Heliyon 9 (2023) e20229

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

M-TAG: A modular teaching-aid for Geant4

Liam Carroll [a,b,*], Shirin A. Enger [a,b]

[a] *Medical Physics Unit, Department of Oncology, Faculty of Medicine, McGill University, Montreal, Quebec, Canada*
[b] *Lady Davis Institute for Medical Research, Jewish General Hospital, Montreal, QC, H3T 1E2, Canada*

ARTICLE INFO

Keywords:
Monte Carlo simmulation
Geant4 simulation toolkit
Teaching aid

ABSTRACT

Geant4 is a versatile Monte Carlo radiation transport simulation toolkit with a steep learning curve. This work introduces a user-code called M-TAG (Modular Radiation Teaching-Aid for Geant4), built on top of Geant4. M-TAG is designed to help gradually introduce the Geant4 toolkit to new users.

The goal of Geant4 is to record quantities from the simulated radiation as it is transported through geometries. M-TAG simplifies the inclusion of new geometric elements and detector components in the simulation by including new classes. M-TAG also provides basic validated examples for some common detector development tasks. Geant4 intercom modules, called messenger classes, manage these classes. To validate M-TAG, simulations were performed to calculate the range of positrons in water. One hundred million decays at the center of a water-filled sphere with a radius of 1 m were allowed for fluorine-18, carbon-11, oxygen-15 and gallium-68. These results were compared to literature values. An inexperienced Geant4 user was tasked with creating a simulation model for a plastic scintillator-based detector and conducting basic tests to assess the effectiveness of M-TAG as a teaching tool. The simulation involved calculating the dose to the detector's sensitive volume using a 2x2 cm planar monoenergetic photon source spanning energies from 20 to 100 keV. One billion particles were simulated twice: once with the actual detector geometry and once with the sensitive volume replaced by water. The validity of M-TAG was also verified by computing dose ratios and comparing them with mass-attenuation ratios obtained from NIST XCOM data sets.

The mean positron travel distances were within the distribution of literature values. Simulated positron energy spectra means were within 1.8% of literature means. Simulated dose ratios agreed with literature values within uncertainties.

We have developed and verified a modular Geant4 teaching aid called M-TAG. It was used to introduce a new user to Geant4, who used it to perform further validation simulations.

1. Introduction

Geant4 [1] is an open-source and flexible Monte Carlo toolkit that allows the user to simulate radiation transport and interaction with matter. It is widely used in medical research, with a search on PubMed for "Geant4" showing over 800 results since 2017 [2] in fields such as medical imaging, radiobiology and medical physics. The widespread use of Geant4 is due to its open-source nature,

* Corresponding author.
E-mail address: liam.carroll@mail.mcgill.ca (L. Carroll).

https://doi.org/10.1016/j.heliyon.2023.e20229
Received 5 April 2023; Received in revised form 13 September 2023; Accepted 14 September 2023

Available online 19 September 2023

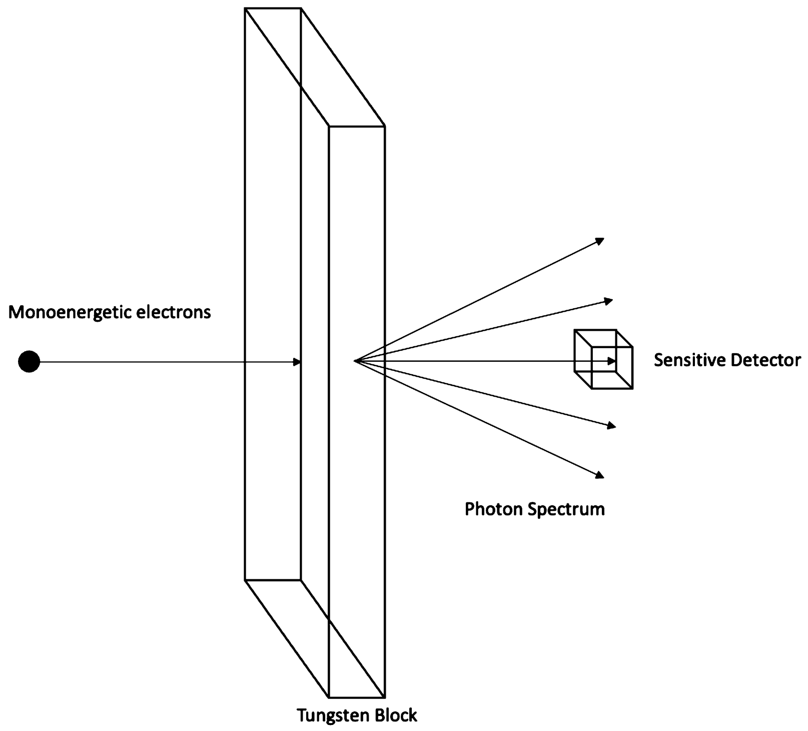


Fig. 1. Geometry of toy Geant4 example showing the electron source, tungsten block, photons and sensitive detector.

the flexibility of its design, its capability to simulate all radiation qualities and a wide variety of physics models. This allows Geant4 to be adapted to many use cases covering a variety of physics fields [1,3,4]. Much of Geant4's flexibility is due to a strict adherence to object-oriented programming philosophies, allowing for easy toolkit extension without compromising older features.

Since its first release in December 1998 [1], there have been many improvements and additions to the toolkit [3,4], where for each release, physics models and cross-sections are improved, features added, and customization options implemented. These features come at the cost of a steep learning curve for new users, with the user manual in portable document format being 469 pages long. Of course, a new user does not need to know everything about Geant4 to start designing simulations, but certain aspects must be well understood.

As mentioned above, Geant4 exists as a set of libraries written in C++, allowing the user to model and simulate specific applications. The most basic Geant4 user-defined simulations consist of a set of mandatory classes that can model the experimental setup in a simulation world describing the geometries of interest, including the material description and mass density for each geometric component for the radiation to traverse through and interact with (DetectorConstruction Class), a source of radiation including its type, beam properties and energy (PrimaryGenerationAction Class), physics models and interactions cross sections to transport the radiation and handle its interaction with matter (PhysicsList class). There are also many optional classes, such as messenger classes, which allow the user to define commands that can control the simulation at run-time and the G4UserAction classes, which allow the user to control and query the simulation at different simulation states. To briefly demonstrate the complexity of Geant4, a simple example to model an experimental setup where monoenergetic electrons hit a tungsten target and generate X-rays is described below. First, in the DetectorConstruction class, all necessary materials are defined and constructed. The simulated geometry is modeled based on a mother–daughter concept where the entire geometry is placed inside a World Volume, which is the top of the hierarchy outside of which no particle is tracked. Each geometric component describes three layers, i.e., G4Solid, G4LogicalVolume and G4PhysicalVolume. The G4Solid describes the shape and size of the element. After this, in the G4LogicalVolume, the component's material, sensitivity, and, if needed, magnetic field are assigned. Finally, in the G4PhysicalVolume, spatial positioning (and rotation) of the component occurs, and the geometrical component is placed/rotated in its mother volume concerning the local coordinate system of the mother volume [5]. The PrimaryGenerator class is used to describe the monoenergetic electron beam. Energy, momentum and source location can all be described using the G4ParticleGun or G4GeneralParticleSource. This hypothetical example is shown in Fig. 1.

A detector element must be added if the user wishes to know the energy spectrum of the resulting X-ray photons at some distance past the tungsten block. While Geant4 produces a tremendous amount of information throughout a simulation, it is up to the user to determine what values/quantities to save or "score". There are many different methods to score values. For simplicity, we will focus on sensitive detectors. The G4VSensitiveDetector class describes what occurs when a particle moves through a linked logical volume. This class allows users to score information from the incoming particle (energy, particle type, path length in the detector, energy deposited, etc.). The scored quantities can then be inputted into histograms created using the G4UserRunAction class or saved

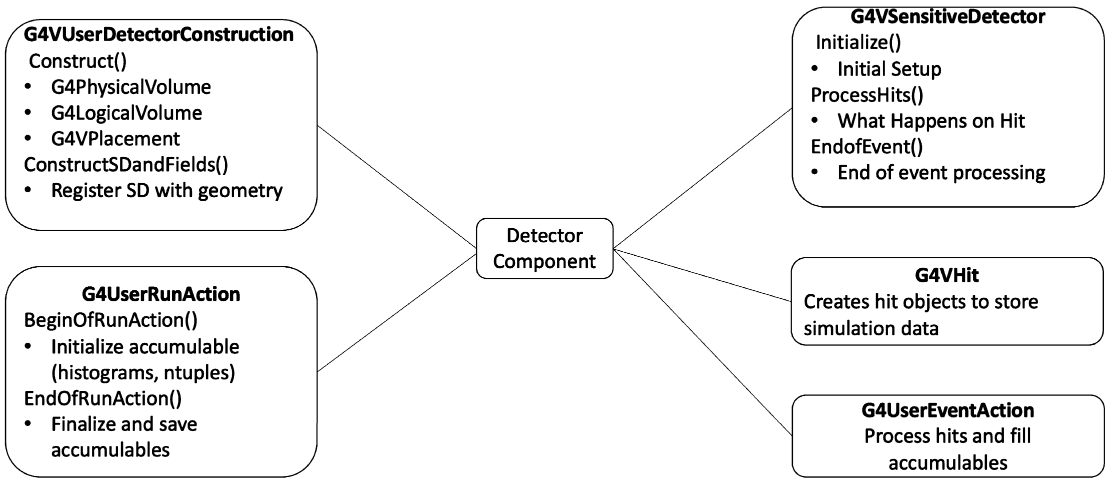


Fig. 2. Classes and functions required to add a basic detector element to a Geant4 simulation.

for further processing using a G4VHit class. The G4VHit class can then be accessed directly in the G4VSensitiveDetector class or the G4UserEventAction class [6]. Geant4 has built-in data structure classes to help save information to a file (histograms, profiles and Ntuples) [7]. These data structures must be initialized in the user runAction. Fig. 2 summarizes the different classes and functions required to initialize a sensitive detector.

Returning to the example, if the user wishes to know the energy spectrum of the generated X-ray photons at some distance behind the tungsten block, they would first have to define a new geometric element. The user could define a 1x1x1 cm^2 air-filled cube to sample the photon spectrum. They would then need to create a G4VSensitiveDetector class and link it to the G4LogicalVolume. The user would initialize a histogram in the G4UserRunAction class to save results.

The above example illustrates the complexity of creating a single detector element in Geant4. This is just one part of the toolkit and demonstrates the complexity of running a radiation transport simulation using Geant4. To use Geant4 confidently, new users must first have a working knowledge of C++ and object-oriented programming; then, they must learn the intricacies of Geant4. With this steep learning curve, it takes new users months to produce useful simulations. Some tools exist to alleviate this problem. Three such tools are GEANT4 Application for Tomographic Emission (GATE) [8], Tool for Particle Simulation (TOPAS) [9] and Geant4-based Architecture for Medical-Applications (GAMOS) [10]. These three tools are similar in that they are intended for users with little to no familiarity with Geant4 and require little coding experience by the user. They function by building on top of the Geant4 toolkit to provide an easy-to-use interface. TOPAS GATE or GAMOS scripting languages are used to build and run the simulation. The scripts call functions designed to control the core Geant4 functionalities and additional functionalities provided by the tool-kits. The users are insulated from the Geant4 source code, which allows for quick deployment or new simulations, provided the tool covers the applications. This illustrates the main advantages and disadvantages of such simulation tools, allowing new users to produce effective simulations quickly without needing to learn the Geant4 toolkit. Suppose the Geant4 wrappers GATE and TOPAS do not cover an application. In that case, the users will either have to wait for the developers to offer a solution or learn Geant4 to code it themselves.

This project aimed to develop a modular teaching aid for Geant4 simulations (M-TAG) to bridge the gap between coding in the Geant4 toolkit and using tools such as GATE, TOPAS and GAMOS. The main goal of implementing M-TAG was to provide a teaching aid for the Geant4 toolkit that allows a new user to learn Geant4 while being exposed to the toolkit one element at a time. Users are required to code in C++ using Geant4 functions and classes. The M-TAG framework provides additional functions and classes that allow users to load pre-built simulation geometries and detector elements without re-compiling their user-code. It is important to note that M-TAG is not a fully featured software package such as GATE, TOPAS and GAMOS.

2. Materials and methods

M-TAG began as a Geant4-based user-code designed to simulate wearable radiation detectors for positron emission tomography (PET) [11–13]. During the development and optimization of this detector, several different detector and phantom designs were modeled and simulated. To streamline the simulation process, the user-code was modified to allow for a quick redefinition of the simulation geometry due to rapid change in the detector design and to enable the design and simulation of other detector models. Further efforts were made to make the code as agnostic as possible concerning simulation geometry to allow for code sharing between lab members. Finally, features were added to the code to use as a teaching aid for new lab members.

The current version of M-TAG has two goals. First, M-TAG allows easy code sharing between students working on similar projects. Second, M-TAG is designed as a teaching aid to introduce the Geant4 toolkit to new users gradually.

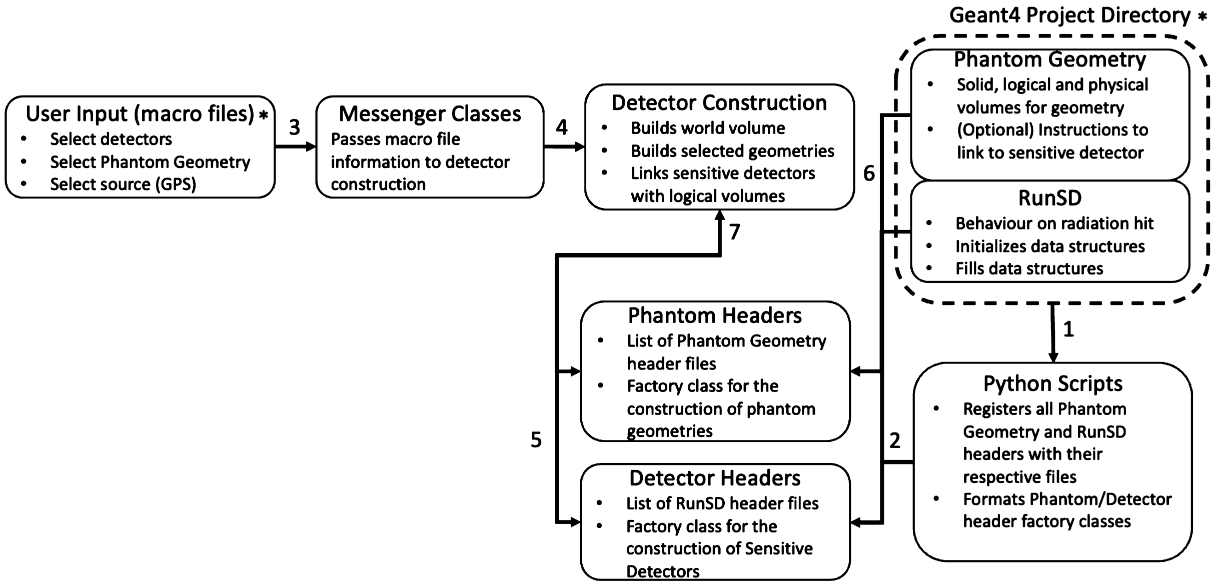

Geant4 Project Directory *
Phantom Geometry
• Solid, logical and physical volumes for geometry
• (Optional) Instructions to link to sensitive detector
RunSD
• Behaviour on radiation hit
• Initializes data structures
• Fills data structures
User Input (macro files) *
• Select detectors
• Select Phantom Geometry
• Select source (GPS)
Messenger Classes
Passes macro file information to detector construction
Detector Construction
• Builds world volume
• Builds selected geometries
• Links sensitive detectors with logical volumes
Phantom Headers
• List of Phantom Geometry header files
• Factory class for the construction of phantom geometries
Detector Headers
• List of RunSD header files
• Factory class for the construction of Sensitive Detectors
Python Scripts
• Registers all Phantom Geometry and RunSD headers with their respective files
• Formats Phantom/Detector header factory classes
1
2
3
4
5
6
7

Fig. 3. User inputs and flow of information through M-TAG classes. * Show mandatory user input. (1) Before project compilation, a Python script registers phantom geometry and RunSD header files in the project directory, and (2) formats the factory classes used to build them. (3) At run time, user macro files select which detector RunSD and phantom geometries will be used. This information is passed to messenger classes, which (4) pass the information to the detector construction class. (5) The detector construction uses the phantom header and detector header factory classes to build the required phantoms/detectors. (6) The factories access the phantom geometry and RunSD classes to build the components, and (7) pass the object back to the detector construction.

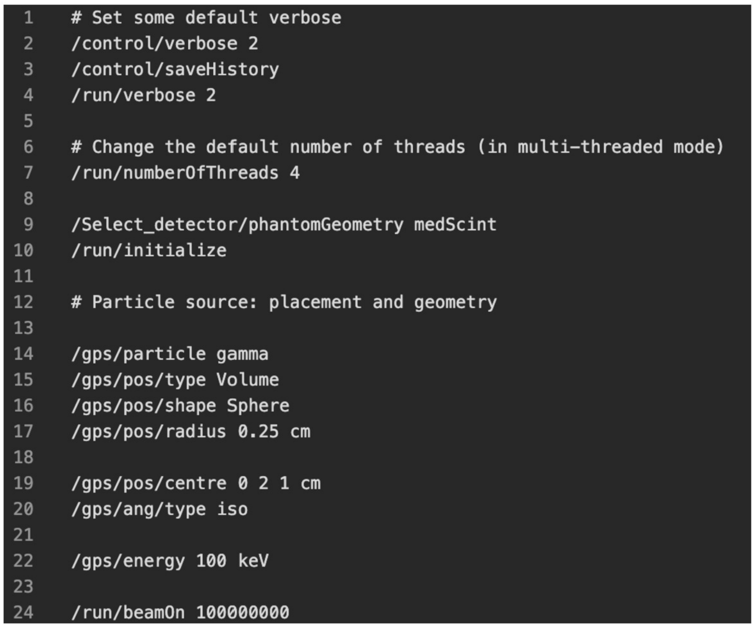
2.1. *Software description*

M-TAG was designed using the same object-oriented programming philosophies as Geant4 to extend its functionality without disabling any base Geant4 features. M-TAG focuses on compartmentalizing portions of the Geant4 toolkit. Specifically, M-TAG facilitates the addition of new geometric components and scoring elements into Geant4, thus modifying the geometry and scoring parameters of the simulation.

A virtual class called *PhantomGeometry* is introduced to be inherited by the user to describe geometric elements in Geant4 simulations. One PhantomGeometry instance can describe as many individual components as desired. This allows multi-part components, such as complex detectors, to be described in one file. The PhantomGeometry class contains two virtual functions. *CreatePhantom* initializes the included geometric elements, and *CleanPhantom* allows the user to modify the geometry after a run. A similar virtual class, called *RunSD*, was written to define detector elements in Geant4. While there are many ways to score information using the Geant4 package, two of the most used methods are through a G4SensitiveDetector or a G4MultiFunctionalDetector. Both of these classes define how an associated logical volume will record information when struck by quanta of radiation. These classes are designed to be inherited and completed by the user. When creating a new instance of RunSD, the user can also inherit the G4SensitiveDetector or G4MultiFunctionalDetector class. This allows the user to use either scoring method. As mentioned above, Geant4 data structures used to save information to files are normally initialized in the userRunAction class. The RunSD class provides additional functions allowing these objects to be created, filled, and saved without needing to write code in the userRunAction class. The M-TAG userRunAction class calls and manages these functions to ensure proper multi-threading behavior. Together, the RunSD and PhantomGeometry classes are called M-TAG modules. Each M-TAG module is saved in a separate file, which can be added to the source code of any M-TAG project.

In M-TAG, the G4VUserDetectorConstruction class tracks the M-TAG modules created by the user and then loads them in the correct order. The user must create their world volume in which all other geometric elements will be added. The user can then apply the base Geant4 functions to create additional geometric components or use the M-TAG functions to add M-TAG modules. M-TAG modules can be added by using either macro files or run-time commands. These commands will be passed to the G4VUserDetector-Construction, which can then build the appropriate M-TAG module objects.

Fig. 3 shows the flow of information through the M-TAG classes. Before M-TAG modules can be used in a simulation, they must be included and compiled in the M-TAG header files. To facilitate this, a Python script has been included that automates the inclusion of any M-TAG modules in the phantom headers and detector headers files. In addition, the Python script writes factory classes that allow for the creation of any M-TAG modules that have been included. In this way, a user must only place the M-TAG module files in their project directory, run the Python script and compile their code. Users then have direct access to the M-TAG modules using run-time commands. Macro files or run-time commands can specify which modules should be included at run-time. These commands are handled by a messenger class that passes the information to the detector construction class. Detector construction can then call the factory classes of the phantom headers and detector headers files to construct the chosen M-TAG modules.



```
1   # Set some default verbose
2   /control/verbose 2
3   /control/saveHistory
4   /run/verbose 2
5
6   # Change the default number of threads (in multi-threaded mode)
7   /run/numberOfThreads 4
8
9   /Select_detector/phantomGeometry medScint
10  /run/initialize
11
12  # Particle source: placement and geometry
13
14  /gps/particle gamma
15  /gps/pos/type Volume
16  /gps/pos/shape Sphere
17  /gps/pos/radius 0.25 cm
18
19  /gps/pos/centre 0 2 1 cm
20  /gps/ang/type iso
21
22  /gps/energy 100 keV
23
24  /run/beamOn 100000000
```

Fig. 4. Basic macro file showing how a user would use a phantomGemoetry and RunSD M-TAG module.

M-TAG uses a validated Geant4 modular physics list, including the Geant4 Penelope [14] reference physics list as well as Geant4's general particles source [3] and radioactive decay. The decay data was obtained from the Evaluated Nuclear Structure Data File [15] (Brookhaven National Laboratory, National Nuclear Data Centre). This physics list allows users to define their source using macro files or run-time commands. Penelope [14] electromagnetic physics models and cross-sections were chosen for the radiation transport calculations due to the accuracy of this model at low energies normally required for medical radiation detector simulations. However, as the physics is described using the G4VModularPhysicsList class from Geant4, the users can easily add and remove physics processes and particle types from the implemented physics processes to tailor the simulation to their application.

2.2. M-TAG usage

To use M-TAG, several specific steps are required. It is important to note that M-TAG does not employ a graphical user interface (GUI). Code is written and modified using a code editor such as VS Code.

Select graphical user interfaces (e.g., OpenGL, Heprep) can be called and utilized within a macrofile to display the constructed geometry or particle simulation process.

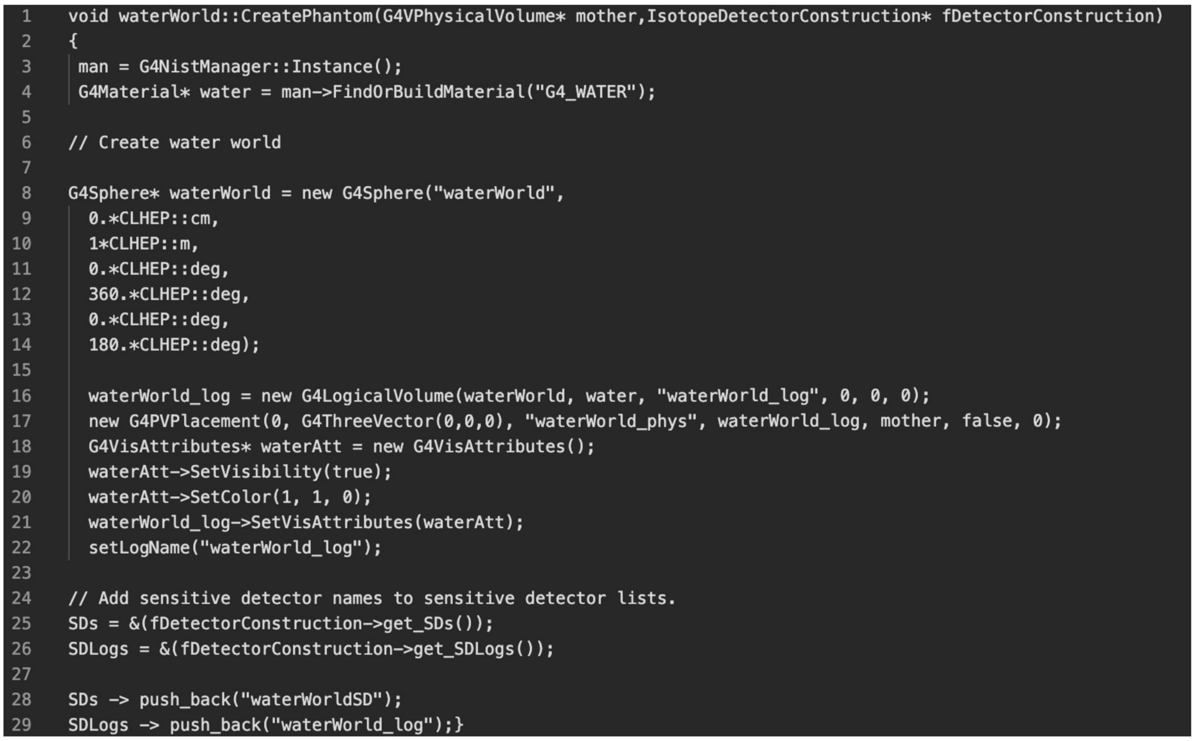
Regarding the simulation setup, each detector construction in M-TAG requires a separate source code (.cc) and a header file (.hh). The source code defines the detector geometry, material, and placement, while the header file contains the necessary classes used in the source code. The sensitive volume of the detector is also defined in the source code and incorporated into the sensitive detector list of M-TAG using a class member access operator.

The simplest way to use M-TAG is to call its functions using a macro file. M-TAG uses the same macro file structure as general Geant4 user-codes, with the addition of additional M-TAG commands. Fig. 4 shows a basic macro file. Line 9 calls the phantomGemoetry class *medScint*, which builds the Hyperscint HS-RP100 scintillating point-detector [16] (MedScint Inc., Quebec City, Canada) and assigns it as a sensitive detector. The rest of the macro file is comprised of native Geant4 commands. Lines 14-22 use the well-documented general particle source commands to describe the simulation particle source. Macrofiles play a critical role in the setup process, enabling the user to:

- Select and call specific detector geometries from the sensitive detector list, allowing easy switching between different detector constructions.
- Toggle visualization on/off for better understanding.
- Set the number of particles to simulate.

This is the most basic M-TAG usage, allowing new users to quickly run simulations using pre-built M-TAG modules. M-TAG macro files use the same syntax as native Geant4 macro files. This allows users to use native macro commands, such as command-based scoring.

Users can modify and tailor the code as per their requirements. By examining existing files and detector constructions, users can comprehend the functionality of each line and subsequently create their custom simulations based on these examples. In this way, M-TAG compliments the available Geant4 documentation and existing examples.



```
1   void waterWorld::CreatePhantom(G4VPhysicalVolume* mother,IsotopeDetectorConstruction* fDetectorConstruction)
2   {
3    man = G4NistManager::Instance();
4    G4Material* water = man->FindOrBuildMaterial("G4_WATER");
5
6   // Create water world
7
8   G4Sphere* waterWorld = new G4Sphere("waterWorld",
9     0.*CLHEP::cm,
10    1*CLHEP::m,
11    0.*CLHEP::deg,
12    360.*CLHEP::deg,
13    0.*CLHEP::deg,
14    180.*CLHEP::deg);
15
16    waterWorld_log = new G4LogicalVolume(waterWorld, water, "waterWorld_log", 0, 0, 0);
17    new G4PVPlacement(0, G4ThreeVector(0,0,0), "waterWorld_phys", waterWorld_log, mother, false, 0);
18    G4VisAttributes* waterAtt = new G4VisAttributes();
19    waterAtt->SetVisibility(true);
20    waterAtt->SetColor(1, 1, 0);
21    waterWorld_log->SetVisAttributes(waterAtt);
22    setLogName("waterWorld_log");
23
24  // Add sensitive detector names to sensitive detector lists.
25  SDs = &(fDetectorConstruction->get_SDs());
26  SDLogs = &(fDetectorConstruction->get_SDLogs());
27
28  SDs -> push_back("waterWorldSD");
29  SDLogs -> push_back("waterWorld_log");}
```

Fig. 5. Simple example showing the usage of the *CreatePhantom* class.

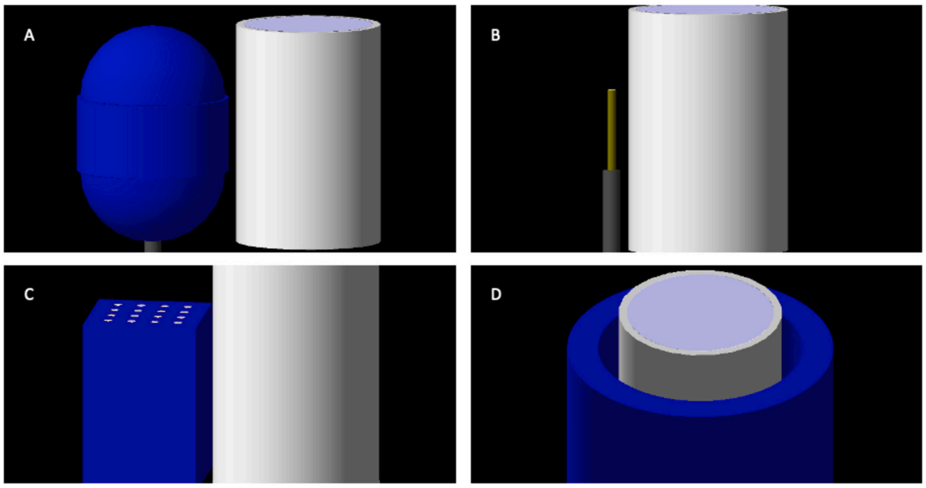


Fig. 6. Simulation geometries created using M-TAG. (A) Scintillating fiber in a plastic bolus. (B) Bare scintillating fiber-based detector. (C) 16-channel scintillating fiber-based detector. (D) Spiral scintillating fiber-based detector.

More advanced users will want to design their own RunSD and phantomGemoetry classes to implement new simulation setups. Phantom Geometry classes must inherit the public *phantomGeometry* class and describe the geometry in the *CreatePhantom* function. Fig. 5 shows a simple example of using the *CreatePhantom* class to create a simple water-filled sphere. Lines 25-29 then assign a sensitive detector called waterWorldSD the logical volume of this phantom. In this way, M-TAG will assign waterWorldSD to the logical volume whenever this phantom is added to a simulation.

Figs. 6 and 7 show multiple examples of simulation worlds created by new Geant4 users using M-TAG all of these examples were coded by users with no prior Geant4 experience and no C++ knowledge. Fig. 6 A-D show different simulated prototypes of a scintillating fiber-based detector. Fig. 7 shows a simulated model of the Hyperscint detector.

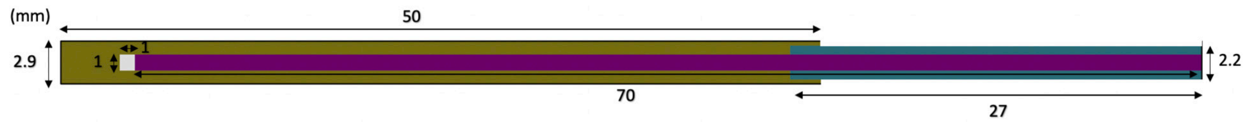


Fig. 7. Cross-section of the simulated hyperscint detector showing sensitive volume in white, optical fiber in magenta, acrylic cover in yellow and polyethylene sheath cover in blue. All dimensions are in mm.

2.3. M-TAG validation

A first validation calculation of M-TAG was performed by calculating the range in water for positrons emitted from four common radioisotopes used in dynamic PET; fluorine-18 (^{18}F), carbon-11 (^{11}C), oxygen-15 (^{15}O) and gallium-68 (^{68}Ga). The resulting positrons were allowed to annihilate. The obtained results were compared with published data. To do this, a sphere with a radius of 1 m was filled with water of unit mass density (1.0 g/cm^3) and a single point source was defined at the center of the simulation. 100 million decay events were simulated. For each primary particle, the positron emission energy and the location of the creation of the annihilation photons were recorded. The location information was used to determine the positron range.

Positrons travel through a medium with a torturous path, undergoing many interactions, including elastic scattering, inelastic collisions and bremsstrahlung emission. As positrons slow down, they eventually annihilate with an electron in the medium to produce two anti-coincident photons. There is the possibility of an intermediate step where a positron will capture an electron from the surrounding media to form a positronium [17]. This process is normally not accounted for in radiation transport simulations; hence, it is not accounted for in Geant4. A positron's range in water depends on its original energy, but there is no exact analytical expression to calculate this. As a result, there have been many simulation studies to calculate average and maximum positron travel ranges through water for common PET radioisotopes [17–20].

The M-TAG code was compared to four alternative methods of calculating the positron range in water. PeneloPET is a Monte Carlo simulation code based on the PENELOPE [14] Monte Carlo code using the Penelope cross-sections [20]. In addition to this simulation, an analytical expression described by Cal-Gonzalez et al. [20] was used to estimate the range of positrons in water. A third comparison is made to the GATE platform [21], an application built on top of the Geant4 toolkit. Lehnert et al. [18] used GATE version 5.0 with the Geant4 standard energy package to simulate positron range from annihilation photon density distributions. Champion and Le Loirec [17] developed a Monte Carlo code that simulates positronium formation and uses it to calculate positron range, taking positronium formation into account.

For validation purposes, simulations were performed with production energy cuts at 0.1 and 1 keV for all radioisotopes. This is much larger than the positronium formation threshold (5.8 eV) [17]. Using an energy cut of 1 keV decreases simulation speed by factor 10, as shown in previous work by our group [11]. Additional validation for M-TAG is presented in section 2.4.

2.4. M-TAG test case

Since M-TAG was developed to facilitate Geant4 instruction, we used it to help a new student learn Geant4. An undergraduate physics student with basic object-oriented programming experience was shown the basics of Geant4 and asked to simulate the Hyperscint HS-RP100 scintillating point-detector [16] (MedScint Inc., Quebec City, Canada) in low-energy photon beams. The student was introduced to the basic particle tracking concepts (particle, sensitive detector, hit) in Geant4 and simulation hierarchy (Run, Event, Track, Step). Using the M-TAG code, the student could implement the hyperscint detector in Geant4.

Fig. 7 shows a cross-section of the simulated detector with all dimensions. The hyperscint detector was modeled in the air. The optical transmission cable was modeled as a poly methyl methacrylate cylinder and is shown in magenta. The sensitive volume is placed on its tip, shown in white, which is modeled as either polyvinyltoluene (PVT) or water, depending on the simulation. Surrounding the sensitive volume and a portion of the transmission cable is a light-tight acrylic layer shown in yellow. The blue portion is a light-tight polyehtylene sheath for the optical fiber. A 2 cm x 2 cm planar photon source was created parallel to the optical fiber length and placed centered on the sensitive volume of the detector. The source-to-surface distance was 17.1 cm. For both sensitive volume materials (water and PVT), 10 simulations were performed with mono-energetic photon sources ranging from 10-100 keV in 10 keV increments. The total dose was scored, and uncertainties were calculated using the history-by-history method [22]. The ratio of dose to water and plastic for each energy was calculated. These values were compared to mass energy absorption ratios for the given materials [23]. The highest simulated photon energy was 100 keV. At this energy, the maximum range of secondary electrons is 0.0139 cm and 0.0143 cm for PVT and water, respectively. With this small range, we can assume that the mass-energy absorption ratios will equal the dose ratios, i.e., collision kerma equals absorbed dose.

Following the recommendations of the American Association of Physicists in Medicine Task Group 268 report, simulation parameters shared between all simulations are presented in Table 1 [24].

3. Results

3.1. M-TAG validation

The recorded positron emission energies for ^{18}F, ^{11}C, ^{15}O, and ^{68}Ga were plotted on histograms and compared to theoretical positron emission energy spectra as calculated by Champion and Le Loirec [17]. Calculated energy uncertainties were below 0.001%. Fig. 8 presents this comparison, and Table 2 compares the mean for these distributions.

Table 1
Summary of parameters used for Monte Carlo simulation.

Item	Description	References
Toolkit	M-TAG (Geant4 toolkit version 11.01)	Allison et al. [3]
Cross-sections	Penelope	Baro et al. [14]
Validation	Calculated positron range in water to previously calculated and experimental values	Champion and Le Loirec [17], Lehnert et al. [18], Partridge et al. [19], Cal-Honzalez et al. [20]
Source description for M-TAG validation	positron-emitting radioisotopes allowed decaying as a point source at center of simulation geometry	N/A
Source description for hyperscint detector	2x2 cm^2 planar monoenergetic photon source	N/A
Production and Tracking Cut-off for all particles	1 keV	Carroll et al. [11]
Statistical uncertainty	history-by-history method	Walters et al. [22]

Table 2
Comparison of simulated and literature [17] positron emission energy means.

Isotopes	^{18}F	^{11}C	^{15}O	^{68}Ga
Simulated Mean (keV)	250.1	385.7	735.3	829.1
Literature Mean (keV)	252.0	390.0	730.0	844.0
% Difference	+0.7%	+1.1%	-0.7%	+1.8%

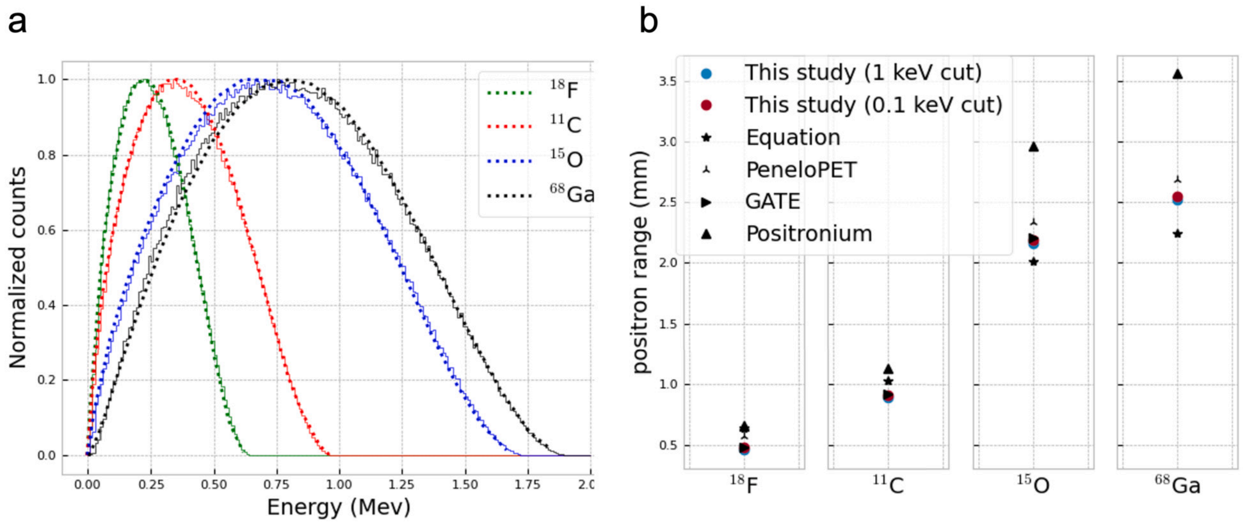


Fig. 8. Results from the validation simulations. (a) Comparison of simulated positron emission energy spectra (solid lines) to those calculated by Champion and Le Loirec [17] (dotted lines). (b) Comparison of simulated positron ranges in this study with 1 and 0.1 keV energy cutoff analytical equation for range [20], PeneloPET simulation [20], GATE simulation [18], and simulation taking into account positronium formation [17].

The mean positron ranges for all scenarios were plotted in Fig. 8 b. Calculated range uncertainties were below 0.001%. For all isotopes, ranges simulated by Champion [17], which was the only simulation to include positronium formation, were larger than all other studies. The positron ranges simulated in this work were consistently near the lower end of all ranges and agree very well with the GATE simulation [18] with percent differences ranging between 0.2 and 3.1%. This is expected as the GATE software builds upon Geant4, but uses a different Geant4 version. This shows that the user-code implementation correctly calculates positron ranges. Our user-code and the GATE code agree with the other results for isotopes with low positron energies. This study simulated positron ranges with two particle energy cuts (0.1 and 1 keV). All ranges simulated with the 0.1 keV cut were larger than those simulated with a 1 keV cut. 3.73%, 2.75%, 1.73% and 1.47% larger for ^{18}F, ^{11}C, ^{15}O, ^{68}Ga respectively).

3.2. M-TAG test case

Fig. 9 shows the simulation data compared to known values. All simulated dose ratios agree within error to the mass-energy absorption ratios.

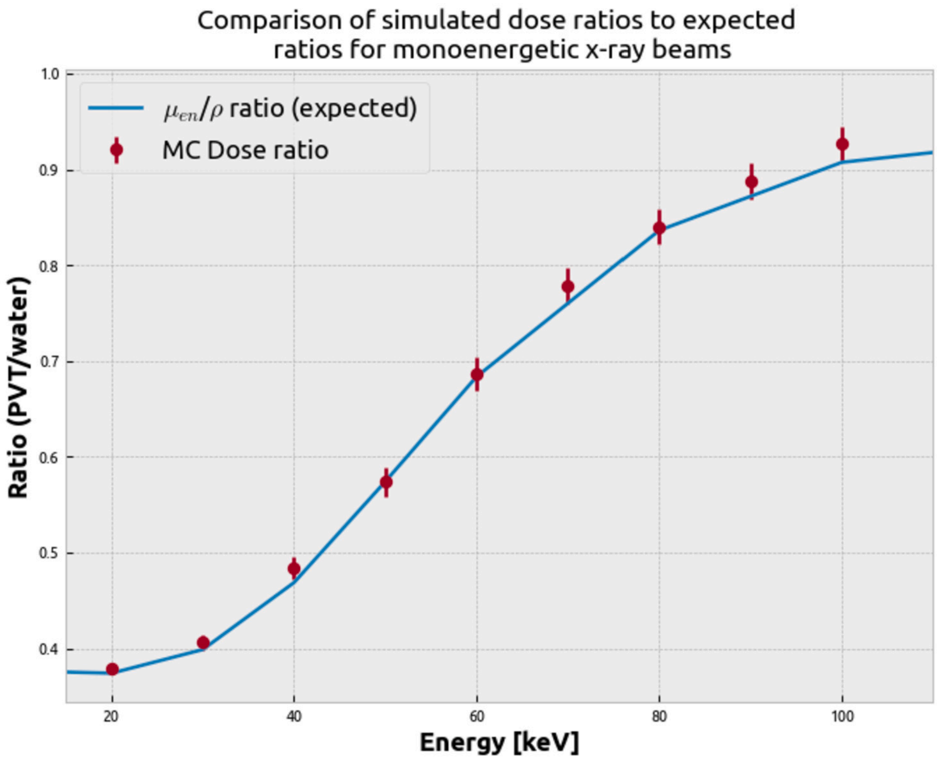


Fig. 9. Results from the hyperscint simulation. The simulated dose ratios (PVT/water) are shown as data points, and the mass-energy absorption coefficients are shown as a smooth line. We expected the results for this simulation to agree with the mass-energy absorption coefficient ratios.

4. Discussion

Geant4 is a Monte Carlo-based radiation simulation toolkit that gives the user total control over all simulation aspects. It is extensively used in many research fields with a growing user-base in medical radiation simulations. The same features that make Geant4 so popular also make learning it difficult. While Geant4 provides a user-guide and many examples of working simulations, there is a lack of Geant4 teaching aids.

M-TAG was developed to accelerate the design and testing of radiation detectors for medical applications and as a teaching aid to introduce researchers to the Geant4 toolkit. The core physics engine is tuned to be accurate for medically relevant energies. The definition of the simulation world is designed such that individual physical volumes can each be defined as separate classes that can be added and removed from the simulation world easily. The same is done for detector components of the simulation, where each detector type is its own class containing all the information required to place the detector in the simulation, create the data structures to store data points and save data once the simulation is complete. All of the above can be done interactively through Geant4 macro files, which act as input files for the simulation, or at run-time using run-time commands. In this way, very little Geant4 experience or knowledge in C++ programming is required to use the built-in M-TAG classes. At the same time, more advanced users can define and add additional geometries and detector elements. M-TAG modules add additional modularity to Geant4 that allows users to quickly share simulation building blocks, such as geometric elements and detectors, with each other. This feature also helps new users to include complicated simulation geometries quickly.

The first validation test case for M-TAG was to calculate the range of positrons in water compared to literature values. This choice was made since the design of M-TAG started as a Geant4 user-code to simulate positron detectors to be combined with PET. Positron generation and transport by the M-TAG implementation of Geant4 were validated by using it to calculate travel distance in water from positrons emitted in the decay of common PET radioisotopes. The range of the resulting positrons was compared to published values, as shown in Fig. 8 b. Our results fell within the distribution of published data and were near the low end compared to the literature values. This is expected as we use a large energy threshold cutoff of 1 keV. This choice was made to save computation time as described in previous work [11]. Of note is the good agreement of the positron ranges from this work compared to the GATE simulations (ranging between 0.2 and 3.1%). This is the expected result as both M-TAG and GATE use Geant4 to perform the Monte Carlo calculations and shows that M-TAG properly calls Geant4 functions and classes.

The true test for the code was whether it could be used to introduce a new user to Geant4. A new Geant4 user, an undergraduate student in Physics performing a research project during the summer, could construct a model of the hyperscint scintillator-based detector and use it to perform basic Geant4 simulations. The hyperscint simulations provide a second set of validation simulations for the M-TAG code as they agree within uncertainty to the mass-energy absorption coefficient ratios obtained from NIST. This simulation setup can now be used to model more realistic radiation sources to compare with results obtained from measurements. Besides this new user, M-TAG has been used by two other lab members to learn Geant4 and perform their research. Some of their simulations can be seen in Fig. 6. Describing the methods and results of these studies is beyond the scope of this paper. This anecdotal evidence suggests that M-TAG is useful as a teaching aid for Geant4. Further comparisons could be made with a larger new-user cohort.

While Geant4 is a highly customizable simulation toolkit, many other software packages specialize in radiation transport simulation. Three such examples are the GATE [21], TOPAS [9] and GAMOS [10] packages that build on top Geant4 and are designed to be used for medical radiation simulations. The key difference between them and M-TAG is that they completely insulate the user from the Geant4 code with a scripting language used to build simulations using scripting languages. Note that designing a simulation using these scripting languages is unlike using Geant4. GATE TOPAS and GAMOS functions then interpret user input files and call Geant4 functions to build the simulation. These are highly featured software packages but do not include all Geant4 functionality. GATE TOPAS and GAMOS are much easier to learn than Geant4, they have an extensive library of functions and options, and if a user's use-case is covered by one of these software packages, it is usually quicker to implement using it. Suppose a new user wants to perform a simulation that GATE or TOPAS does not cover. In that case, they have two choices: 1) contact the developers and hope that they implement the missing feature on time, and 2) find another tool, such as Geant4, that will allow them to implement the missing feature. Both of these options are time-consuming. In this instance, learning Geant4 first would have been the more efficient option.

In addition to the scripting language, GAMOS has a plug-in capability, allowing users to add custom C++ code to extend the base GAMOS functionality. All capabilities in the GAMOS program can be accessed using GAMOS commands without needing to code any C++. The advantage of the plug-in system is that it allows users to extend GAMOS without needing to understand the underlying GAMOS kernel. [10] For example, a user could write a class describing their own G4VSensitiveDetector, called *MyDetector.cc*, and *MyDetector* would then be a selectable detector option when defining simulations with GAMOS. This plug-in system allows new users to code quickly, allows advanced to extend and customize GAMOS and facilitates code sharing. A user must still learn GAMOS and Geant4 to extend its functionality. M-TAG is designed so users start coding in C++ using native Geant4 commands. The only exception to this is if a new user uses only existing M-TAG modules for their simulation.

M-TAG is a Geant4 user-code and teaching aid that can be used to introduce new users to Geant4 and C++. It has validated examples and a framework for easy detection and geometry definition. M-TAG has limitations, especially compared to GATE, TOPAS and GAMOS. M-TAG is designed to be easily extendable by users and expects new users to quickly start coding in C++. This does require knowledge of object-oriented programming concepts. In addition, there are very few safety rails in the code. Users can modify the physics list, geometry and other simulation aspects. This can lead to incorrect results. Other tools provide a safe working environment where breaking the simulation is difficult. At this time, there are also limited examples available.

5. Conclusion

This work presents a modular Geant4-based simulation package called M-TAG, developed as a teaching aid for Geant4 and validated for use at medically relevant energy levels. M-TAG provides a method to allow new Geant4 users to quickly develop simulations and extract useful data without masking any Geant4 features, which makes it a useful tool for both new and advanced Geant4 users. The M-TAG source code can be provided upon request.

CRediT authorship contribution statement

Liam Carroll: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Shirin A. Enger: Conceived and designed the experiments; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This research was undertaken, in part, thanks to funding from the Canada Research Chairs Program (grant # 252135) as well as Collaborative Health Research Project (NSERC+CIHR grant 170620).

References

[1] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, et al., Geant4—a simulation toolkit, Nucl. Instrum. Methods Phys. Res., Sect. A, Accel. Spectrom. Detect. Assoc. Equip. 506 (3) (July 2003) 250–303. Available from: https://www.sciencedirect.com/science/article/pii/S0168900203013688.

[2] PubMed "Geant4", https://pubmed.ncbi.nlm.nih.gov/?term=Geant4&filter=years.2017-2022, 2022.

[3] J. Allison, K. Amako, J. Apostolakis, P. Arce, M. Asai, T. Aso, et al., Recent developments in GEANT4, Nucl. Instrum. Methods Phys. Res., Sect. A, Accel. Spectrom. Detect. Assoc. Equip. 835 (November 2016) 186–225.

[4] J. Allison, K. Amako, J. Apostolakis, H. Araujo, P.A. Dubois, M. Asai, et al., Geant4 developments and applications, IEEE Trans. Nucl. Sci. 53 (1) (February 2006) 270–278.

[5] Geant4 Collaboration, Class categories and domains — book for application developers 11.0 documentation. Available from: https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/html/Fundamentals/classCategory.html.

[6] Geant4 Collaboration, Hits — book for application developers 11.0 documentation. Available from: https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/html/Detector/hit.html#sensitive-detector.

[7] Geant4 Collaboration, Analysis manager classes — book for application developers 11.0 documentation. Available from: https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/html/Analysis/managers.html#analysis-manager.

[8] S. Jan, G. Santin, D. Strul, S. Staelens, K. Assié, D. Autret, et al., GATE: a simulation toolkit for PET and SPECT, Phys. Med. Biol. 49 (19) (2004) 4543–4561. Available from: http://stacks.iop.org/0031-9155/49/i=19/a=007?key=crossref.be1d0bbaf1d502515584e510c77393cb.

[9] J. Perl, J. Shin, J. Schümann, B. Faddegon, H. Paganetti, TOPAS: an innovative proton Monte Carlo platform for research and clinical applications, Med. Phys. 39 (11) (2012) 6818. Available from: pmc/articles/PMC3493036/.

[10] P. Arce, J. Lagares, LHR, Gamos: a framework to do Geant4 simulations in different physics fields with an user-friendly interface, Elsevier; Available from: https://www.sciencedirect.com/science/article/pii/S0168900213012709?casa_token=kSUyvoXD-9EAAAAA:me0Mag_7L0WGCnhF2jXtrL9FXK6QqsWobv8-30lqjKzkOA7FPBObSGR4Nm1y7_PD692ziEnJdKU, 2014.

[11] L. Carroll, S.A. Enger, Simulation of a novel, non-invasive radiation detector to measure the arterial input function for dynamic PET, Med. Phys. (October 2022). Available from: https://onlinelibrary-wiley-com.proxy3.library.mcgill.ca/doi/full/10.1002/mp.16055.

[12] L. Carroll, E. Croteau, G. Kertzscher, O. Sarrhini, V. Turgeon, R. Lecomte, et al., Cross-validation of a non-invasive positron detector to measure the arterial input function for pharmacokinetic modelling in dynamic positron emission tomography, Phys. Med. (2020) 76.

[13] V. Turgeon, G. Kertzscher, L. Carroll, R. Hopewell, G. Massarweh, S.A. Enger, Characterization of scintillating fibers for use as positron detector in positron emission tomography, Phys. Med. 65 (September 2019) 114–120. Available from: https://www.sciencedirect.com/science/article/abs/pii/S1120179719301899.

[14] J. Baró, J. Sempau, J.M. Fernández-Varea, F. Salvat, PENELOPE: an algorithm for Monte Carlo simulation of the penetration and energy loss of electrons and positrons in matter, Nucl. Instrum. Methods Phys. Res., Sect. B, Beam Interact. Mater. Atoms 100 (1) (May 1995) 31–46.

[15] J.K. TULI, Evaluated nuclear structure data file – a manual for preparation of data sets. Available from: http://www.osti.gov/servlets/purl/779777-iAo1Uo/native/, February 2001.

[16] Y. Chen, X. Ma, F. Ma, E. Jean, F. Therriault-Proulx, L. Beaulieu, Comparative optic and dosimetric characterization of the HYPERSCINT scintillation dosimetry research platform for multipoint applications, Phys. Med. Biol. 66 (8) (April 2021) 085009. Available from: https://iopscience.iop.org/article/10.1088/1361-6560/abf1bd.

[17] C. Champion, C. Le Loirec, Positron follow-up in liquid water: II. Spatial and energetic study for the most important radioisotopes used in PET, Phys. Med. Biol. 52 (22) (November 2007) 6605–6625. Available from: http://stacks.iop.org/0031-9155/52/i=22/a=004?key=crossref.1daf17f2e1836466fb122a3a39da746d.

[18] W. Lehnert, M.C. Gregoire, A. Reilhac, S.R. Meikle, Analytical positron range modelling in heterogeneous media for PET Monte Carlo simulation, Phys. Med. Biol. 56 (11) (May 2011) 3313. Available from: https://iopscience.iop.org/article/10.1088/0031-9155/56/11/009.

[19] M. Partridge, A. Spinelli, W. Ryder, C. Hindorf, The effect of $\beta+$ energy on performance of a small animal PET camera, Nucl. Instrum. Methods Phys. Res., Sect. A, Accel. Spectrom. Detect. Assoc. Equip. 568 (2) (December 2006) 933–936.

[20] J. Cal-González, J.L. Herraiz, S. España, P.M.G. Corzo, J.J. Vaquero, M. Desco, et al., Positron range estimations with PeneloPET, Phys. Med. Biol. 58 (15) (2013) 5127–5152. Available from: http://iopscience.iop.org/article/10.1088/0031-9155/58/15/5127/pdf.

[21] D. Strul, G. Santin, D. Lazaro, V. Breton, C. Morel, GATE (geant4 application for tomographic emission): a PET/SPECT general-purpose simulation platform, Nucl. Phys. B, Proc. Suppl. 125 (September 2003) 75–79.

[22] B.R.B. Walters, I. Kawrakow, D.WO. Rogers, History by history statistical estimators in the BEAM code system, Med. Phys. 29 (12) (November 2002) 2745–2752. Available from: http://doi.wiley.com/10.1118/1.1517611.

[23] NIST, XCOM database. Available from: https://www.physics.nist.gov/PhysRefData/Xcom/html/xcom1.html.

[24] I. Sechopoulos, D.W.O. Rogers, M. Bazalova-Carter, W.E. Bolch, E.C. Heath, M.F. McNitt-Gray, et al., RECORDS: improved reporting of Monte Carlo RaDiation transport studies: report of the AAPM research committee task group 268, Med. Phys. 45 (1) (January 2018) e1–e5. Available from: https://onlinelibrary.wiley.com/doi/10.1002/mp.12702.