PHITS DynamicMC User Manual

Cyclotron and Radioisotope Center
(CYRIC)
Tohoku University, Japan

June 16, 2023

Contents

1 Background

The DynamicMC method is a proposed technique to evaluate dosimetric quantities in a 3-dimensional radiation field. In everyday life we are continuously exposed tp ionizing radiations from variety of sources and the deposited energy from these ionizing radiations can have detrimental effect on our health. Furthermore, occupational workers that are dealing with radioactive materials and nuclear radiation may be at a higher risk. Considering these, it would be important to determine the transport and energy deposition of these ionizing radiations in a more realistic fashion. The DynamicMC platform offers users with a great flexibility to model multiple irradiation setups in a 3-dimensional radiation field; this functions similar as taking static snapshots of a system at different timestamps (*i.e.*, t_1, t_2,...,t_n). These timestamps resemble the dynamic variation of the dosimetric quantities and hence the name DynamicMC.

Previously, we have developed DynamicMC for modelling relative movement of ORNL phantom in a radiation field for the Monte Carlo N-Particle (MCNP) package (Health Physics. 2023, 124(4):301-309). The present work reports the development of a new version of the DynamicMC program for Particle and Heavy Ion Transport code System (PHITS) Monte Carlo (MC) package. Considering the wide use of PHITS MC package and its rapidly growing community, it would be rather pertinent to further develop the DynamicMC to provide support for PHITS users. In this user manual the structure of the PHITS DynamicMC program, its functions and operations are discussed. In addition, information regarding building from source code was provided for users that are interested to modify and further develop the DynamicMC for PHITS MC package.

Users are free to modify, redistribute or use the present program without any restrictions, please see the license section for more information regarding the GNU General Public License version 3.

2 Structure of the program

The present program has seven main parts, namely, (1) graphical user interface (GUI), (2) graphical coordinate input, (3) dynamic phantom generator, (4) simple script/code viewer, (5) a simple collision detection module, (6) shielding generator option and (7) sumtally generator module. In addition, the

present version of the program supports two operational mode that are; (1) monoenergetic (*Mono E*) and radioisotope (*RI*) modes and two coordinate input options that are; (1) manual and (2) graphical modes.

The monoenergetic (*Mono E*) mode offers the functionality to model protons, neurons, photons, electrons and alpha particles with discrete energies. The radioisotope (*RI*) mode offers the possibility of modelling different radioisotopes with exact energy distribution and decay mechanism; this mode uses the *e-type = 28* mode offered by PHITS MC package to accomplish this task.

The GUI was developed in C++ programming language using the QT5 libraries (`https://www.qt.io/`). The GUI has an embedded graphical coordinate input module that provides the ability to manipulate the movement of radioactive source, human phantom and shielding material; this graphical coordinate input was developed in C++ programming language using OpenGL application programming interface (`https://www.opengl.org/`).

The main part of the program is the phantom generator that can be used to generate user-defined phantom irradiation instances. The phantom irradiation scenario will be generated by taking the user inputs from the GUI. Users have the ability to model variety of different radioactive source or beam types and energies for cases with or without shielding materials. All these options were explained in more details in following parts.

3 The Graphical User Interface (GUI) functions

The GUI for the new version of DynamicMC has been designed to have be more features. A snapshot of the program interface is shown in Fig. 1. The users can set the type of the incident particle, beam energy, tallies, maxcas, maxbch, step value (used to define increments in the dynamics movements) and five different sliders to manipulate the x and y positions of human phantom and the radioactive source, and the source z position with respect to the height of the modelled human phantom for the graphical coordinate input option. Considering, the manual coordinate option, users have the ability to define the position of the radioactive source and human phantom. In addition, the shield option provides the users with the possibility of modelling a rectangular parallelepiped shield with user defined materials and density; this option is only available for graphical coordinate input option. The users have the ability to manipulate the x, y and z positions and also the length and width of

the modelled shield using X, Y, H, L and W sliders, respectively. The black color window under *"Dynamics"* shown in Fig. 1 is the interactive graphical coordinate input based on the OpenGL. The red and blue squares represent the radioactive source and human phantom, respectively. The green rectangle represents the modelled shielding between the radioactive source (*i.e.*, red square) and the human phantom (*i.e.*, blue square). The buttons embedded into the GUI are *Gen*, *Run PHITS*, *View code*, *Quit*, *F5*, *Dev info.*, *RI*, *Mono E*, *manual* and *graphical*.

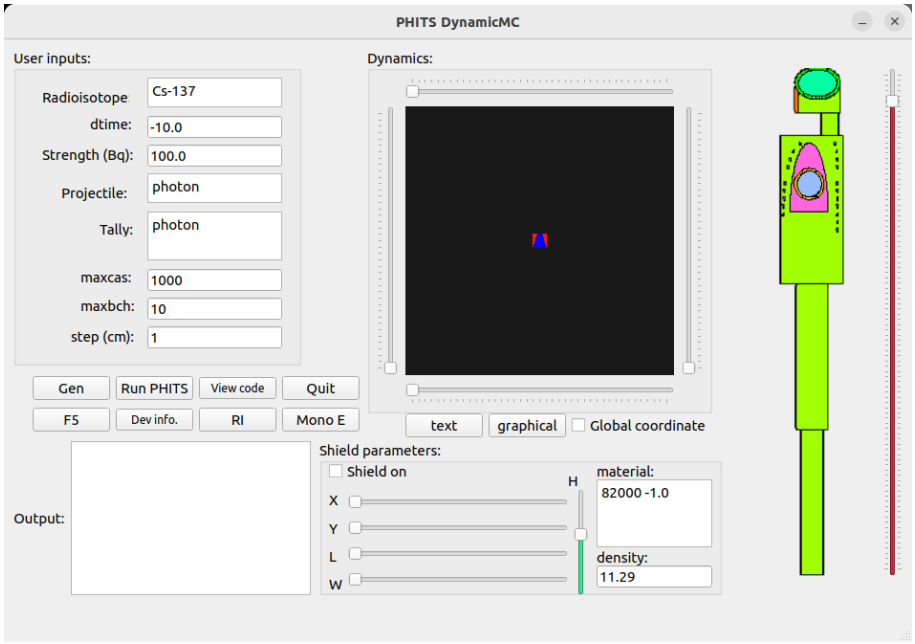

Figure 1: Snapshot of the GUI program in action.

There are two different operational modes the users can set to generate their models. The first is the *RI* mode which stands for RadioIsotope mode (which is set to be the default mode in the present program and model) and can be accessed by clicking the *RI* button. The other option would be the *Mono E* mode that enables the users to model different incident particles with user defined energy and tally options. After clicking the *Mono E* button, the GUI structure will automatically change as shown in Fig. 2.

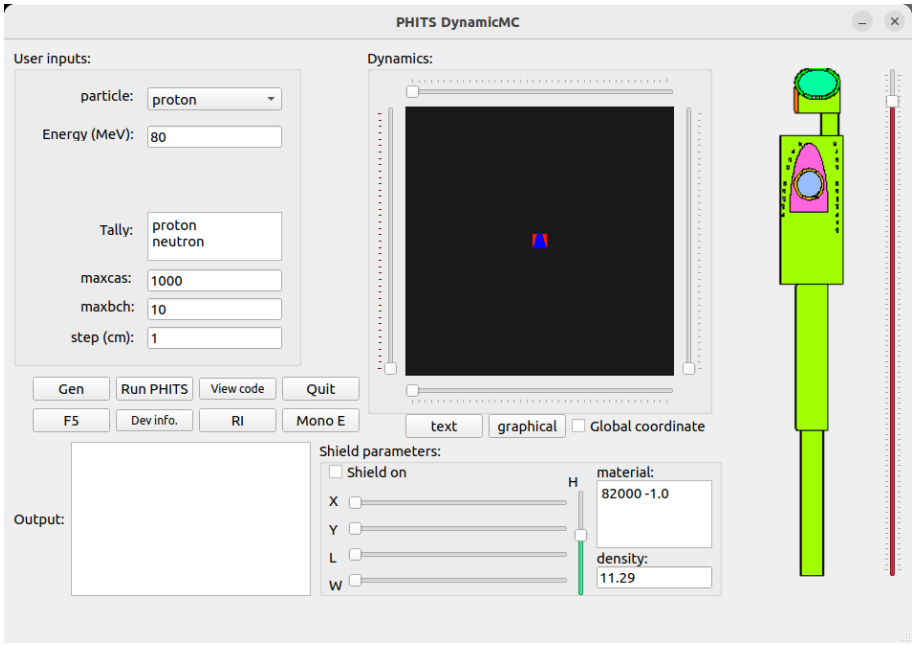

Figure 2: The *Mono E* mode GUI structure after clicking the *Mono E* button.

The *RI* mode will provide the users with the possibility of defining the radioisotope in form of a string (Cs-137, Co-60 and etc.). The *dtime* and *Strength in Bq* are the option for time evolution and source strength. The next two options are the type of the projectile emitted from the radioisotope and the particle that needs to be tallied. The preset example for the *RI* mode is for the modelling of Cs-137 radioisotope.

The *Mono E* mode is shown in Fig. 2 and the users are only required to choose a particle type from the menu. The menu supports four different incident particle types that are; (1) proton, (2) neutron, (3) photon and (4) alpha particles. The next entry would be the energy of incident particle in MeV that users can set. The preset example for the *Mono E* mode is 80 MeV proton irradiation with proton and secondary neutron tallies.

The total number of iterations can also be set by the users using *maxcas* and *maxbch*. The *step* variable would be used to determine the increment of the dynamic movements and it has been discussed in more details in the following sections.

4 Anthropomorphic phantom generator

The phantom that was used in our program is shown in Fig. 3. This phantom is less computationally expensive compared to the previous version that contained almost every organs in the human body.

The modelled phantom composed of internal organs in the upper part of the human body, mainly the chest and above (see Fig. 4).

Entire model of the phantom was written in FORTRAN90 programming language. The position of the phantom that is controlled by the user input, will be passed to the phantom generator module and position of every organ would automatically change based on the x and y locations taken from the dynamics part of the GUI. Another important feature that was implemented in the phantom generator module is the autogenerated tallies that will be implemented based on the number of tallies set by the user. The 2-dimensional energy deposition tallies for the entire modelled setup would be generated. In addition,dose deposition tallies in left and right lung, heart tissue and blood, spine and discs, brain and head-face skin would also be autogenerated using region based mesh for a specific cell containing these organs. The generated phantom faces toward the positive y-axis and the direction at which the phantom faces is shown as the *narrow upper side* of the blue mark shown in the black color OpenGL window (see Fig. 1).

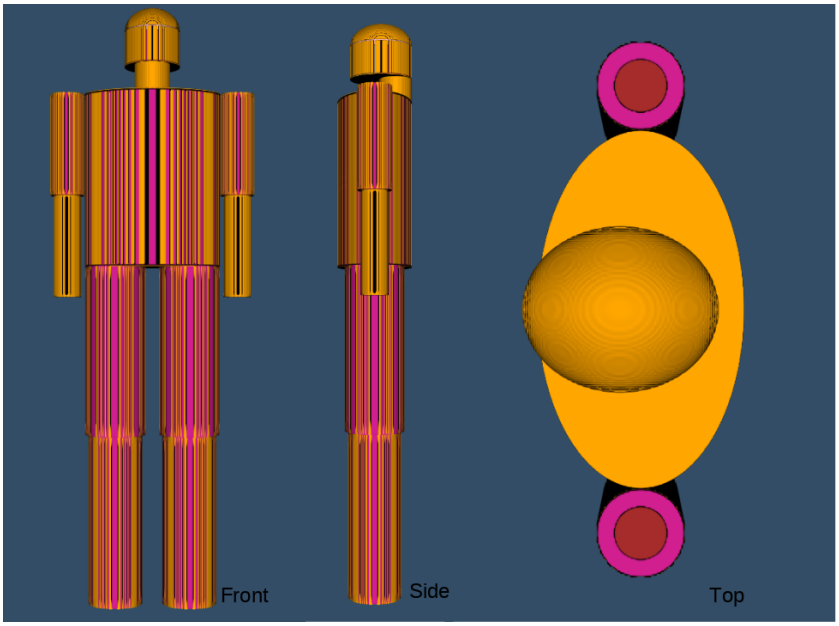

Figure 3: The anthropomorphic phantom embedded into the present version of the DynamicMC.

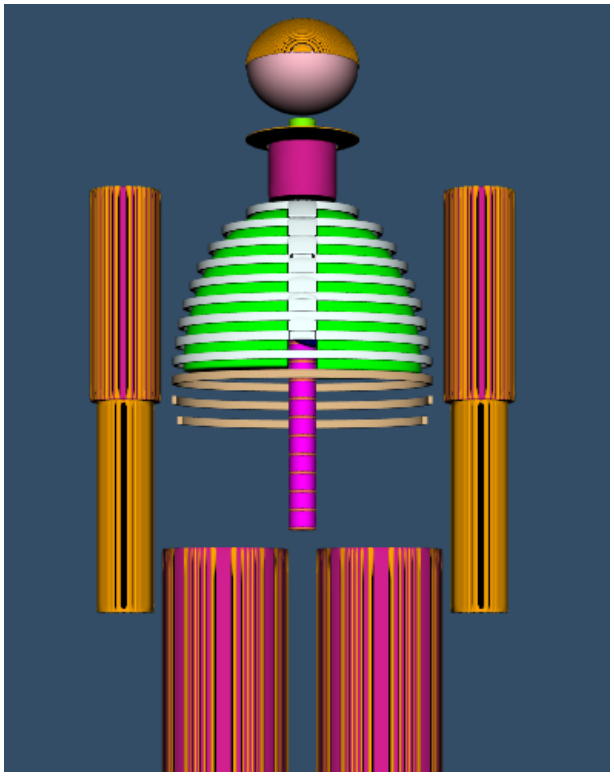

Figure 4: Internal organs considered in the present anthropomorphic phantom.

5 Dynamics of radioactive source and human phantom

The dynamic part of the present DynamicMC is controlled using an embedded OpenGL function for the graphical input option. Considering the graphical coordinate input function, the sliders around the black color window under *Dynamics* tab will be used to manipulate the x and y coordinate of the human phantom and the radioactive source as shown in Fig. 5, the white arrow added to the figure shows the direction at which human phantom faces; this can also be seen from the narrow upper part of the blue marker. The top and left slider (red color) can be used to set the x and y coordinate of the radioactive source. The bottom and right slider (blue color) can be used to set the x and y coordinate of the phantom. After setting these, user must click on the back color window to update the positions.

It needs to be noted that after changing the the source or the phantom position users must click on the OpenGL window to update the location positions accordingly. After pressing *Gen* for an specific source and phantom position, one must either set the source and phantom position once again by using the sliders or clicking the *F5* button which has the same meaning as *refresh* function in most

widely used computer programs. Clicking the *F5* button after run, the position of the source and phantom that was previously used will be loaded into the program buffer. It is a good idea to click the OpenGL window (the black color window under dynamics) to refresh the OpenGL position plot to ensure that the relative source to phantom position remained unchanged.

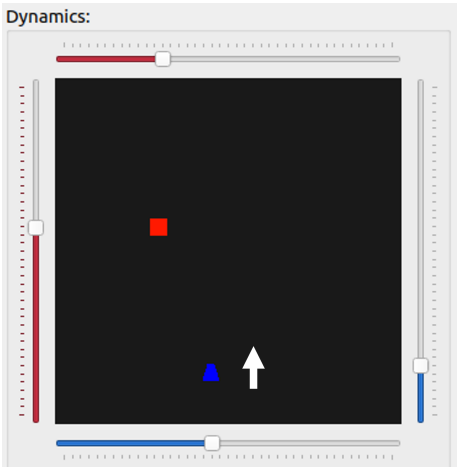

Figure 5: Position selection and manipulation in graphical mode using the OpenGL module. (Note: the white arrow was added to the figure and shows the direction at which the phantom faces which is same as the narrow upper part of the blue marker, the current view is from the top)

Considering Fig. 5, the red and blue color markers represent the source and phantom, respectively. The OpenGL window shown in Fig. 5 provides the users with the ability to get a relative estimate about the source, phantom and shielding material location. Considering the manipulation of the dynamic movement for both source and phantom, the slider value change for every tick can be set by the *step* variable. In addition, the numerical value for x and y positions will be shown in the *Output* window as the user change the positions using the sliders.

Considering the text coordinate input function, the users have the ability to manually enter the x, y and z location of the radioactive source and also the x and y location of the human phantom. The text coordinate input option is shown in Fig. 6.

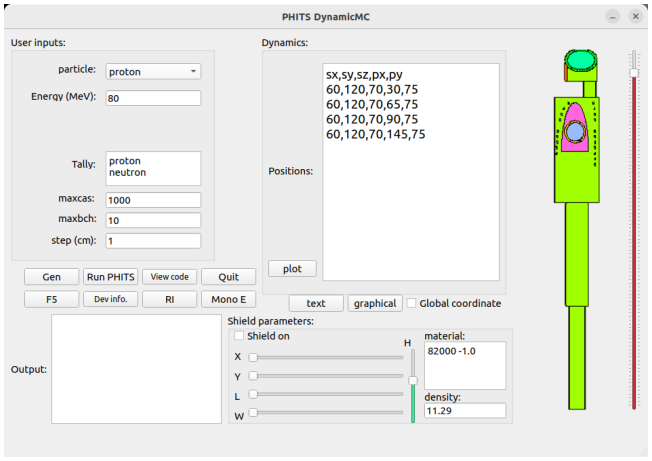

Figure 6: The text coordinate input option accessed by clicking the *text* button.

6 Modelling radiation transport without shielding material

In this section, a step-by-step guide to model a radiation transport example without any shielding material will be discussed using the graphical and text coordinate option. Snapshots from the program for every step is provided for the users to follow using the graphical coordinate option.

1) Run DynamicMC program (you will be seeing Fig. 7)

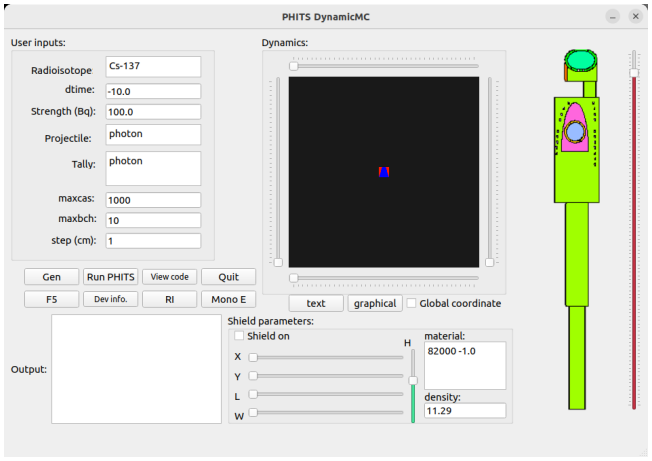

Figure 7: Initial run window of DynamicMC.

2) We manipulate y coordinate of the source using the left side slider as shown in Fig. 8.

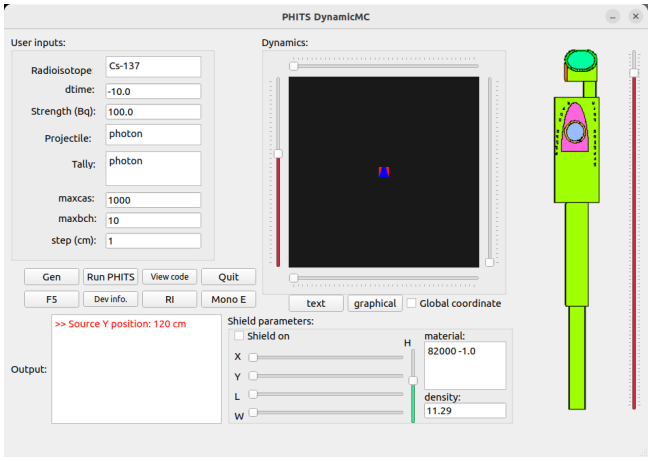

Figure 8: Changing the y coordinate of the source.

3) We manipulate x coordinate of the source using the top side slider as shown in Fig. 9.

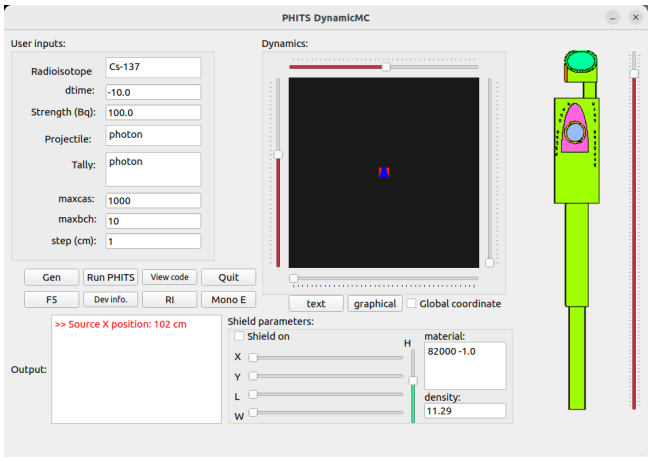

Figure 9: Changing the x coordinate of the source.

4) We manipulate x coordinate of the phantom using the bottom side slider as shown in Fig. 10.

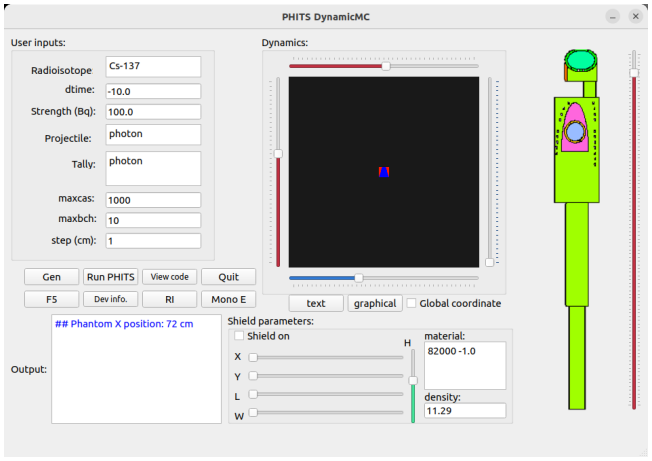

Figure 10: Changing the x coordinate of the phantom.

5) We manipulate y coordinate of the phantom using the right side slider as shown in Fig. 11.

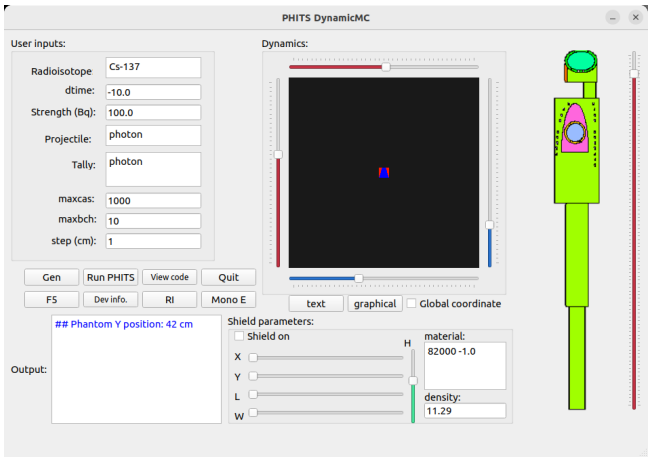

Figure 11: Changing the y coordinate of the phantom.

6) We now update the OpenGL window by left clicking on the black color window and the red (source) and blue (phantom) markers will be update to a correct relative location as shown in Fig. 12.

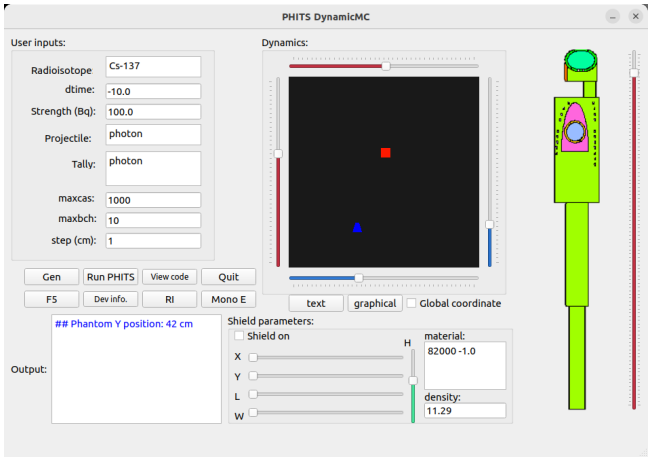

Figure 12: Updating the OpenGL window by left clicking on the black color window.

7) We manipulate z coordinate of the phantom using the slider next to the phantom image as shown in Fig. 13.

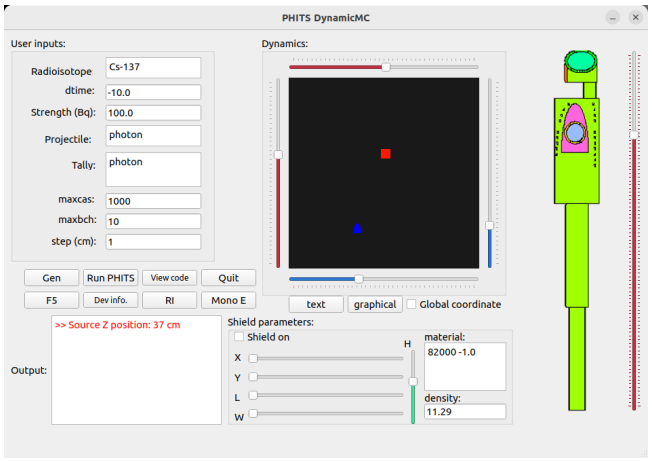

Figure 13: Changing the z coordinate of the phantom.

8) We now would like to enable the Global coordinate option using the checkbox under the *Dynamics* window; this will essentially give us a larger view of the entire model for the 2-dimensional energy deposition. This is shown in Fig. 14.

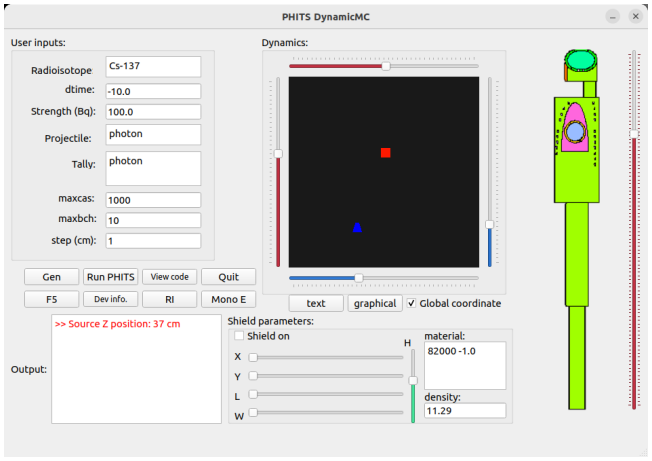

Figure 14: Enabling the global coordinate option.

9) The last step would be to generate our irradiation model by clicking on *Gen* button as is shown in Fig. 15.

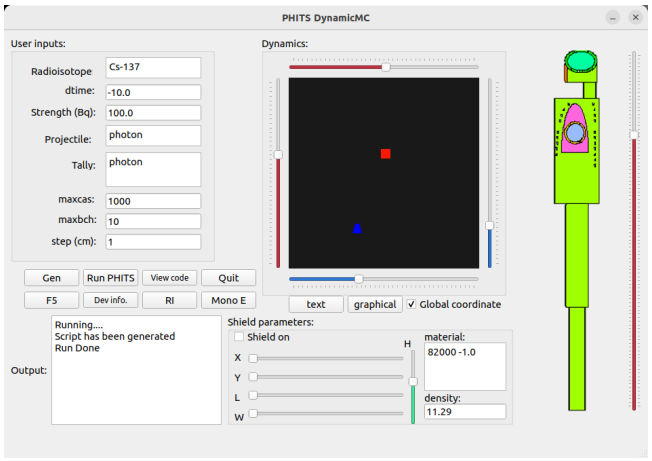

Figure 15: Running and generating the model.

The generated model will be saved under *working_directory* in a *run* folder and in a file named *0000n_code.inp*, where *n* is the number of input files ranging from 1 to 99999. The *run* folders will increase in number when the users open the program and therefore all your runs for a particular session will be kept under a specific run folder. Considering this is your first model, then the generated input file will be named *00001_code.inp* and will be saved under *working_directory/run1/* and all your configuration will be saved in the input file which is ready for a PHITS run. The separation of the *working_directory* and *program_files* make the data sorting and extraction task much easier for the users.

Considering the text coordinate input function, the users can click on the *text* button and enter the radioactive source and phantom position as shown in the preset example (see Fig. 16). It needs to be noted that *sx*, *sy* and *sz* are the x, y and z position of the radioactive source, respectively and *px* and *py* are the x and y coordinate of the human phantom, respectively. The user inputs must be separated by commas and the format must be kept as shown in the preset example. The model can be generated in a similar fashion as the graphical coordinate input option. The step value in the text coordinate input mode would be used to set the tally region and the value of the step must be set as $200\times$*step* cm. Therefore, considering the largest x or y value that user inputs to the text coordinate box, the step value must be set larger than maximum value/200 so that the tally can embrace the modelled domain.

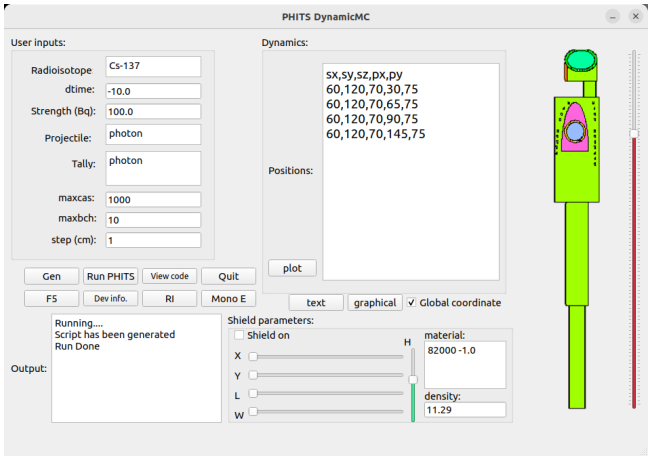

Figure 16: The *text* coordinate input module in action.

7 Position plotter module for text coordinate input

Considering the text coordinate input option, users no longer have access to the graphical OpenGL interface and therefore it would rather be tedious to get a relative estimate of source and phantom position. In order to tackle this issue, we have developed a custom position plotter that helps the users to plot the position of the radioactive source and human phantom in 2D. After generating the input scripts for different positions, the *plot* button under *Dynamics* window can be clicked to invoke the position plotter as shown in Fig. 17.

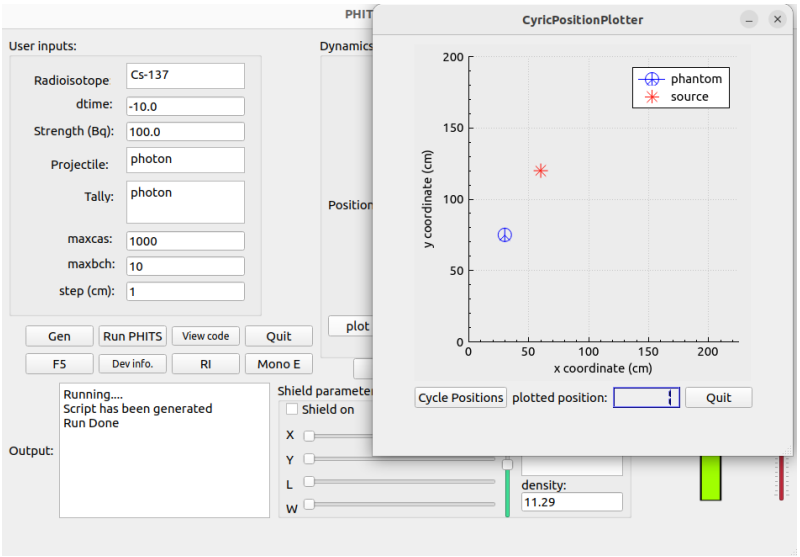

Figure 17: The custom made position plotter program.

The position plotter automatically determines the maximum value in the set of user defined positions and generate the plot for n number of dynamic points. Clicking the *Cycle Positions* button, users can cycle through different irradiation positions.

8 Modelling radiation transport with shielding material

1) Considering the case with shielding material, steps discussed in the previous section for the graphical coordinate input needs to be repeated for source and the phantom, as shown in Fig. 18. it needs to be noted that the shielding case only works under the graphical mode.

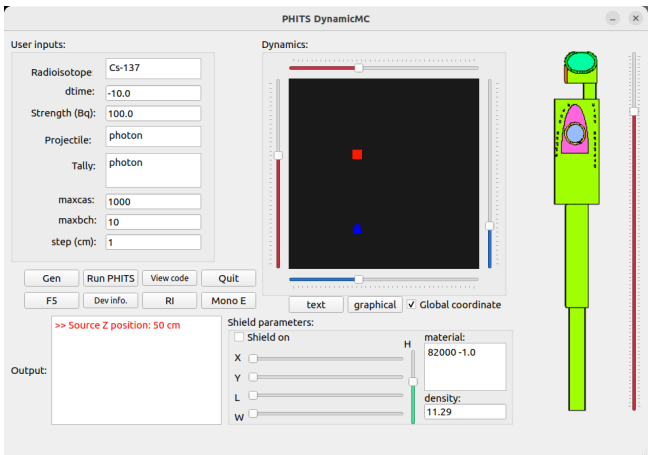

Figure 18: Preparation of setup without shielding.

2) Now we need to turn on the shielding option by clicking the checkbox named *Shield on* as shown in Fig. 19.

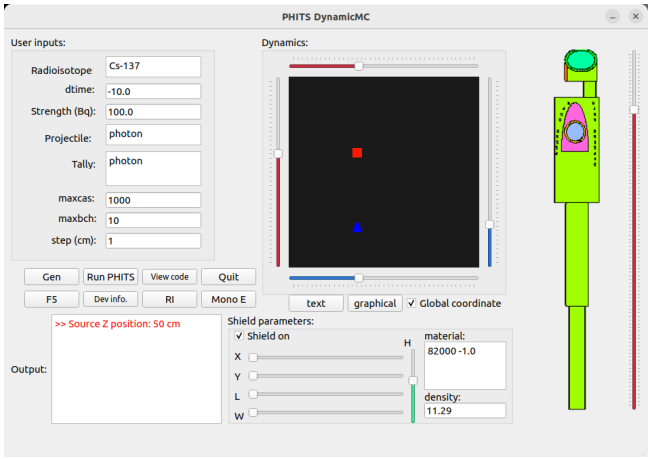

Figure 19: Turning on the shielding option.

3) We manipulate x coordinate of the shield using the X slider as shown in Fig. 20.

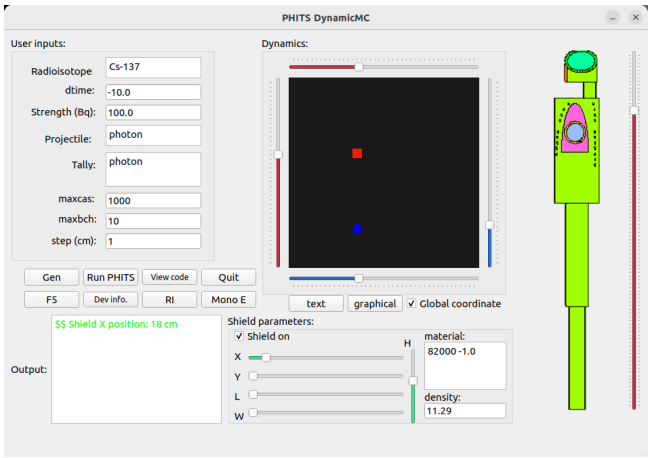

Figure 20: Changing the x coordinate of the shield.

4) We manipulate y coordinate of the shield using the Y slider as shown in Fig. 21.

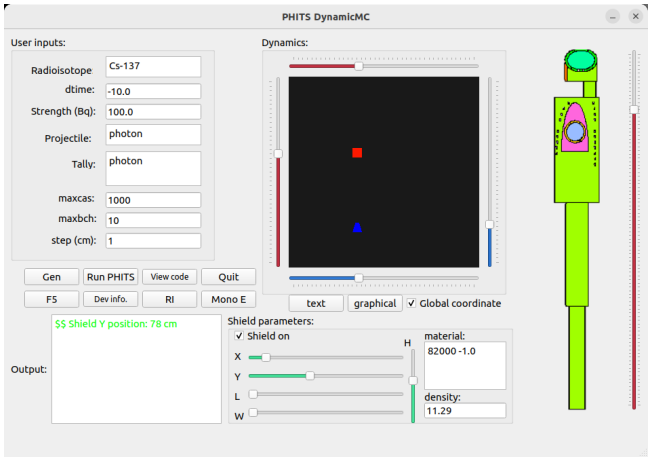

Figure 21: Changing the y coordinate of the shield.

5) We manipulate length of the shield using the L slider as shown in Fig. 22.

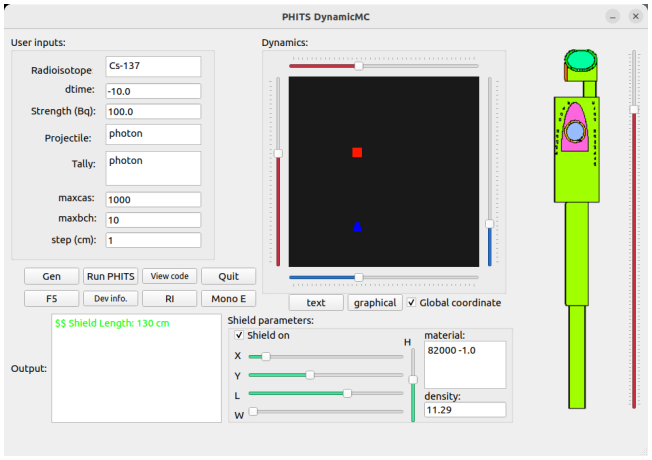

Figure 22: Changing the length of the shield.

6) We manipulate width of the shield using the W slider as shown in Fig. 23.

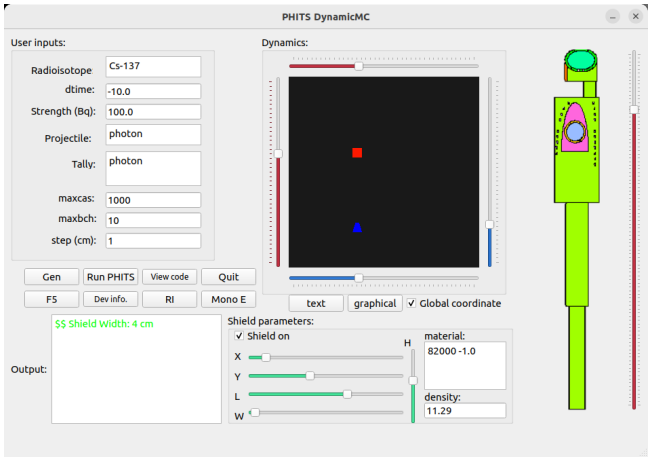

Figure 23: Changing the width of the shield.

6) We now update the OpenGL window by left clicking on the black color window and the red (source) and blue (phantom) squares along with the green (shield) rectangle will be update to a correct relative location as shown in Fig. 24.

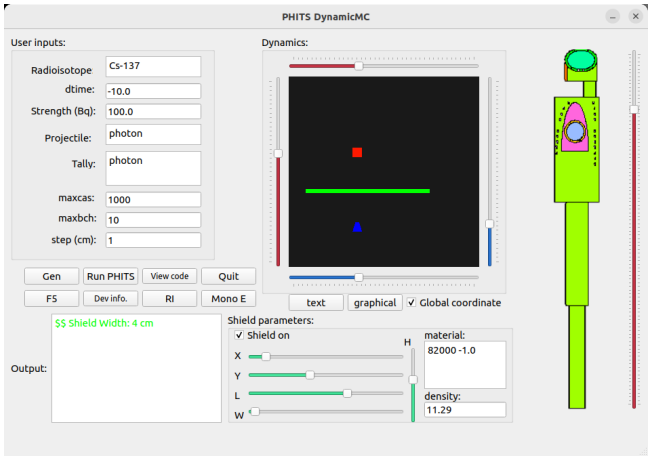

Figure 24: Updating the OpenGL window by left clicking on the black color window.

In order to manipulate the height of the shield, users need to change the value of the H slider. The shielding material can be changed using the $ZZZAAA$ notation at which Z and A are the atomic and mass number of an specific element or isotope. Subsequently, the density of the shield can be set right under the material definition which is in g/cm^3. The current example is set to be natural lead (Pb) with the atomic number of 82 and density of 11.29 g/cm^3. As usual you must click on the *Run* button to generate the input script. The naming will be similar to that discussed in the previous step. Users can run the generated script using PHITS.

9 The 2 and 3 body collision detection module

In order to avoid any errors in the location selection for both without and with shielding cases, we have developed and implemented 2 and 3 body collision detection modules, respectively. The two cases for collision detection uses the user defined x and y location of source, phantom and the shield to determine if there will be any overlaps. In case of an overlap, the program will report an error message as shown in Figs. 25 and 26 and the run will be terminated without generating any input files.

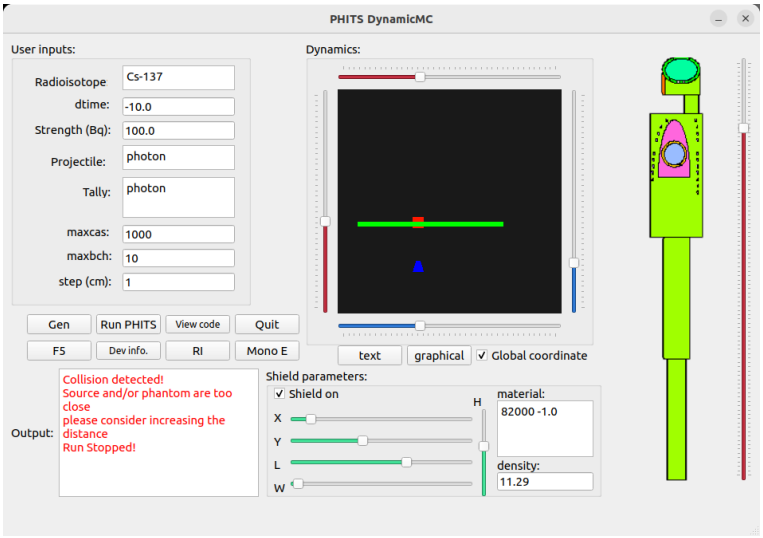

Figure 25: Collision detection between the source and the shield.

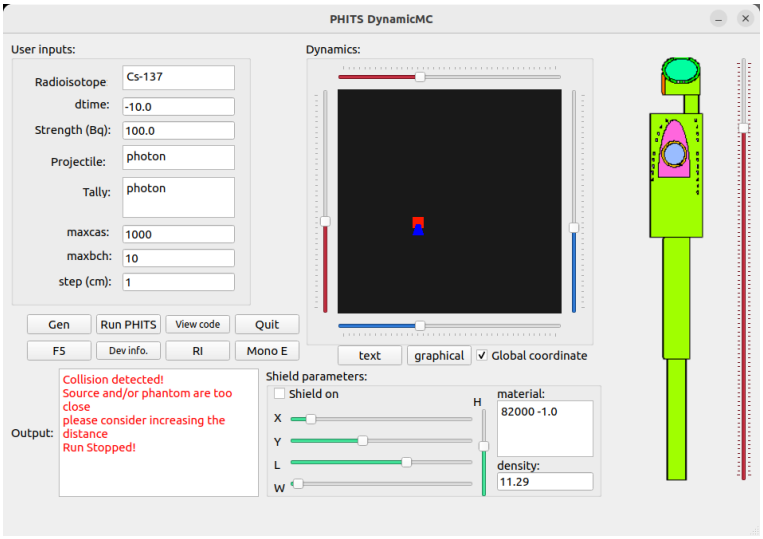

Figure 26: Collision detection between the source and the phantom.

In case of getting this error, users must maintain a good distance between the source, phantom and the shield. This is important since the present work considers an external source therefore the source cannot be placed inside the phantom domain. Similarly, shielding material cannot be placed inside the phantom or the source.

10 The simple script/code viewer

A simple script/code viewer was developed and can be used to only view the latest generated script. After the user generated the script by clicking the *Run* button, the *View code* button can be used to invoke the built-in simple script/code viewer to inspect the elements in the generated script. The snapshot of the simple script/code view is shown in Fig. 27.

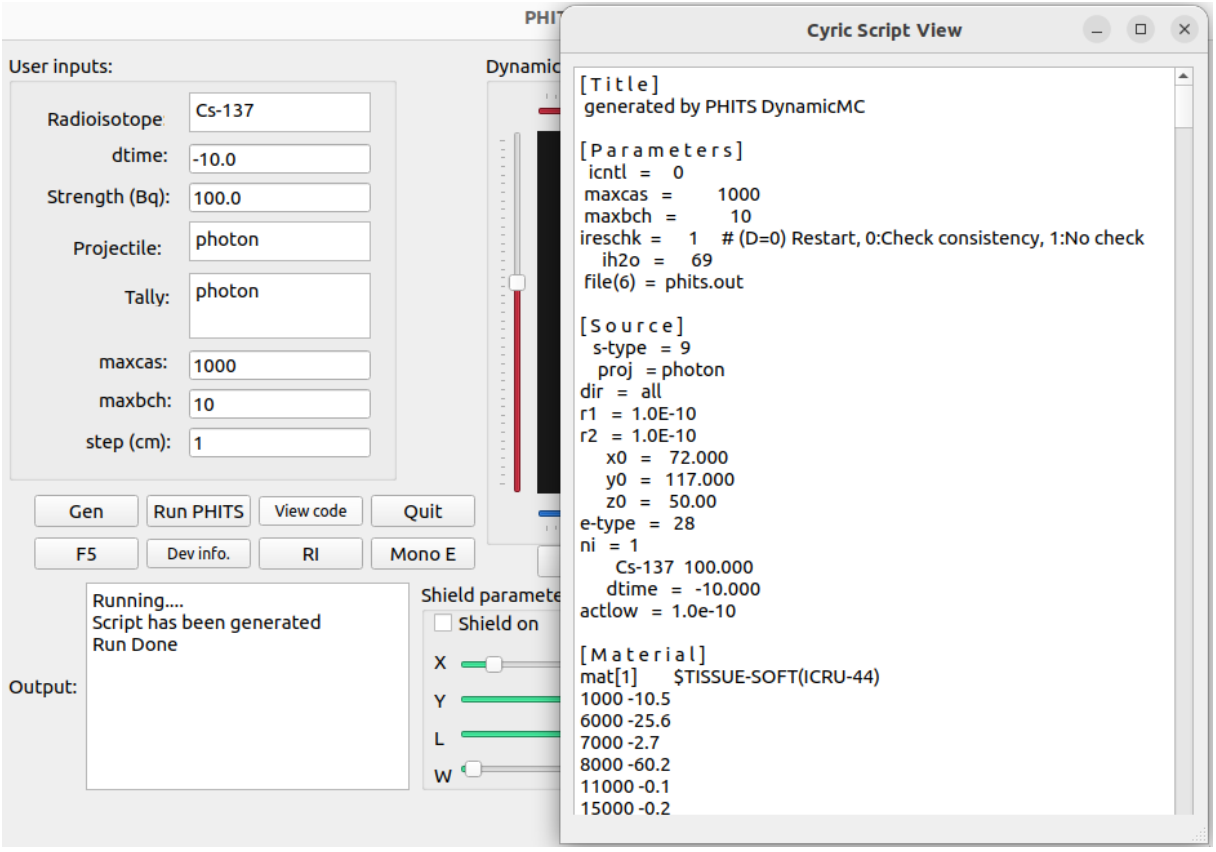

Figure 27: The simple script/code viewer in action.

It needs to be noted that the script viewer cannot edit the generated script/code; this is mainly to avoid any potential issues. If you wish to edit the generated script you can use any text editors that you prefer on your desired operating system.

11 Autogenerated tallies and custom organ tally plotter

The present program generates tallies according to the user-defined particles in the tally section; this is valid for both *RI* and *Mono E* modes and also *text* and *graphical* options. The 2D energy deposition tallies will be generated from top, side and front views so that the radiation dispersion and subsequent energy deposition can be observed. In addition, dose deposition organ tallies for set of selected organs would be generated that provide the value of dose in each of these organs in the unit of Gy/source. Lastly, the *sumtally* would be automatically generated to provide the weighted average dose results for each of the selected organ for n number of irradiation positions.

The *sumtally* results would be important when determining the weighted averaged dose in selected organs as the human phantom and/or the source moves relative to one another; this in fact resembles the realistic scenario at which individuals would move in a radiation contaminated environment or in general *a 3-dimensional radiation field*. The *sumtally* will be generated for all dynamic points that were entered by the user and the results will be saved in *n+1* run directory under *working_directory* folder. For example, if the user simulated 4 different positions, there will be *00001_code.inp*, *00002_code.inp*, *00003_code.inp* and *00004_code.inp* files in *run1* folder under *working_directory*, given the fact that it is the first run. Then the program runs these results and generate an extra input file that determines *sumtally* results in a file named *00005_code.inp*. In the *sumtally* file, the *icntl = 13* and the path to organ tallies would be automatically set based on the run number. Users can test this easily using the text coordinate input mode with the preset example. In order to enhance the usability of the present program, we have developed a custom organ tally plotter program that is bundled with the PHITS DynamicMC. The present tally plotter program is only functional for organ tallies, both for every single run and also for *sumtally* output files. The tally plotter is accessible under TallyPlot directory in the program bundle (see Program directories and data curation section for more information). The snapshot of the tally plotter is shown in Fig. 28.

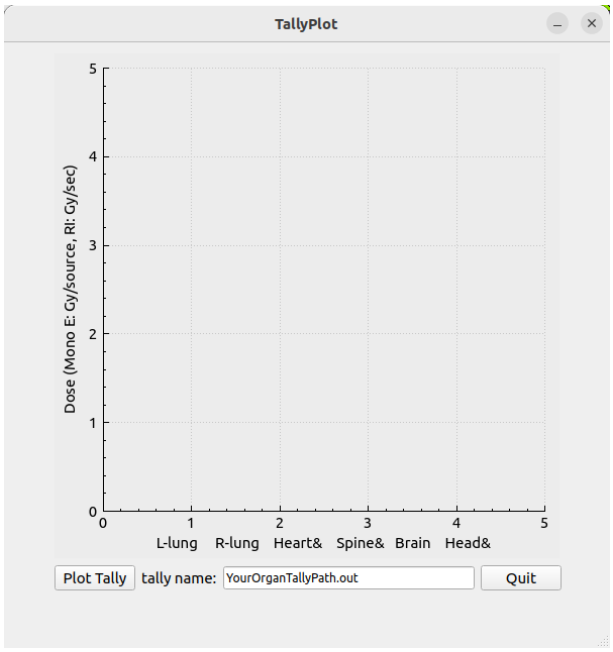

Figure 28: The custom organ tally plotter program in action.

The custom tally plotter takes the path to the *.out* organ tally results files generated by PHITS MC package. Users must clear the tally name and copy and paste the path to the *.out* file of organ tally. For example on GNU/Linux machine the path will be;
file:///home/Your_Username/Desktop/phitsDYMC/working_directory/run8/00001/organ_deposit_photon.out
In this case, we are plotting the organ tally in the first position (00001) in the 8th run folder.

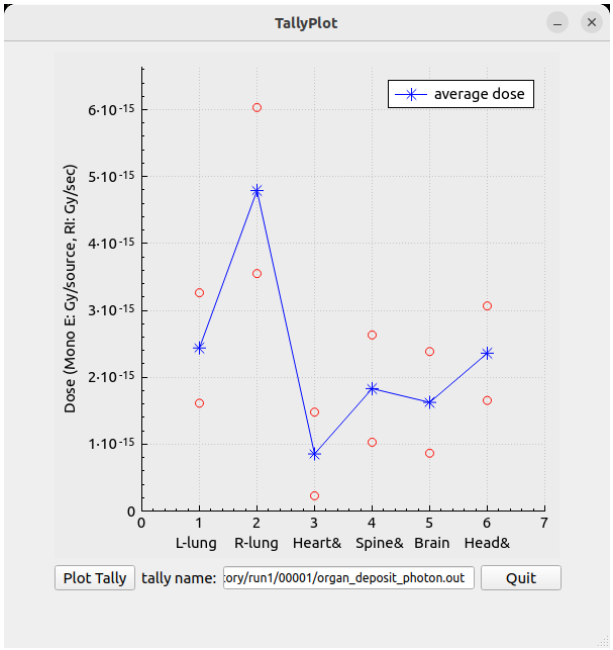

Figure 29: The custom organ tally plotter program showing tally results.

The blue asterisk symbol connected by the blue line represent the average tally value and the red circles above and below the blue asterisk represent the upper and lower values of the error bars, respectively. The organs are (1) left lung (L-lung), (2) right lung (R-lung), (3) heart tissue and blood (Heart&), (4) spine and discs (Spine&), (5) brain (Brain) and (6) Head and face and skin (Head&). The present tally plotter provides a quick visualization of the dose in different organs which could be helpful for users to preform preliminary analysis of the their obtained results.

12 Building from source code on GNU/Linux

It needs to be noted that the present program was developed and designed to operate on GNU/Linux operating systems. Therefore, in order to get the best performance and all the features, we strongly recommend the users to run the DynamicMC on GNU/Linux operating systems. In this section, we

provided step-by-step guide to compile the present program from the source code. In order to build the entire program from the source, firstly there are number of packages that needs to be installed prior to compilation of the PHITS DynamicMC. We listed these tools briefly;

- make
- gcc compiler
- g++ compiler
- gfortran compiler
- qtbase5-dev
- cmake (only if you wish to use cmake instead of qmake, here we focus on qmake)
- freeglut3
- freeglut3-dev

After installing these, you must check your *qmake* version. Sometimes *qmake* would be set as *qmake-qt5*. Here we consider a case for *qmake* and then users can run the configuration script named *configure.sh*. Use *chmod +x configure.sh* and then *./configure.sh*. You need to wait for the entire program to compile and depending on your system the compile time can vary. There are large number of code and files that needs to be compiled and linked which may increase the compile time. For testing purpose, we have compiled the program on a System76 `https://system76.com/` Pangolin GNU/Linux machine running Ubuntu 22.04 LTS with gcc compiler version 11.3.0 with AMD Ryzen 7 6800u CPU and found the compile time to be 58.589s.

13 For Microsoft Windows users

Microsoft Windows users have two options to run the PHITS DynamicMC, the best option would be to use Windows Subsystem for Linux (WSL) that enables the users to install a GNU/Linux distribution on their Windows machine. The installation and setup process of WSL is rather simple and there are large amount of tutorial materials available online. We strongly recommend users to run the PHITS DynamicMC on a GNU/Linux operating system to get most of the features that PHITS DynamicMC

offers. After installing the WSL on your Windows machine, please follow the steps in the previous section (Building from source code on GNU/Linux) to get PHITS DynamicMC running.

Another options is to use our Windows version of the PHITS DynamicMC natively on your Windows machine, please note that PHITS DynamicMC was developed on and for GNU/Linux operating system, mainly due to the fact that most high performance computing are being performed on GNU/Linux machines. We have compiled the program and built Windows executables. The present program does not require any installation and can be executed in a portable fashion. All .dll files that are required by the program were included in the program directory. In addition, the generated scripts will be placed under *working_directory*. Depending on the complexity of your setup, the number of output files generated from the tallies could be rather large.

Download the Win64 zip file of the program, unzip and run the *PHITS_DYMC.bat* in the main program folder.

Note: Due to the fundamental difference between the GNU/Linux and Windows operating systems, the program callbacks and features may vary slightly. In additions, software bugs could vary between the GNU/Linux and Windows versions. In our case, we tend to provide continuous updates and features for the GNU/Linux version of the program and therefore we strongly recommend the users to run the PHITS DynamicMC on WSL when running Windows on their computer; this will make the debugging and support task easier for developers and the users.

14 Program directories and data curation

The main program folder contains the three main directories that are; (1) program_files, (2) working_directory and (3) TallyPlot (see Fig. 30). Furthermore, there will be few more files in the main program folder, such as changelog (that records any changes or bug fixes in different version of the program), configure.sh (that could be used to compile the entire program from source), PHITS_DYMC.sh (used to run the PHITS DynamicMC program using a simple bash call to the main program executable) and a README file (helps users with a brief and quick start). The native Windows version of the program has similar directories with a difference that the PHITS_DYMC.sh was changed to PHITS_DYMC.bat that serves the same purpose for running the program on Windows machines.

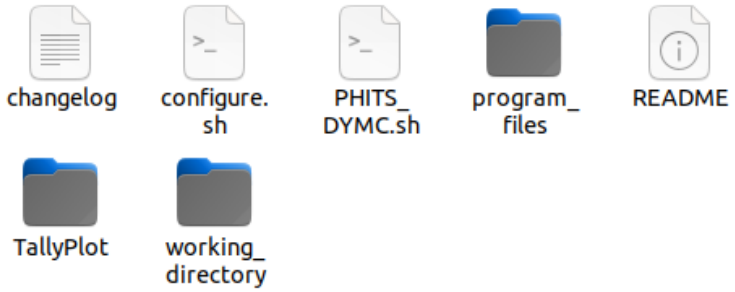

Figure 30: The main program folder containing different directories.

Running the program .sh or .bat will start the PHITS DynamicMC and users can follow the steps discussed in the previous section of this user manual to simulate their desired setups. When executing the PHITS DynamicMC, a run folder will be automatically generated under the working_directory which will be numbered from run1 to runN. All users generated scripts and PHITS run results will be saved in a specific run folder. Note, if you wish to separate different dynamic simulations in different run folders, then after every run you must close the PHITS DynamicMC program and re-run the program so that your runs will be separated. However, for all number of dynamic positions, the scripts and results will all be saved in one run folder. The example content of the working_directory is shown in Fig. 31.

Figure 31: Example of contents under the working_directory.

In the example shown in Fig. 31, 8 different scenarios were modelled and each were saved in a separate run folder. Looking into one of these run folder that was generated when running the preset example of the text coordinate input, The steps to run and recreate this would be;

- run PHITS DynamicMC
- click on text button
- click on Gen button

- click on Run PHITS button
- wait for the run to complete
- close the program
- go to working directory
- find the latest run folder that was generated (you can simply sort the folders by their modified time and the latest one will correspond to the latest run folder)

After following the above steps, the content of the run folder will be as shown in Fig. 32.

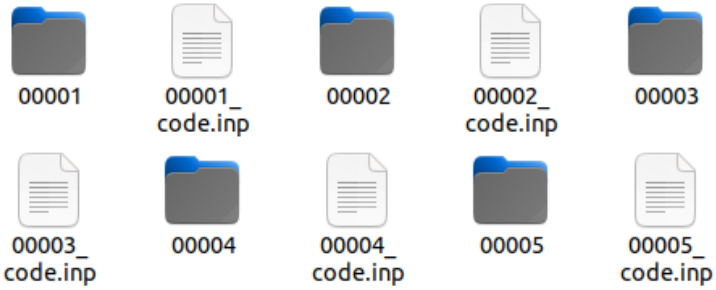

Figure 32: Content of the run folder generated by running the text input preset example.

The generated PHITS input script for position 1 is saved in 00001_code.inp and similarly for all other 3 remaining positions (please note that the preset example only models 4 different source and phantom positions). The extra input file which in this case named 00005_code.inp is the *sumtally* input that is automatically generated and performs weighted average of the organs tally results, thanks to PHITS MC package for having the *sumtally* option. The results for different positions is saved in their respective number for example position 00001_code.inp results is saved in 00001 folder and so on. It is always a good practice to backup your working_directory for your important runs.

15 Bug report, contribution and addition of features

Please report any bugs or issues to Ben at: `ben.sh@tohoku.ac.jp` or `ben.sh@my.cityu.edu.hk`
In addition we welcome any suggestions or potential contributions from interested users, regarding adding new features to the latest version of the DynamicMC. We welcome both ideas or potential contributions.

Please kindly set the subject of your email "PHITS DynamicMC" and it would be easier for us to get in contact with you as soon as possible.

16 Developers

The present program was initially conceptualized and developed by Mehrdad S. Beni, Hiroshi Watabe and Peter K.N. Yu. It was further developed by collaborating with our expert colleagues in the field from several institutes. Full list of contributors and their academic affiliations that helped in improving the quality and number of features in the present program are given below;

Hiroshi Watabe [1], Tatsuhiko Sato [2], Kwan Ngok Yu [3], Milena Zivkovic [4], Dragana Krstic [4], Dragoslav Nikezic [4,5], Kyeong Min Kim [6], Taiga Yamaya [7], Naoki Kawachi [8], Hiroki Tanaka [9], A.K.F. Haque [10], M. Rafiqul Islam [11], Mehrdad Shahmohammadi Beni [1,3]

[1]Division of Radiation Protection and Safety Control, Cyclotron and Radioisotope Center, Tohoku University, 6-3 Aoba, Aramaki, Aoba-ku, Sendai, Miyagi 980-8578, Japan, [2]Nuclear Science and Engineering Center, Japan Atomic Energy Agency, 2-4 Shirakata, Tokai, Ibaraki 319-1195, Japan, [3]Department of Physics, City University of Hong Kong, Tat Chee Avenue, Kowloon Tong, Hong Kong, China, [4]Faculty of Science, University of Kragujevac, Serbia, [5]State University of Novi Pazar, Serbia, [6]Korea Institute of Radiological & Medical Sciences, 75, Nowon-ro, Nowon-gu, Seoul, Korea, [7]National Institutes for Quantum Science and Technology, Anagawa 4-9-1, Inage-ku, Chiba-shi, Chiba 263-8555, Japan, [8]National Institutes for Quantum Science and Technology, 1233 Watanuki, Takasaki, Gunma 370 1292, Japan, [9]Institute for Integrated Radiation and Nuclear Science, Kyoto University, 2-1010 Asashiro-Nishi, Kumatori-cho, Sennan-gun, Osaka 590-0494, Japan, [10]Department of Physics, University of Rajshahi, Bangladesh, [11]Institute of Nuclear Medical Physics, AERE, Bangladesh Atomic Energy Commission, Dhaka, 1349, Bangladesh

We truly appreciate all the contributions from our colleagues in this work!

17 License

18 About the logo/icon

The CYRIC logo that is used as the program icon and also shown in the top left corner of this user manual is an actual positron emission mammography (PEM) scan of a 3D printed design that we have prepared. This was part of a fun experimental project entitled *Dynamic PEM* that was designed and performed by Hiroshi Watabe and Mehrdad S. Beni.
The F-18 radioisotopes were injected at a slow rate into the 3D printed CYRIC structure with a micro-pump as the PEM scan was running. We were able to observe the movement of radionuclides inside the dedicated channel that we 3D printed. It was a memorable piece of experimental work with some dynamic spices!

Here is how we did it, see Fig. 33!

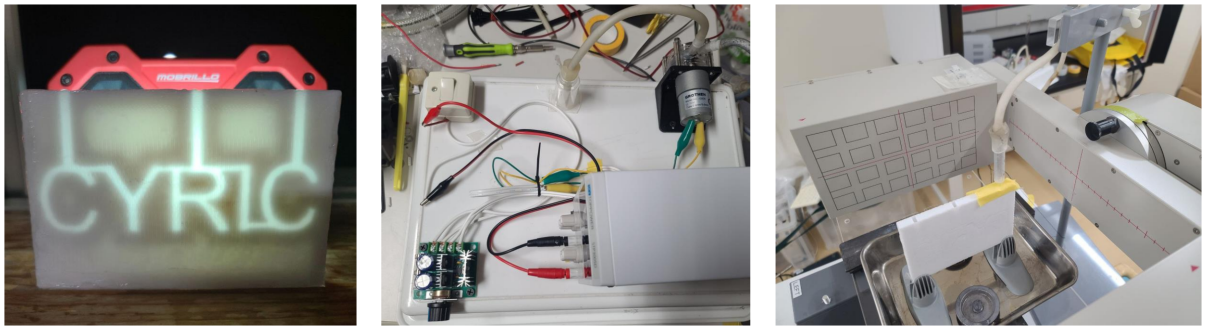

Figure 33: The fun experiment.