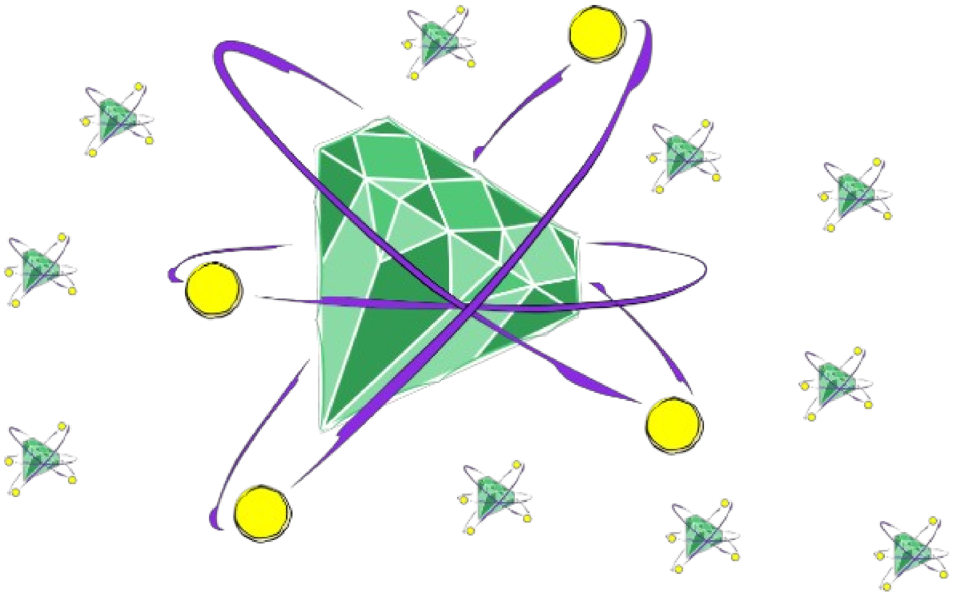
Inserm

CHU BREST CENTRE HOSPITALIER UNIVERSITAIRE

UBO Université de Bretagne Occidentale

LATIM Laboratoire de Traitement de l'Information Médicale
Laboratory of Medical Information Processing

WMLQ 2024
4-7 June 2024, Warsaw Poland



GGEMS

GPU Geant4-based Monte Carlo Simulations

Didier BENOIT, Julien BERT

LaTIM – INSERM UMR 1101

MONTE CARLO
IN
MEDICAL APPLICATIONS

Monte Carlo Simulation in Medical Applications

Radiation Therapy

- Adaptive / on-line treatment planning

External beam radiotherapy

MR-LINAC

Intraoperative radiotherapy

- Complex optimization

Non-coplanar trajectories

Medical Imaging

- Dose monitoring

X-ray interventional radiology

Nuclear medical imaging

- System simulation (generating data)

Monte Carlo Simulation in Medical Applications

Basic Principles

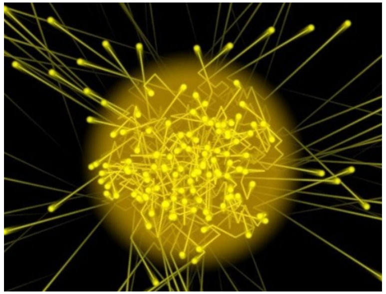
- Determine input variables and range
- Generate random samples
- Run the model for each sample
- Collect and analyze output data

Pros & Cons

- Robust
- Understanding probabilities
- Accurate
- Computational intensity
- Resource requirements
- No real time

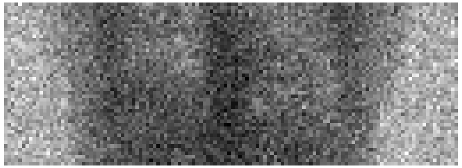
Range :

- Number 1 → 6

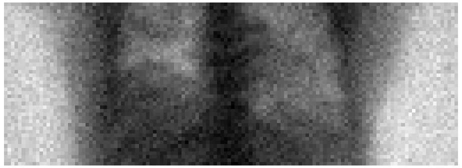


Range :

- Position
- Angle
- Energy



10^6 photons



100×10^6 photons

Monte Carlo Simulation Softwares

Most commonly used softwares in the field of medical imaging

GEANT4
A SIMULATION TOOLKIT

Very generic code for particle physics
- All particles
- Lot of physical processes
- Complex geometries

Multithreading
Too generic for medical application
Only CPU

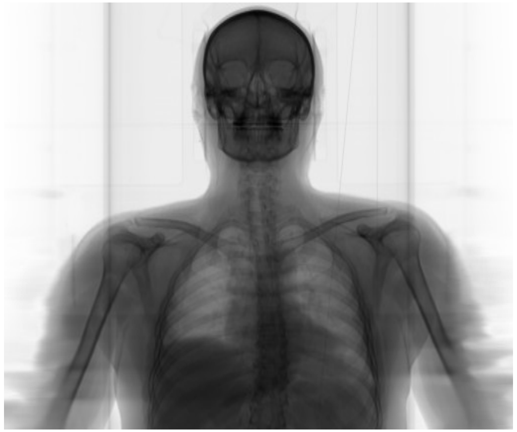
GATE

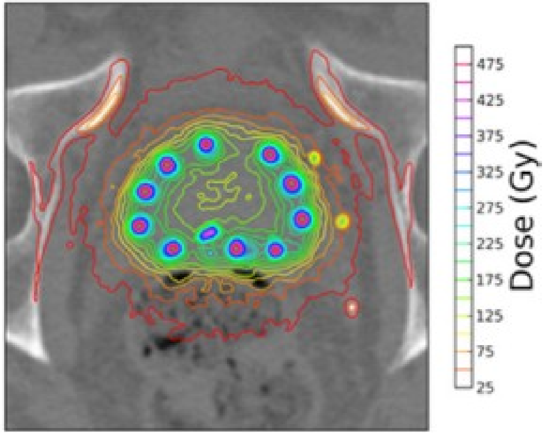
Based on Geant4
Dedicated for medical imaging and radiotherapy simulations
Multithreading
Only CPU

Slow simulation time for realistic system → No real time
Large resource requirement (cluster)

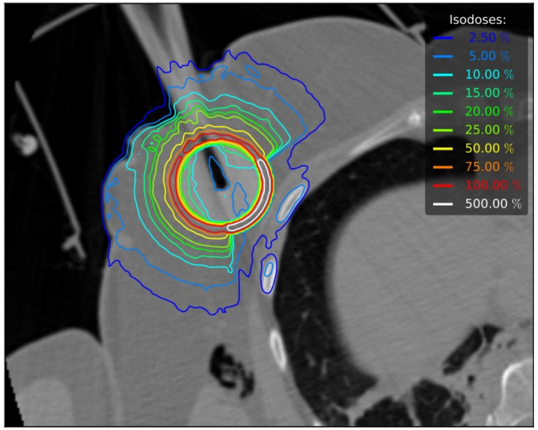
Monte Carlo Examples in Medical Applications

Radiography
12x10^9 photons
24 days
1 CPU core





Brachytherapy
15 hours
1 CPU core



IORT
20 hours
1 CPU core

GGEMS Motivations

- Fast simulation

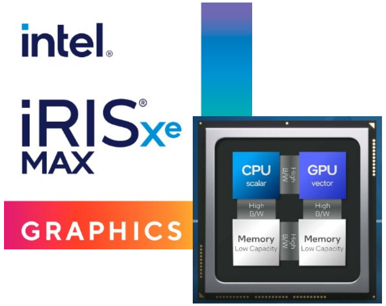
- Multi-OS
- Multi-architecture
  - ✓ CPU + GPU
- C++ and Python
- Easy to maintain

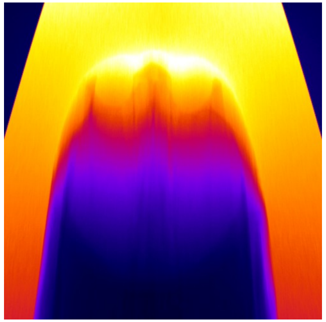
|| Programmation

- Few particles (γ, e-, e+)
- Fast physic models (G4)

|| Physics

- Dedicated to medical applications
  - ✓ Data generation for reconstruction
  - ✓ Data generation for IA learning
  - ✓ Dosimetry applications





HIGH PERFORMANCE COMPUTING (C/C++)

High Performance Computing (C/C++)

CPU

POSIX Threads (Linux)

+

Intel SIMD
SSE : 128-bits register
AVX : 128-bits register
AVX2 : 256-bits register
AVX512 : 512-bits register

Only CPU !!!

GPGPU

Only GPU !!!
Only NVIDIA Graphic Card !!!

CPU + GPGPU

Multi-OS

CPU (multi-threads)
- AMD
- INTEL

GPU
- NVIDIA
- Intel
- AMD

OpenCL Introduction

OpenCL
Hardware Structure

Platform
↓
Device

OpenCL
Software Structure

Context
↓
Kernel

→

Intel Processor

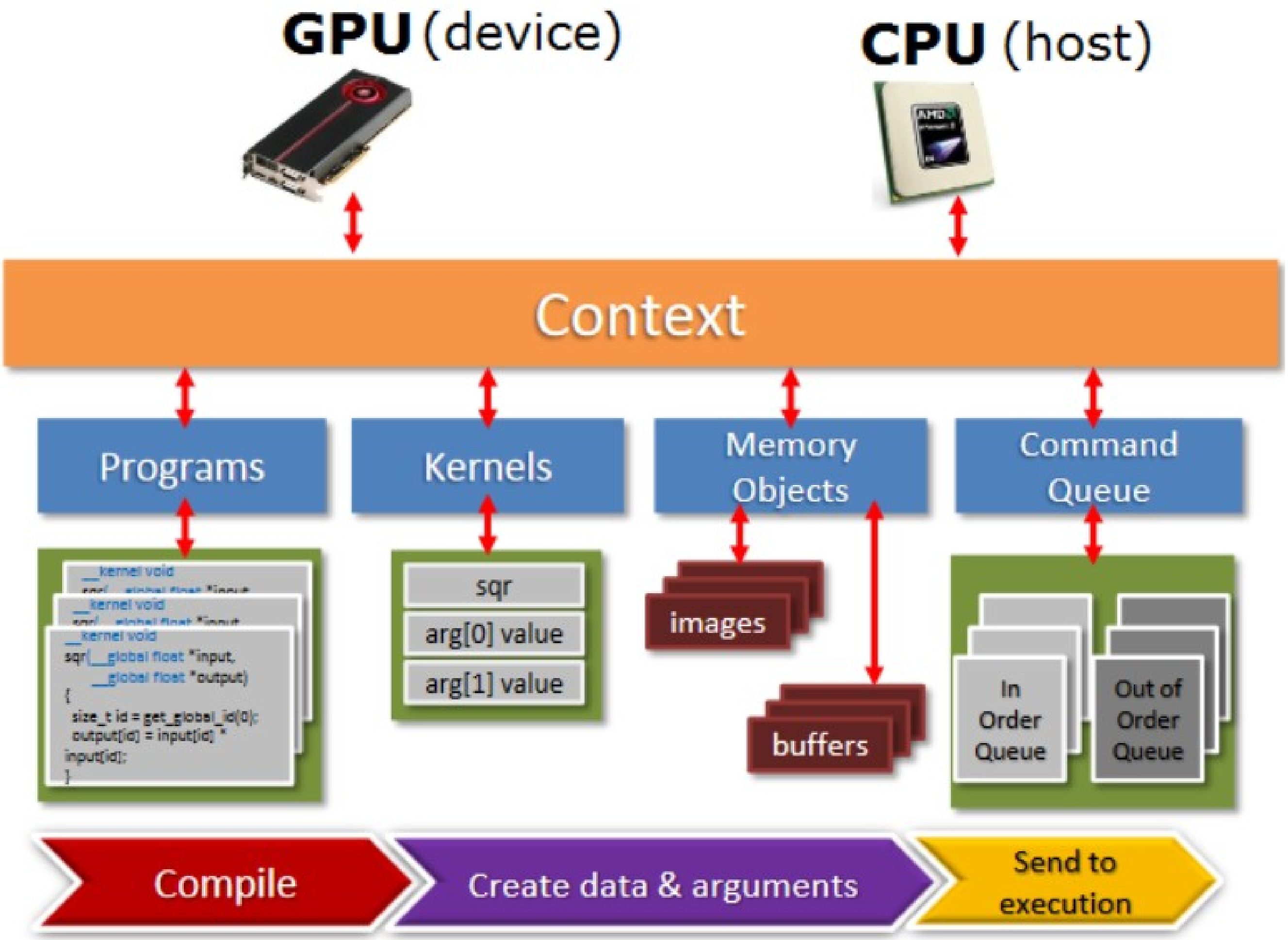
NVIDIA Cards

→

Platforms
- Intel
- NVIDIA

Devices
- I5 Core CPU
- IRIS GPU
- RTX 4090
- RTX 4070

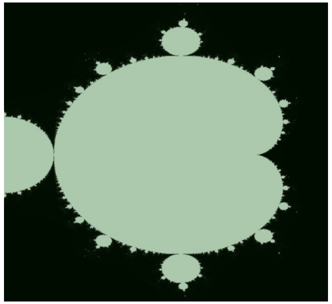
OpenCL Program Flow



D. Benoit and J. Bert, LaTIM – INSERM UMR 1101, Brest, France

OpenCL Comparison Performance

SIMD Benchmark from https://github.com/PolarNick239/FPGABenchmarks/tree/master/benchmarks/mandelbrot



I modified benchmark to implement OpenCL, CUDA and AVX-512

Image results : 16000x16000 pixels

Computer 1

32,8 s	**CPU**	**16 threads**
16,3 s	CPU + SSE	16 threads
7,4 s	CPU + AVX	16 threads
3,8 s	CPU + AVX512	16 threads
5,7 s	CUDA GPU 0	
0,9 s	**CUDA GPU 1**	
5,9 s	OpenCL GPU 0	
0,7 s	**OpenCL GPU 1**	
2,1 s	**OpenCL CPU**	**16 threads**

CPU : Intel Xeon W-2245 @ 3,9 GHz
GPU 0 : Quadro P400
GPU 1 : NVIDIA RTX 4000 Ada

Computer 2

30,3 s	**CPU**	**16 threads**
15,2 s	CPU + SSE	16 threads
8,3 s	CPU + AVX	16 threads
3,1 s	**CUDA GPU 1**	
4,6 s	OpenCL GPU 0	
2,0 s	**OpenCL GPU 1**	
6,1 s	**OpenCL CPU**	**16 threads**

CPU : 12th Gen Intel i5-12500H
GPU 0 : Intel Iris Xe Graphics
GPU 1 : NVIDIA RTX 4060 Laptop

GGEMS SOFTWARE

GGEMS Software

Fast simulation platform for imaging application and particle therapy

Advanced Monte Carlo simulation platform using the OpenCL for CPU/GPU.

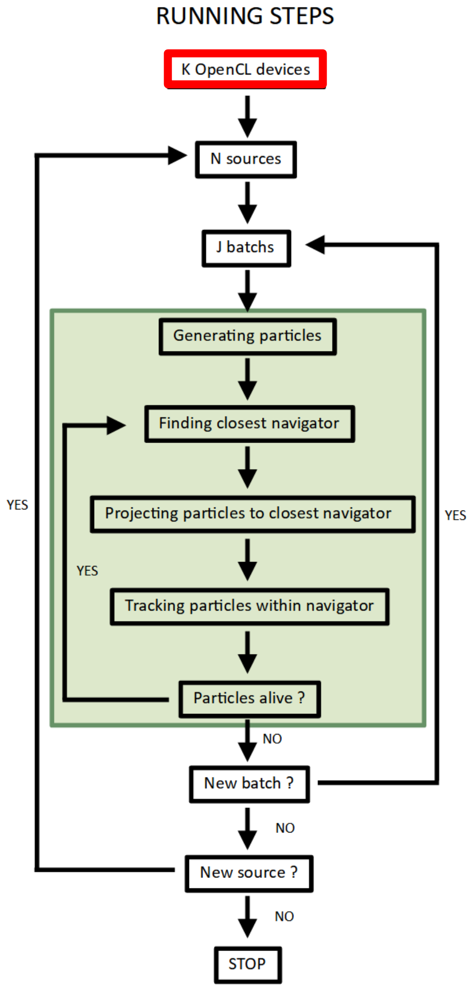
C++ and Python command interface

Not a generic platform as Geant4 or GATE

Main GGEMS features:

- Multithreaded CPU + multi GPU approach
- Physic models from Geant4, and particle tracking (gamma)
- Particle source (cone-beam)
- Particle navigation in voxelized volume
- OpenGL visualization

Multi-Architecture GPU/CPU

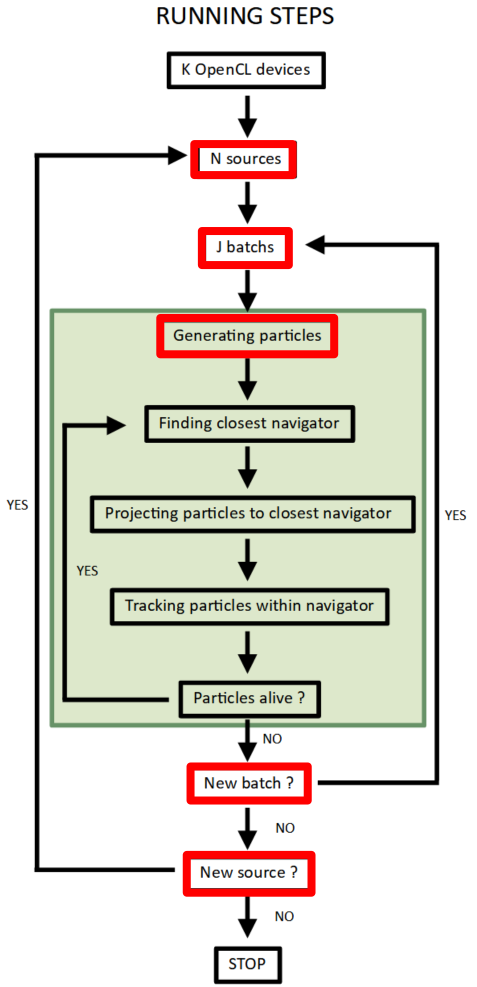


Select OpenCL device(s)
All configurations possible

```python
from ggems import *
opencl_manager = GGEMSOpenCLManager()

opencl_manager.set_device_index(0) # Activate device id 0
opencl_manager.set_device_index(1) # Activate device id 1

opencl_manager.set_device_to_activate('0;1') # Activate devices 0 and 1

opencl_manager.set_device_to_activate('gpu', 'nvidia') # Activate all NVIDIA GPU only
opencl_manager.set_device_to_activate('gpu', 'intel')  # Activate all Intel GPU only
opencl_manager.set_device_to_activate('gpu', 'amd')    # Activate all AMD GPU only
opencl_manager.set_device_to_activate('cpu')           # Activate cpu only
opencl_manager.set_device_to_activate('all')           # Activate all found devices

opencl_manager.set_device_balancing('0.5;0.5')         # 2 devices, 50 % /50 %

opencl_manager.clean()
exit()
```


D. Benoit and J. Bert, LaTIM – INSERM UMR 1101, Brest, France

Particle Source

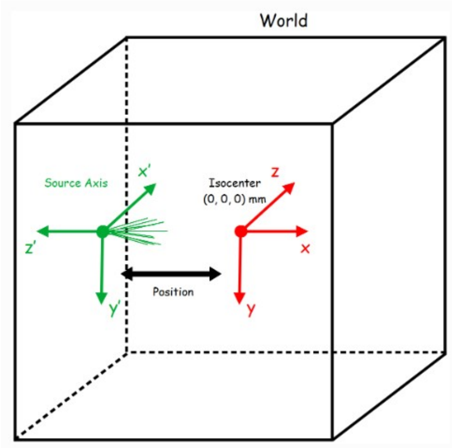


Gamma particle only (for moment)

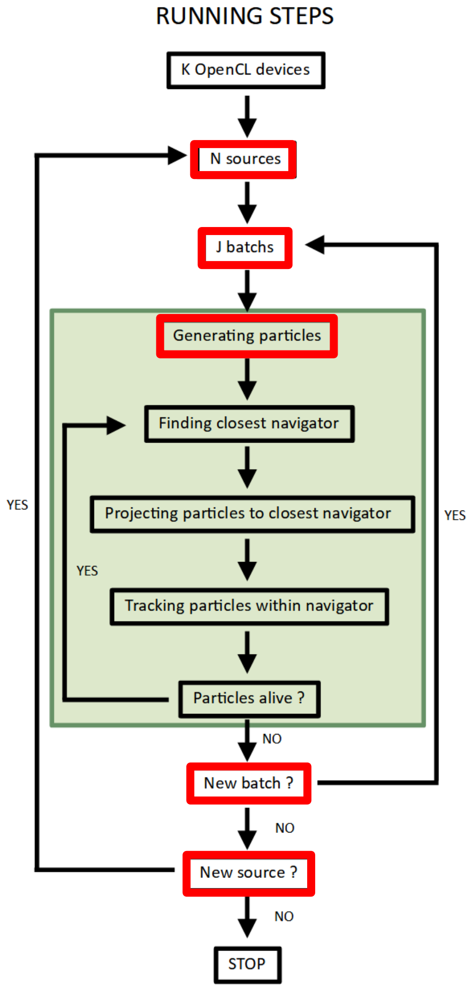
External cone/fan beam shape

Mono or Polychromatic

```
point_source = GGEMSXRaySource('my_source')
point_source.set_source_particle_type('gamma')
point_source.set_number_of_particles(10e9)
point_source.set_position(-595.0, 0.0, 0.0, 'mm')
point_source.set_rotation(0.0, 0.0, 0.0, 'deg')
point_source.set_beam_aperture(12.5, 'deg')
point_source.set_focal_spot_size(0.0, 0.0, 0.0, 'mm')
point_source.set_polyenergy('data/spectrum_120kVp_2mmAl.dat')
```



Particle Source



Many defined sources possible

10^6 primaries simultaneously tracked

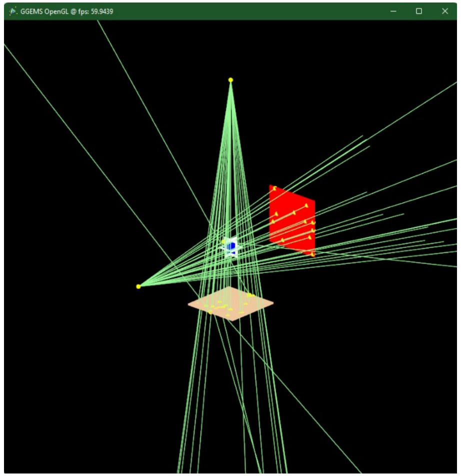
Example :

10^9 primaries → 10^3 batches

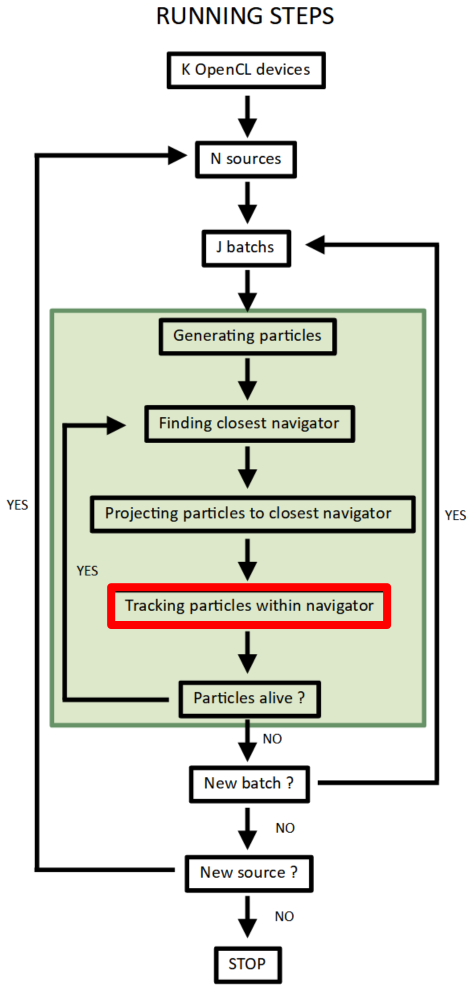
2 OpenCL devices → 500 batches / device

For each primary → A random generator number

```
typedef struct GGEMSPrimaryParticles_t
{
  GGint particle_tracking_id; /*!< Particle id for tracking */

  GGfloat E_[MAXIMUM_PARTICLES]; /*!< Energies of particles */
  GGfloat dx_[MAXIMUM_PARTICLES]; /*!< Direction of the particle in x */
  GGfloat dy_[MAXIMUM_PARTICLES]; /*!< Direction of the particle in y */
  GGfloat dz_[MAXIMUM_PARTICLES]; /*!< Direction of the particle in z */
  GGfloat px_[MAXIMUM_PARTICLES]; /*!< Position of the particle in x */
  GGfloat py_[MAXIMUM_PARTICLES]; /*!< Position of the particle in y */
  GGfloat pz_[MAXIMUM_PARTICLES]; /*!< Position of the particle in z */

  GGint E_index_[MAXIMUM_PARTICLES]; /*!< Energy index within CS and Mat tables */
  GGint solid_id_[MAXIMUM_PARTICLES]; /*!< current solid crossed by the particle */

  GGfloat particle_solid_distance_[MAXIMUM_PARTICLES]; /*!< Distance from previous position to next position */
  GGfloat next_interaction_distance_[MAXIMUM_PARTICLES]; /*!< Distance to the next interaction */
  GGchar next_discrete_process_[MAXIMUM_PARTICLES]; /*!< Next process */

  GGchar status_[MAXIMUM_PARTICLES]; /*!< Status of the particle */
  GGchar level_[MAXIMUM_PARTICLES]; /*!< Level of the particle */
  GGchar pname_[MAXIMUM_PARTICLES]; /*!< particle name (photon, electron, etc) */
  // …
} GGEMSPrimaryParticles;
```



Photon Physical Processes



From Geant4 10.6, gamma processes implemented :

- Compton scattering
- Photoelectric effect
- Rayleigh scattering

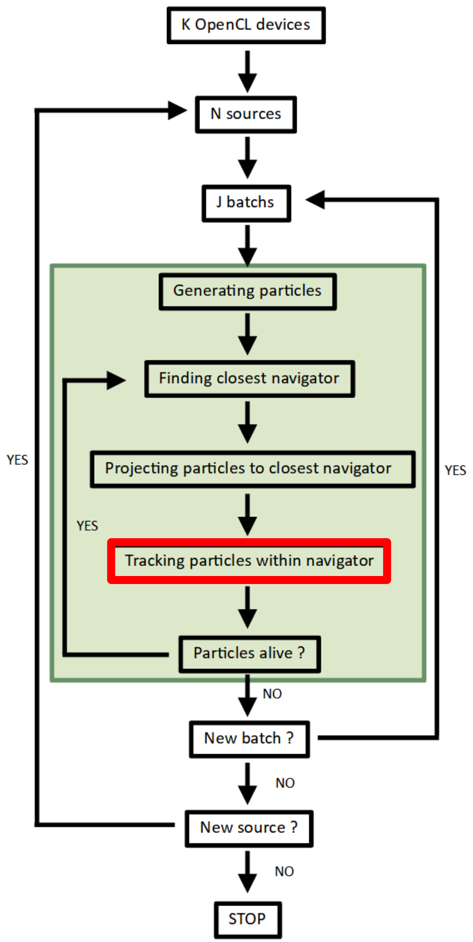
All cross-section tables computed before simulation

Cross-section tables are used during particle tracking

```
processes_manager = GGEMSProcessesManager()
range_cuts_manager = GGEMSRangeCutsManager()

processes_manager.add_process('Compton', 'gamma', 'all')
processes_manager.add_process('Photoelectric', 'gamma', 'all')
processes_manager.add_process('Rayleigh', 'gamma', 'all')

processes_manager.set_cross_section_table_number_of_bins(220)
processes_manager.set_cross_section_table_energy_min(1.0, 'keV')
processes_manager.set_cross_section_table_energy_max(1.0, 'MeV')

range_cuts_manager.set_cut('gamma', 0.1, 'mm', 'all')
```

Photon Physical Processes

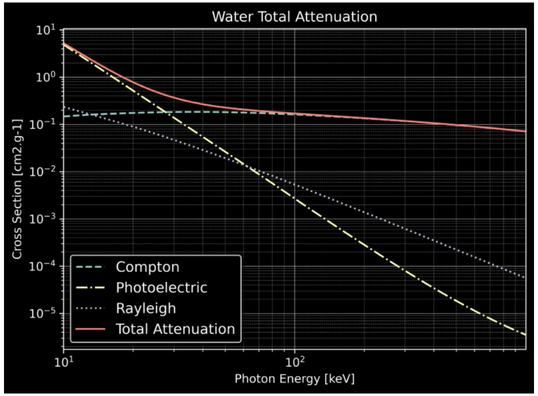
RUNNING STEPS

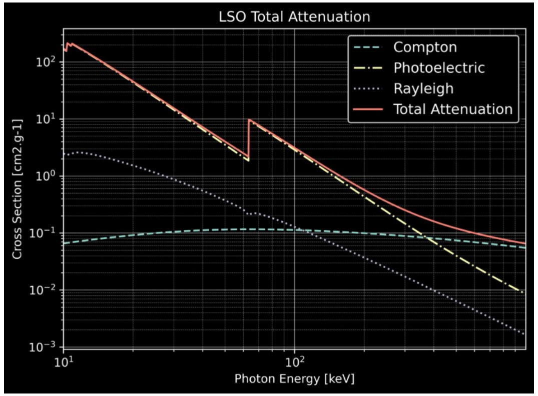


Tools for cross-section tables an total attenuation in GGEMS directory :

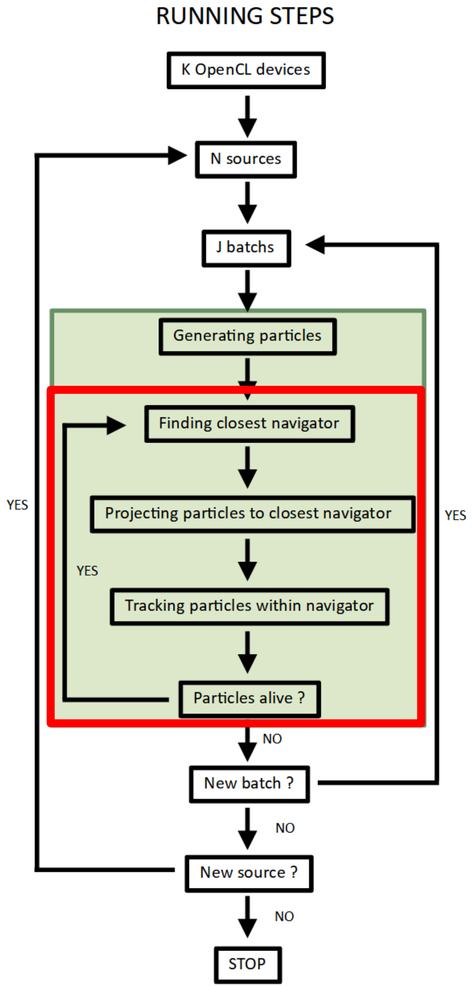
- Example 0 : cross section value
- Example 1 : total attenuation

```
C:\Users\didie\Workspace\ggems\examples\0_Cross_Sections>python cross_sections.py -d 0 -m Water -p Compton -e 0.1
Material: Water
    Density: 1.0  g.cm-3
    Photon energy cut (for 1 mm distance): 2.940556526184082 keV
    Electron energy cut (for 1 mm distance): 351.8771667480469 keV
    Positron energy cut (for 1 mm distance): 342.54461669921875 keV
    Atomic number density: 6.6319030831538785e+22 atoms.cm-3
    Attenuation:  0.17056232690811157  cm-1
    Energy attenuation:  0.025445882230997086  cm-1
At  0.1  MeV, cross section is  0.1622132956981659 cm2.g-1
```



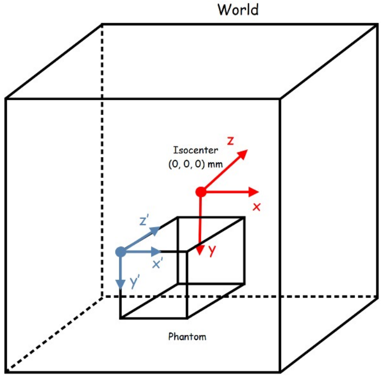


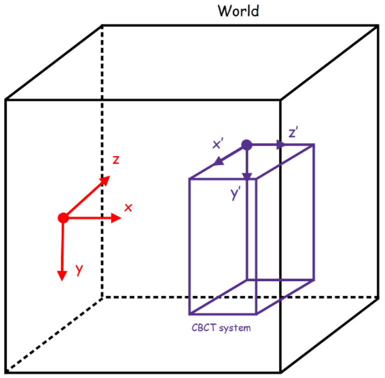
Navigator : Voxelized Volume Definition



All navigators are voxelized
Everything is a navigator
« Material » around navigator is « empty »
Navigator specialized for Phantom and Detector
!!! BEWARE to NAVIGATOR COLLISION !!!

```
phantom = GGEMSVoxelizedPhantom('phantom')
phantom.set_phantom('data/phantom.mhd', 'data/range_phantom.txt')
phantom.set_rotation(0.0, 0.0, 0.0, 'deg')
phantom.set_position(0.0, 0.0, 0.0, 'mm')

ct_detector = GGEMSCTSystem('Stellar')
ct_detector.set_ct_type('curved')
ct_detector.set_number_of_modules(1, 46)
ct_detector.set_number_of_detection_elements(64, 16, 1)
ct_detector.set_size_of_detection_elements(0.6, 0.6, 0.6, 'mm')
ct_detector.set_material('GOS')
ct_detector.set_source_detector_distance(1085.6, 'mm')
ct_detector.set_source_isocenter_distance(595.0, 'mm')
ct_detector.set_rotation(0.0, 0.0, 0.0, 'deg')
ct_detector.set_threshold(10.0, 'keV')
ct_detector.save('data/projection')
ct_detector.store_scatter(True)
```

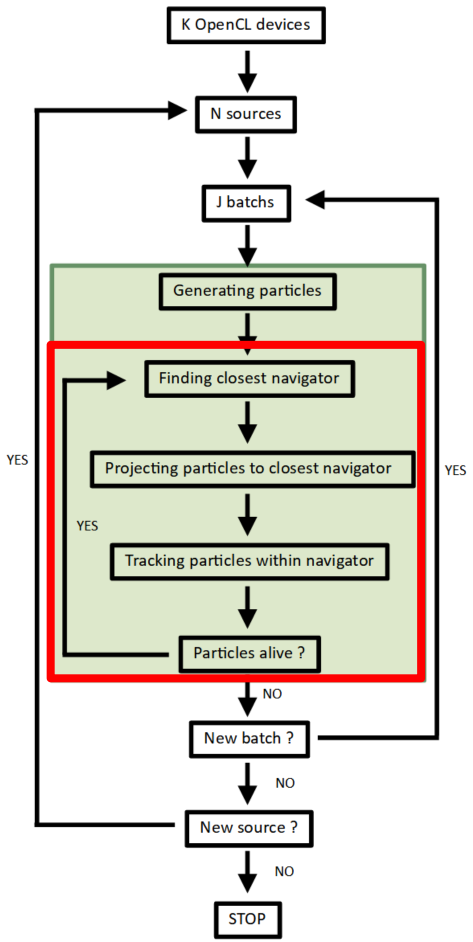


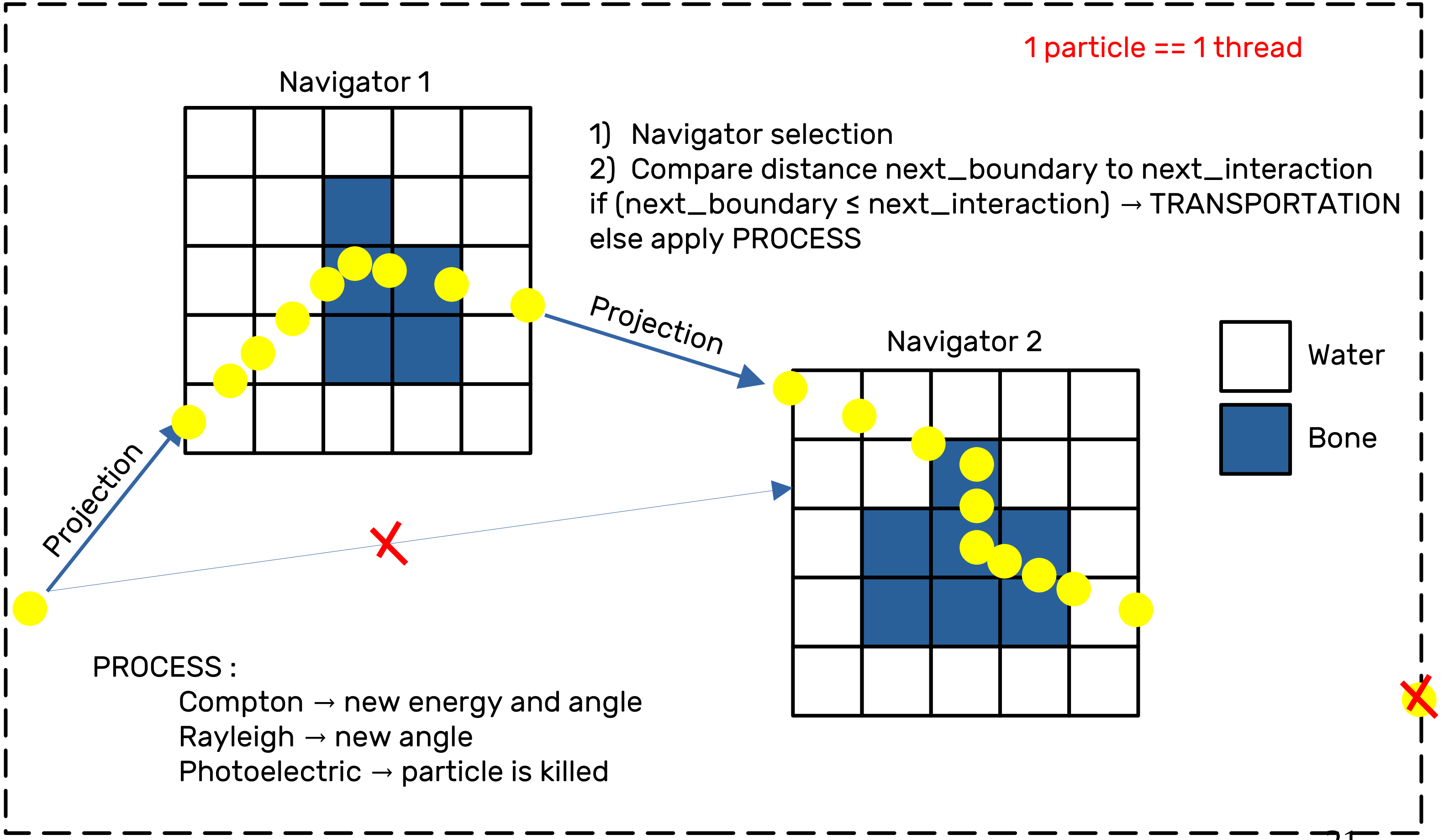


D. Benoit and J. Bert, LaTIM – INSERM UMR 1101, Brest, France

Navigator : Particle Tracking

RUNNING STEPS





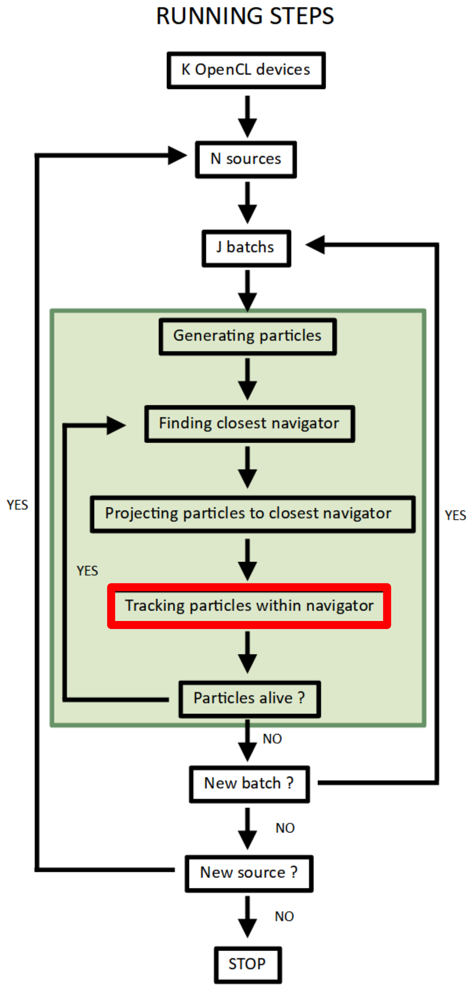
1) Navigator selection
2) Compare distance next_boundary to next_interaction
if (next_boundary ≤ next_interaction) → TRANSPORTATION
else apply PROCESS

PROCESS :

Compton → new energy and angle
Rayleigh → new angle
Photoelectric → particle is killed

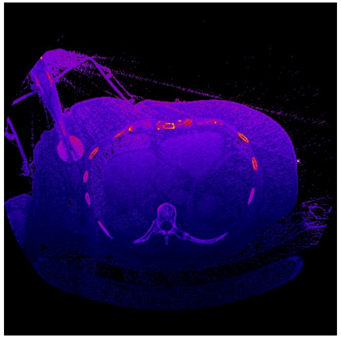
D. Benoit and J. Bert, LaTIM – INSERM UMR 1101, Brest, France

Dosimetry

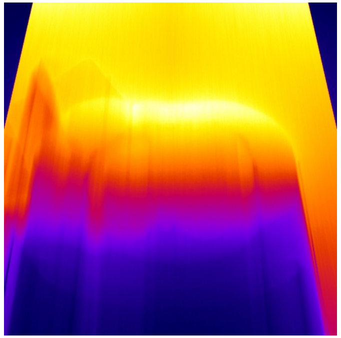


Dosimetry module has to be attached to a volume
TLE (Track Length Estimator) method can be activated to improve statistics

```
dosimetry = GGEMSDosimetryCalculator()
dosimetry.attach_to_navigator('phantom')
dosimetry.set_output_basename('data/dosimetry')
dosimetry.set_dosel_size(0.5, 0.5, 0.5, 'mm')
dosimetry.water_reference(False)
dosimetry.minimum_density(0.1, 'g/cm3')
dosimetry.set_tle(is_tle)

#output
dosimetry.uncertainty(True)
dosimetry.photon_tracking(True)
dosimetry.edep(True)
dosimetry.hit(True)
dosimetry.edep_squared(True)
```

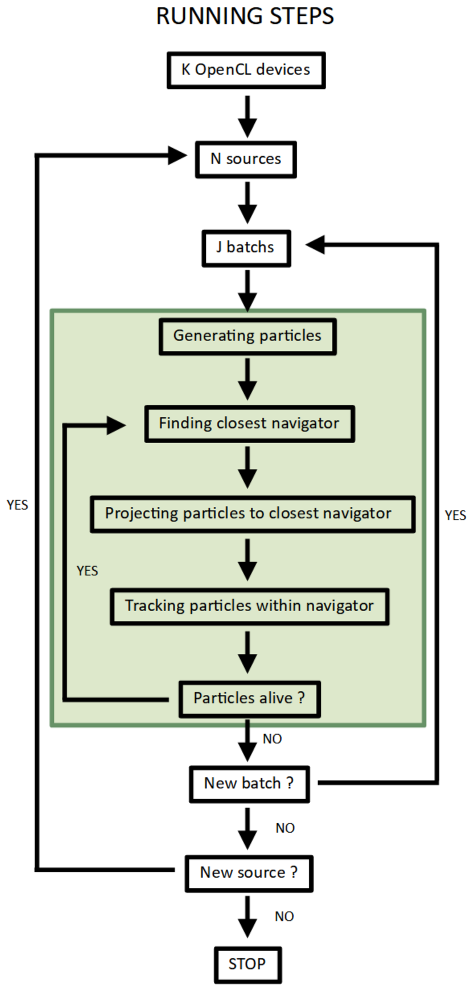


Absorbed Dose



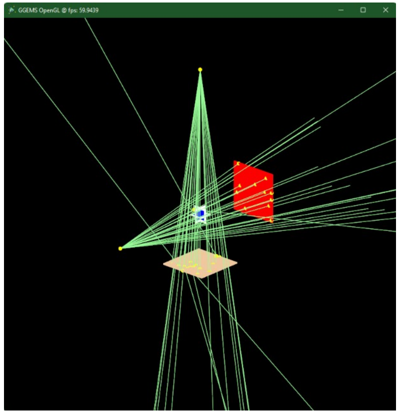
Photon Tracking

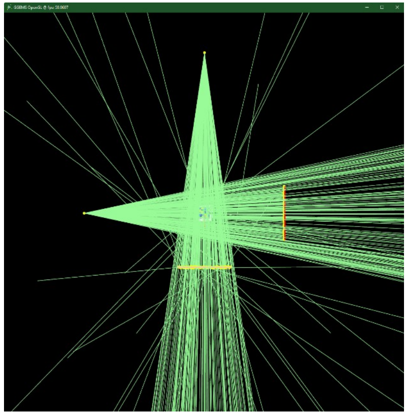
OpenGL Visualization



Simple OpenGL interface to visualize particles and volumes

```
opengl_manager = GGEMSOpenGLManager()
opengl_manager.set_window_dimensions(window_dims[0], window_dims[1])
opengl_manager.set_msaa(msaa)
opengl_manager.set_background_color(window_color)
opengl_manager.set_draw_axis(is_axis)
opengl_manager.set_world_size(3.0, 3.0, 3.0, 'm')
opengl_manager.set_image_output('data/axis')
opengl_manager.set_displayed_particles(number_of_displayed_particles)
opengl_manager.set_particle_color('gamma', 152, 251, 152)
#opengl_manager.set_particle_color('gamma', color_name='red')
opengl_manager.initialize()
```



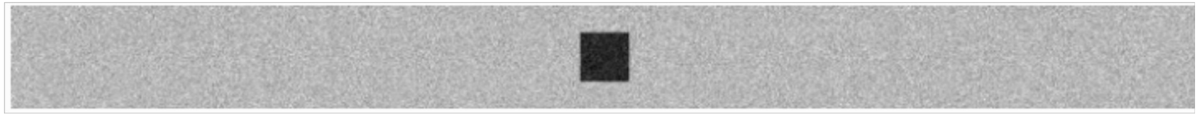


EXAMPLES
&
APPLICATIONS

CT Scanner Application

CT scanner projection

- Curved CT detector
- 10^9 particles generated
- Beam Aperture 12,5°
- Polychromatic source
- ~1900 counts/pixel (in white)
- Output : MHD file



Results

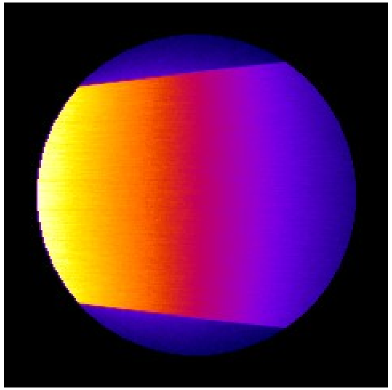
GeForce GTX 1050 Ti	90 s
Quadro P400	360 s
Xeon X-2245 16 threads	**375 s**
1050 Ti (80%) + P400 (20%)	70 s
GeForce RTX 4060	**26 s**

D. Benoit and J. Bert, LaTIM – INSERM UMR 1101, Brest, France

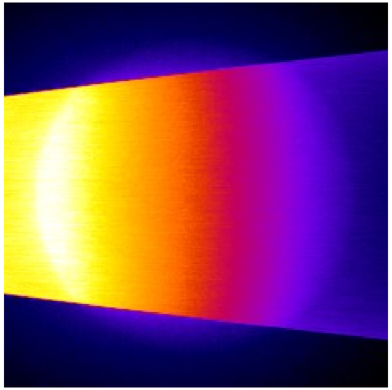
Dosimetry Application

Dosimetry in water cylinder

- $2x10^8$ particles generated
- Beam Aperture 5°
- Polychromatic source
- Output : MHD file



Absorbed Dose



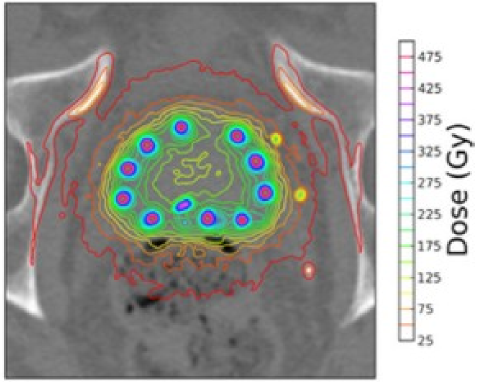
Photon Tracking

Results

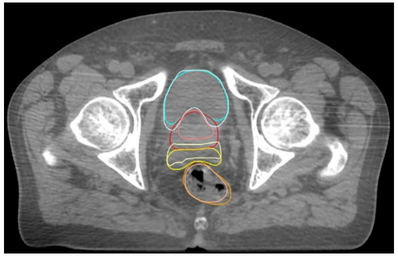
GeForce GTX 1050 Ti	860 s
Quadro P400	2190 s
Xeon X-2245 16 threads	1010 s
GeForce RTX 4060	**200 s**

Other Applications at LaTIM

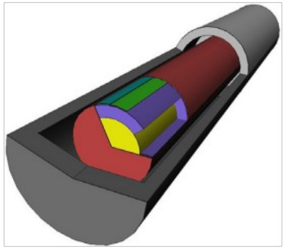
Prostate brachytherapy



(b)



Patient CT

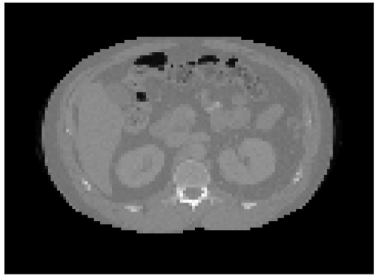


Seed model

MC simulations

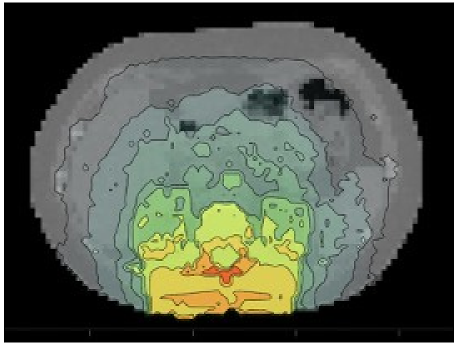
GATE (1 CPU core): **15 h**
GGEMS (GTX 2080 Ti): **10 s**

Interventional radiology



Pre-op. CT

Position
Angulation
Voltage



GATE (1 CPU core)	**5 h**
GGEMS (GTX 1080 Ti)	**6 s**
Average statistical uncertainty	**4.6 %**

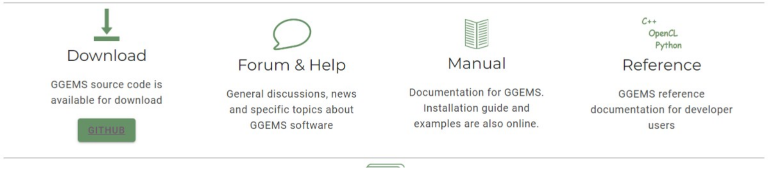
GGEMS Communication & Conclusion

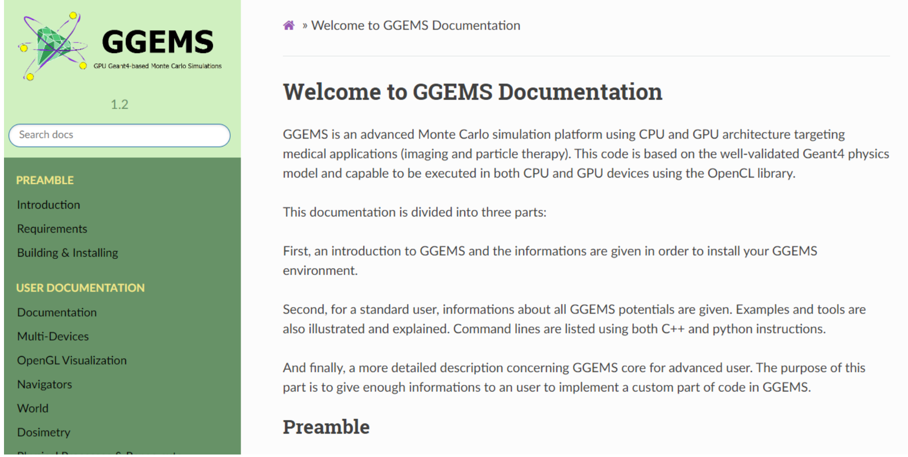
Website : https://ggems.fr

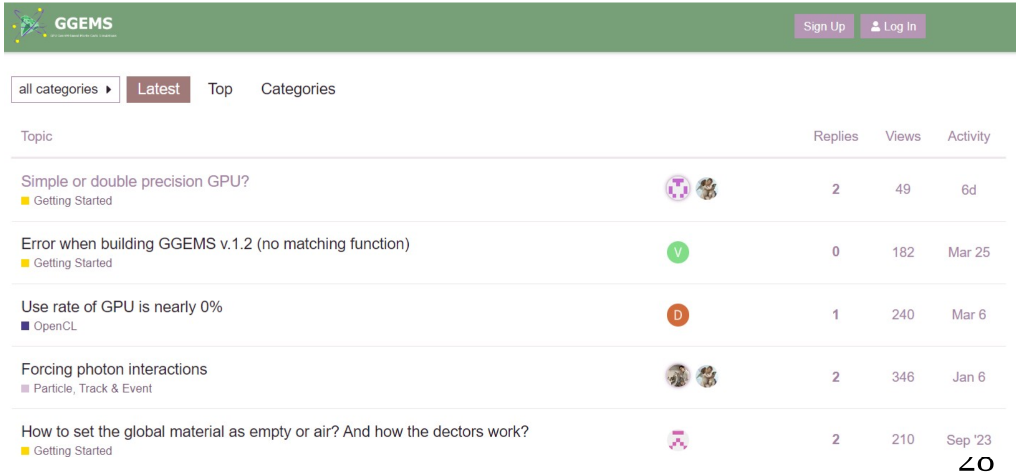
Github : https://github.com/GGEMS/ggems

Documentation : https://doc.ggems.fr/v1.2

Forum : https://forum.ggems.fr







GGEMS Communication & Conclusion

GGEMS is a fast simulation platform for medical application :

- Dosimetry
- Generating data CT/CBCT system

GGEMS is less generic than Geant4 or GATE

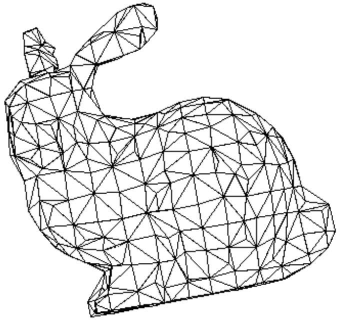
For very realistic → Geant4 or GATE

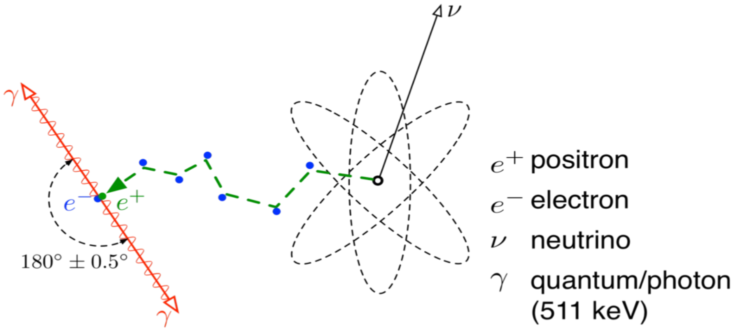
Next release (september 2024) :

- Navigation in meshed phantom
- Generating data for SPECT system
- Simplify installation and Python scripting

Later :

- e+, e- particles and associated physical processes
- Generating data for PET system





D. Benoit and J. Bert, LaTIM – INSERM UMR 1101, Brest, France