each scored quantity **one of several filters** can be used

➤ Only electrons, only particles in a given volume, ...

➤ **Several ways to classify** the different scores

➤ One different score for each volume copy, or volume name, or energy bin, ...

➤ Results can be printed in **one or several formats** for each scored quantity

➤ Standard output, text/binary file, histograms

➤ Scoring can be made in **parallel worlds**

✓ **All scored quantities can be calculated with/without errors**

✓ **All scored quantities can be calculated per event or per run**

▪ Taking into account correlations from particles from same event

# Scoring

## ☺ Everything managed with user commands

For example: **score the number of interactions (collisions) of gammas in each physical volume**

- **Select all volumes for scoring**

/gamos/scoring/createMFDetector nInterDet \*

- **Score number of collisions**

/gamos/scoring/addScorer2MFD nInterScorer **GmG4PSNofCollision** nInterDet

- **One score per each physical volume**

/gamos/scoring/assignClassifier2Scorer **GmClassifierByPhysicalVolume** nInterScorer

- **Only score for gammas**

/gamos/scoring/addFilter2Scorer **GmGammaFilter** nInterScorer

## OUTPUT:

MultiFunctionalDet: nInterDet

PrimitiveScorer: nInterScorer

Number of entries= 6

index: target:1 = 2.111 +/- (REL) 0.024593555

index: primary collimator:1 = 1.066 +/- (REL) 0.049482843

index: flattening filter:1 = 0.028 +/- (REL) 0.28856012

index: jaws\_X:1 = 0.003 +/- (REL) 1

index: jaws\_X:2 = 0.006 +/- (REL) 0.6231189

index: expHall:0 = 0.002 +/- (REL) 0.70675279

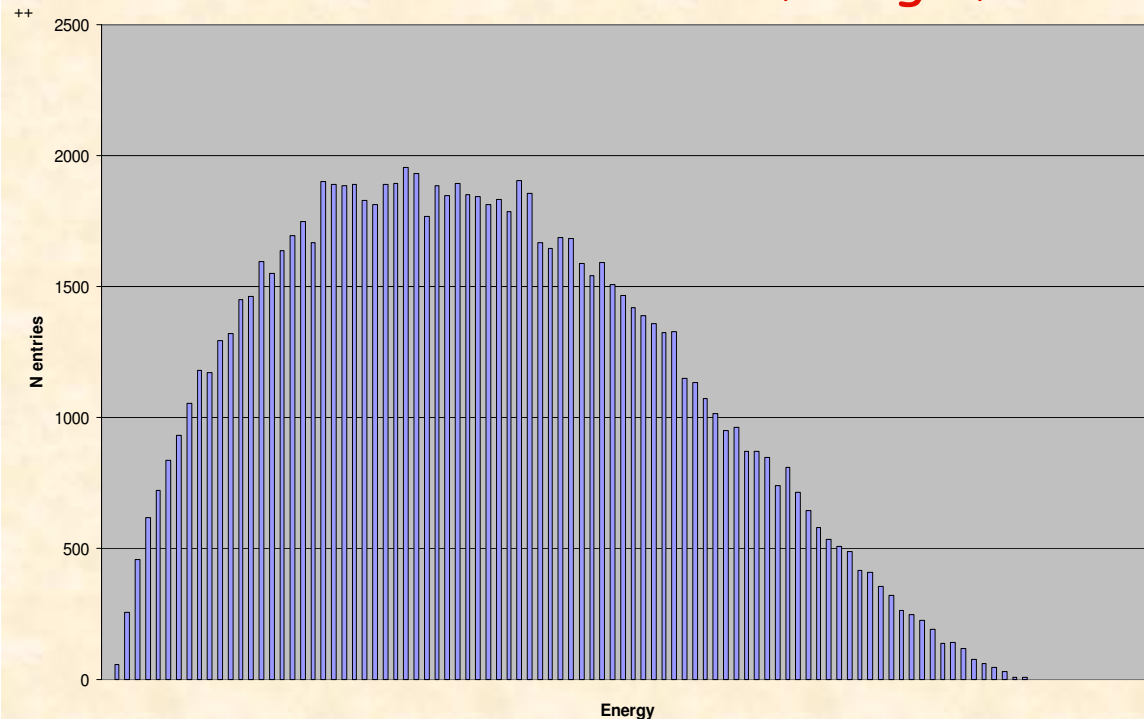
# Histograms

❖ By default histograms are written in ROOT format

❑ But many medical users have never heard of ROOT, while they are experts on Matlab, Origin, ...

⇒ **Own format:** can write histograms in text files (CSV) without any external dependency

✓ Read them in Matlab, Origin, Octave, GNUplot, ..., even MS Excel



```
/gamos/userAction GmGenerHistosUA  
/gamos/analysis/fileFormat CSV
```

**MS Excel graphics of energy of e+ from F18 decay**



# Parameter management

Many classes change their behaviour depending on the value of some parameters, that the user can set with user commands

☺ GAMOS code use parameters extensively to maximise the flexibility

- ☐ Always a default value in case the user does not care

☺ Parameters can be numbers, strings, list of numbers or list of strings

- ☐ Number: units and mathematical expressions can be used

`10*cos(30*deg) (3-0.5*0.6)*becquerel`

- ☐ Strings: wildcards can be used

`*det crys*_*`

☺ GAMOS provides a unique command to manage all these parameters

- A parameter is defined in the input macro

`/gamos/setParam SD:EnergyResol 0.1*mm`

- Any class can use this value in any part of the code

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

**Over 300 parameters (all clearly documented)**



# *Extracting your results*

# Obtaining data

## Many user actions that produce histograms, tables or text/binary files

### Histograms

- **Step:** process, energy, energy lost, energy deposited, step length, number of secondaries, position X/Y/Z/R2/R3/phi/theta, change in position, change in angle, change in time
  - **Track:** creatorProcess, initial energy, final energy, energy lost, energy deposited, track length, number of steps, number of secondary's, initial position X/Y/Z/R2/R3/phi/theta, final position X/Y/Z/R2/R3/phi/theta, change in position, change in angle
  - **Secondary tracks:** creator process, energy, energy of primary, energy secondary/energy primary, angle secondary – primary preStep, angle secondary –primary postStep, change of primary angle
  - **Primary generator:** energy, 1D and 2D positions, angles, time between events
  - **Hits** (for each SD type): number of hits, energy, number of energy deposits, width, time span, distance between hits, positions (Cartesian, cylindrical)
  - **Reconstructed hits** (for each SD type): number of rechits, energy, number of hits, width, time span, distance between hits, positions (Cartesian, cylindrical)
  - **Positron:** initial energy, final energy, final time, final range, final distance to origin, gamma energies, gamma angles, range vs energy, range vs distance, ...
  - **PET:** distance PET line to vertex, hit energies, positron gammas histos (to disentangle detector effects)
  - **PET classification table:** true/scatter/random coincidences/random&scatter, close or far from vertex, recovered Compton hits or not
  - **Radiotherapy phase space:** X, Y, XY, theta, phi, energy, Vx, Vy, Vz
  - **Radiotherapy dose in phantoms:** dose, dose-volume, PDD, profile, user selection of voxels
- ✓ **User can select histogram min, max and number of bins of each histogram with a user command**

# Obtaining data

## Tables with debugging data

- **Table of interactions:** how many times each process type happened, how many times a particle was created by a process type
- **Table of number of tracks and step**
- **Table of where CPU is spent**

## Text or binary files with track and step data

- **Step:** event ID, trackID, particle, process, energy, energy lost, energy deposited, step length, number of secondaries, position X/Y/Z/R2/R3/phi/theta, change in position, change in angle, change in time
- **Track:** eventID, trackID, particle, creatorProcess, initial energy, final energy, energy lost, energy deposited, track length, number of steps, number of secondary's, initial position X/Y/Z/R2/R3/phi/theta, final position X/Y/Z/R2/R3/phi/theta, change in position, change in angle
- **Secondary tracks:** eventID, primaryID, particle, creator process, energy, energy of primary, energy/energy primary, angle secondary – primary preStep, angle secondary –primary postStep, change of primary angle
- ✓ **User can select which variables to write**
- ✓ **Example of writing and reading back a binary file**

# Filters

## Filter (accepts or not) on *G4Step*, *G4Track* or *G4Step* and *G4Track*

- Start/End/Enter/Exit/Traverse/In LogicalVolume/PhysicalVolume/Touchable/Region (24 filters)
- Start/End/Enter/Exit/Traverse/In LogicalVolume/PhysicalVolume/Touchable/Region or their children (24 filters)
- Gamma
- Particle in list
- Charged particles
- Process and particle
- Secondary
- Filter on classifier index
- Electron
- Energy in an interval (pre, post or vertex)
- Neutral particles
- Creator process
- Range
- Positron
- Process name
- Primary
- Step number

(for volume, particles, processes one or several can be given and wildcards can be used, e.g. ``GmCreatorProcessFilter *Ioni compt`` )

### Filters on filters:

- AND
- Inverse
- Filter with history (pass filter in any previous track step)
- Filter with ancestor history (one ancestor pass filter in any previous track step)
- OR
- OnSecondary (apply filter to secondary's)
- XOR

# Classifiers

Classify *G4Step* and return an index

New concept developed for *GAMOS*, not existing in *Geant4*

- ByParticle
- ByNAncestors
- ByTouchable
- ByParticleProcess
- CompoundClassifier
- ByKineticEnergy
- ByLogicalVolume
- ByRegion
- ByProcessName
- By1Ancestor
- ByPhysicalVolume
- ByExtraInfoPhaseSpace
- ByPrimaryParticle



# Obtaining desired results

- ☺ Scorers can be used together with filters or classifiers
- ☺ User actions can also be used together with filters or classifiers

/gamos/userAction GmTrackHistosUA GmPrimaryFilter GmGammaFilter

- only makes histograms for primary particles that are gamma

/gamos/userAction GmTrackHistosUA GmClassifierByParticle

- makes a different set of histograms for each particle type

+200 histograms & 23 scorers + 96 filters & 12 classifiers, all managed with Geant4-like user commands: vast amount of possibilities

- Big chance that you can get your desired output data without C++ !
  - Not only for medical physics !

- And if you are missing something: user actions/scorers/filters/classifiers are plug-in's = easy to add a new one



# Obtaining desired results: example

**PROBLEM:**  $^{90}\text{Sr}$  has a beta-decay to  $^{90}\text{Y}$  which again has another beta-decay,  
I want to separate the dose deposition from each of the two betas

**SOLUTION:** make a dose deposit scorer for particles that come from  $^{90}\text{Y}$   
and another for particles that come directly from  $^{90}\text{Sr}$  (= do not come from  $^{90}\text{Y}$ )

## 1. Do it with C++:

- ❖ Write `G4VUserTrackingAction` that creates `G4VUserTrackInformation` if track is  $^{90}\text{Y}$  and copy it to the secondary tracks
- ❖ Write a `G4MultiFunctionalDetector`
- ❖ Write a `G4PSDoseDeposit` scorer and associate it to the `G4MultiFunctionalDetector`
- ❖ Write a `G4VFilter` that accepts particle with the `G4VUserTrackInformation`, and attach it to the scorer

**SUMMARY:** > 100 lines of C++, that need a good Geant4 expertise

# Obtaining desired results: example

## 2. Do it with GAMOS commands:

- Select in which volume the scoring is done:

```
/gamos/scoring/createMFDetector doseDet sphere
```

- Create a filter that accepts a track if it comes from  $^{90}\text{Y}$  (= if its ancestor is  $^{90}\text{Y}$ ):

```
/gamos/filter Y90_filter GmParticleFilter Y90[0.0]
```

```
/gamos/filter Y90_ancestor_filter GmHistoryAncestorsFilter Y90_filter
```

- Create a filter that accepts a track if non of its ancestor is  $^{90}\text{Y}$  (inverse of previous one):

```
/gamos/filter NotY90_ancestor_filter GmInverseFilter Y90_ancestor_filter
```

- Create two dose deposit scorers, each one with one of the above filters:

```
/gamos/scoring/addScorer2MFD Y90_doseScorer GmG4PSDoseDeposit doseDet
```

```
/gamos/scoring/addFilter2Scorer Y90_ancestor_filter Y90_doseScorer
```

```
/gamos/scoring/addScorer2MFD NotY90_doseScorer GmG4PSDoseDeposit doseDet
```

```
/gamos/scoring/addFilter2Scorer NotY90_ancestor_filter NotY90_doseScorer
```

**SUMMARY: < 10 lines, no C++, no need to know Geant4 details**

# Obtaining desired results: example

## OUTPUT:

```
# MultiFunctionalDet: doseDet
# PrimitiveScorer: Y90_doseScorer
# Number of entries= 2
# index: 1 = 1.2553326e-15 +-(REL) 1 Gy
# index: 2 = 2.6671857e-10 +-(REL) 0.00164726 Gy
#doseScorer_Y90 SUM ALL: 2.6671982e-10 +/- 4.3935662e-13 Gy

# MultiFunctionalDet: doseDet
# PrimitiveScorer: NotY90_doseScorer
# Number of entries= 2
# index: 1 = 6.8124996e-15 +-(REL) 0.19720637 Gy
# index: 2 = 6.8497274e-11 +-(REL) 0.0022286963 Gy
#doseScorer_NotY90 SUM ALL: 6.8504086e-11 +/- 1.5266553e-13 Gy
```

Even if you are profficient in C++ and Geant4, using GAMOS will save you a lot of time...

# Optimisation: time studies

## ✓ User action to get CPU time study

➤ By particle, energy bins, volume, region (or combination of them)

Add two user commands:

```
/gamos/classifier GmCompoundClassifier particleAndEnergyClassifier  
GmClassifierByParticle GmClassifierByEnergy  
  
/gamos/userAction GmTimeStudyUA particleAndEnergyClassifier
```

gamma/0.001-0.01:	User=0.01 Real=0 Sys=0
gamma/0.01-0.1:	User=2.01 Real=2.45 Sys=0.27
gamma/0.1-1:	User=19.12 Real=22.05 Sys=1.51
gamma/1-10:	User=4.25 Real=5.4 Sys=0.3
e-/0.0001-0.001:	User=0.07 Real=0.1 Sys=0
e-/0.001-0.01:	User=0.54 Real=0.69 Sys=0.06
e-/0.01-0.1:	User=4.71 Real=5.41 Sys=0.38
e-/0.1-1:	User=15.59 Real=18.19 Sys=1.79
e-/1-10:	User=82.83 Real=98.62 Sys=7.45

## ✓ Example to get detailed *gprof* profiling about where (in which methods) the time is spent

➤ Time spent in a method and integrated time in all the methods called by it

# *Applications*

*PE7*



# PET

- ❖ Any PET detector can be simulated with simple text format
- ❖ Utility to create simple geometries from a few parameters (number of crystals per block, number of block per ring, radius, ...)
- ❖ Several sensitive detectors types selectable by user commands
  - ❖ User can create it own one and make it a plug-in

- ❖ Automatic creation of hits with detailed information

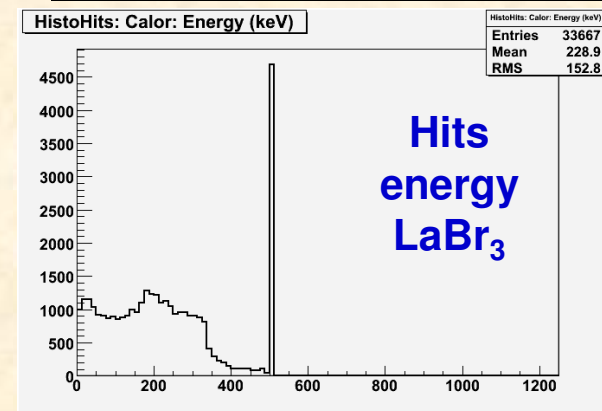
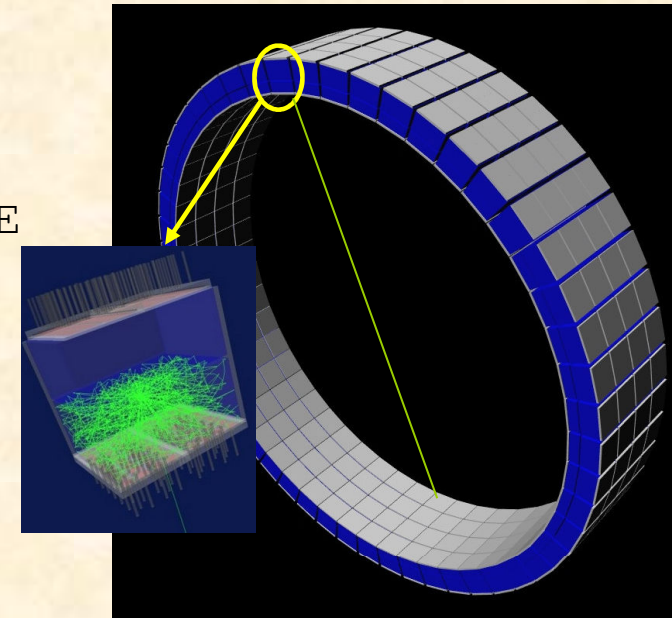
```
/gamos/assocSD2LogVol GmSDSimple SD_TYPE VOLUME_NAME
```

- ❖ Writing hits into text or binary file Retrieving them in next job (for doing parametric studies)
  - ❖ Format compatible with STIR software

- ❖ Detector effects

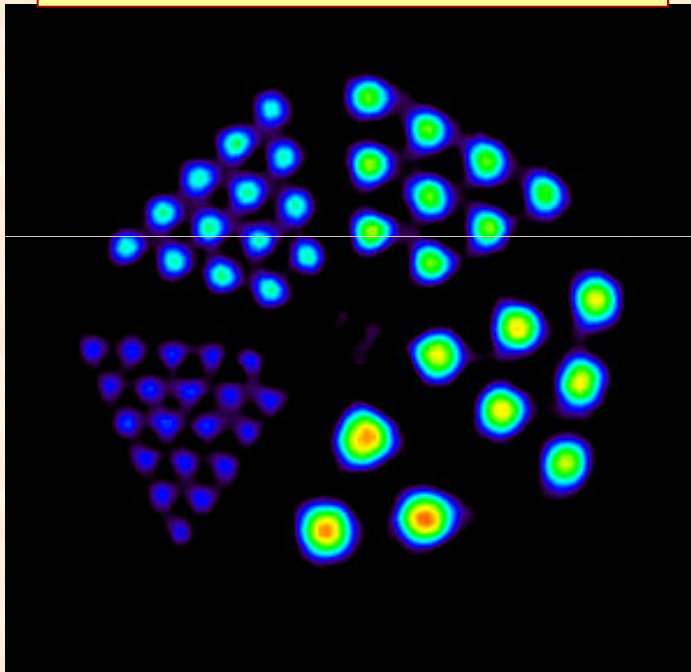
- Energy resolution
- Time resolution
- Dead time (paralyzable / non-paralyzable)
- Measuring time
- PET coincidence time

- ❖ Digitizer and reconstructed hits framework and examples

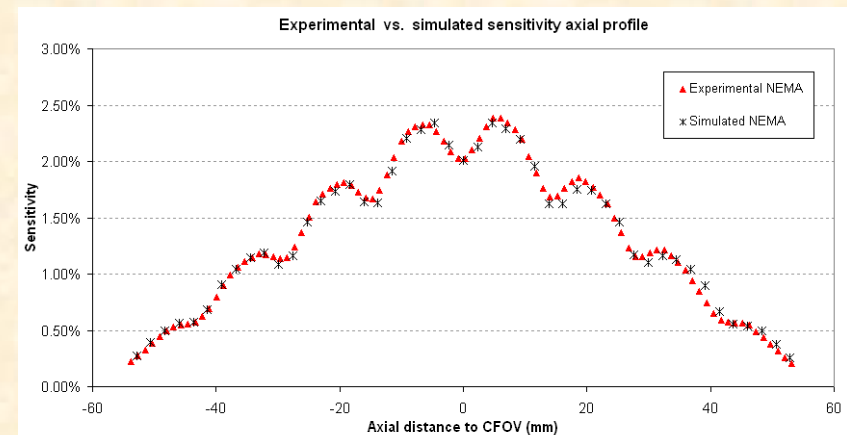


- ❖ Table of PET classification
  - ❖ True / scattered / random coincidences / scattered&random
  - ❖ PET line far or close to vertex
  - ❖ Optionally merge hits if there is more than one because of Compton interactions
- ❖ Histograms of hits or reconstructed hits with a user command
- ❖ Histograms of positron history
- ❖ Histograms of PET information

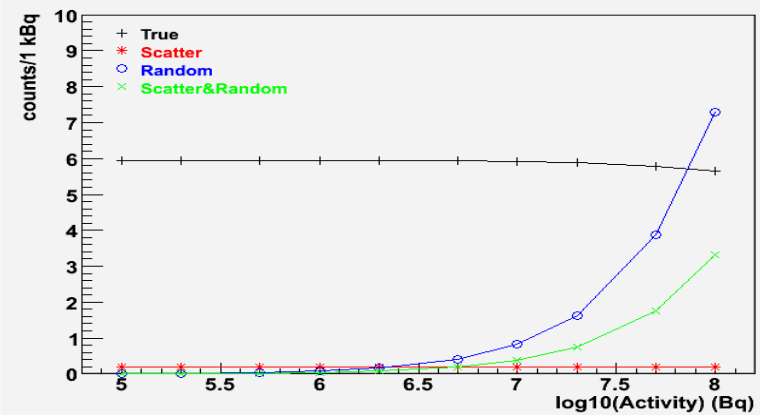
ClearPET derenzo reconstruction



ClearPET sensitivity profile



PixelPET sensitivity of different event types



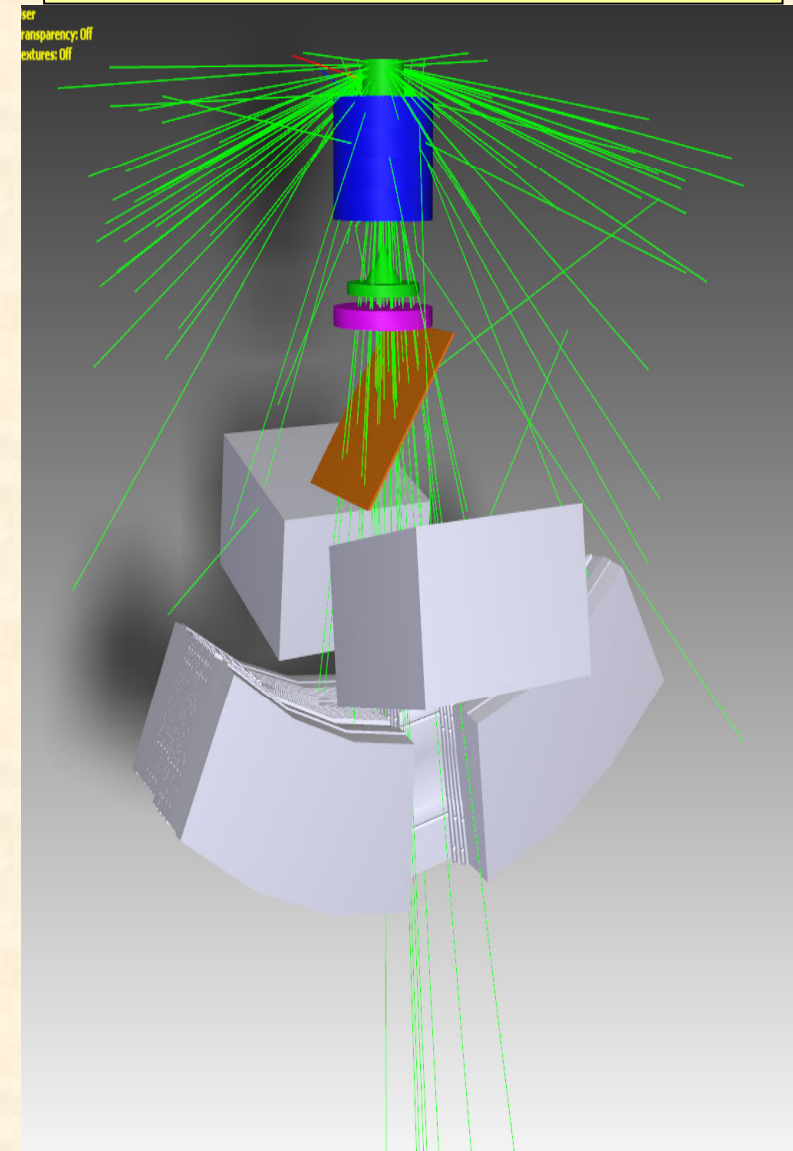
# *Radiotherapy*

## GAMOS framework covers the needs of an external beam

### radiotherapy simulation user:

- ❖ Easy way to build accelerator geometries
- ❖ Reading DICOM patient geometries
- ❖ Generator distributions
- ❖ Writing/reading phase space files
- ❖ Dose deposition in phantoms
  - ❖ Histograms and binary files
- ❖ CPU Optimisation
  - ❖ Optimising cuts
  - ❖ Variance reduction techniques
- ❖ Debugging tools

VRML view of VARIAN 6000 accelerator





# ***Accelerator geometry***

Accelerator geometry can be written with simple Geant4 text format

But some accelerator parts are difficult  $\Rightarrow$  we provide modules that allow to build them with a short list of parameters

## **JAWS:**

- Use radiotherapist point of view: provide field at a plane (10x10 cm, 40x40 cm, ...) :
  - Type (X / Y)
  - Box dimensions
  - Focus Z
  - Box position Z
  - Field Z
  - Projections on field
  - Material
  - Mother volume to place it



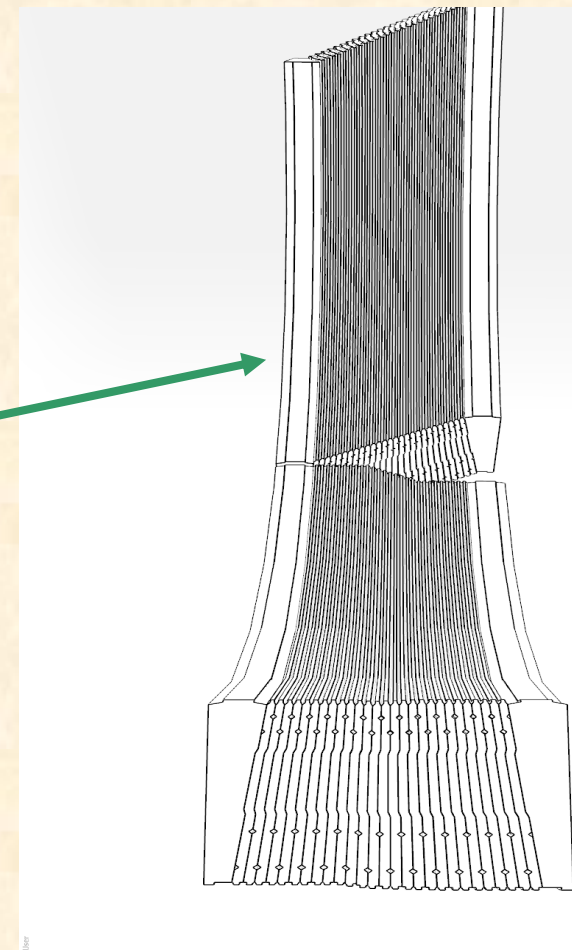
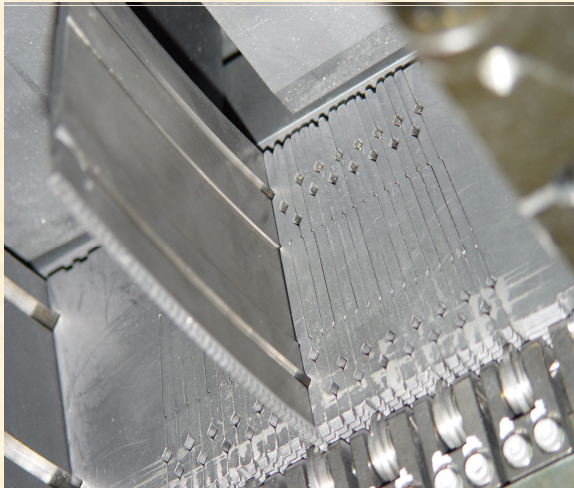
# Accelerator geometry

## MULTILEAF COLLIMATOR:

➤ Use radiotherapist point of view: provide conformal beam

- Leave profiles
- End leaves type (Rounded or straight)
- Focus Z,
- Top MLC Z plane
- Isocenter Z
- Interleaves Gap, Z Gap Definition
- Material
- ...

VRML view of curved MLC



## Write phase space files

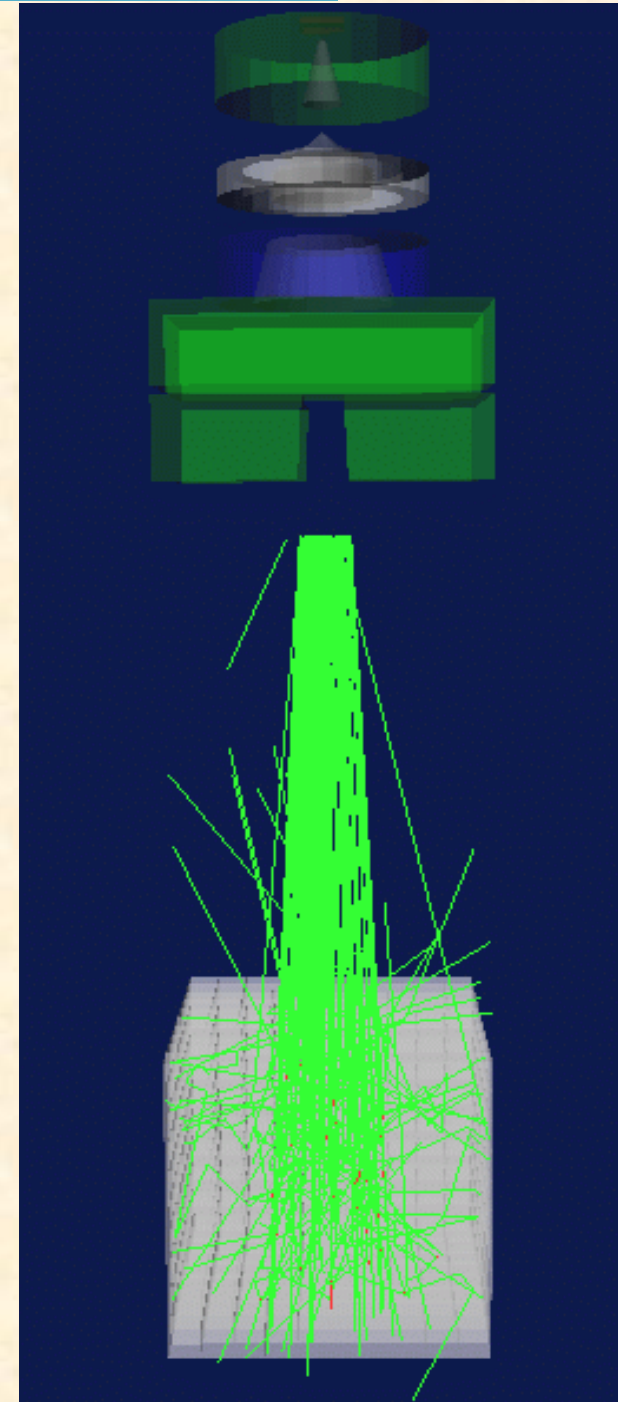
### ❖ IAEA format

- ❑ One or several Z planes in the same job
- ❑ Stop after last Z plane or not
- ❑ Save header after N events  
(not to lose everything if job is aborted)
- ❑ Save extra info:
  - ❑ Regions particle traversed
  - ❑ Regions particle created
  - ❑ Regions particle interacted
  - ❑ Particle origin Z
  - ❑ BIG FLEXIBILITY
    - ❑ More can be easily added (they are plug-in's)
    - ❑ User selects which information and how many bytes each one occupies

# Phase space files

## Use phase space files as generator

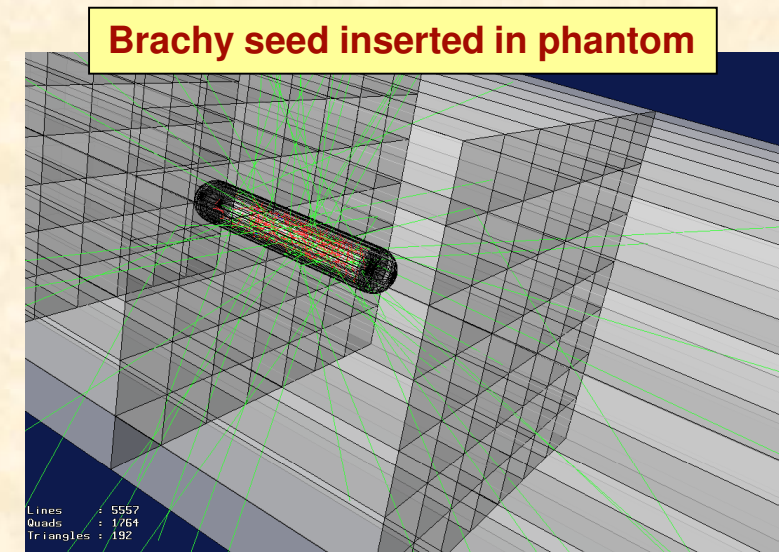
- ❖ Displace or rotate phase space particles
- ❖ Reuse phase space particles
  - ❑ Optional automatic calculation of reuse number
  - ❑ Optional mirroring in X, Y or XY
  - ❑ Particles can be reused without writing phase space files
- ❖ Recycle phase space files
- ❖ Optional skipping of first N events
- ❖ Optional histograms of particles read
- ❖ Filter particles by extra info
- ❖ Also read EGSnrc phase space format





# Phantom voxelised geometries

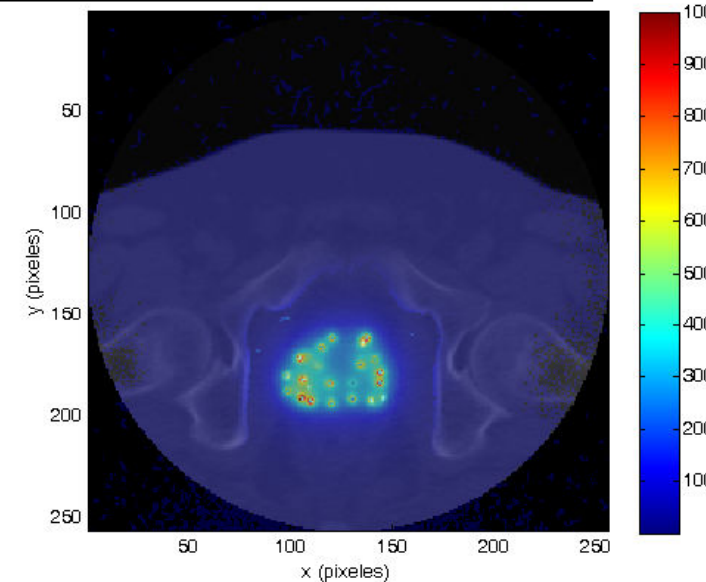
- Use DICOM files
  - ❑ Read GEANT4 format (example advanced/medical/DICOM: DICOM → ASCII)
  - ❑ Read EGSnrc format
- ❖ Split one material in several if voxel densities are different
  - ❑ User defines an interval  $X$  and a material is created for the densities in each  $X$  interval
- ❖ Displace or rotate read-in phantom geometry
- Simple phantoms can be created with a few commands
- ❖ Geant4 allows to build parallel worlds
  - ☹ But only geometry is seen, not materials (=no interactions)
- ☺ We have developed a tool that allows to **insert objects in phantom geometries** and produce the interactions in them
  - ✓ Realistic simulation of brachytherapy sources or ionisation chambers!



# Dose in voxelised phantom

- ❖ Fast Geant4 regular navigation is default
  - ❖ New dose scorer, taking into account effect of energy loss and multiple scattering when distributing dose in voxels
- ❖ Dose with errors can be calculated
- ❖ Different printing formats in each run
  - ❑ Text in standard output
  - ❑ Dose histograms (X, Y, Z, XY, XZ, YZ, dose, dose-volume)
  - ❑ File with dose and error at each voxel
  - ❑ File with dose and dose<sup>2</sup> at each voxel (to properly take into account correlations)
- ❖ Many different doses can be calculated in same job, with different filters, for detailed studies
- ❖ Dose in total or by event
  - ❑ Proper management of original number of events when reading phase space files

Dose from brachytherapy seeds





# Analysis of results

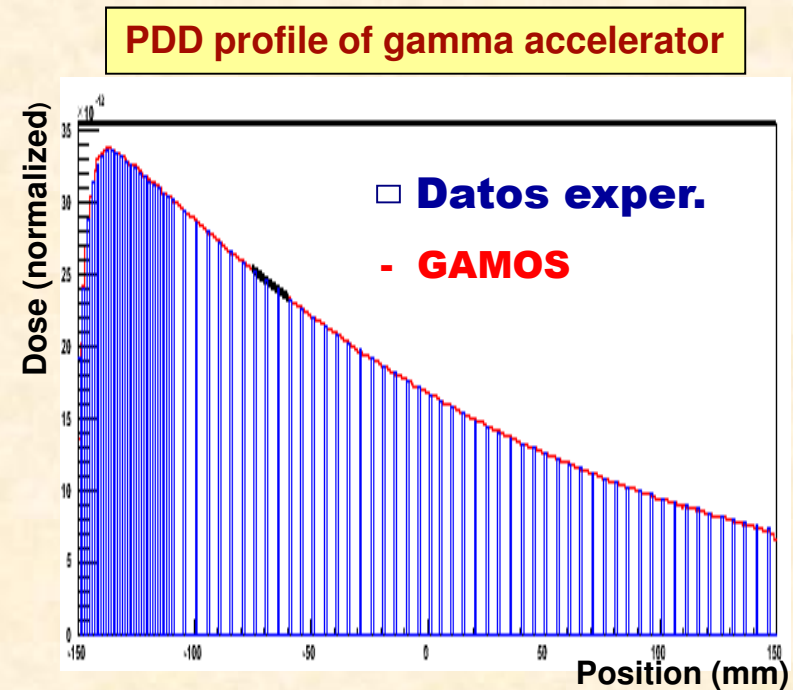
A set of easy-to-use executables and scripts

## Phase space files

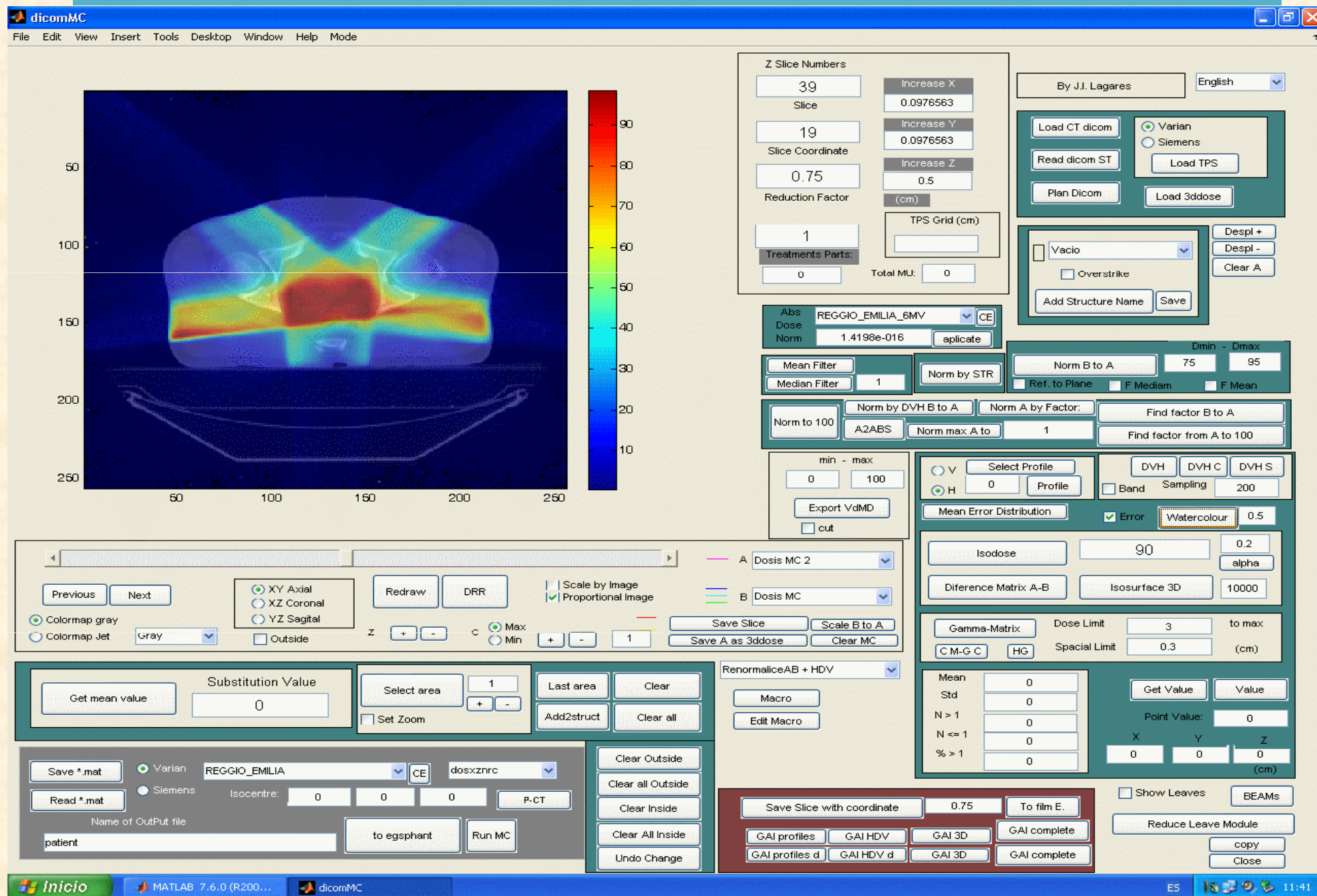
- ❖ Sum phase space files
- ❖ Analyse phase space files
  - Statistics
  - Histograms by particle type at each Z plane
    - (= when writing PS)
- ❖ Compare two phase space files

## Dose files

- ❖ Sum dose files
  - ❖ Analyse dose files
    - Statistics
    - Dose histograms (X, Y, Z, XY, XZ, YZ, dose, dose-volume) (= when writing dose)
  - ❖ Compare two dose files
- GUI under development (MIRAS project)



# Dose visualisation



IMRT dose visualisation with CIEMAT's tool (MIRAS)

# Radiotherapy optimisation

## Cuts and user limits optimisation:

- ❖ Optimisation of cuts and user limits is usually a complicated task
  - ❖ But they can save you a lot of CPU time

In GAMOS we propose an  
**Automatic determination of best production cuts or user limits in  
one job**

## Geant4 electromagnetic parameters:

- ❖ They are not optimised for radiotherapy
  - ❖ They cover a wide range of energy and applications ⇒ they have to be conservative
- ✓ We have started to optimise GEANT4 electromagnetic parameters for radiotherapy: <http://fismed.ciemat.es/GAMOS/RToptim>

# ***CPU time comparison with BEAMnrc***

VARIAN 2100 gamma accelerator, 6 MeV:

10<sup>6</sup> events on Pentium Dual-Core 3 GHz

geant4.9.3.beta01

<b>BEAMnrc</b>	<b>GAMOS (standard)</b>	<b>GAMOS (low energy)*</b>
<b>277 s</b>	<b>291 s</b>	<b>476 s</b>

\* No latest log-log interpolation speed improvements



# CPU time comparison with DOSXYZnrc

Dose in phantom, phase space file as generator

10<sup>4</sup> 5x5x3 mm water phantom and 4.5 10<sup>6</sup> head & neck phantom

10<sup>6</sup> events on Pentium Dual-Core 3 GHz

geant4.9.3.beta01 (regular navigation, skipping voxel boundaries)

	<b>DOSXYZnrc</b>	<b>GAMOS/GEANT4 (standard)</b>	<b>GAMOS/GEANT4 (low energy)</b>
<b>water</b>	<b>234 s</b>	<b>150 s</b>	<b>338 s</b>
<b>patient</b>	<b>300 s</b>	<b>179 s</b>	<b>324 s</b>

## Dependency with number of materials

### DOSXYZnrc:

- 4 densities: **297 s**
- All densities: **300 s**

### GEANT4: (1 density = 1 material)

- 4 materials: **179 s**
- 23 materials: **316 s**
- 68 materials: **394 s**
- 196 materials: **516 s**

➤ Need to implement density-changing materials in Geant4...



# Particle splitting

## UNIFORM BREMSSTRAHLUNG SPLITTING

- All bremsstrahlung photons are replicated the same number of times

## Z-PLANE DIRECTION BREMSSTRAHLUNG SPLITTING

- User defines a Z plane with limits in X & Y (represents entrance of phantom)
- Same as uniform BS, but if gamma does not aim at Z plane, Russian roulette is played

## EQUAL-WEIGHT SPLITTING

- Similar as Z-plane direction BS, but splits every gamma produced, not only from bremsstrahlung
- Russian roulette is played with  $e^-/e^+$ , so that very few reach Z plane
- Aim is that all particles that reach phantom have the same weight
- Based on EGSnrc DBS technique

# Particle splitting: RESULTS

EM physics “standard”:

## UNIFORM BREMSSTRAHLUNG SPLITTING

- Maximum efficiency gain: 2.2 times

## Z-PLANE DIRECTION BREMSSTRAHLUNG SPLITTING

- Maximum efficiency gain: 6.5 times

## EQUAL-WEIGHT SPLITTING

- Maximum efficiency gain: 45 times

- BEAMnrc results with same accelerator: gain 80

- Algorithm in GAMOS is not fully implemented yet...

# Installation

# Documentation

# Installation

GAMOS is freely available from CIEMAT web

- User registers and downloads installation scripts

We provide a no-choice but very easy way: **one-line installation**

```
sh installGamos.sh $HOME/gamos
```

- No need to manually download and install packages
  - No need to define environmental variables
  - ✓ Checks that your system has the needed components
  - ✓ Downloads, installs and compiles CLHEP, Geant4, (optionally) ROOT and GAMOS in the selected directory
    - *GAMOS compiles = Geant4*
  - Optionally an expert user can make several choices, Geant4-like
  - ✓ Installation tested on Scientific Linux, Fedora Core, Debian and Ubuntu, and on MacOS
- > 50 tests are run on three different OS to check each new release

## User's Guide:

- Installation
- All available functionality
- How to provide new functionality by creating a plug-in

## Software reference manual (doxygen):

- Documentation of the classes and their dependencies

## Examples:

- A simple one and a few more complicated ones

```
/gamos/setParam GmGeometryFromText:FileName mygeom.txt  
/gamos/geometry GmGeometryFromText  
/gamos/physics GmEMPhysics  
/gamos/generator GmGenerator  
/run/initialize  
/gamos/generator/addSingleParticleSource my_source gamma 6.*MeV  
/run/beamOn 1000
```




## Four tutorials

- Radiotherapy tutorial
- PET tutorial
- Histograms and scorers tutorial
- Plug-in tutorial
- Propose about 15 exercises each
  - ❖ Increasing in difficulty
  - ✓ Reference output provided
  - ✓ Solutions provided
  - ⇒ User can do them by her/himself
- 4 GAMOS tutorial courses have been given in Europe and America
  - Next tutorials:
    - European School of Medical Physics (October 2009)
    - Univ. Santiago de Chile (February 2010)


# Summary


- GAMOS is an easy and also flexible framework
  - You can do many things with GAMOS utilities, but you may easily replace any GAMOS component with a Geant4 or own one
- GAMOS provides wide range of possibilities to get detailed results
- GAMOS has several optimisation options
- Full PET and external beam radiotherapy applications have been developed
- ✓ GAMOS core is application-independent
  - ✓ Other applications are being developed
  - ✓ First talks about merging GAMOS and GRAS...
  - ✓ Proposal for putting GAMOS code in Geant4 environments repository...
  - ✓ Even if you are not medical physicist, it will probably save you a lot of time

**http://fismat.ciemat.es/GAMOS**  
or **http://geant4.cern.ch → Applications → Medical**



File Edit View History Bookmarks Tools Help





# GAMOS

Geant4-based Architecture for Medicine-Oriented Simulations

User Name:   
Password:   
  
[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](#)

Visitor locations



## 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.

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](#)
- [SEAL](#)
- [CLHEP](#)
- [ROOT](#)

### News

- **Last GAMOS release 2.0.1!**