GATE

A GEANT4-based toolkit for Medical Physics

What is GATE?

“GATE is an advanced opensource software developed by the international OpenGATE collaboration and dedicated to numerical simulations in medical imaging and radiotherapy. GATE is based on the Geant4 toolkit.”

“It currently **supports** simulations of Emission Tomography (Positron Emission Tomography - **PET** and Single Photon Emission Computed Tomography - **SPECT**), Computed Tomography (**CT**), **Optical Imaging** (Bioluminescence and Fluorescence) and **Radiotherapy experiments**. Using an e**asy-to-learn macro mechanism** to configurate 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. It can also be used for dose calculation in radiation therapy, brachytherapy or any other application.”

Summary

- GATE runs as a generic GEANT4 application
- No programming skills required!
- Everything is setup using script commands, basically macro files — no graphical interface, sorry…
- Different strategies for imaging applications and for dosimetry/radiotherapy
- See documentation here

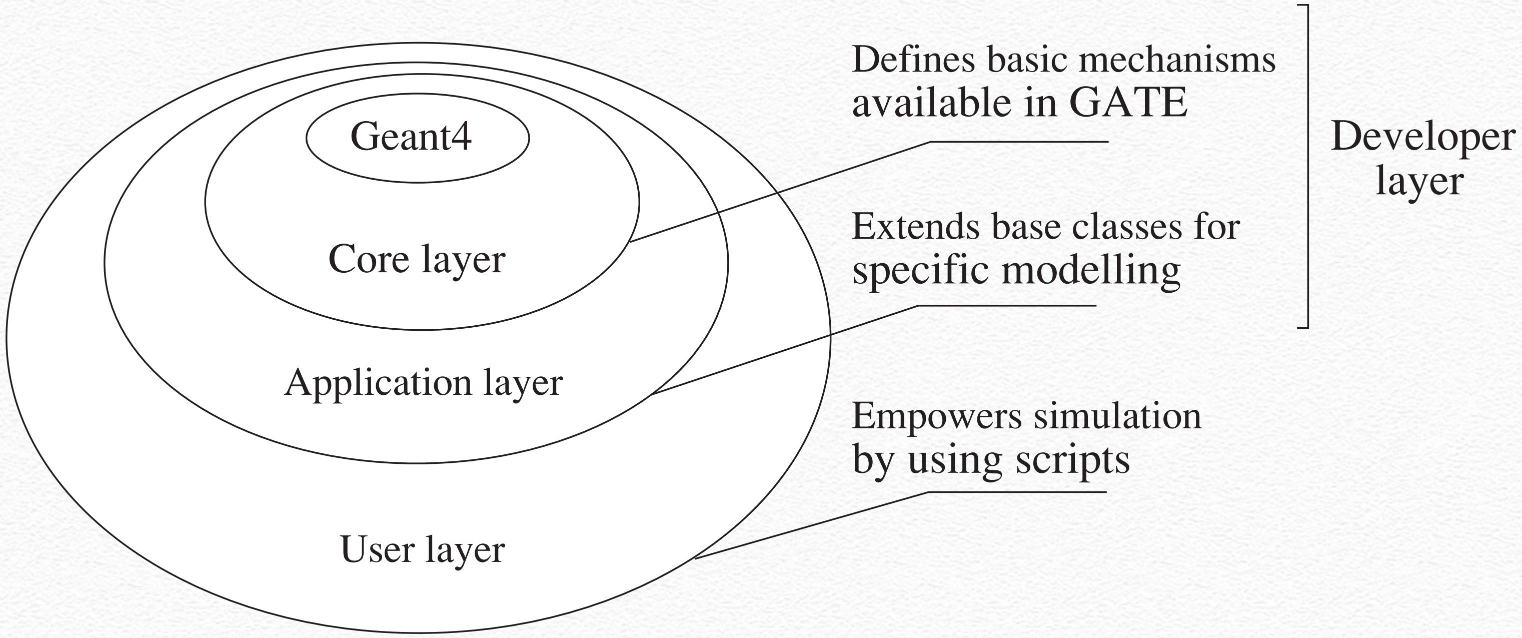


Figure 1. Sketch of the layered architecture of GATE.

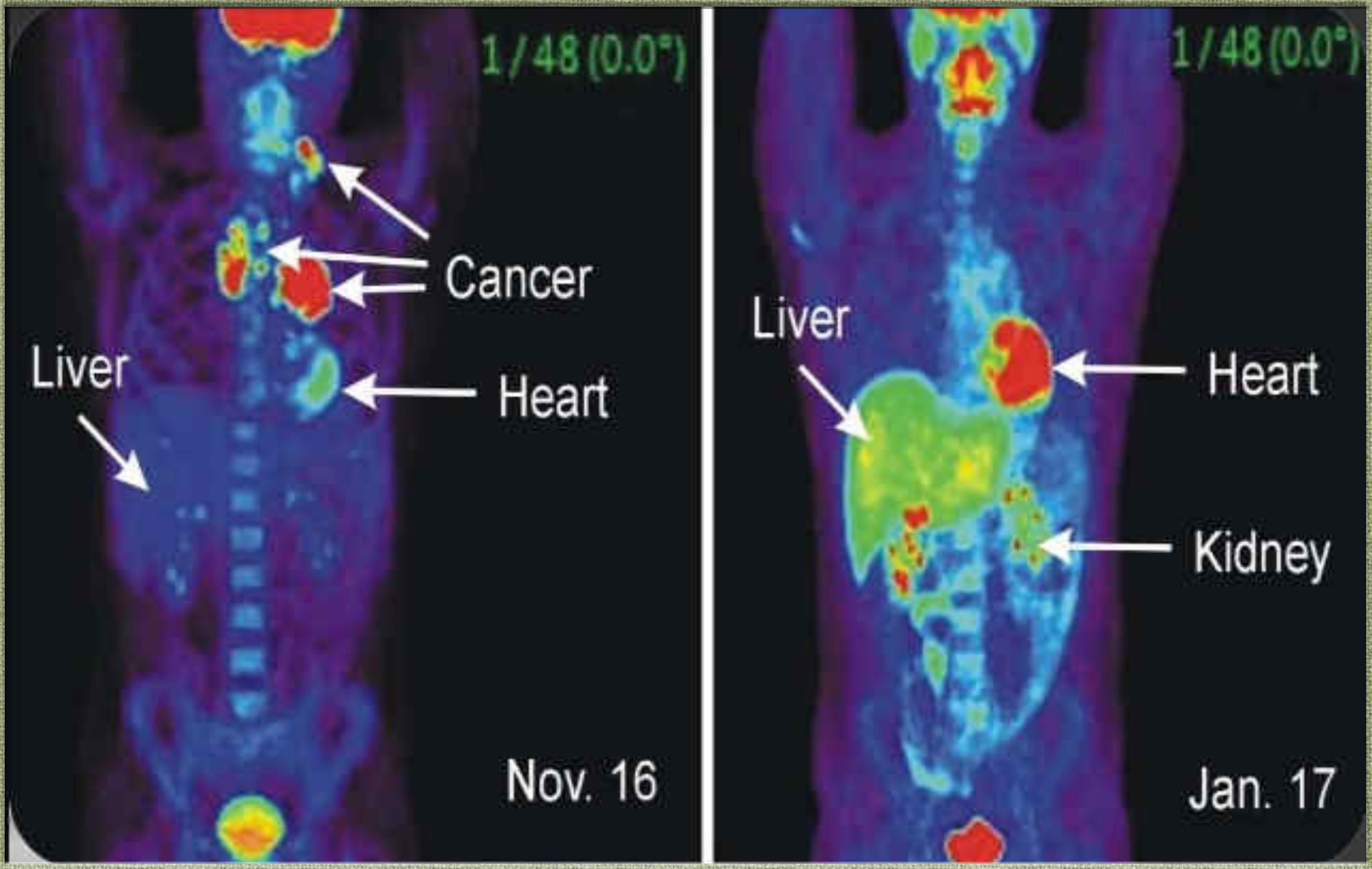
GATE-RTion

- Dedicated tool for pencil beam facilities (proton and light ion beam therapy)
- Provide a long term stable version of GATE, with all the necessary features for dosimetric applications
- Provide a collection of tools for interface with the clinical environment
- Develop guidelines for clinical users for the use of the provided tools

More details: https://gate.uca.fr/download/gate-rtion#/admin

Useful links

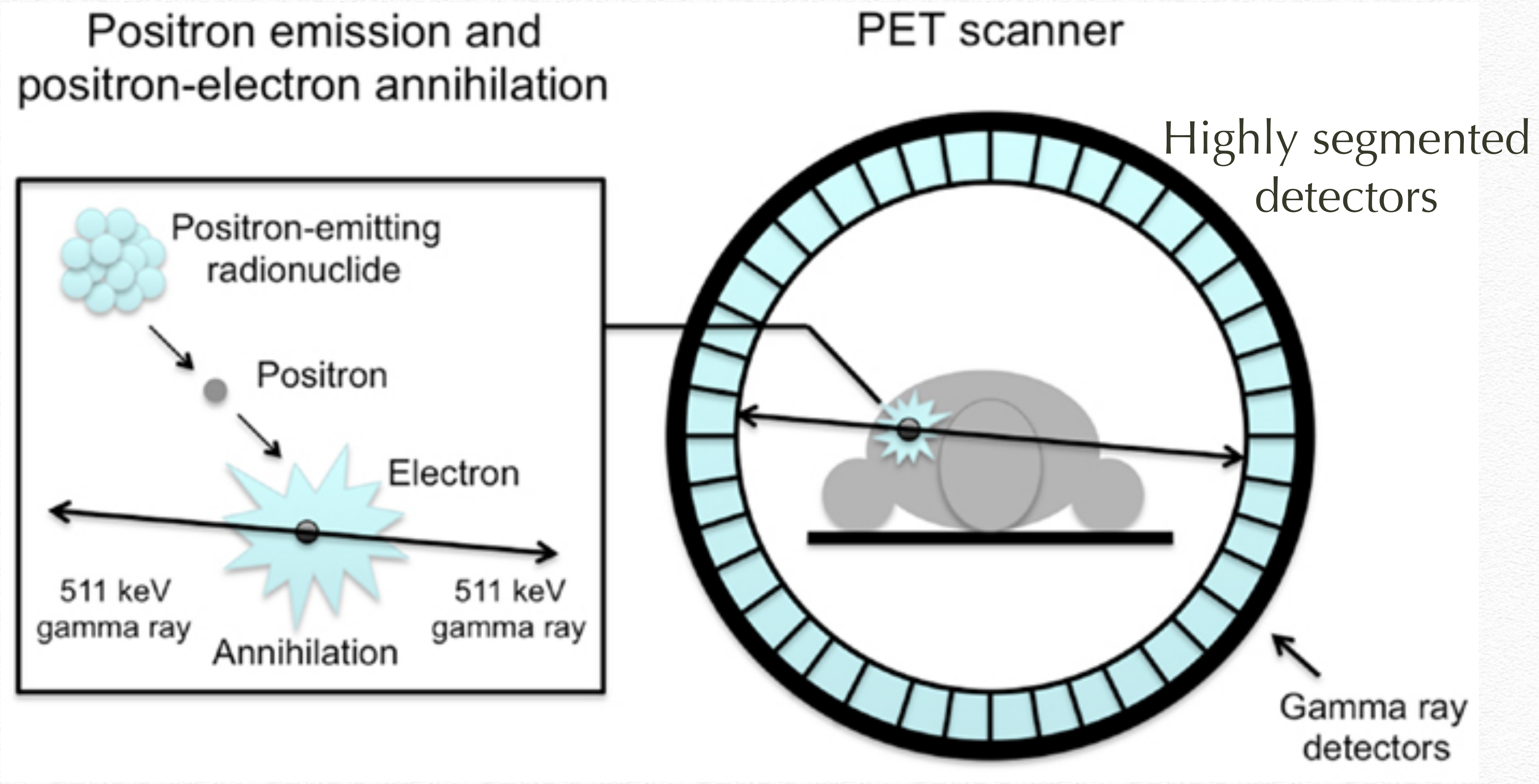
- GATE website with documentation and download instructions
- A long list of publications with overviews on Gate and many specific applications in Medical Physics, both for imaging and treatment planning
- Repository with plenty of examples



PET

Positron Emission Tomography

The concept



- Patient injected with a positron emitting radioactive isotope (e.g. ^{11}C, ^{13}N, $^{15}0$, and ^{18}F), attached to a molecule preferably absorbed by the organ to study
- Isotope emits a positron, which thermalises and annihilates emitting two (almost) back-to-back 511 keV gammas
- One or more rings of detectors detect these gammas. Coincidences are used to produce an image of the emission region by overlapping many lines

Gate PET example

- The benchmarks folder has plenty of examples
- The benchPET consists of:
  - 8 detector heads (that can be rotated)
  - 400 detector blocks each
  - each block is a dual layer of LSO-BGO crystals
  - cylindrical water phantom with two linear sources (^{18}F, 109.8 min HL and ^{15}O, 2.03 min HL) 100 kBq each
  - total acquisition time is 4 min, in 2x 2 min frames
  - the heads rotate by 22.5 deg between frames
  - coincidence time window is set to 120 ns (to allow a large number of random coincidences)
  - takes **~12 hours** in a 1 GHz CPU (!)…

Image from this paper

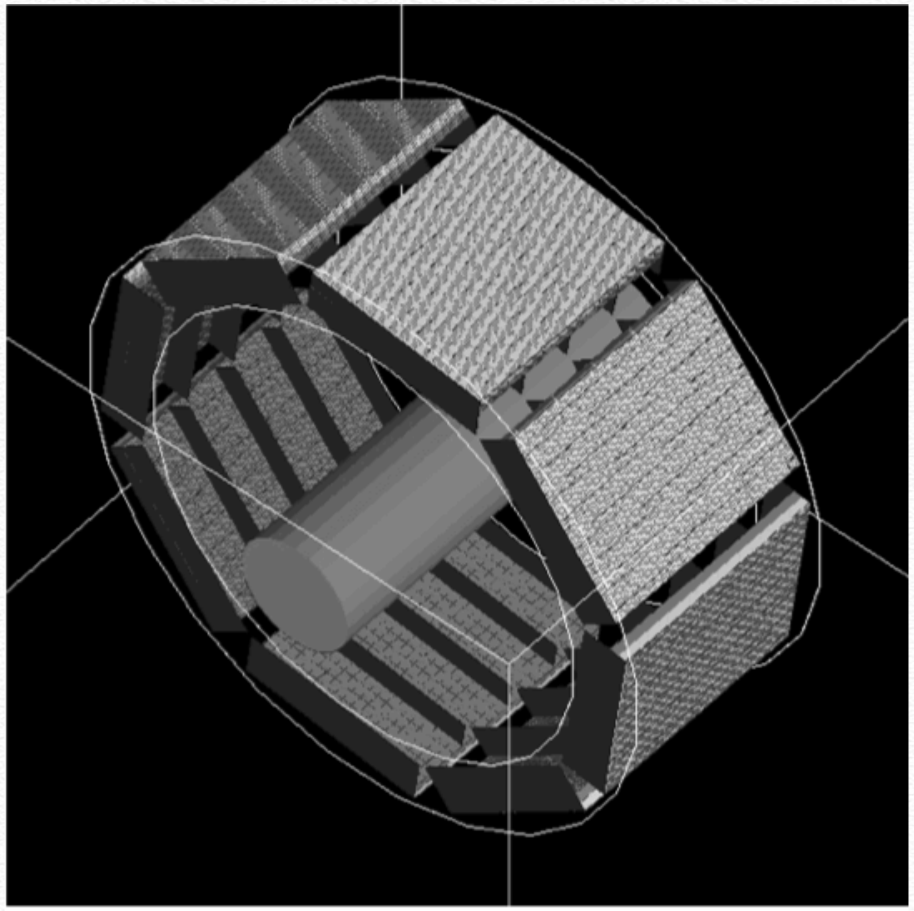


Figure 4. Illustration of the PET benchmark set-up.

PET Example

- We obviously don't have that long just to run the example, so let's build our own simplified PET system that we can play with
- Download file myPET.zip from www.lip.pt/~alex and unzip it inside your working folder
- To keep things organised, we are going to use different macro files:
  - main.mac
  - geometry_step?.Mac
  - physics.mac
  - digitizer.mac
  - source.mac
  - output.mac
  - vis.mac

How to run Gate

- Instead of installing yet another software, let's use Docker
- You can do this in your own computer as long as you install Docker. Instructions on how to do this for different OSs are here.
- The Gate collaboration provides a docker image with GEANT4 10.6 and Gate 9.0
- To make things easier, download the scripts setup_docker.sh and run_docker.sh from here and copy them inside your working folder (make dure you're in the gate branch!)
- To start the Gate container, open a terminal in your working folder

```
source setup_docker.sh
source run_docker.sh
cd /usershared/myPET
```

- You should now be in the folder we just unzipped

Step-by-step…

- Look inside main.mac — this is where we will control the flow and call the other macros
- You'll notice most commands are commented out. Let's slowly construct the geometry… using files `geometry_step1.mac` through `…step5.mac`
- To run the simulation, type `Gate main.mac`
- Unfortunately the GEANT4 in the Gate image was not compiled with OpenGL support. So we must use VRML2 files for visualisation
  - Uncomment the line that executes `vis.mac` from `main.mac`
  - Run the simulation again, you'll see a new file g4_00, with .wrl termination
  - Open it with `view3dscene`

Step-by-step…

- Look inside main.mac — this is where we will control the flow and call the other macros
- You'll notice most commands are commented out. Let's slowly construct the geometry… using files geometry_step1.mac through …step5.mac
- To run the simulation, type Gate main.mac
- Unfortunately the GEANT4 in the Gate image was not compiled with OpenGL support. So we must use VRML2 files for visualisation
- Then uncomment the use of the remaining macros, analysing the commands inside them
- To have decent statistics run the simulation with at least 1M events (using */gate/application/setTotalNumberOfPrimaries*)
    - Remember to comment out the execution of vis.mac

Analysis

- Open root (in a different terminal) and start a ***new TBrowser*** to navigate the output file
- There are several TTrees inside the .root file:
  - Hits — provides extensive information about **all the hits** in **all the crystals**
  - **Singles** — information for **each** crystal
  - **Coincidences** — information about events where a coincidence was detected
  - OpticalData (we don't use this)

Analysis

- Some suggested plots:
  - Have a look at the time distribution
    Singles->Draw(“time”)
  - Check the position distribution of the source
    Singles->Draw(“sourcePosZ:sourcePosY:sourcePosX”)
    Singles->Draw(“sourcePosZ:sourcePosX**2 + sourcePosY**2”)
  - Check the interactions in the crystals
    Singles->Draw(“energy") — and fit a gaussian, is the resolution as expected?
    Singles->Draw(“globalPosZ:globalPosY:globalPosX")
    Singles->Draw(“globalPosX:globalPosZ")
    Singles->Draw(“globalPosY:globalPosX")
    Singles->Draw(“energy:sqrt(globalPosX**2+globalPosY**2)”,"","colz")
  - **Suggestion:** start an analysis script and add these plots as you go along. It will make your like much easier!
    You can start with exercises.C from the website

Coincidences

- A few interesting plots to try:
  - `Coincidences->Draw(“sourcePosZ1")`
    `Coincidences->Draw(“sourcePosZ2")`
    Do you expect these distributions to be identical?
  - The above distributions also show that not all the source emissions have the same probability of being detected — this is a direct effect of us having only one ring (solid angle effect)
    - <u>How are we even seeing coincidences from |source_Z| > 10 mm?</u>
  - Compare this distribution with `sourcePosZ` in the `Singles` TTree
  - Have a look at the photon travel time. How does that compare with our “coincidence window”?
    `Coincidences->Draw(“(time2-time1)*1.e12")` — photon travel time difference (in picoseconds)
  - `Coincidences->Draw(“rsectorID2:rsectorID1","","colz")`
    Photons are detected in approximately opposite modules (as expected) but not exactly — **Why?**

Coincidences

- A few interesting plots to try:
  - Coincidences->Draw("sourcePosZ1")
    Coincidences->Draw("sourcePosZ2")
    Do you expect these distributions to be identical?
  - The above distributions also show that not all the source emissions have the same probability of being detected — this is a direct effect of us having only one ring (solid angle effect)
    - How are we even seeing coincidences from |source_Z| > 10 mm?
  - Compare this distribution with sourcePosZ in the Singles TTree
  - Have a look at the photon travel time. How does that compare with our "coincidence window"?
    Coincidences->Draw("(time2-time1)*1.e12") — photon travel time difference (in picoseconds)
  - Coincidences->Draw("rsectorID2:rsectorID1","","colz")
    Photons are detected in approximately opposite modules (as expected) but not exactly — **Why?**
  - This is caused by Compton or Rayleigh scattering (in the phantom or crystals). Try adding cuts for no Compton/Rayleigh in the phantom)
    (This is done using the 2nd parameter in the *Draw* function)

Exercises

- Use the information on the Compton scatters in the phantom to get the energy of events with no energy loss in the phantom
  - Compare the energy spectrum of gammas with at least 1 Compton scatter in the phantom with the total — do you think we should adjust our energy window?
- What is the fraction of decays that produced a coincidence?
- Are all coincidences from gammas in the same decay?
- What is the fraction of coincidences with no scatter in the phantom?
- Estimate the distance between the detected gammas in coincidence, with and without phantom scatters
- Estimate the acollinearity angle. Again, check the effect on this angle in the case of no phantom scatters
- Double the radius of the phantom and redo the previous estimates. Also have a look at the coincidence time, has it changed?

A more realistic simulation

- So far we have not used the extra potential offered by GATE
- Instead of running a fixed number of events, let's set a running time and use the activity of our source
- Edit the main.mac macro to have an acquisition interval from 0 to 10 sec. Comment the line to have a fixed number of events
- Set the initial activity of the source as 100k Bq
- With 100k Bq x 10 sec we should still have ~1M events

False coincidences

- Have a look at the absolute time distribution — it now spans between the limits we defined
- Do you expect us to have false coincidences with the current setup?
  - How can you check this?
  - Hint: plot the coincidence time distribution
- Increase the coincidence time to see more false coincidences (e.g. 100 ns or 1 μs)
  - Redo the distance and angle distributions between coincident gammas now that you have false coincidences

A realistic source

- Instead of our ideal source, let's use ^{15}O
- This is a source frequently used in PET
- It decays by positron emission (99.9%), which then annihilates inside the patient
- The half-life of this source is ~2 min
- GATE can realistically simulate the effect of the decay of the source during the exam

A realistic source

- Edit the main.mac macro to use source_O15.mac to define the primaries
- Set the scan duration for 10 min. Set the duration of each time slice as 10 sec
  - GATE calculates the source activity in each time slice
- Use an initial activity of 1000 Bq
- Be patient, this will take some time
  (these are in fact 60 simulations!)
- Analyse the output file to check the decay of the source activity
  - Does it agree with the half-life of the source?

A moving patient

- GATE allows you to add movement to any of the geometry elements
- You can use it to rotate the PET ring and modules
- Or to simulate movement of the patient during the exam (e.g. breathing)
- Types of movement: translation, rotation, orbiting, wobbling (oscillating translation)

A moving patient

- We will make our phantom “wobble” in the x, y and z directions
- We can set the amplitude, period/frequency and initial phase
- For this, use the geometry_step6.mac macro file
- Note that it is important for the source to be attached to the phantom, to move with it
- Set your acquisition interval to 20 sec, in slices of 1 sec
  - GATE repositions the geometry at each time slice
- Increase the initial activity of the source by 10x to increase the statistics
- Can you tell from the data that the phantom/source is moving?