GATE status and future perspectives

David Sarrut, on behalf of the collaboration



Current release is 9.2 – April 2022

- Mostly consolidation of 9.1
- With Geant4 11
 - Gate is only evaluated with this Geant4 version
 - May work with previous, but no guarantee (and not maintained)
- More than 70 contributors (since 2012 only)
- Automated compilation check on Linux + OSX
- 20+ benchmarks
- Docker + VirtualMachine

https://github.com/OpenGATE/Gate

Future developments

Introducing ...



Python GATE*!

Goal

A (future) new GATE version

- Macro files are replaced by Python scripts
- Geant4 multithreading
- Simplified installation (one line)
- Numerous enhancements (speed, IA integration, digitizer, etc)

Still ... we keep

- Open-source, open-mind
- OpenGate community
- Collaborative work
- Research oriented in Medical Physics (imaging, dosimetry)

Current GATE serie 9.2 (c++) will be continued

Future GATE serie 10.0 (python/c++) is an additional experimental project

Transition period: two versions.

Interests

- Easy, one line installation
- Writing simulation with Python is much more efficient than with macros
 - Sharing simplified: users-developed modules (e.g. IEC phantoms, detectors, linac etc.)
 - Easy access to whole Python ecosystem (Al!)
- Speed:
 - G4 engine: still same time, no additional time penalty
 - G4 multithread
 - Current GATE overhead: hopefully reduced
- Faster development time, better separation:
 - Python: user parameters, options, I/O etc
 - C++: core processing only
- Advanced features:
 - Dynamic trigger: call user function during runtime (time penalty)

```
# create the simulation
                                                                                                                      # default source for tests
sim = gam.Simulation()
                                                                                                                      source = sim.add_source('Generic', 'mysource')
                                                                                                                      source_energy_mono = 130 * MeV
# · main · options
ui = sim.user_info
                                                                                                                      source.particle = 'proton'
ui.q4 verbose = False
                                                                                                                      source.position.type = 'sphere'
ui.g4_verbose_level = 1
                                                                                                                      source.position.radius = 10 * mm
ui.visu = False
                                                                                                                      source.position.translation = [0, 0, -14 * cm]
                                                                                                                      source.activity = 10000 * Bq
# add a material database
sim.add material database(paths.data / 'GateMaterials.db')
                                                                                                                      source.direction.type = 'momentum'
                                                                                                                      source.direction.momentum = [0, 0, 1]
# units
m·=·gam.g4_units('m')
                                                                                                                      # · cuts
cm = gam.g4_units('cm')
                                                                                                                      c = sim.get_physics_user_info().production_cuts
MeV = gam.g4_units('MeV')
                                                                                                                      c.patient.electron = 3 * mm
Bq = gam.g4_units('Bq')
mm = gam.g4_units('mm')
                                                                                                                      # · add · dose · actor
#··change·world·size
                                                                                                                      dose = sim.add_actor('DoseActor', 'dose')
world = sim.world
                                                                                                                      dose.save = paths.output / 'test009-edep.mhd'
world.size = [1 \cdot * \cdot m, \cdot 1 \cdot * \cdot m, \cdot 1 \cdot * \cdot m]
                                                                                                                      dose.mother = 'patient'
# add a simple fake volume to test hierarchy
                                                                                                                      dose.size = [99, 99, 99]
# translation and rotation like in the Gate macro
                                                                                                                      dose.spacing = [2 * mm, 2 * mm, 2 * mm]
fake = sim.add volume('Box', 'fake')
                                                                                                                      dose.img_coord_system = True
fake.size = [40 * cm, 40 * cm, 40 * cm]
                                                                                                                      dose translation = [2 * mm, 3 * mm, -2 * mm]
fake.material = 'G4_AIR'
                                                                                                                      dose.hit_type = 'random'
fake.color = [1, 0, 1, 1]
fake.rotation = Rotation.from_euler('x', 20, degrees=True).as_matrix()
                                                                                                                      # add stat actor
# image
                                                                                                                      stats = sim.add_actor('SimulationStatisticsActor', 'Stats')
patient = sim.add_volume('Image', 'patient')
                                                                                                                      stats.track_types_flag = True
patient.image = paths.data / 'patient-4mm.mhd'
patient.mother = 'fake'
                                                                                                                      # create G4 objects
patient.material = 'G4_AIR' · · # · material · used · by · default
patient.voxel_materials = [[-2000, -900, 'G4_AIR'],
                                                                                                                      sim.initialize()
-----[-900, -100, 'Lung'],
-----[-100, ·0, ·'G4_ADIPOSE_TISSUE_ICRP'],
                                                                                                                      # · print · info
.....[0, 300, 'G4_TISSUE_SOFT_ICRP'],
                                                                                                                      print(sim.dump volumes())
.....[300, 800, 'G4_B-100_BONE'],
[800, 6000, 'G4_BONE_COMPACT_ICRU']
# or alternatively, from a file (like in Gate)
                                                                                                                      # verbose
vm = gam.read_voxel_materials(paths.gate_data / ' 'patient-HU2mat-v1.txt')
                                                                                                                      sim.apply q4 command('/tracking/verbose 0')
vm[0][0] = -2000
assert vm == patient.voxel_materials
                                                                                                                      # start simulation
patient.voxel_materials = vm
#·write·the·image·of·labels·(None·by·default)
                                                                                                                      sim.start()
patient.dump_label_image = paths.output / 'test009_label.mhd'
```

```
import gam_gate as gam
import gam_g4 as g4
import pathlib
import os
```

```
#-add-a-material-database
sim.add_material_database(paths.data-/-'GateMaterials.db')

#-units
m = gam.g4_units('m')
cm = gam.g4_units('cm')
MeV = gam.g4_units('MeV')
Bq = gam.g4_units('Bq')
mm = gam.g4_units('mm')

#--change-world-size
world-=-sim.world
world.size-=-[1:*-m,-1:*-m,-1:*-m]
```

```
# add a simple fake volume to test hierarchy
# · translation · and · rotation · like · in · the · Gate · macro
fake = sim.add_volume('Box', 'fake')
fake.size = [40 * cm, 40 * cm, 40 * cm]
fake.material = 'G4_AIR'
fake.color = [1, 0, 1, 1]
fake.rotation = Rotation.from_euler('x', 20, degrees=True).as_matrix()
# image
patient = sim.add_volume('Image', 'patient')
patient.image = paths.data / 'patient-4mm.mhd'
patient.mother = 'fake'
patient.material = 'G4_AIR' ** # * material * used * by * default
patient.voxel_materials == [[-2000, -900, 'G4_AIR'],
·····[-900, ·-100, ·'Lung'],
       ·······[-100, ·0, ·'G4_ADIPOSE_TISSUE_ICRP'],
       ······[300, 800, 'G4_B-100_BONE'],
        .....[800, 6000, 'G4_BONE_COMPACT_ICRU']
# or alternatively, from a file (like in Gate)
vm = gam.read_voxel_materials(paths.gate_data // 'patient-HU2mat-v1.txt')
vm[0][0] \cdot = \cdot -2000
assert vm == patient.voxel_materials
patient.voxel_materials = vm
#·write·the·image·of·labels·(None·by·default)
patient.dump_label_image = paths.output // 'test009_label.mhd'
```

```
# create the simulation
sim = gam.Simulation()
# · main · options
ui = sim.user info
ui.q4 verbose = False
ui.g4_verbose_level = 1
ui.visu = False
ui.random_engine = 'MersenneTwister'
ui.random seed = 'auto'
ui.number_of_threads = 5 paths.data / 'GateMaterials.db')
# units
m·=·gam.g4_units('m')
cm = gam.g4_units('cm')
MeV = gam.g4_units('MeV')
Bq = gam.g4_units('Bq')
mm = gam g4_units('mm')
# · · change · world · size
world = sim.world
world.size = [1 * m, 1 * m, 1 * m]
```

```
import gam_gate as gam
import gam_g4 as g4
import pathlib
import os
```

```
# · add · a · material · database
sim.add material database(paths.data / 'GateMaterials.db')
#·units
m·=·gam.g4_units('m')
cm = gam g4_units('cm')
MeV = gam.g4_units('MeV')
Bq = gam.g4_units('Bq')
mm = gam.g4_units('mm')
#··change·world·size
world = sim world
world.size = [1 * m, 1 * m, 1 * m]
# add a simple fake volume to test hierarchy
# · translation · and · rotation · like · in · the · Gate · macro
fake = sim.add_volume('Box', 'fake')
fake.size = [40 * cm, 40 * cm, 40 * cm]
fake.material = 'G4_AIR'
fake.color = [1, \cdot 0, \cdot 1, \cdot 1]
fake.rotation = Rotation.from_euler('x', 20, degrees=True).as_matrix()
# image
patient = sim.add_volume('Image', 'patient')
patient.image = paths.data / 'patient-4mm.mhd'
patient.mother = 'fake'
patient.material = 'G4_AIR' ** # * material * used * by * default
patient.voxel_materials == [[-2000, -900, 'G4_AIR'],
-----[-900, --100, -'Lung'],
       ······[-100, ·0, ·'G4_ADIPOSE_TISSUE_ICRP'],
       ·····[0, ·300, ·'G4_TISSUE_SOFT_ICRP'],
            ·····[300, ·800, ·'G4_B-100_BONE'],
        .....[800, 6000, 'G4_BONE_COMPACT_ICRU']
# or alternatively, from a file (like in Gate)
vm = gam.read_voxel_materials(paths.gate_data // 'patient-HU2mat-v1.txt')
vm[0][0] \cdot = \cdot -2000
assert vm == patient.voxel_materials
patient.voxel_materials = vm
#·write·the·image·of·labels·(None·by·default)
patient.dump_label_image = paths.output // 'test009_label.mhd'
```

```
# create the simulation
sim = gam.Simulation()
# · main · options
ui = sim.user info
ui.g4_verbose = False
ui.g4_verbose_level = 1
ui.visu = False
ui.random_engine = 'MersenneTwister'
ui.random_seed = 'auto'
ui.number_of_threads = 5
sim.add material database(paths.data / 'GateMaterials.db')
# units
m = gam_g4_units('m')
cm = gam.g4_units('cm')
MeV = gam.g4 units('MeV')
Bq = gam.g4_units('Bq')
mm = gam g4_units('mm')
# · · change · world · size
world = sim.world
world.size = [1 * m, 1 * m, 1 * m]
```

```
#·create·the·simulation
                      # add a simple fake volume to test hierarchy
sim = gam.Simulation()
                      # translation and rotation like in the Gate macro
# main options
ui = sim.user_info
                      fake = sim.add volume('Box', 'fake')
ui.g4_verbose = False
ui.g4_verbose_level = 1
                      fake.size = [40 * cm, 40 * cm, 40 * cm]
ui.visu·=·False
                                                                                                                             cm]
                      fake.material = 'G4 AIR'
# add a material database
sim.add material database(pat
                      fake.color = [1, 0, 1, 1]
#·units
                      fake.rotation = Rotation.from euler('x', 20, degrees=True).as matrix()
m·=·gam.g4 units('m')
cm = gam.g4_units('cm')
                                                                                                                            cuts
MeV = gam.g4_units('MeV')
Bq = gam.g4_units('Bq')
                      # image
mm = gam.g4_units('mm')
                      patient = sim.add volume('Image', 'patient')
#··change·world·size
world = sim.world
                      patient.image = paths.data / 'patient-4mm.mhd'
world.size = [1 * m, 1 * m,
                      patient.mother = 'fake'
#·add·a·simple·fake·volume
                      patient.material = 'G4_AIR' · · # · material · used · by · default
# · translation · and · rotation ·
fake = sim.add volume('Box',
                      patient.voxel materials = [[-2000, -900, 'G4 AIR'],
fake size = (40 \cdot * \cdot cm, \cdot 40 \cdot * \cdot c)
fake.material = 'G4 AIR'
                          -----[-900, -100, 'Lung'],
fake.color = [1, 0, 1, 1]
fake.rotation = Rotation.fro
                                             ·····[-100, 0, 'G4_ADIPOSE_TISSUE_ICRP'],
# image
                                                                                                                            Actor', 'Stats')
                                                  [0, 300, 'G4 TISSUE SOFT ICRP'],
patient = sim.add_volume('Ima
patient.image = paths.data /
                                                 .....[300, 800, 'G4_B-100_BONE'],
patient.mother = 'fake'
patient.material = 'G4_AIR'
                                               .....[800, 6000, 'G4_BONE_COMPACT_ICRU']]
patient.voxel_materials == [[
                      # or alternatively, from a file (like in Gate)
                      vm = gam.read_voxel_materials(paths.gate_data / 'patient-HU2mat-v1.txt')
                      vm[0][0] = -2000
# · or · alternatively, · from · a · f
                      assert vm == patient.voxel materials
vm·=·gam.read_voxel_material
vm[0][0] \cdot = -2000
                      patient.voxel materials = vm
assert vm == patient.voxel_ma
patient.voxel_materials = vm
```

```
# · create · the · simulation
                                                                                       # default source for tests
sim = gam.Simulation()
                                                                                        source = sim.add source('Generic', 'mysource')
# · main · options
ui = sim.user_info
ui.g4_verbose = False
                     # default source for tests
ui.g4_verbose_level = 1
ui.visu·=·False
                                                                                                                       14 · * · cm]
                     source = sim.add_source('Generic', 'mysource')
# add a material database
sim.add material database
                     source energy mono = 130 * MeV
# units
m·=·gam.g4_units('m')
                     source.particle = 'proton'
cm·=·gam.g4_units('cm')
                                                                                                                       tion cuts
MeV = gam.g4_units('MeV')
Bq = gam.g4_units('Bq')
                     source.position.type = 'sphere'
mm = gam.g4_units('mm')
#··change·world·size
                     source.position.radius = 10 * mm
world = sim.world
                                                                                                                       ep.mhd'
world.size = [1 * m, 1 * n
                     source.position.translation = [0, 0, -14 * cm]
# add a simple fake volume
# translation and rotation
                     source.activity = 10000 * Bq
fake = sim_add volume('Box
fake.size = [40 * cm, 40
                                                                                                                       * · mm ]
fake.material = 'G4 AIR'
                     source.direction.type = 'momentum'
fake.color = [1, 0, 1, 1]
fake rotation = Rotation.
                     source direction momentum = [0, 0, 1]
# image
                                                                                                                       sticsActor', 'Stats')
patient = sim.add_volume(
patient.image = paths.data
patient.mother = 'fake'
patient.material = 'G4_AIF
                     # · cuts
patient.voxel_materials =
                     c = sim get_physics_user_info().production_cuts
                     c.patient.electron = 3 * mm
# · or · alternatively, · from ·
vm·=·gam.read_voxel_mater.
                                                                                                                       ('0'
vm[0][0] \cdot = \cdot -2000
assert vm == patient.voxel_materials
                                                                                       # start simulation
patient.voxel_materials = vm
#·write·the·image·of·labels·(None·by·default)
```

patient.dump_label_image = paths.output / 'test009_label.mhd'

sim.start()

```
# · create · the · simulation
                                                                                      # · default · source · for · tests
sim = gam.Simulation()
                                                                                       source = sim.add_source('Generic', 'mysource')
# · main · options
                                                                                       source.energy.mono = 130 * MeV
ui = sim.user_info
                                                                                       source.particle = 'proton'
ui.g4_verbose = False
                                                                                       source.position.type = 'sphere'
ui.g4_verbose_le
ui.visu = False
              # add dose actor
# add a materia
sim.add materia
              dose = sim.add_actor('DoseActor', 'dose')
#·units
m·=·gam.g4_units
              dose.save = paths.output / 'test009-edep.mhd'
cm·=·gam.g4_uni
MeV · = · gam • g4_un:
Bq = gam.g4_uni
              dose.mother = 'patient'
mm = gam.g4_unit
              dose.size = [99, 99, 99]
# · · change · world
world = sim wor
world.size = [1
              dose spacing = [2 \times mm, 2 \times mm, 2 \times mm]
# add a simple
# translation a
              dose img coord system = True
fake = sim_add
fake size = \[40]
              dose.translation = [2 * mm, 3 * mm, -2 * mm]
fake.material =
fake.color = 1
              dose.hit_type = 'random'
#·image
                                                                                                                                 Stats')
patient = sim.ad
patient.image =
patient.mother
              # add stat actor
patient.material
patient.voxel_ma
              stats = sim.add_actor('SimulationStatisticsActor', 'Stats')
              stats.track_types_flag = True
# or alternative
vm·=·gam.read
vm[0][0] \cdot = \cdot -2000
assert vm == patient.voxel_materials
                                                                                       #·start·simulation
patient.voxel_materials = vm
#·write·the·image·of·labels·(None·by·default)
```

patient.dump_label_image = paths.output / 'test009_label.mhd'

sim.start()

```
# create the simulation
                                                                                                               # · default · source · for · tests
sim = gam.Simulation()
                                                                                                               source = sim.add_source('Generic', 'mysource')
                                                                                                               source energy mono = 130 * MeV
# · main · options
ui = sim.user_info
                                                                                                               source.particle = 'proton'
ui.g4_verbose = False
                                                                                                               source.position.type = 'sphere'
ui.g4_verbose_level = 1
                                                                                                               source.position.radius = 10 * * mm
ui.visu·=·False
                                                                                                               source.position.translation = [0, 0, -14 \times cm]
                                                                                                               source.activity = 10000 * * Bq
# add a material database
sim.add_material_database(paths.data / 'GateMaterials.db')
                                                                                                               source.direction.type = 'momentum'
                                                                                                               source.direction.momentum = [0, 0, 1]
#·units
m·=·gam.g4 units('m')
                                                                                                               # · cuts
cm·=·gam.g4_units('cm')
                                                                                                               c = sim.get_physics_user_info().production_cuts
MeV = gam.g4_units('MeV')
Bq = gam.g4_units('Bq')
mm = gam.g4_units('mm')
                                     # create G4 objects
#··change·world·size
world = sim_world
world.size = [1 * * m, 1 * m, 1 * m]
                                     sim.initialize()
# add a simple fake volume to test h
# · translation · and · rotation · like · in · t
fake = sim.add_volume('Box', 'fake')
fake.size = [40 * cm, 40 * cm, 40 *
fake.material = 'G4_AIR'
                                     # start simulation
fake.color = [1, 0, 1, 1]
fake.rotation = Rotation.from_euler(
                                     sim.start()
#·image
patient = sim.add_volume('Image', 'p
patient.image = paths.data / 'patien'
patient.mother = 'fake'
patient.material = 'G4_AIR' · · # · mater
patient.voxel_materials = [[-2000, --
-----[-900, --100, 'Lung'],
-----[-100, 0, 'G4_ADIPOSE_TISSUE_ICRP'],
                                                                                                               # · print · info
.....[0, 300, 'G4_TISSUE_SOFT_ICRP'],
                                                                                                               print(sim.dump_volumes())
.....[300, 800, 'G4_B-100_BONE'],
.....[800, 6000, 'G4_BONE_COMPACT_ICRU']
                                                                                                               # · verbose
#·or·alternatively, ·from·a·file·(like·in·Gate)
vm = gam.read voxel materials(paths.gate data / 'patient-HU2mat-v1.txt')
                                                                                                               sim.apply_g4_command('/tracking/verbose 0')
vm[0][0] \cdot = \cdot -2000
assert vm == patient.voxel_materials
                                                                                                               # start simulation
patient.voxel materials = vm
#·write·the·image·of·labels·(None·by·default)
                                                                                                               sim.start()
patient.dump_label_image = paths.output // 'test009_label.mhd'
```

Status

Still not equivalent to current GATE. Estimate: 30%

- Available (among others):
 - Time slicing, basics volumes, voxelized volumes, boolean volumes, repeaters
 - All G4 physics lists, production cuts in volume, spliting/Russian Roulette (partial)
 - Generic source, voxelized source, beta+ source (!), confine, motion source, GAN source
 - DoseActor, PhaseSpaceActor (root), Adder (digitizer), AcceptanceAngle
- Not yet implemented (among others):
 - STL volume
 - Tracking cuts, advanced cuts, Optical, Polarization, DNA,
 - Back to back gamma source, Y90 source
 - Digitizer modules, LET, TLE, ARF, FFD

Estimation: can reach 80% within 1-2 years. Last 5% will be very hard

Warning: "it is hard to make prediction, especially about the future";)

Wanna try?

Set your Python environment (optional, but highly recommended)

pip install gam-gate
gam_gate_tests

See: https://github.com/OpenGATE/gam-gate

Docs https://gam-gate.readthedocs.io (incomplete, only started)

And tests/examples (> 50!):

https://github.com/OpenGATE/gam-gate/tree/master/gam_tests/src

Code example

https://github.com/OpenGATE/gam-gate/blob/master/gam_tests/src/test004_simple.py https://github.com/OpenGATE/gam-gate/blob/master/gam_tests/src/test009_voxels.py

Under the hoods



Folder g4_bindings Geant4 binding from C++ to Python

(expose functions, classes); pybind11

Folder gam_lib

Core classes (running): source, scorers etc



Folder gam_gate

User UI (initialisation)

Some information

- Not all Geant4 fct, classes are exposed to Python (can be extended)
- Object destruction policy C++ vs Python
 - If, C++ pointer, need to keep a point in Python side
- Not possible to create or start 2 G4 simulation in the same script
- Image management via ITK:
 - Accessible via both C++ and Py sides
 - If transfer: need to copy!
- Warning for actors and scorers
 - Need to comply to multithread (need lock on shared variables)
 - Need to comply to multi-run and timed simulation

Python side

- One main object : Simulation
- 4 sub-main concepts: volume source physics actors
- All users parameters like "dictionary" structure
- Stored in user_info object