MONTE CARLO SIMULATION PLATFORM AND SOFTWARE STACK IN DOSE-3D PROJECT*

Jakub Hajduga[a,b,†], Bartłomiej Rachwał[c], Tomasz Szumlak[a]
Marcin Filipek[a], Tomasz Fiutowski[a], Wioleta Górska[a,b]
Paweł Jurgielewicz[a], Damian Kabat[b], Kamila Kalecińska[a]
Łukasz Kapłon[b,d,e], Maciej Kopeć[a], Stefan Koperny[a]
Dagmara Kulig[b], Bartosz Mindur[a], Jakub Moroń[a]
Gabriel Moskal[b,f], Antoni Ruciński[b,d], Piotr Wiącek[a]

[a]AGH University of Science and Technology
Faculty of Physics and Applied Computer Science
Mickiewicza 30, 30-059 Kraków, Poland
[b]Department of Medical Physics
Maria Sklodowska-Curie National Research
Institute of Oncology Kraków Branch
Garncarska 11, 31-115 Kraków, Poland
[c]Cracow University of Technology
Faculty of Materials Science and Physics
Warszawska 24, 31-155 Kraków, Poland
[d]Department of Experimental Particle Physics and Applications
Faculty of Physics, Astronomy and Applied Computer Science
Jagiellonian University in Kraków
Łojasiewicza 11, 30-348 Kraków, Poland
[e]Total-Body Jagiellonian-PET Laboratory, Jagiellonian University
Łojasiewicza 11, 30-348 Kraków, Poland
[f]Department of Chemical Technology
Faculty of Chemistry of the Jagiellonian University
Gronostajowa 2, 30-387 Kraków, Poland
[g]Institute of Nuclear Physics Polish Academy of Sciences
Radzikowskiego 152, 31-342 Kraków, Poland

Received 20 September 2022, accepted 10 November 2022, published online 29 November 2022

We present building blocks that make up the software stack being developed as part of the Dose-3D project that aims at constructing a next-generation active medical phantom for spatial therapeutic dose distribution reconstruction. The architecture of the custom `G4RT` application is

* Presented at the 4th Jagiellonian Symposium on *Advances in Particle Physics and Medicine*, Cracow, Poland, 10–15 July, 2022.

† Corresponding author: `jhajduga@agh.edu.pl`

discussed, and Python-based packages for high-level data processing and analysis are introduced.

DOI:10.5506/APhysPolBSupp.15.4-A8

1. Introduction

1.1. Project overview

The Dose-3D [1] project aims at creating the next generation of medical phantom. More specifically, it will be a three-dimensional modular [2] and configurable active detector featuring high granularity. The active material of the said detector is a plastic scintillator. A voxelised phantom will be capable of directly measuring the spatial distribution of the deposited therapeutic dose. The project will use plastic scintillators produced by casting, and polymerization in the bulk [3, 4] and 3D printed by digital light processing [5]. The scintillators will be read out by the photomultiplier via polymer optical fibres, and the light-tight head housing with scintillators is also 3D printed using fused deposition modelling. However, in addition to preparing the device itself, it is necessary to develop the high-end software for dose simulation, configuration, and control of the entire device and, finally, the data processing and analysis.

1.2. Software-stack

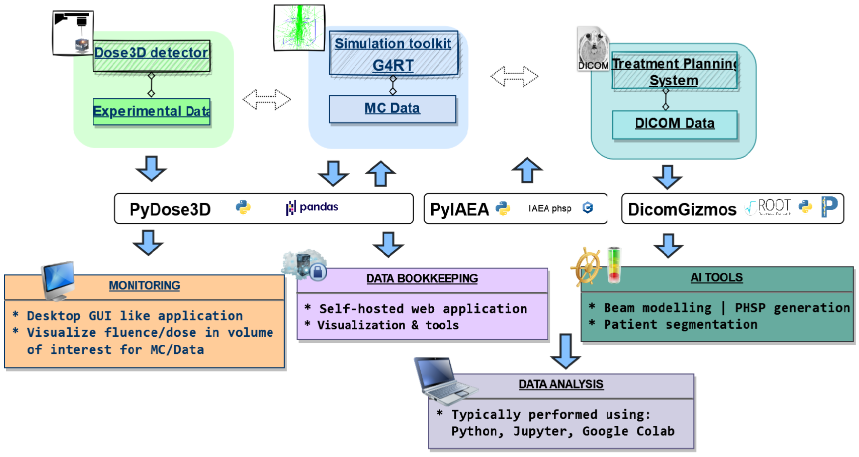
As a part of the comprehensive software stack being developed in the project, a number of data analyses and processing packages are being developed. The core elements of this project are tools for analyzing the raw data, machine learning, Python/C++ modules, along with the medical physics-specific software for DICOM (Digital Imaging and Communications in Medicine) [6] standard. By using programmes developed for the project, we were able to avoid dependence on payware, heavy software or software with functions not required by the project. In addition, when using third-party software, it will still be necessary to develop our own software for handling data of various types from different sources; therefore, once we have standardised output data, it seems much more convenient to handle it in a manner which is familiar and fully under our supervision. As Monte Carlo (MC) simulations are considered the gold standard for high-accuracy data generation, the development of a custom MC simulation platform is essential for the project. By using the in-house simulation platform called G4RT implemented on top of the Geant4 [7] framework, we will produce data that mimics real apparatus. It was recognised that the development of our own software to meet the needs of the project was essential, as the solutions available on the market (such as GATE, Topaz or more commercial solutions such as PRIMO) do not allow for full proof of world creation due to various limitations (GATE and Topaz use parameterised interfaces based on macros, not allowing for full freedom of world creation and code control) or for the openness of data flow (in PRIMO — gantry geometrics or some of their solutions are proprietary). The data will be used to optimize the parameters of the Dose-3D cell and ensure proper calibration of the prototyped phantom. As a team, we have plans to publish the application along with the source code for the general public before the finishing of the project. Until this software is ready to be published, we will continue to develop it and create more refined versions.

2. High-level software stack

During the designing and development of the software, it was decided to implement a high-level interface developed as independent packages, which has made it possible to separate various key functionalities from each other and allows some functionality to be used independently of the overall environment. In order to ensure the uniformity of the data format on the input and output of the individual parts of the workflow, as well as to enable data exchange between Python and C++ environments, it was decided to use Dataframes. The use of Dataframes implemented in the pandas package makes it possible to ensure that we are using the correct type of individual data sets for the package and will also allow us to implement unit tests in the future. Currently, data integrity and homogeneity checking is carried out by project staff.

2.1. Global software dependencies and data flow in the environment

When designing the execution environment for the G4RT simulator, it was crucial to plan an appropriate dependency structure and data flow. This was to ensure both well-structured and documented functionality and for proper management of the data formats and information flow. Consistently, the parts of our environment can be used independently of each other as standalone packages — this brings benefits such as the ability to perform data analysis in a Jupyter environment [8] or to perform work on data in DICOM format without running the simulation itself. This gives the ability to perform more specific tasks through smaller and lighter sub-applications of the environment. The path taken during the development of the Dose-3D software stack was set to meet the challenge of creating tools and applications that are easy to maintain, allows data preservation and grant people with different levels of programming experience a chance to work. The structure discussed here is shown in Fig. 1. One of the challenges we met along the way of creating a workflow and unifying the data format was to connect environments working in both Python and C++ with each other. Following, it was necessary to make sure that the data being compared — in the case of analysis — shared the same structure (units, order of magnitude, scale) as the others. To achieve this, we studied in detail the data formats proposed by PRIMO, NIO (National Research Institute of Oncology) for experimental measurements (carried out there on the water phantom of square fields), and then decided on a method for unifying their format — the use of the previously mentioned pandas data frames here helped achieve data homogeneity at this level. When working on data in both DICOM and IAEA phsp (International Atomic Energy Agency — phase-space) [9] formats, it was necessary to understand the data standard and then implement a set of tools to parse the collected data. Since both formats can be used in G4RT, it used elements of the pybind11 library, which allowed Python libraries to be used inside the G4RT code, as well as allowing the Python tools themselves to call out directly code developed in C++ — allowing data to flow directly between the two without the need for intermediate formats.



```
* Desktop GUI like application
* Visualize fluence/dose in volume
  of interest for MC/Data
```

```
* Self-hosted web application
* Visualization & tools
```

```
* Beam modelling | PHSP generation
* Patient segmentation
```

```
* Typically performed using:
  Python, Jupyter, Google Colab
```

Fig. 1. Dependencies between different parts of the Dose-3D software environment.

3. G4RT application

3.1. Data flow inside application

The G4RT application uses a novel and modern approach to building applications in C++. However, this necessitated a change in approach to the typical application written on Geant4. One of the most noticeable differences is the abandonment of the so-called messenger classes for user interface development. In the case of the classic use of messenger classes, they are the only method for communication between the application and the user (configure applications and provide some interactivity). As the developers of Geant4 themselves note: "Writing a messenger is not complicated, but it is very tedious and a hurdle to maintain" [10]. Currently, in the Geant4 repository, there are over 500 different messengers. The entire handling of the application API is put over to a new type of class, so-called services. The whole idea of services allows you to create fewer elements that can communicate with the user by taking configuration and storing it as class elements, where also, all of this process happens at the application development level. The services can be divided into two types — those responsible for preprocessing as well as the main service — the global configuration service. They manage of user communication and management of all configurable elements of the application, such as materials, geometry parameterization or run-time. To better illustrate the dependencies in the G4RT, an information flow model for a single simulation run is presented in Fig. 2. In addition, Fig. 3 presents the factory-like mechanism handled with a number of services for determining the initial conditions of a simulation based on a plan from the Treatment Planning System (TPS).

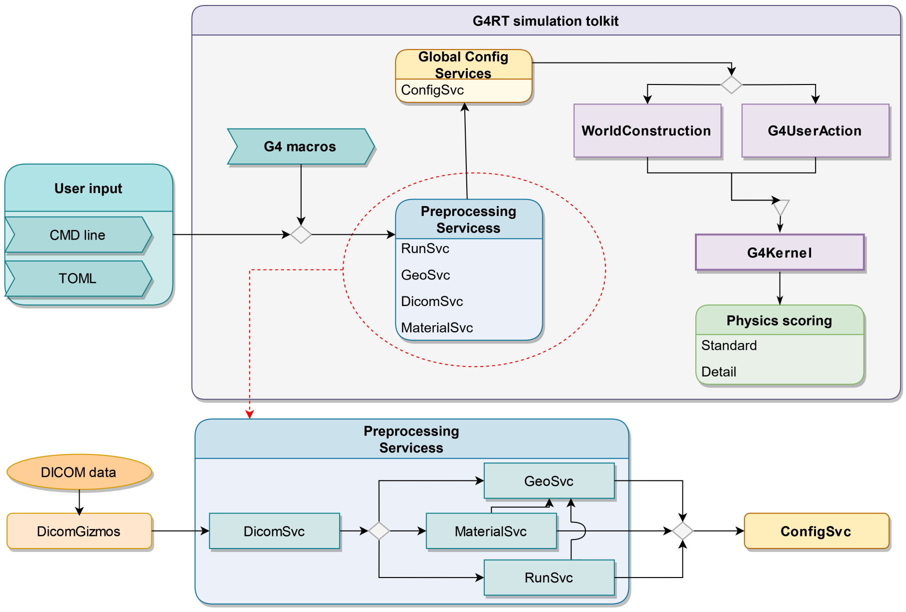


Fig. 2. The information flow model for a single simulation run within G4RT application.

3.2. Particle beam and patient geometry

Another important element that has been proposed is the geometry building system — the geometry service. Being implemented in G4RT, the architecture of this service allows easy world scaling for gantry elements as well as for patient models. The configuration of the entire system is divided into a beam modelling stage and a phantom generation stage, as shown in Fig. 3. As can be seen in the graph, the G4RT application allows for different ways to generate the particle beam as well as the patient model. This will allow for the reproduction of Treatment Plans as well as defines the simulation platform as a reference application for the Dose-3D experiment.

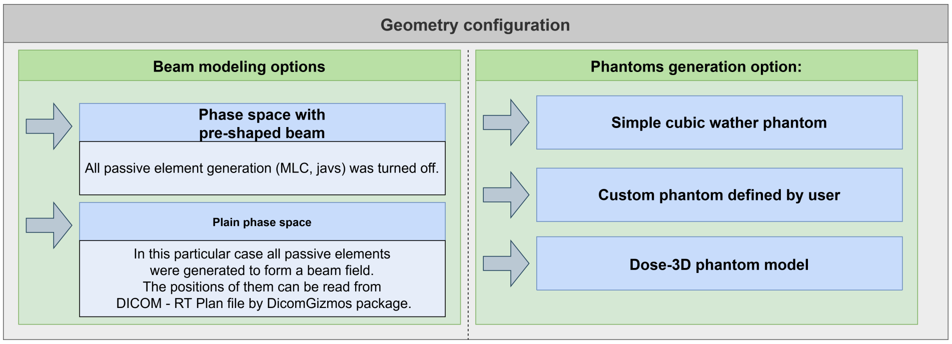

Geometry configuration
Beam modeling options
Phase space with pre-shaped beam
All passive element generation (MLC, javs) was turned off.
Plain phase space
In this particular case all passive elements were generated to form a beam field. The positions of them can be read from DICOM - RT Plan file by DicomGizmos package.
Phantoms generation option:
Simple cubic wather phantom
Custom phantom defined by user
Dose-3D phantom model

Fig. 3. Chart showing the different methods of configuring beam and patient geometry.

4. Python-based utilities

4.1. PyDose3D package

A set of key high-level data analysis tools has been gathered together in the Python package called PyDose3D. This allows for standardizing the tools and plotting styles for the data analysis across the whole collaboration. As an example of the module, such as a tool that allows for a simple and comprehensive presentation of dose data, their analysis as well as comparison with each other have been collected (Figs. 4 and 5). In order to provide an example comparison of possible data sets, it was decided to present a comparison of the dose distribution from the squared field in the water phantom in PDD (percentage depth dose curve) profile.

4.2. In-house interfaces for external libraries

A set of other Python-based tools developed during the project includes those for the DICOM file format — DicomGizmos package — which provides the ability to handle the input data for MC simulation (DICOM-RT from

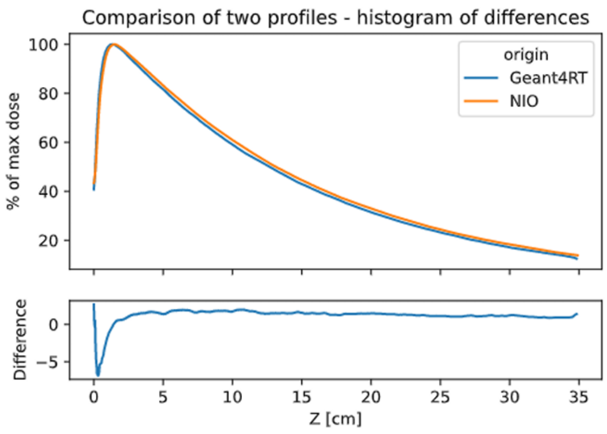


Fig. 4. PDDs obtained from Geant4-RT simulation and measurement fields with dose differences in the function of depth.

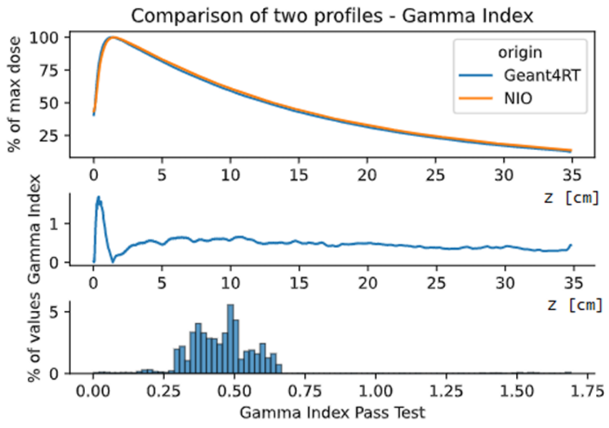


Fig. 5. PDDs obtained from Geant4-RT simulation and measurement fields with gamma index values distribution and in the function of depth.

TPS), as well as enabling the creation of *para*-CT images based on GDML (Geometry Description Markup Language) geometry format. The last functionality allows geometry created within typical Geant4 applications to be exported and used in medical software. In addition, CT generated in this way is free of noise and ensures the preservation of sharp edges, which can prove useful within simple simulations or studies performed in TPS systems. The PyIAEA package consists of a refactored of the original IAEA library as well as the Geant4 event-oriented history of particles. This phase-space reader has been tightly coupled with Python, which was accomplished with minimal dependencies on external libraries and tools. In Fig. 6, the end-user interface is shown that can be utilized for IAEA phase-space exploration (possible to be performed with Pandas Dataframe format).

```
from iaea.utils import Reader
if __name__ == "__main__":
    data = "path-to-data/iaea-phsp"      # file w/o extension
    reader = Reader(data)                # initialize the reader utility
    reader.SetColumns(['X','Y','E'])     # specify variables to read-in
    df = reader.GetParticles(10000)      # read-in specified statistics to Pandas Dataframe
```

Fig. 6. A code snippet demonstrating the simplicity and intuitiveness of a high-level Python interface to support IAEA libraries.

5. Summary

The groundwork for the architecture and development of the software stack was described. The in-house MC simulation platform is being developed as well as Python-based utilities were described in detail.

The POIR.04.04.00-00-15E5/18 project is carried out within the "TEAM-NET" programme of the Foundation for Polish Science, co-financed by the European Union under the European Regional Development Fund.

REFERENCES

[1] Dose-3D Collaboration «TN-Dose-3D Project website» https://dose3d.fis.agh.edu.pl/en/projekt-dose-3d-z-programu-team-net-fnp-eng/.

[2] P. Jurgielewicz *et al.*, *Nucl. Instrum. Methods Phys. Res. A* **1045**, 167607 (2023).

[3] Ł. Kapłon, G. Moskal, *Bio. Algorithm Med. Syst.* **17**, 191 (2021).

[4] Ł. Kapłon *et al.*, *Radiat. Meas.* **158**, 106864 (2022).

[5] D. Kulig *et al.*, Comparison of cell casted and 3D-printed plastic scintillators for dosimetry applications, submitted to *Radiat. Prot. Dosim.*

[6] R.N. Graham, R.W. Perriss, A.F. Scarsbrook, *Clin. Radiol.* **60**, 1133 (2005).

[7] S. Agostinelli *et al.*, *Nucl. Instrum. Methods Phys. Res. A* **503**, 250 (2003).

[8] M. Beg *et al.*, *Comput. Sci. Eng.* **23**, 36 (2021).

[9] M.A. Cortés-Giraldo, J.M. Quesada, M.I. Gallardo, R. Capote, *Int. J. Radiat. Biol.* **88**, 200 (2012).

[10] N. Mori, *Nucl. Instrum. Methods Phys. Res. A* **1002**, 165298 (2021).