Design of Software for Visualized Import of Geometric Models Based on QT and CADmesh

Weiyang Zhang¹, Shiwei Jing^{1,*}

2026-06-03

ChinaRxiv

Abstract

To address the low efficiency, high barrier to entry, and difficulty in constructing complex models associated with traditional manual coding in Geant4 modeling, this paper designs and implements a visual modeling tool based on the QT framework and the CADmesh interface. Using a three-tier modular architecture, the tool realizes the full workflow of STL/OBJ model import, parameter adjustment, one-click conversion, and simulation integration. Tests show that the tool operates stably, reduces the conversion time of complex models by more than 95%, achieves geometric errors below 1.4%, and keeps physical simulation deviations below 3%. At present, it supports only STL/OBJ formats, and the processing efficiency for ultra-large-scale models remains to be further optimized. The tool enables zero-code modeling, effectively lowers the barrier to use, and provides efficient and reliable technical support for nuclear physics particle transport simulation.

Full Text

Design of Software for Visualized Import of Geometric Models Based on QT and CADmesh

Weiyang Zhang1, Shiwei Jing1,*

1 School of Physics, Northeast Normal University, Changchun 130024, China

* E-mail: jingsw504@nenu.edu.cn

Abstract:
To address the problems of low efficiency, high barriers to use, and difficulty in constructing complex models in traditional manual coding-based modeling for Geant4, this paper designs and implements a visual modeling tool based on the QT framework and the CADmesh interface. Using a three-layer modular architecture, the tool implements the full workflow of STL/OBJ model import, parameter adjustment, one-click conversion, and simulation integration. Tests show that the tool operates stably; the conversion time for complex models is reduced by more than 95%, the geometric error is less than 1.4%, and the deviation in physical simulation is less than 3%. At present, only STL/OBJ formats are supported, and the processing efficiency for ultra-large-scale models still requires further optimization. The tool enables code-free modeling, effectively lowers the threshold for use, and provides efficient and reliable technical support for nuclear-physics particle-transport simulations.

Keywords: Geant4; CADmesh; QT; visual modeling; particle simulation

Software Design of Geometric Model Visualization Import Based on QT and CADmesh

Weiyang Zhang1, Shiwei Jing1,*

1 School of Physics, Northeast Normal University, Changchun 130024, China

Abstract:
Aiming at the problems of low efficiency, high threshold and difficulty in constructing complex models in traditional manual coding modeling for Geant4, this paper designs and implements a visual modeling tool based on the QT framework and CADmesh interface. Adopting a three-layer modular architecture, the tool realizes the whole process including STL/OBJ model import, parameter adjustment, one-click conversion and simulation integration. Tests show that the tool runs stably, the conversion time of complex models is reduced by more than 95%, the geometric error is less than 1.4%, and the deviation of physical simulation is less than 3%. At present, only STL/OBJ formats are supported, and the processing efficiency of ultra-large-scale models needs to be further optimized. The tool achieves coding-free modeling, effectively lowers the application threshold, and provides efficient and reliable technical support for nuclear physics particle transport simulation.

Keywords: Geant4; CADmesh; QT; Visual modeling; Particle simulation

1 Introduction

In modern science and engineering, accurate radiation simulation and particle-transport studies are indispensable. Whether in the investigation of particle-motion laws in nuclear physics experiments or in frontier explorations in high-energy physics, accurately simulating the interactions between particles and matter is an essential component[1]. This process not only forms the basis for deeply analyzing the nature of physical phenomena, but also provides key support for optimizing system design and ensuring the safety of personnel and equipment, and is of great significance to technological development in related fields.

Geant4, a Monte Carlo simulation toolkit developed and maintained by the European Organization for Nuclear Research, has been widely applied across many disciplines by virtue of its powerful particle-transport simulation framework[2]. This toolkit adopts object-oriented technology and an iterative incremental software process, supports functional extension and improvement, and does not affect the use of code in existing production modes[3]. Geant4 has shortcomings in constructing complex geometric models. Its native modeling relies on combinations of simple constructive solid geometry (CSG) entities such as spheres and cuboids, generating complex models through text- or code-controlled geometric transformations and Boolean operations[4]. When faced with irregular complex structures in engineering practice, this modeling approach involves a lengthy

workflow, is time-consuming and labor-intensive, and is prone to errors introduced by manual operation, making it difficult to meet the current, increasingly demanding requirements for high-precision simulation.

To solve the problem of constructing complex geometries in Monte Carlo modeling, automatic modeling methods based on computer-aided design (CAD) software data conversion have gradually become a research direction[5]. This method converts complex three-dimensional models built in CAD software and directly uses them as geometric input for Monte Carlo calculation programs, effectively improving the efficiency and accuracy of complex-model construction. As an open-source project, CADmesh has become a key tool for bridging CAD models and Geant4 by virtue of its support for multiple file formats and its extensible features such as ASSIMP and TETGEN readers[6]. Its core advantage lies in the fact that, without relying on third-party intermediate formats such as GDML, it can directly import triangular-facet geometry files in formats such as STL and PLY into Geant4, simplifying the modeling workflow. At present, it has been widely applied in ionizing-radiation simulation scenarios such as radiotherapy dose calculation and radiation-protection simulation, providing technical support for research institutions, medical-device manufacturers, and others.

Although CADmesh provides a feasible path for connecting CAD models with Geant4, unresolved issues remain in practical applications. On the one hand, the compatibility between formats such as STL and OBJ output by different CAD software and Geant4 lacks systematic verification, and the absence of a universal data interface can easily lead to problems such as facet loss and coordinate offsets during format conversion, affecting model accuracy[7]. On the other hand, existing tools for importing CAD data into Geant4 depend on intermediate formats, with complex operations and uneven simulation efficiency[8]. The MCAM program developed by the domestic FDS team has made some progress in Geant4 automatic modeling, but it still has not solved the problem of balancing the accuracy and efficiency of complex-model import[9]. In addition, how to optimize the import process for complex models[10] and achieve deep integration of design, analysis, and optimization remains a difficulty in the current field. To address the above pain points, this paper develops a visual modeling tool based on the QT framework[11], implements a one-stop operation from model import to simulation execution, lowers the threshold for use, improves modeling efficiency, and provides an efficient solution for particle-transport simulation of complex geometries.

2 Overall Software Design

2.1 Requirements Analysis

Based on practical research scenarios in the field of nuclear physics and in combination with the characteristics of Geant4 modeling, the software functional requirements include three main aspects: convenience, automation, and visu-

alization. The file-import function must support the import of STL and OBJ format files, be compatible with model files exported by mainstream CAD software such as FreeCAD and Blender[11], automatically identify file formats, and run successfully. After import, the model-visualization function must support functions such as model rotation and scaling,

This facilitates observation of geometric-structure details. The parameter-visualization adjustment function provides model size scaling, spatial translation, and rotation, displays the model morphology in real time, applies parameter modifications immediately, and opens the modified graphics with one click. The one-click conversion function can automatically parse the geometric data of the imported model, such as vertex coordinates, facet topology, and material information; call the CADmesh interface to generate Geant4-compatible C++ scripts [12]; and support the export of script files and compilation configuration files (`CMakeLists.txt`), without requiring users to manually write parsing code. The result-output and log-recording functions ensure that the exported Geant4 script contains a complete detector construction class, `G4VUserDetectorConstruction`, material definitions, and physical-volume placement logic, allowing direct integration into a Geant4 simulation project. Automatic recording of conversion logs is still maintained, including the imported file path, parameter settings, conversion time, error information, etc.; logs can be exported as `.txt` files to facilitate troubleshooting. The interface layout conforms to the operating habits of scientific researchers. Core function buttons—import, conversion, adjustment, conversion, and export —are all concentrated on the main interface, with no hidden menus. The entire process from model import to Geant4 script export can be completed in $\leq$ 5 steps, and an operation manual containing illustrated tutorials and help documents is provided.

Because this software is operated on a virtual machine, it supports the Ubuntu 20.04/22.04 Linux versions and is compatible with Geant4 10.7 and above, CADmesh 3.0 and above, QT 5.12 and above, as well as the relevant hardware requirement configurations.

2.2 Overall Framework Design

This software is based on the Linux system Ubuntu 22.04 LTS and is deployed and developed through a VMware Workstation 17 virtual machine. Technically, it is built around the Geant4 ecosystem and the QT framework. QT 5.15.2 is adopted. Relying on the event-handling mechanism of its QtCore module and the interface components of its QtWidgets module, the design of visual interaction and business logic is realized; the `G4VUserDetectorConstruction` class library of Geant4 11.1.2 and the CADmesh 3.0 interface are integrated; and dynamic link libraries are directly called to implement geometric-model adaptation. The stability of the parallel-computing module in the Linux environment is significantly better than that under Windows, and seamless integration between the tool and the simulation workflow can be achieved through Shell scripts.

The software adopts a three-layer architecture and a modular design pattern. From top to bottom, it is divided into the UI interaction layer, the core logic layer, and the bottom interface layer. Communication between layers is carried out through standardized interfaces, reducing coupling between modules and facilitating subsequent functional expansion and maintenance. The final overall program flow design is shown in Figure 1.

Figure contents:

- User starts software
- UI interaction layer
  - Input operation instruction
  - Encapsulate user request
  - Receive parsing request
- Request type determination
  - File parsing
    - Read STL/OBJ
    - Verify integrity
  - Format conversion
    - Convert to CADmesh format
  - Script generation
    - Generate Geant4 construction-class script
- Bottom interface layer
  - Call CADmesh interface
  - Call third-party library
  - Integrate processing results
- UI interaction layer
  - Receive processing result
  - Update interface display
- User views result

Figure 1 Overall program flowchart of the QT visualization software

The UI interaction layer, i.e., the user-facing layer, is responsible for receiving user operation commands, conversion status, and log information, and for providing a parameter-adjustment interface; it serves as the bridge for interaction between the user and the software. The file-operation component mainly handles file selection, batch import, and log export, and supports drag-and-drop file import. The visualization preview component provides rotation, translation, and scaling operation modes, and displays key information such as the number of model vertices, the number of facets, and the volume. The parameter-adjustment component includes three subfunctions—scaling, translation, and rotation—and adopts a parameter-control mode in which parameters are adjusted through a separate interface. The function-control component includes function buttons such as model drawing-format conversion and one-click conversion. The status-feedback component displays the current operation status, such as importing, previewing, converting, completed, and error, and shows messages in real time such as "File imported successfully," "Model repair completed," and

"Conversion successful."

The logic layer, i.e., the processing layer, parses user operation commands and coordinates the functional modules to complete business processing; it is the main part of the software. The file-parsing module is responsible for reading STL or OBJ file data, parsing vertex coordinates and facet topology, checking file integrity, and handling file-format errors. The parameter-processing module receives the scaling, translation, and rotation parameters transmitted from the UI layer, converts them into geometric transformation matrices in the Geant4 world coordinate system, applies them to the model vertex data, and updates them in real time. The format-conversion module converts the parsed model data into a format recognizable by CADmesh, calls the CADmesh interface to generate geometric solids, and supports mutual conversion between STL and OBJ formats. The script-generation module automatically generates Geant4 detector-construction class scripts (`.cc/.hh`) based on geometric-solid data, material information, and transformation matrices. These scripts contain complete code for material definition, geometric-solid encapsulation, logical-volume creation, physical-volume placement, and related operations, and are compatible with Geant4 compilation specifications. In addition, a management module mainly records key events during software operation, such as file import, parameter modification, conversion steps, and error information, sorted by timestamp; it supports displaying only error logs or all logs, as well as exporting them.

The low-level interface layer is the dependency-calling layer, which encapsulates third-party library interfaces, provides low-level technical support for the logic layer, and shields interface differences among different libraries. For the Qt framework interface, QtCore is called to handle events and data structures, and the QtWidgets interface-component module is used to implement interface rendering and interaction logic. The CADmesh interface calls the `CADmesh::FromSTL()` and `CADmesh::FromOBJ()` functions to parse models, and calls the `TessellatedSolid()` function to generate Geant4 geometric solids, setting parameters such as the facet-merging threshold and geometric tolerance. The Geant4 interface encapsulates interfaces for classes such as the `G4NistManager` material-library call, `G4TessellatedSolid` geometric-solid encapsulation, and `G4LogicalVolume` logical-volume creation, ensuring that the generated scripts are compatible with the Geant4 ecosystem.

2.3 Core Design Principles

Each module is responsible for only a single business function. For example, the file-parsing module only handles file reading and parsing. Modules communicate through standardized interfaces, with no direct dependencies, making them easy to modify and test independently. Extensibility is achieved through a plug-in design: modules in the core logic layer can all be extended in the form of dynamic-link libraries (`.so`). In the future, functions such as GDML format support and AI-assisted model optimization can be added without modifying the original architecture. All complex low-level operations, such as CADmesh

Figure 2 shows the overall interface layout of the software. The window title is "Drawing Converter." The menu bar contains "File (F)," "Edit (E)," "Help (H)," and "Tools (T)." The main title reads "Neutron-Simulation Geometry Construction." Buttons include "Sphere," "Cube," "Cylinder," "Cuboid," "Input File," "Start Conversion," "Modify Size and Position," "Run Again," and "Save Log."

Figure 1: Figure 2 shows the overall interface layout of the software. The window title is "Drawing Converter." The menu bar contains "File (F)," "Edit (E)," "Help (H)," and "Tools (T)." The main title reads "Neutron-Simulation Geometry Construction." Buttons include "Sphere," "Cube," "Cylinder," "Cuboid," "Input File," "Start Conversion," "Modify Size and Position," "Run Again," and "Save Log."

interface calls and Geant4 script generation, are encapsulated in the core logic layer, so users can complete the process through simple operations in the UI layer. The low-level interface layer performs compatibility processing for different versions of third-party libraries, such as Geant4 10.7/11.0 and CADmesh 3.0/3.1, automatically identifies the library version, and calls the corresponding interface to avoid version conflicts.

2.4 Research on Module Development

The software interface is developed based on the QtWidgets module and adopts a "main interface + subpanel" layout. The overall style is concise and clear, operations are convenient, and it is suitable for most usage scenarios. The overall interface layout is shown in Figure 2.

Figure 2 Overall layout of the software interface

(1) Interface layout design

The top menu bar contains four menus: File (F), Edit (E), Help (H), and Tools (T). The File menu includes options for creating a new file, opening a file, and saving as, and supports shortcut-key operations, such as using Ctrl+O to open a file. The Edit menu includes cut, copy, and paste options, supporting rapid file operations. The Help menu mainly contains the About Software option; clicking "About Software" opens a local PDF document.

On the left side of the central functional control area are simple selectable shapes. The center contains four buttons—Input File, Start Conversion, Modify Size and Position, and Run Again—as well as a field for displaying the file name. The button states change dynamically according to the operation. After generation, a log display area appears at the bottom: log information is displayed with timestamps; error messages are marked in red, and normal messages are marked in black. Log copying, clearing, and filtering are supported.

(2) Data conversion and script generation

The file-parsing submodule parses STL and OBJ formats, extracts the vertex coordinates, facet topology, and material information of the model, and constructs a unified model data structure. In the final output stage, it is responsible for converting the repaired model data into a CADmesh interface invocation format, automatically generating a Geant4 detector-construction class script, and supporting direct integration into a Geant4 simulation project.

Geometric transformation converts the scaling, translation, and rotation parameters set at the UI layer into the `G4Transform` transformation matrix in Geant4 and applies it to the model vertex data. For CADmesh interface invocation, the corresponding `CADmesh::FromSTL()/CADmesh::FromOBJ()` is called according to the model format, STL or OBJ, to generate a `G4TessellatedSolid` geometric solid. Material matching matches the parsed material information with the Geant4NIST material library; if no matching material is found, a custom material is automatically created. The Geant4 script generated by the script-generation logic contains a complete detector-construction class, including header-file (`.hh`) declarations and member-variable definitions. The source file (`.cc`) constructs functions such as material definition, geometric-solid creation, logical-volume creation, and physical-volume placement.

(3) Geant4 integration module design

The Geant4 integration module is responsible for seamlessly integrating the scripts generated by the software with the Geant4 simulation project, thereby realizing physical-volume placement of the model and invocation of particle-transport simulation. Its core functions include three subfunctions: simulation-environment configuration, simulation-task invocation, and result feedback. Among them, the simulation-environment configuration subfunction is responsible for automatically detecting the user' s local Geant4 installation path, compiler version, dependent CADmesh, and Qt configuration, generating the corresponding `CMakeLists.txt` build script, and ensuring that the generated detector-construction class script can be compiled normally. Simulation tas

The task-invocation subfunction is responsible for integrating the detector-construction class scripts generated by the software with the Geant4 simulation project, enabling automated invocation for compilation, execution, and visualization. Users do not need to manually execute terminal commands: the project is compiled, the simulation is run, and visualization is then displayed. The result-feedback subfunction is responsible for collecting log information, progress data, and output files during the simulation process, providing users with comprehensive result display and export functions.

The environment-checking logic first obtains the Geant4 installation path by reading the system environment variable `G4INSTALL`; if it is not set, the user is prompted to enter it manually. It then checks whether a compiler is installed in the system; if not, the user is prompted to install it via commands. Next, dependency-library checking determines whether the `CADmesh` dynamic-link li-

brary exists and whether the Qt library is correctly configured; if not found, the user is prompted to set the `LD_LIBRARY_PATH` environment variable. The automatic compilation function uses Qt' s `QProcess` class to invoke system Shell commands and execute the CMake compilation workflow; it supports automatic detection of compilation errors and feeds them back to the user. In addition, based on the user' s Geant4 installation path and project directory, a compilation script (`compile.sh`) is automatically generated, including steps such as creating the build directory and executing the `cmake` and `make` commands. After compilation is completed, the simulation-running function automatically invokes the Geant4 executable file, supports parameters such as the number of simulated particles to be transported and the output-file path, reads the simulation log in real time, and displays the simulation progress. The software provides a simulation-parameter configuration panel, supporting settings for particle type, particle energy, number of emitted particles, and output-file format; the parameters are automatically written into the Geant4 macro file. The visualization and feedback functions support one-click opening of the Geant4 visualization window within the software, allowing users to view the model geometry and particle-transport process. Visualization parameters such as color, viewing angle, and trajectory display can be configured through the software interface. Through the signal-and-slot mechanism, the compilation progress and simulation progress are transmitted to the UI layer in real time and displayed in progress bars. After the simulation is completed, the output files are automatically identified.

3 Software Testing and Validation

3.1 Selection of Test Geometries

(1) Sphere: radius 0.5 m, no hollow structure. When exported from FreeCAD, the meshing tolerance was set to two levels, 0.01 mm (high precision) and 1 mm (low precision), corresponding respectively to the two formats STL and OBJ, for a total of 4 test files;

(2) Cylinder: base radius 0.3 m, height 1 m, axis along the Z axis. When exported from FreeCAD, the meshing tolerance was set to two levels, 0.01 mm (high precision) and 1 mm (low precision), corresponding respectively to the two formats STL and OBJ, for a total of 4 test files.

(3) BSA complex model: composed of 5 parts, namely solid lead/graphite cylinders with H=20CM*D=30CM and H=28CM*D=30CM, a diameter of 60 cm, and a height of 15 cm; a hollow graphite cylinder with an outer diameter of 60 cm, inner diameter of 30 cm, and height of 52 cm; and hollow graphite circular platforms with an outer diameter of 60 cm, D3-30cm, D2-10cm, and height of 10 cm. The meshing tolerance was set to 0.01 mm (high precision) and 1 mm (low precision).

(4) Detector: the core target body `coalSample` (coal sample) is cylindrical (radius 10 cm, half-height 4.1 cm), placed at the center of the scene, and

serves as the target of neutron interactions. The detector `solidNaI` (NaI crystal) is cylindrical (radius 3.8 cm, half-height 10 cm), located 18.2 cm above the coal sample, and is used to record the energy data after neutrons pass through the coal sample.

3.2 Stability Analysis of Test Geometries

Each test case was repeated 10 times, and the import success rate, average conversion time, stability, and facet-loss rate were statistically analyzed to verify the robustness of the software. The test results are shown in Table 1. From the tabulated data, it can be concluded that the software supports the import and conversion of STL and OBJ formats, covering models from simple to complex, with no loss of core functionality. No crash records occurred in any test case, indicating that the software can meet the usage requirements of multiple scenarios, and all experimental geometries were imported successfully. Regarding the robustness conclusion, in 10 repeated tests there were no crashes, no memory overflows, and the facet-loss rate was 0%; the software

It runs stably, and its robustness meets the requirements for use in nuclear-physics simulation research.

Table 1 Analysis of Test Geometry Results

Model type	Precision level	Import success rate	Average conversion time (s)	Stability	Facet loss rate
Sphere	High precision	100%	8.78	No crash, no memory overflow	0%
Sphere	Low precision	100%	3.69	No crash, no memory overflow	0%
Cylinder	High precision	100%	2.97	No crash, no memory overflow	0%
Cylinder	Low precision	100%	2.28	No crash, no memory overflow	0%
BSA complex model	High precision	100%	3.65	No crash, no memory overflow	0%

Model type	Precision level	Import success rate	Average conversion time (s)	Stability	Facet loss rate
BSA complex model	Low precision	100%	2.35	No crash, no memory overflow	0%

3.3 Comparison with Traditional Performance

Using the conventional workflow of writing Geant4 code and integrating and compiling Geant4 scripts as the benchmark, the performance advantage of the software in average conversion time was compared. Ubuntu 22.04 LTS was selected as the operating system, and the software versions were Geant4 11.1.2, CADmesh 3.0, and QT 5.15.2. The geometry selected was the BSA complex model (number of facets: 436034; high precision) and the coal-sample detector model. In the traditional method, the C++ geometric parameters were first modified manually, then Geant4 was recompiled, and finally the simulation was run and the total time recorded. In the software method, the parameters were adjusted in the visualization interface, followed by one-click reconversion and simulation, and the total time was recorded. The time-reduction rate and efficiency-improvement factor were calculated. All data were the averages of three repeated tests, with outliers removed.

Testing showed that the traditional method requires manually writing 300–1000 lines of code covering material definitions and geometry placement, with average coding time accounting for 60% of the total; by contrast, the software realizes one-click generation through modular encapsulation, completely avoiding the manual-coding stage and greatly improving work efficiency. In addition, parameter adjustment in the traditional method requires modifying the core code and recompiling, with compilation time accounting for $ $70%; especially for complex models, compilation time can reach 20 minutes. Through visual parameter adjustment and one-click reconversion, the software directly generates new Geant4 scripts without recompiling the entire project, shortening the time by more than 95% and improving the conversion efficiency for complex models by a factor of 18.2.

3.4 Accuracy Verification

To verify the accuracy and effectiveness of the CADmesh modeling method, neutron transport was taken as the experimental object. Geometric models were constructed respectively by two approaches: traditional manual modeling in Geant4 and CADmesh import modeling. The same physical processes and simulation parameters—particle source and detector settings—were used. A 14.1 MeV point source was selected as the particle source, and particle-transport simulations were carried out for the “software-converted model” and the “Geant4

Figure 3 BSA complex model construction diagram, with labels: hollow circular frustum, graphite outer layer, aluminum cylindrical block, iron cylindrical block.

Figure 2: Figure 3 BSA complex model construction diagram, with labels: hollow circular frustum, graphite outer layer, aluminum cylindrical block, iron cylindrical block.

native-coded model,"respectively. The construction diagram of the BSA complex model is shown in Figure 3.

Figure 3 BSA complex model construction diagram

After importing into Geant4, a 14.1 MeV point source was set at the bottom center point, and additionally directly above, at a distance from the top

At a position 1 cm from the end, a 1 cm cube was set up; 1 million particles were specified, and the neutron energy spectrum passing through the small cube was tallied. A schematic of the parameter settings is shown in Figure 4.

Figure label: 1cm × 1cm × 1cm, neutron energy spectrum tally.

Figure label: A 14.1 MeV point source is placed at the center.

Figure 4 Schematic diagram of parameter settings for neutron transport simulation

After the transport calculation, the resulting neutron energy spectrum distribution is shown in Figure 5.

Figure 5 Neutron energy spectrum distribution of the BSA complex model

From the trend of the energy spectrum distribution, the overall shapes of the obtained neutron energy spectrum curves are in close agreement: in the low-energy region ($10^{-8} \sim 10^{-6}$ MeV), the counts all exhibit the characteristic of increasing rapidly with increasing energy, corresponding to the concentration region for the proportions of thermal neutrons and epithermal neutrons; in the intermediate-energy region ($10^{-6} \sim 10^{-2}$ MeV), the amplitudes of count fluctuations are consistent, and the deviation in peak position is less than 5%; in the high-energy region ($10^{-2} \sim 10^{0}$ MeV), the spectral attenuation trends are consistent, and the relative error in the count proportion of fast neutrons (> 1 MeV) is only 2.3%. The detector model construction diagram is shown in Figure 6.

[In Figure 6: "Detector crystal" ; "Target body"]

Figure 6 Model diagram for constructing the coal-sample detector

After the transport calculation, the resulting neutron energy-spectrum distribution is shown in Figure 7. This is a visualization of the neutron energy

distribution and contains two core information modules. In the upper plot, the horizontal axis is neutron energy (unit: MeV; logarithmic scale; range 10^{-11}–10^{1} MeV), and the vertical axis is neutron counts, showing the number distribution of neutrons at different energies. The lower plot divides neutrons into four categories by energy—thermal neutrons, epithermal neutrons, intermediate-energy neutrons, and fast neutrons—and shows the proportion of each category, such as fast neutrons at 29.4%, intermediate-energy neutrons at 49.6%, and thermal neutrons at 3.8%. The shape of the spectrum directly reflects the energy change of neutrons after passing through the coal sample: fast neutrons (high energy) account for 29.4%, indicating that some neutrons were not significantly slowed down by the coal sample; intermediate-energy neutrons account for the highest proportion (49.6%) and constitute the main energy interval after neutron moderation by the coal sample; thermal neutrons (low energy) account for only 3.8%, indicating that the moderating capacity of the coal sample is limited, which is consistent with the physical characteristics of bituminous coal.

Figure 7 Neutron energy-spectrum distribution of the coal-sample detector

This result shows that the complex geometric model imported via CADmesh can accurately reproduce the structural characteristics of the coal sample, and that its geometric fidelity is sufficient to support the accuracy requirements of particle-transport simulation, with good consistency relative to the physical results of traditional manual modeling. In terms of statistical parameters, the average energy obtained from CADmesh modeling is 2.42 MeV, while that from traditional modeling is 2.38 MeV; both meet the accuracy requirements of Monte Carlo simulation. It should be particularly noted that the two

···the difference in modeling efficiency between the methods is significant: traditional manual modeling requires writing approximately 800 lines of code to construct structures such as internal pores, taking about 7 hours; whereas CADmesh-based modeling only requires importing a predesigned CAD model and completing adaptation through a format-conversion tool, with modeling time reduced to only 1 hour—an 86.1% improvement in efficiency compared with traditional manual modeling.

4 Software Advantage Analysis

4.1 Comparison with Similar Tools

To evaluate the overall performance of this software more objectively, this section selects current mainstream or representative CAD-to-Geant4 conversion tools and conducts a systematic comparison in terms of supported formats, ease of use (whether coding is required), precision control, open-source/commercial status, and main application scenarios. The comparison results are shown in Table 2.

Table 2 Comparative analysis of similar CAD–Geant4 conversion

tools

Comparison dimension	This software	MCAM	FastRAD	GUIMesh
Supported formats	STL, OBJ	STEP, STL	STEP, STL	STEP
Ease of use	Code-free visual operation	Requires script writing	Commercial interface, graphical	Requires writing GDML scripts
Precision control	Adjustable facetization tolerance	Depends on user experience	Commercial optimization	Depends on STEP export precision
Open-source/commercial	Open source	Partially open source	Commercial	Open source
Main application scenarios	General STL/OBJ models, suitable for non-programming users	Nuclear devices such as fusion reactors and reactors	Medical physics and industrial radiation protection	Scientific research and teaching

As can be seen from Table 2, this software has clear advantages in ease of use and simplification of the operational workflow. It does not require users to write any code and is especially suitable for researchers without a programming background. In terms of supported formats, this software focuses on general mesh formats such as STL/OBJ, which are highly compatible with the native CADmesh ecosystem; by contrast, tools such as MCAM and FastRAD provide more comprehensive support for parametric formats such as STEP and IGES, making them suitable for complex engineering scenarios that require accurate surface descriptions. In terms of precision control, this software provides facetization-tolerance adjustment and model-repair functions, allowing users to configure parameters flexibly according to precision requirements. Compared with MCAM, which relies on user experience for adjustment, this software offers a higher degree of automation. In the open-source ecosystem, this software inherits the open-source characteristics of CADmesh, enabling users to carry out secondary development based on the source code; compared with the commercial software FastRAD, it is more flexible and extensible. In summary, this software is complementary in positioning to existing tools. It provides specialized optimization for rapid, zero-code, high-precision import scenarios targeting STL/OBJ formats, effectively filling the gap between ease of use and general applicability in existing tools.

4.2 Advantage Analysis

Compared with traditional Geant4 modeling methods, this software has significant advantages in ease of use, performance, compatibility, and practicality, especially for researchers without a programming background. With a low threshold and less code writing required, experimental physicists and nuclear engineers without programming experience can operate it directly, solving the problem of the high programming barrier in traditional methods. The operational workflow is simplified: the complex process of CAD export, format parsing, model repair, parameter adjustment, script generation, Geant4 integration, and compilation and execution is reduced to three steps—importing the file, adjusting parameters, and one-click conversion—decreasing the number of operational steps by 70%. When errors occur during file parsing, model repair, conversion, or compilation, the software displays the specific causes and provides suggested solutions, reducing the difficulty of troubleshooting. Its practical functional design enables the entire workflow, from file import to Geant4 simulation execution, to be completed within the software, eliminating the need to switch among multiple tools and improving workflow continuity. It also automatically records runtime logs, supports export and search, identifies simulation output files, and provides export functions, making it convenient for users to organize experimental data. Its precision is comparable to that of traditional methods, with both geometric precision and physical precision reaching···

Once the requirements are met, it can be used directly for nuclear-physics simulations. A three-layer architecture—UI layer, logic layer, and interface layer—is adopted, with low coupling between modules. In the future, functions such as support for GDML/STEP formats, AI-assisted mesh optimization, and cross-platform adaptation can be added in the form of plug-ins. The generated Geant4 scripts conform to standards, can be directly integrated into existing Geant4 simulation projects, and support manual modification and secondary development by users.

5 Conclusion

To address the problems of the high threshold, low efficiency, and difficult iteration of complex geometric modeling in Geant4, this work completed the full-process design and implementation of visualization-based import software using the QT framework and the CADmesh interface. Functional completeness and ease of use for non-programmers were established. The software provides file import, parameter adjustment, one-click conversion, and Geant4 integration functions. It adopts a three-layer architecture consisting of the UI layer, logic layer, and interface layer, together with a modular design, achieving low coupling and high extensibility among modules. It is compatible with mainstream Linux distributions and with Geant4 and CADmesh versions, laying the foundation for stable software operation. The interface design uses a refined layout and supports interactive model operations and real-time status feedback; the format-conversion module implements precise parsing of STL and OBJ formats

and integrates model-repair algorithms to ensure the integrity of model topology; the Geant4 integration module implements automated invocation of compilation, simulation, and visualization, generates scripts compatible with Geant4 specifications, and requires no manual coding throughout the entire process. Through multidimensional testing and verification, functional tests covered models ranging from simple to complex scales, with no crashes or memory-overflow problems. Performance comparisons show that conversion of complex models is more convenient, and the time required for parameter adjustment is reduced by more than 95%. Accuracy verification indicates that the geometric deviation between models converted by the software and models natively coded in Geant4 is small. For non-programmers, a zero-code workflow is designed, simplifying the complex modeling process into four steps. Together with intuitive parameter adjustment and error prompts, this significantly lowers the technical threshold; the modular architecture supports future functional expansion, and the design avoids tool switching, improving workflow continuity.

References:

[1] Li Chuanlong, Luo Zhiping, Bi Yuanjie, et al. Automatic Monte Carlo modeling method based on CADMesh and its application[J]. Atomic Energy Science and Technology, 2015, 49(09): 1711-1714.

[2] Alpat A B, Coban A, Kaya H, et al. MRADSIM-Converter: A new software for STEP to GDML conversion[J/OL]. Computer Physics Communications, 2023, 286: 108688.

[3] Poole C M, Cornelius I, Trapp J V, et al. A CAD interface for GEANT4[J/OL]. Australasian Physical & Engineering Sciences in Medicine, 2012, 35(3): 329-334.

[4] Jun S. Geant4 Introduction and Applications Workshop[C/OL]//Geant4 Introduction and Applications Workshop. US DOE, 2024.

[5] Poole C M, Cornelius I, Trapp J V, et al. Fast Tessellated Solid Navigation in GEANT4[J/OL]. IEEE Transactions on Nuclear Science, 2012, 59(4): 1695-1701.

[6] Han M C, Kim C H, Jeong J H, et al. DagSolid: a new Geant4 solid class for fast simulation in polygon-mesh geometry[J/OL]. Physics in Medicine and Biology, 2013, 58(13): 4595-4609.

[7] Apostolakis J, Wright D H, Collaboration G. An Overview of the Geant4 Toolkit[C/OL]//AIP Conference Proceedings: Vol. 896. AIP, 2007: 1-10.

[8] Emmitt J, Mcalister A, Bawden N, et al. XRF and 3D Modelling on a Composite Etruscan Helmet[J/OL]. Applied Sciences, 2021, 11(17): 8026.

[9] Gao Z, Yu Z, Holst M. Quality tetrahedral mesh smoothing via boundary-optimized Delaunay triangulation[J/OL]. Computer Aided Geometric Design, 2012, 29(9): 707-721.

[10] Liu Minjun, Shi Rui, Yang Guang, et al. Simulation study of α-particle energy spectra based on Geant4 and software design and implementation[J]. High Power Laser and Particle Beams, 2023, 35(12): 140–148.

[11] Wendling M, Zijp L J, Mcdermott L N, et al. A fast algorithm for gamma evaluation in 3D [J/OL]. *Medical Physics*, 2007, 34(5): 1647–1654.

[12] Mcpic, V. J. T., et al. A CAD interface for GEANT4 [J]. *Australasian Physical & Engineering Sciences in Medicine*, 2012, 35(3): 329–34.

(Corresponding author: Shiwei Jing, Email: jingsw504@nenu.edu.cn)

Author Contribution Statement:

Weiyang Zhang: designed the research plan, conducted experiments, collected and analyzed data, and drafted the manuscript;
Shiwei Jing: proposed the research idea, designed the research plan, conducted experiments, collected and analyzed data, and revised the final version of the manuscript.

Note: Figure translations are in progress. See original paper for figures.