GATE Simulation of LINAC and Radiotherapy

Darshana Suresh - Shalini Nath - Saai Lakshmi

CONTENTS

The Sources

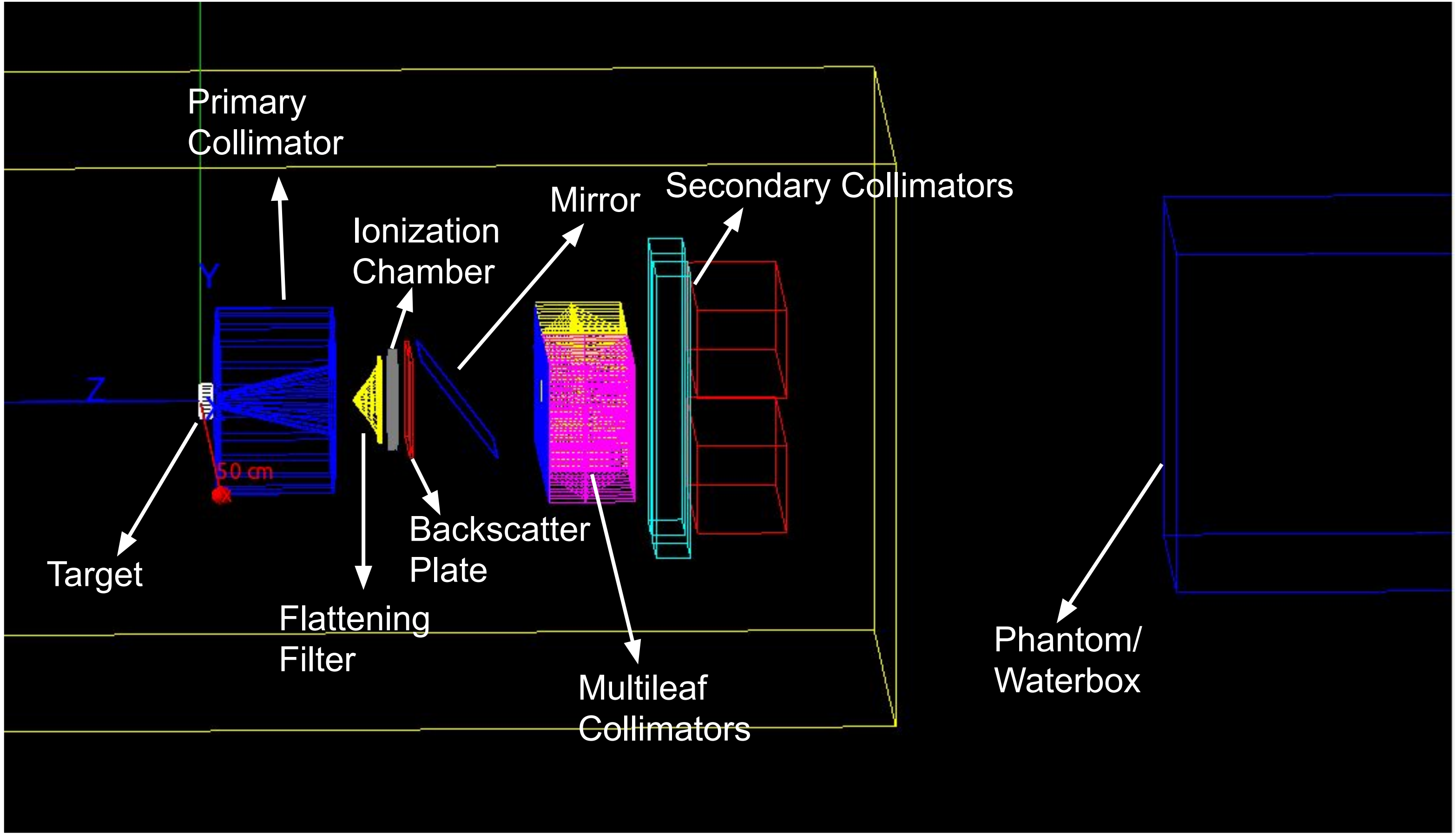
Code set we’re working on :

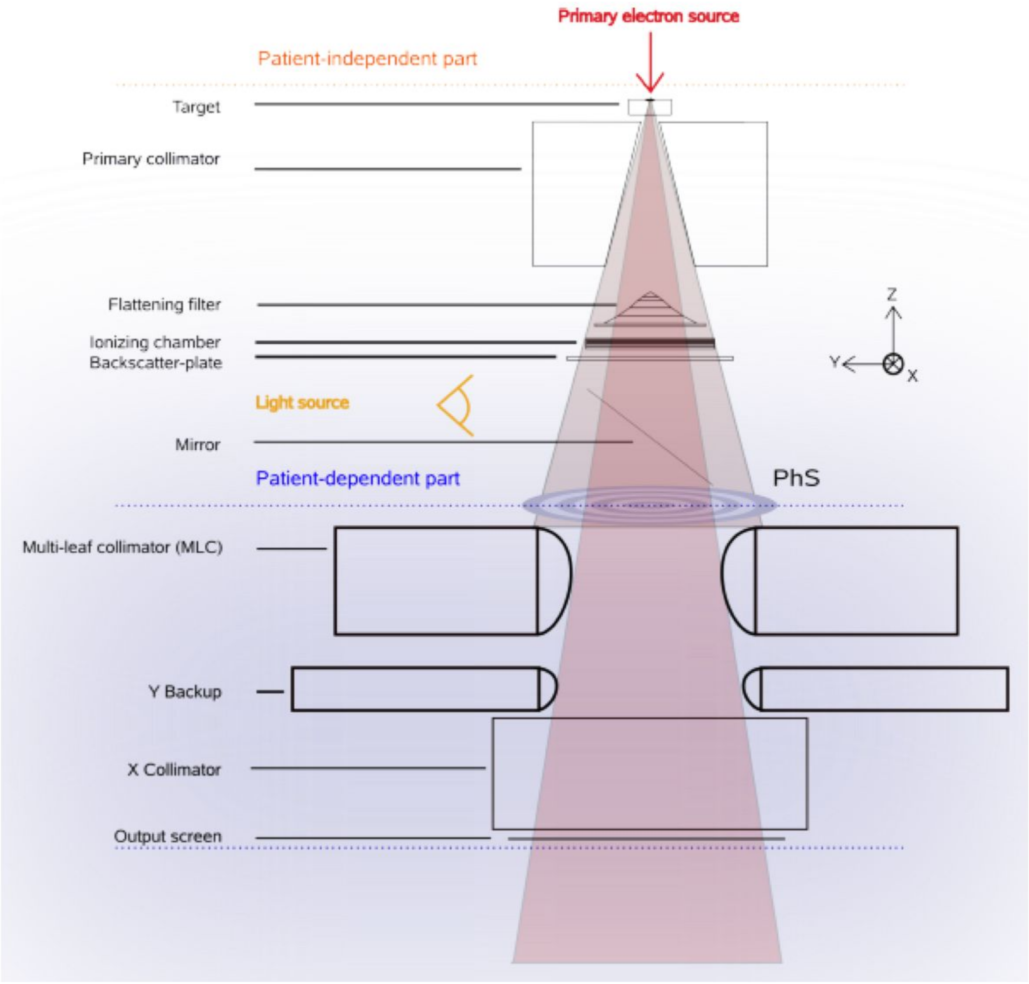
https://github.com/OpenGATE/GateContrib/tree/master/dosimetry/Radiotherapy/example12

This code is the simulation of the LINAC head divided into two parts, as illustrated in the coming slides. We have been using **Gatev8.2** to run the code and bring additional changes to simulate the radiotherapy process. The updated code is put up in github -

https://github.com/Darshana-Suresh/photon-linac

Geometric Simulation in GATE





Simulation in 2 parts

Source:
https://dsarrut.gitbooks.io/gate-exercises/content/exercise4-linac.html

PART 1 - PATIENT INDEPENDENT
[from the target to the phase space]

PART 2 - PATIENT DEPENDENT
[from the phase space to the phantom; here, ‘Output screen’]

PART 1 - Contents

- Creation of geometric simulation of LINAC head
- Production threshold values for generating secondary particles (details)
- KillActor to kill particles outside the region
- Bremsstrahlung splitting for photons (details)

INPUTS

- Specifications to be recorded in Phase Space file (Options in GATE)
- Specifications of primary beam source (details)
- Total number of primary particles, here, electrons.

OUTPUTS

1. output-writePhS-stat.txt
   - Stores number of events: primary particles
   - Tracks: snapshot of a particle
   - Steps: delta information to a track; track is being updated by steps
   - And other information as shown -->

Reference -

https://indico.ihep.ac.cn/event/4287/contribution/1/material/slides/0.pdf

output > output-writePhS-stat.txt

```
# NumberOfRun    = 1
# NumberOfEvents = 5000
# NumberOfTracks = 5710
# NumberOfSteps  = 34601
# NumberOfGeometricalSteps  = 27340
# NumberOfPhysicalSteps     = 7261
# ElapsedTime              = 6.08433
# ElapsedTimeWoInit        = 0.40001
# StartDate                = Wed Mar 18 10:41:03 2020
# EndDate                  = Wed Mar 18 10:41:09 2020
# StartSimulationTime        = 0
# StopSimulationTime         = 1
# CurrentSimulationTime      = 1
# VirtualStartSimulationTime = 0
# VirtualStopSimulationTime  = 1
# ElapsedSimulationTime      = 1
# PPS (Primary per sec)      = 12499.7
# TPS (Track per sec)        = 14274.6
# SPS (Step per sec)         = 86500.3
```

OUTPUTS

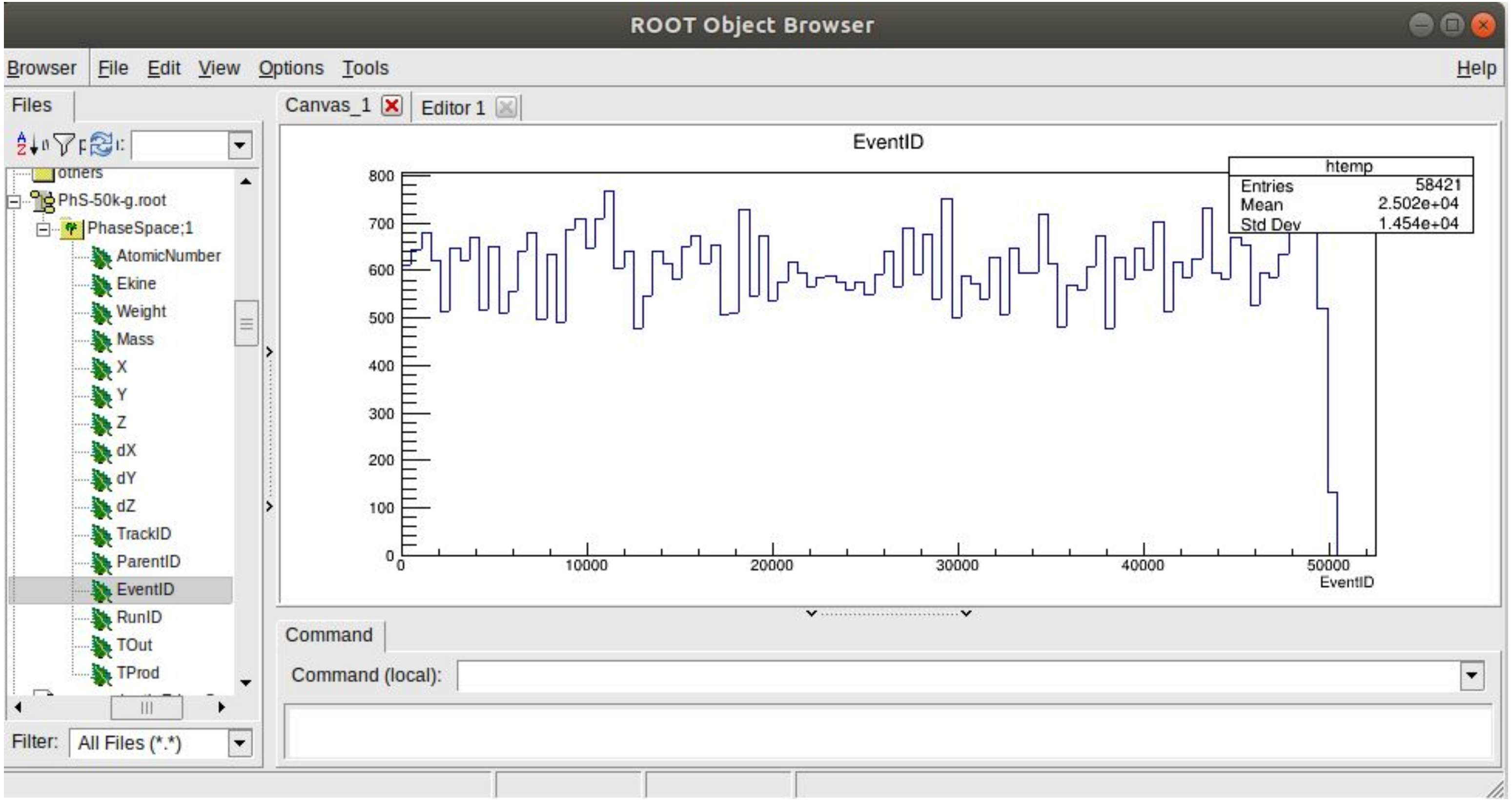
2. output-PhS-g.root

- Contains details of particles that collide with the phase space region
- Stored in root format and can be analyzed using the ROOT software (next slide).
- PhS-Analysis.C shows analysis of the root file through graphs.

Root Installation Source -

https://twiki.cern.ch/twiki/bin/view/Main/OtherSettings

OUTPUTS - Phase space file opened in ROOT



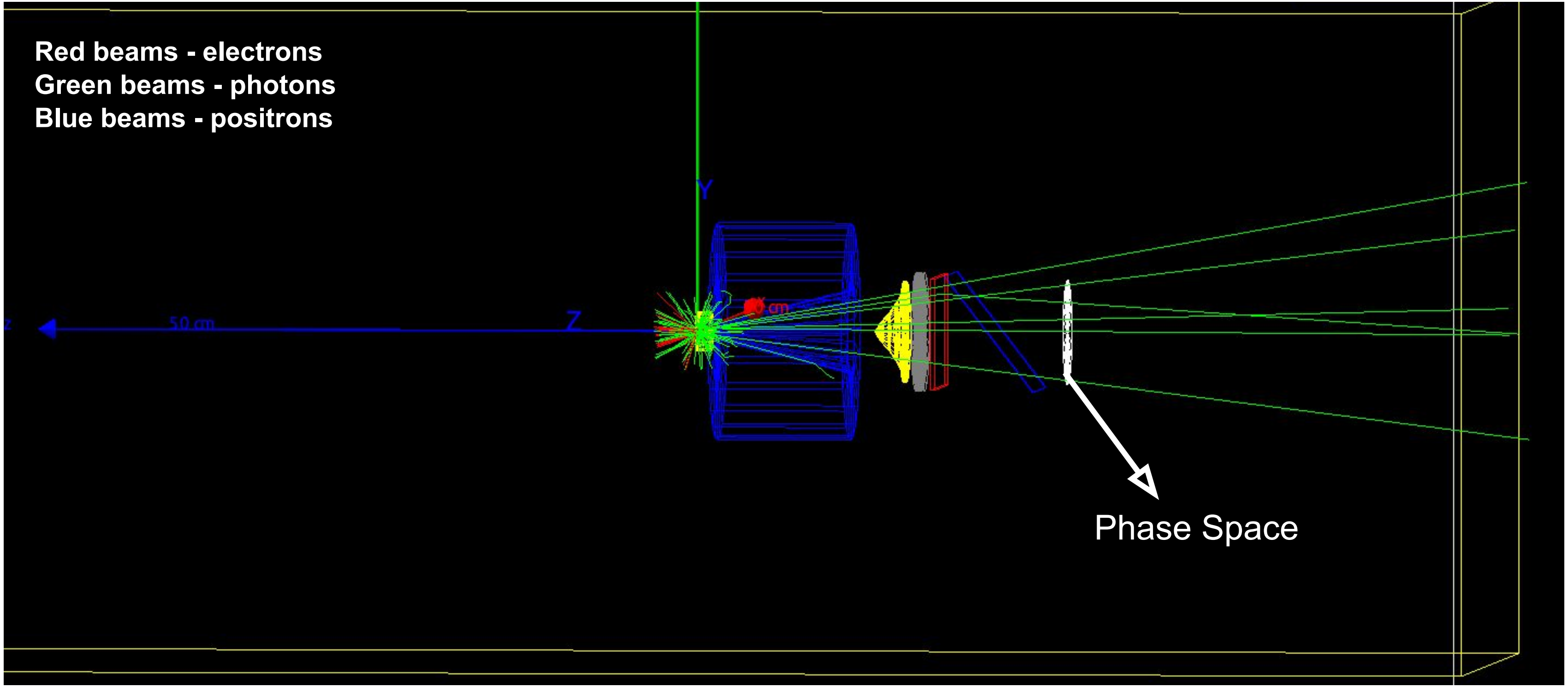
Use of Monte Carlo Method - Bremsstrahlung splitting

- Bremsstrahlung is the physics process by which X-rays are produced by the deceleration of electrons.
- Bremsstrahlung splitting is a variance reduction technique used by Geant4 to simulate this physics process.
- This technique makes use of the Russian Roulette and Splitting method for the simulation of x-ray beam production. This is a usage of **Monte Carlo** method.

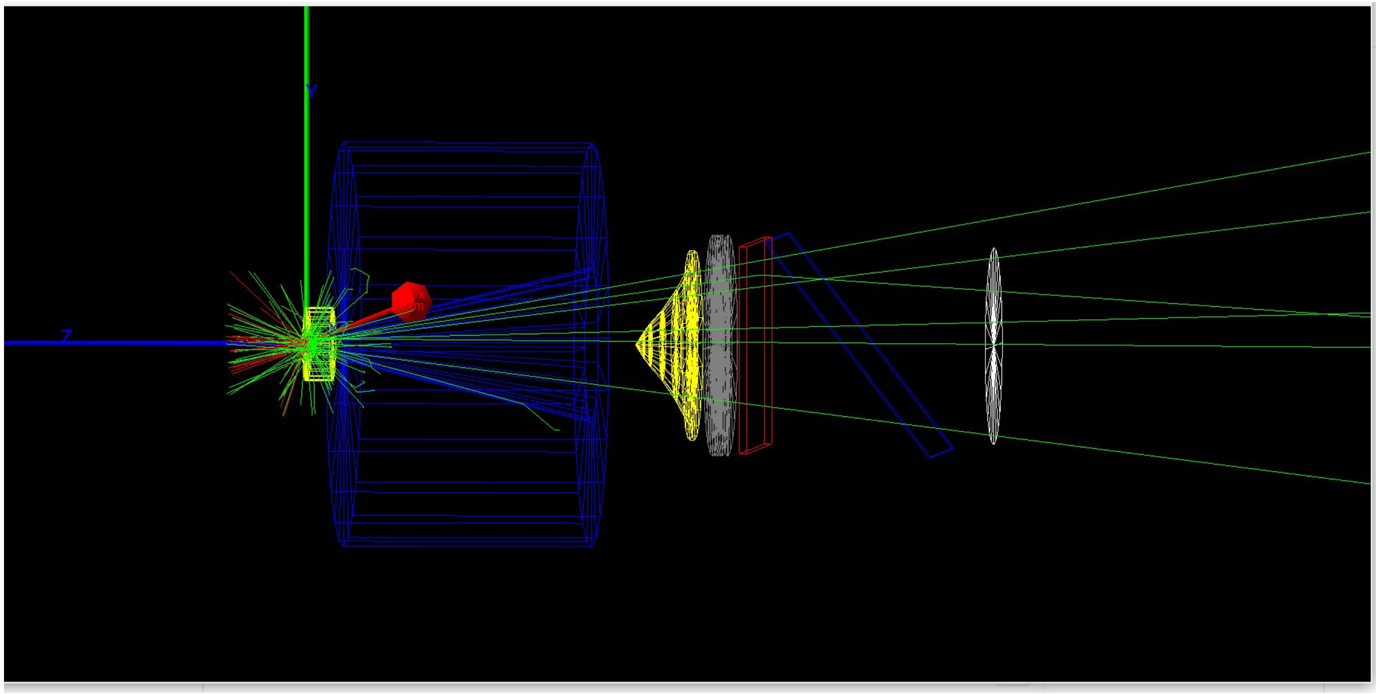
Russian Roulette and Splitting

- In simple terms, if a generated photon moves back towards the source (unwanted region), then its history is recorded with a very less probability of 1/N. That is, the Russian Roulette is applied here.
- But if the generated photon moves away from the source (desired region), then it is split into N photons.
- This method drastically reduces the time of simulation as the favourable generations are multiplied in the same event instead of waiting for more photons to be generated.

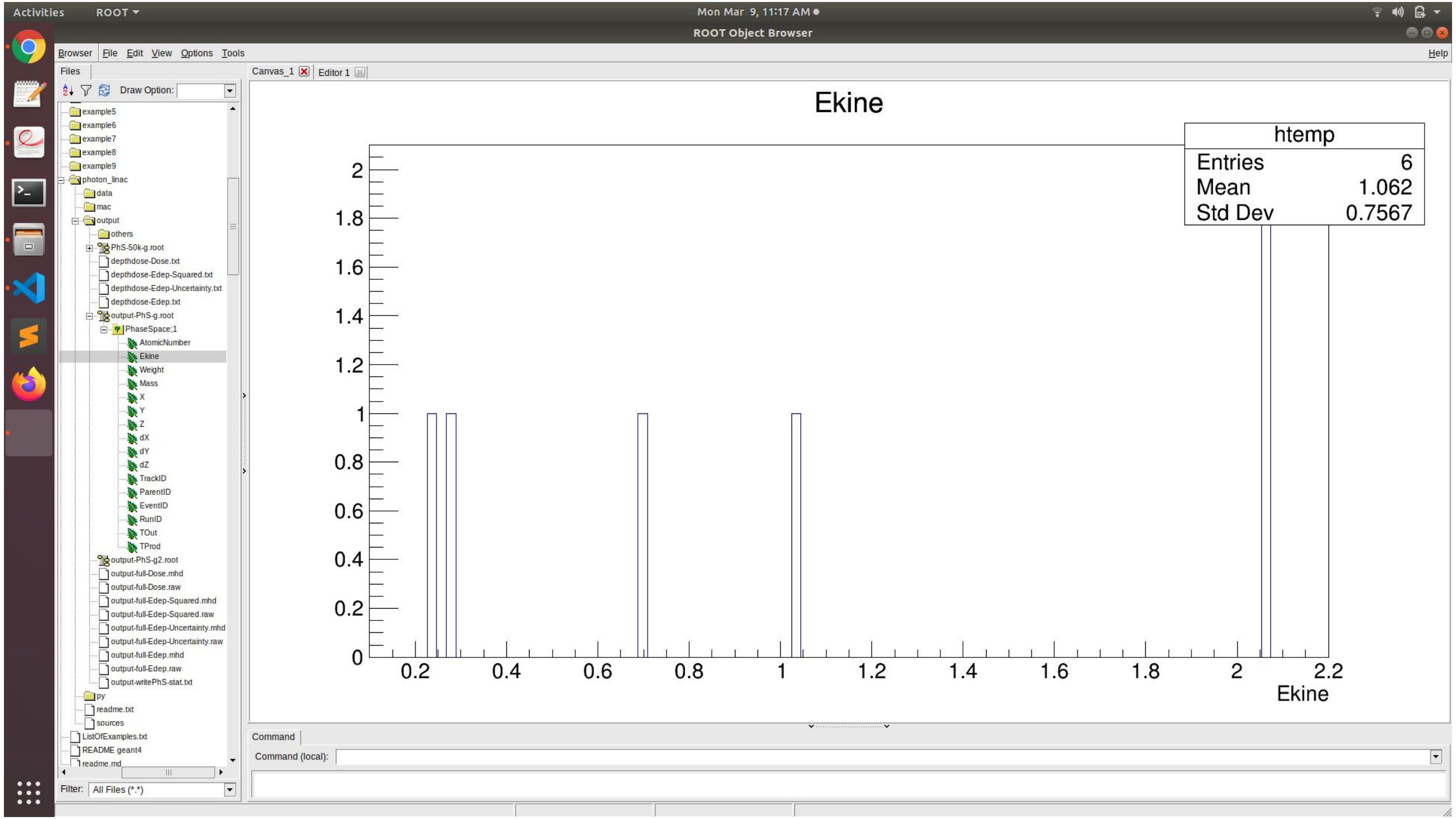
Without Bremsstrahlung splitting - 500 primaries (electrons)



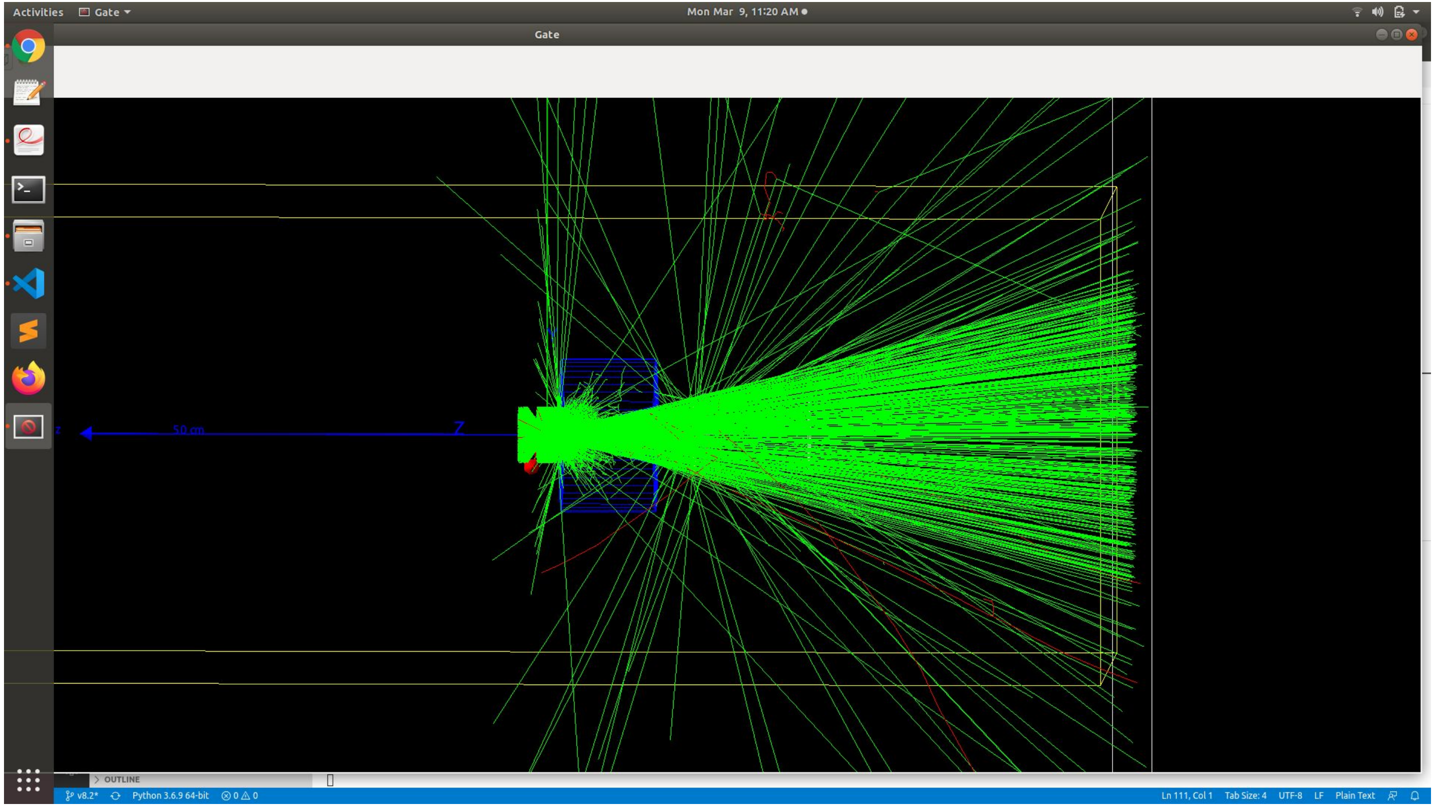
Closer look



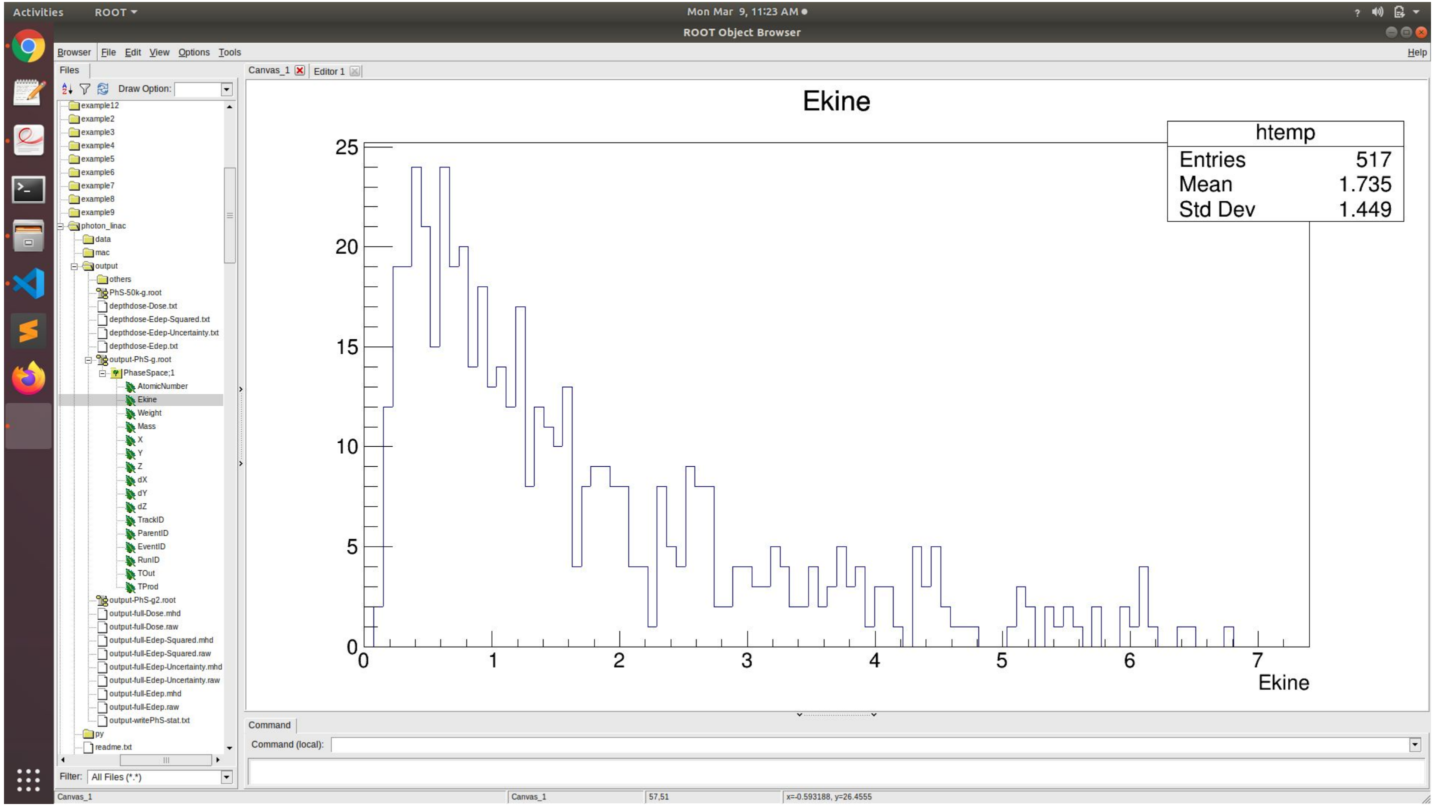
Phase Space has only 6 entries (Only 6 photons formed)



With Bremsstrahlung Splitting - 500 primaries (electrons)



Phase Space has 517 entries (517 photon generations)



PART 2 - Contents

- Creation of remaining geometry
- KillActors to kill particles straying away
- Generation of output files for analysis

INPUTS

- Phase Space root file as source
- MLC placements for each leaf
- Output specifications in terms of depth dose and dose profiles (options in GATE)

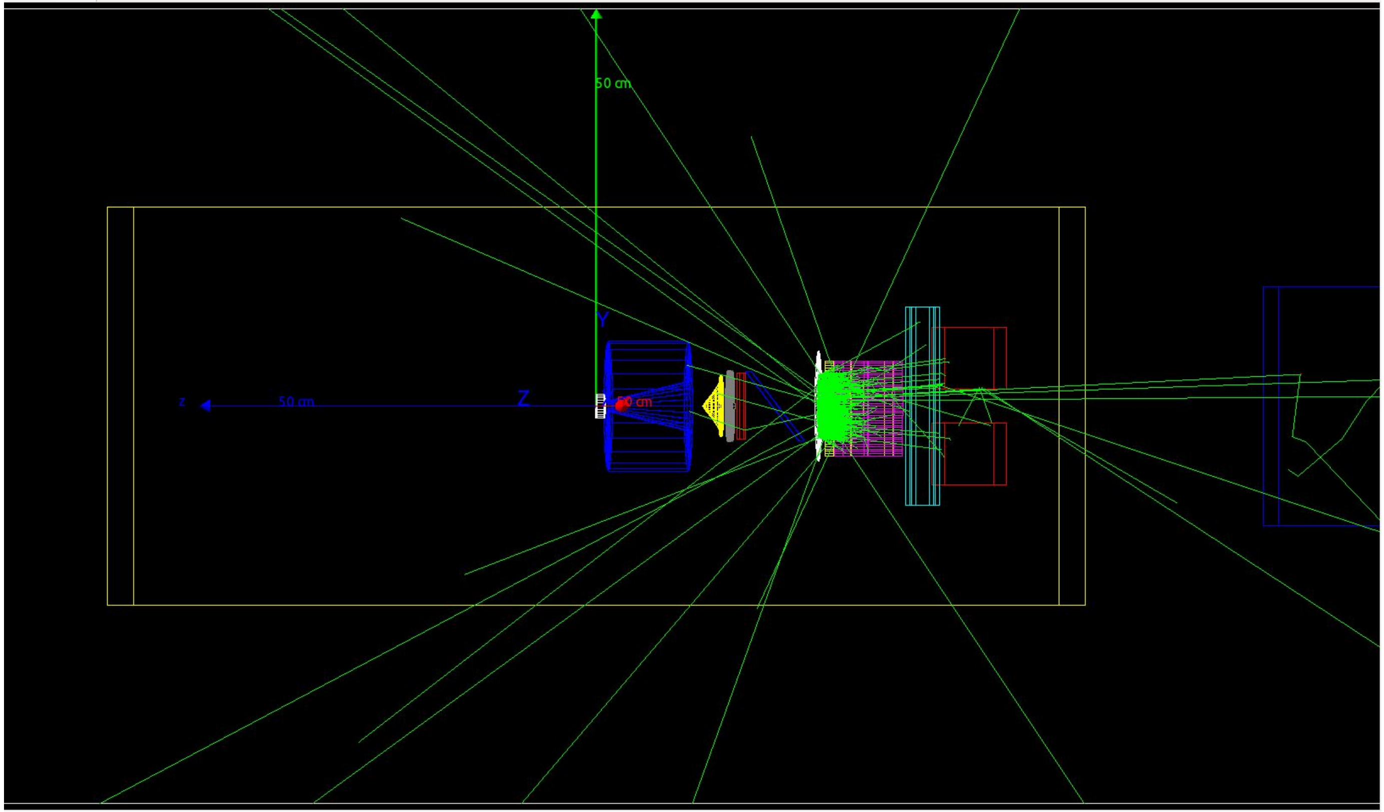
Kill Actors

KillActors are detectors attached to a certain volume that kills any particle that enters it. This is used to kill the beams straying away to unwanted regions.

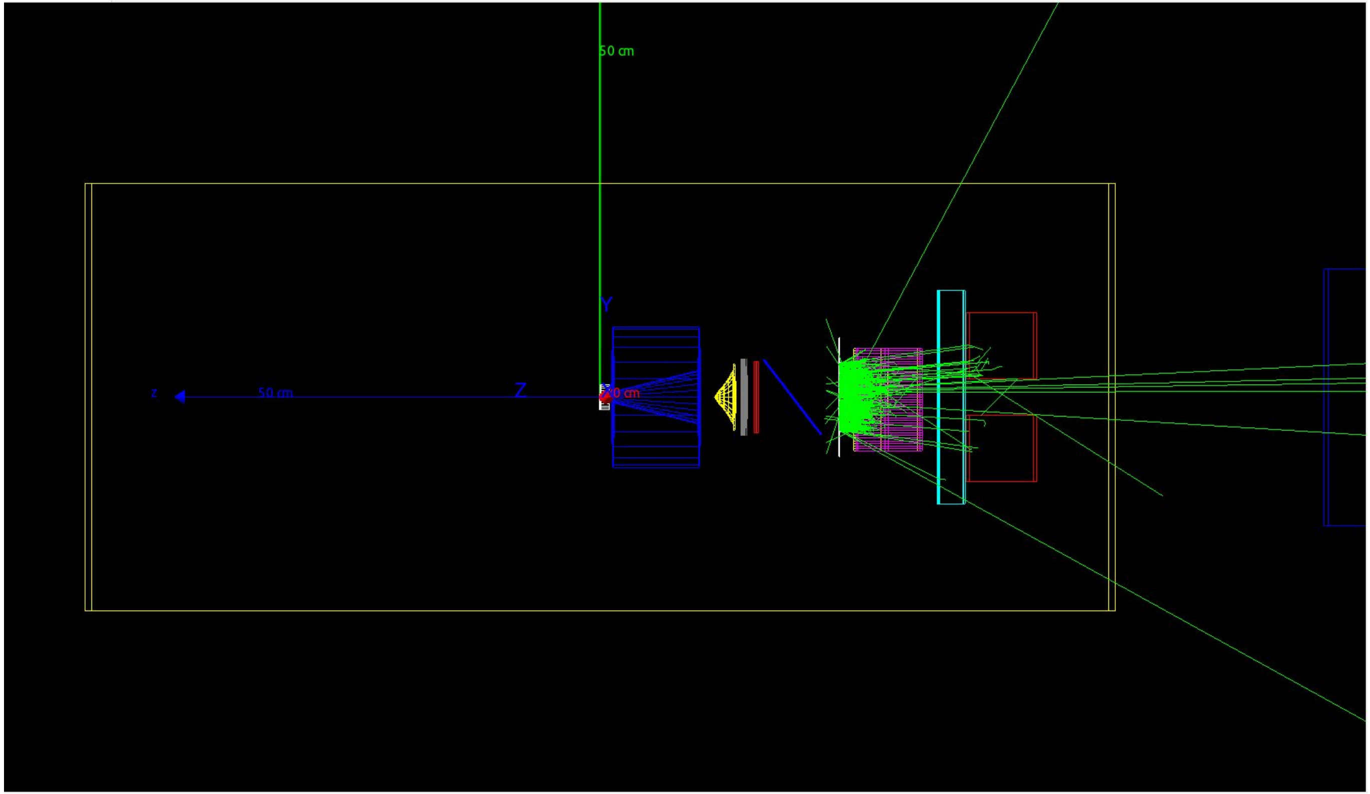
Killing beams in unwanted directions will **save simulation time** as their trajectories need not be followed or recorded any further.

The next slides demonstrate its use.

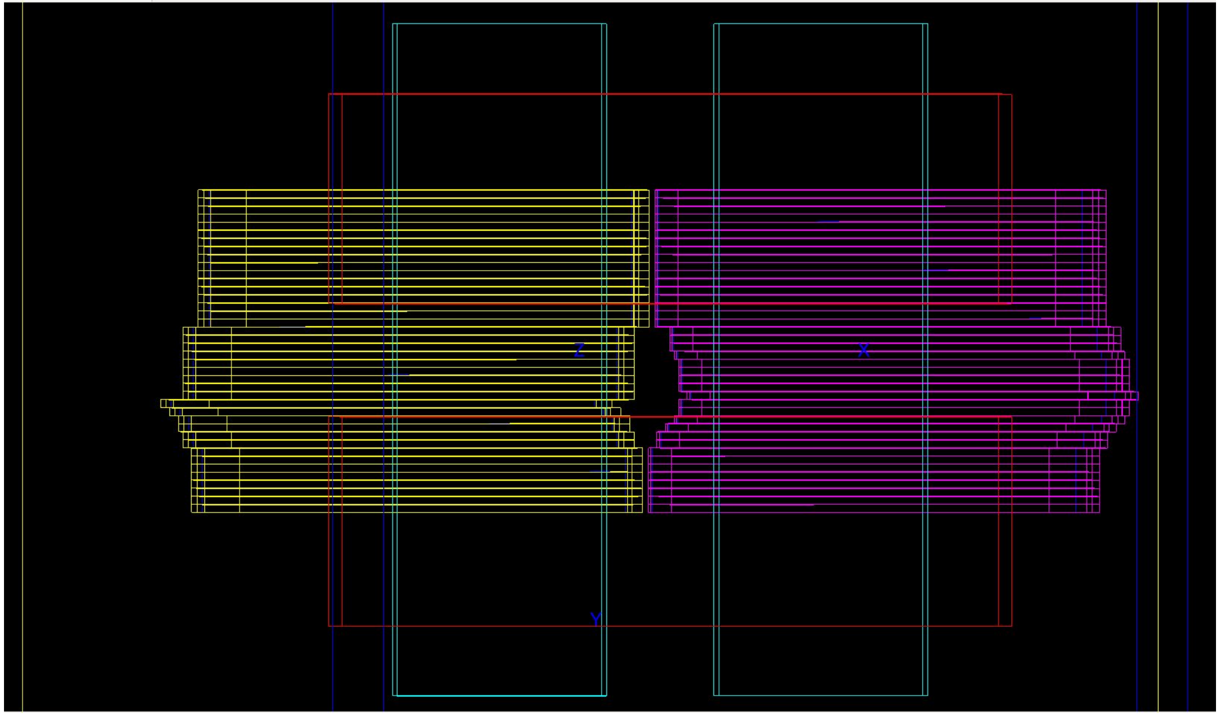
Scattered particles from PhS - without KillActor



Scattered particles from PhS - with KillActor

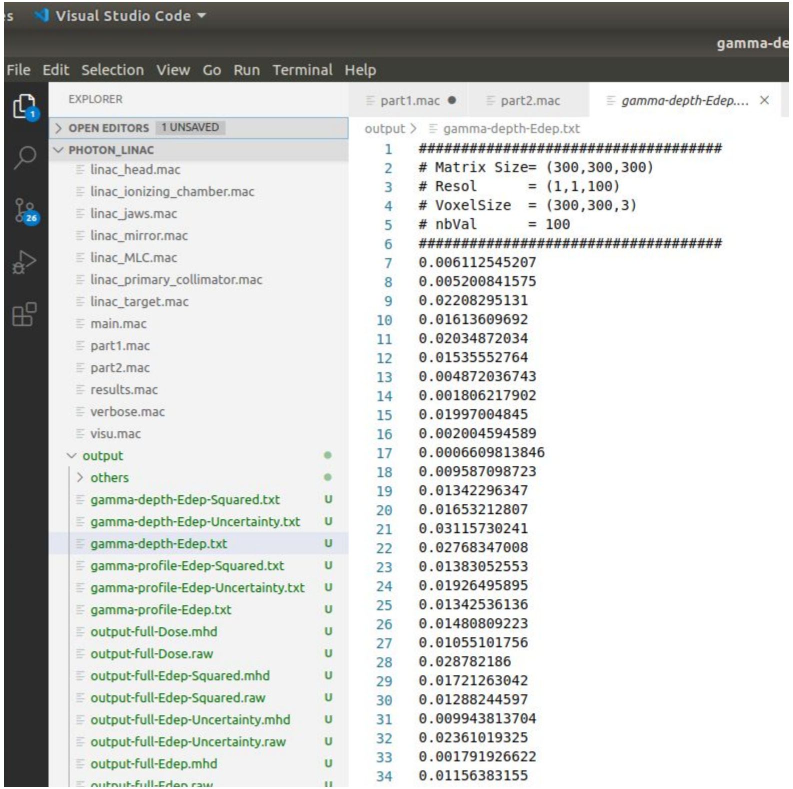


MLCs placements - a visual



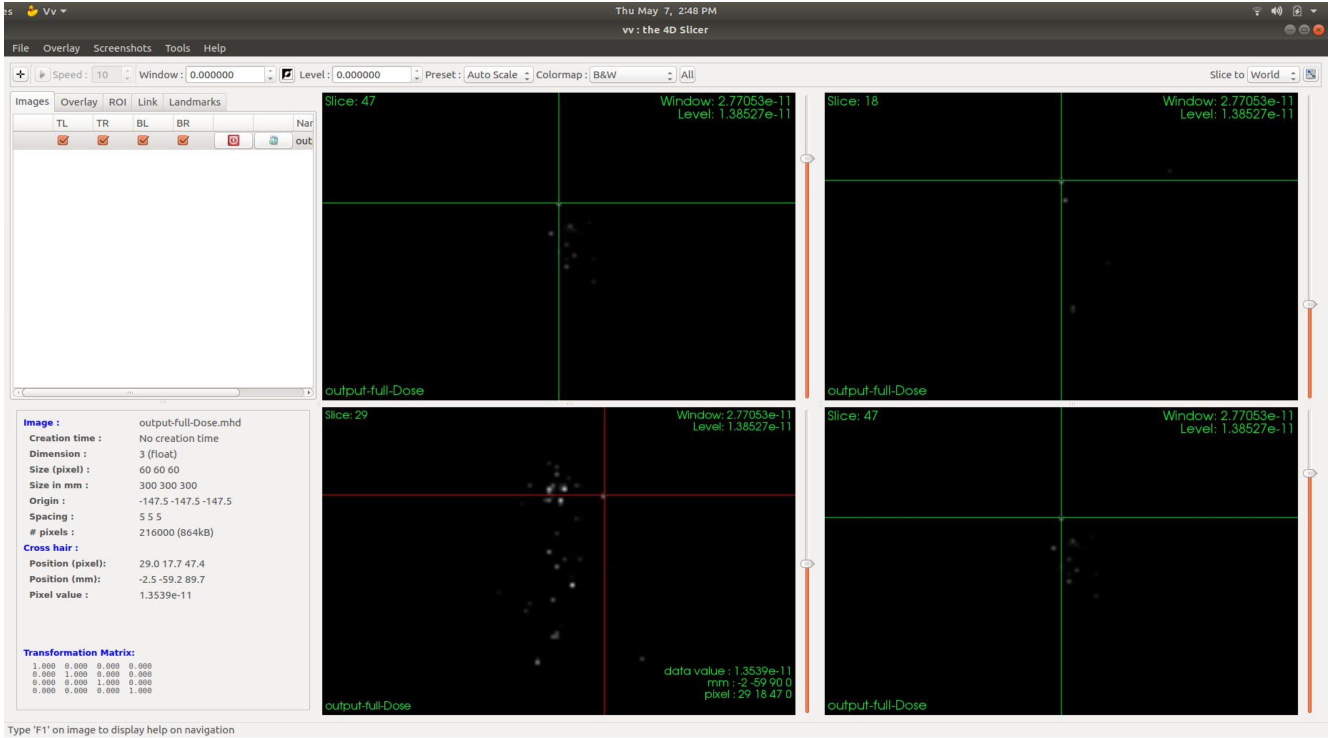
OUTPUTS

1. The energy/dose deposited on the waterbox (phantom) as per the given specifications
2. The square of the energy/dose depositions.
3. The uncertainty of the results (reduces as the number of trials increases)



OUTPUTS

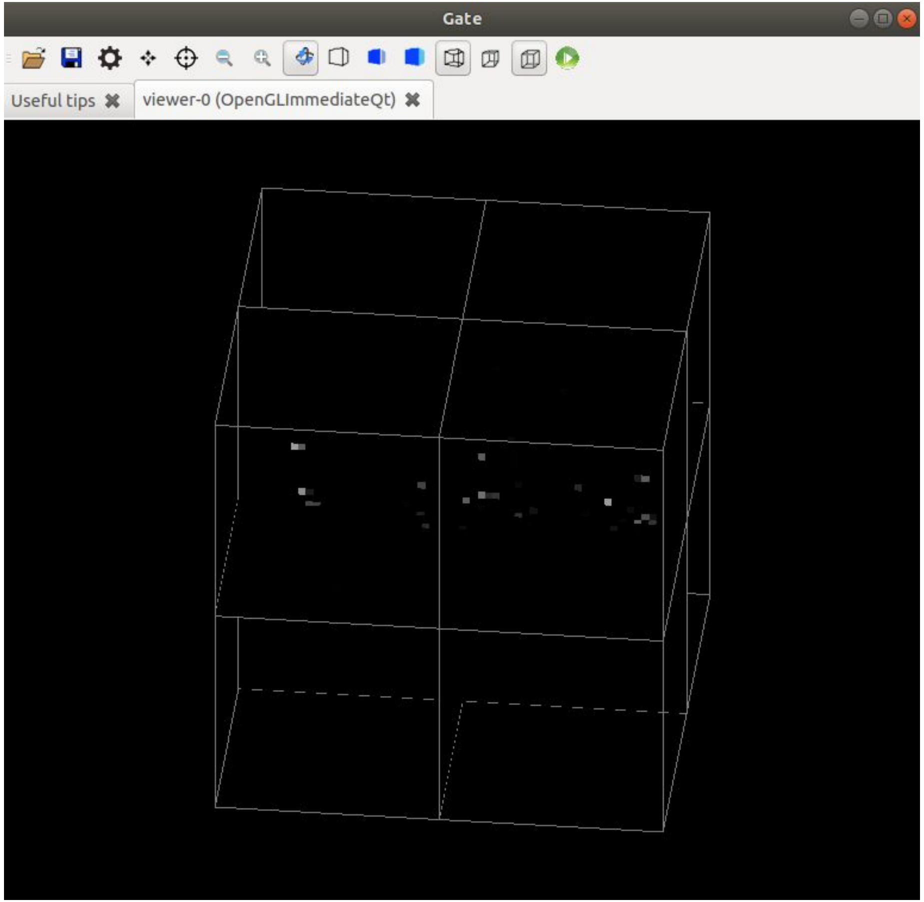
The 3-D figures of the energy depositions are stored in mhd-raw format, which can be viewed using vv software.



OUTPUTS

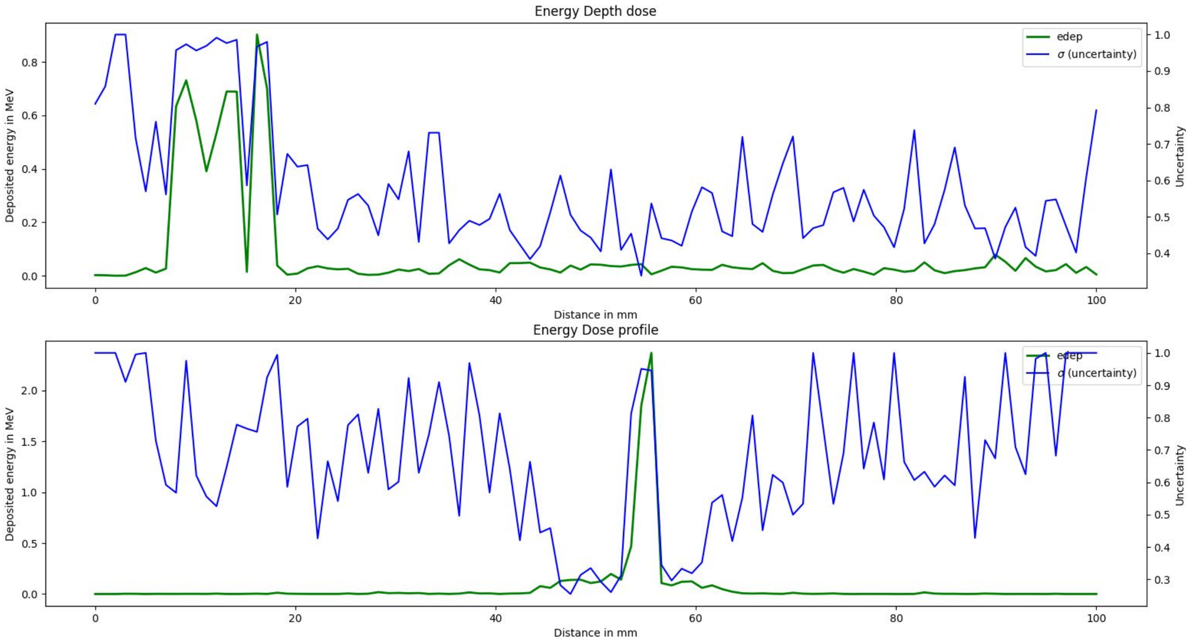
It can also be viewed using GATE, (here, using the code results.mac)

But it is difficult to analyse the result from this 3D visual. The greyish white spots are the energy/dose depositions in the phantom.



OUTPUTS

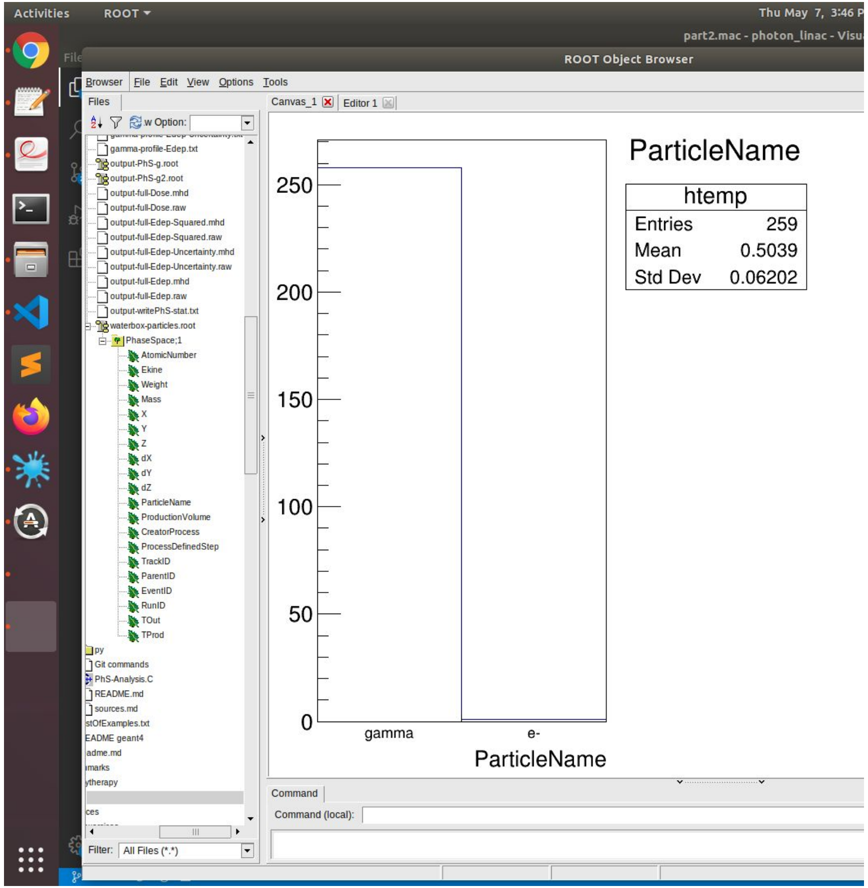
3. The dose outputs can be saved in text files to obtain graphical outputs of depth dose and dose profile in python -



OUTPUTS

4. The details of particles that reach the phantom or water box can be viewed using ROOT (waterbox-particles.root)

Here 259 photons have reached the phantom. -->



Additions from Our End

- Geometry of MLC and Secondary Collimators, including their positions.
- KillActors in part 2 to remove straying secondary particles.
- Detectors attached to phantom to analyze the particles that collide with it.
- Gate code to visualize the 3D output files. (results.mac)
- Python code to plot graphs from the results. (plot.py)