

# GATE Simulation of LINAC and Radiotherapy

Darshana Suresh - Shalini Nath - Saai Lakshmi

# CONTENTS

1. [Source Code](#)
2. [Geometric Simulation](#)
3. [Part 1 - INPUTS](#)
4. [Part 1 - OUTPUTS](#)
5. [Usage of Monte Carlo simulation](#)
6. [Part 2 - INPUTS](#)
7. [Part 2 - OUTPUTS](#)
8. [Additions from Our End](#)

# The Sources

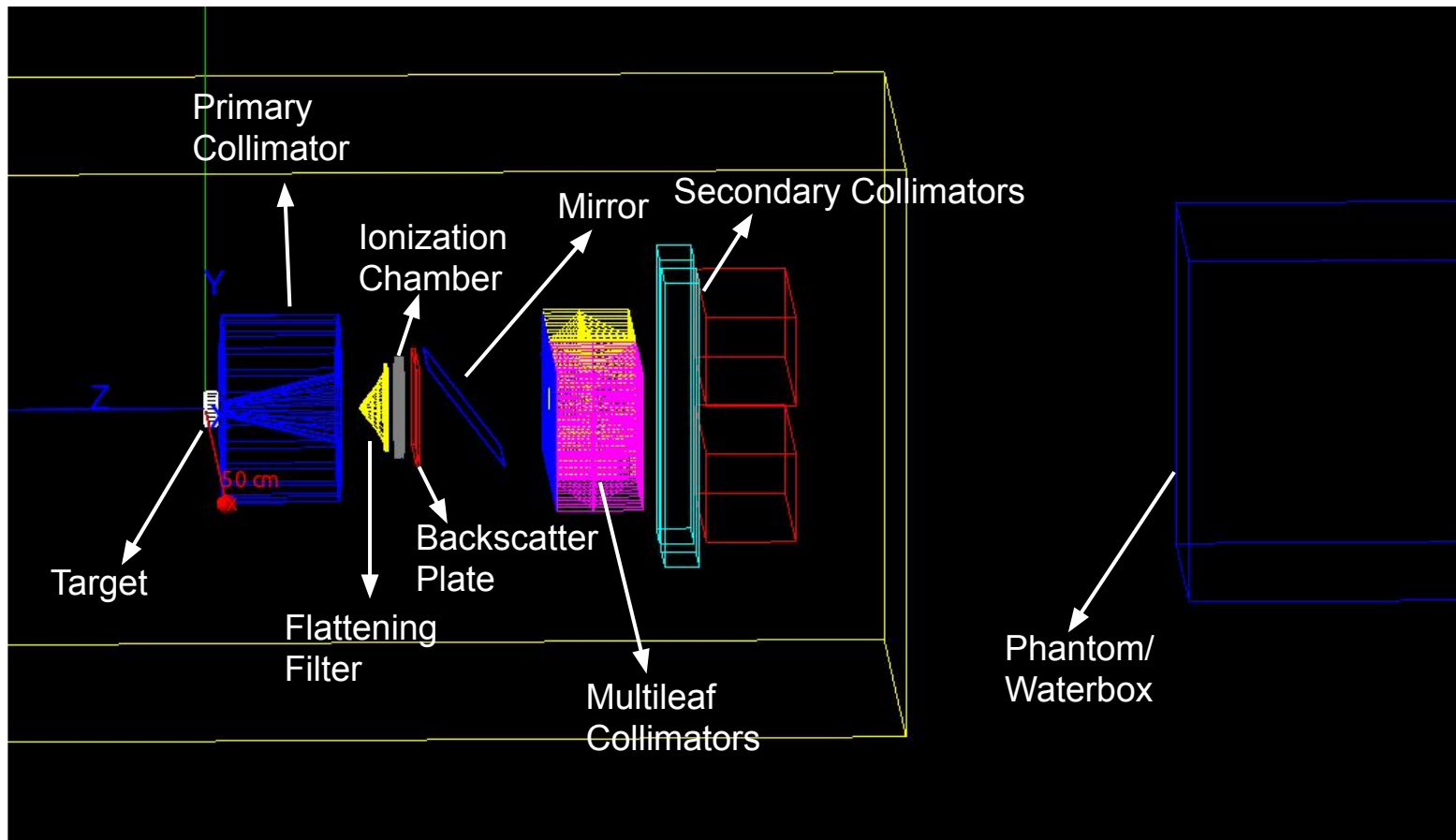
Code set we're working on :

<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



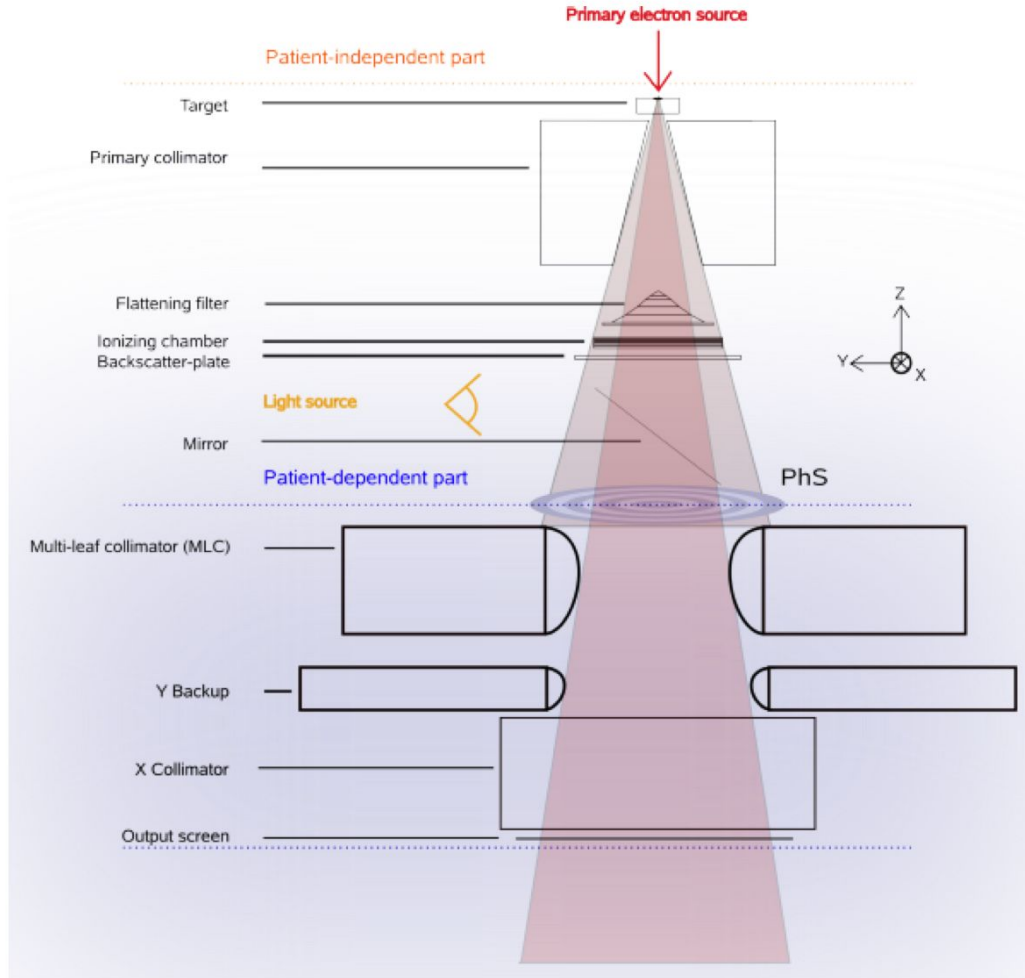
# Simulation in 2 parts

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
1  # NumberOfRun      = 1
2  # NumberOfEvents   = 5000
3  # NumberOfTracks    = 5710
4  # NumberOfSteps     = 34601
5  # NumberOfGeometricalSteps = 27340
6  # NumberOfPhysicalSteps   = 7261
7  # ElapsedTime          = 6.08433
8  # ElapsedTimeWoInit     = 0.40001
9  # StartDate            = Wed Mar 18 10:41:03 2020
10 # EndDate              = Wed Mar 18 10:41:09 2020
11 # StartSimulationTime   = 0
12 # StopSimulationTime    = 1
13 # CurrentSimulationTime = 1
14 # VirtualStartSimulationTime = 0
15 # VirtualStopSimulationTime = 1
16 # ElapsedSimulationTime = 1
17 # PPS (Primary per sec) = 12499.7
18 # TPS (Track per sec)   = 14274.6
19 # SPS (Step per sec)    = 86500.3
20
```

# OUTPUTS

## 2. output-PhS-g.root

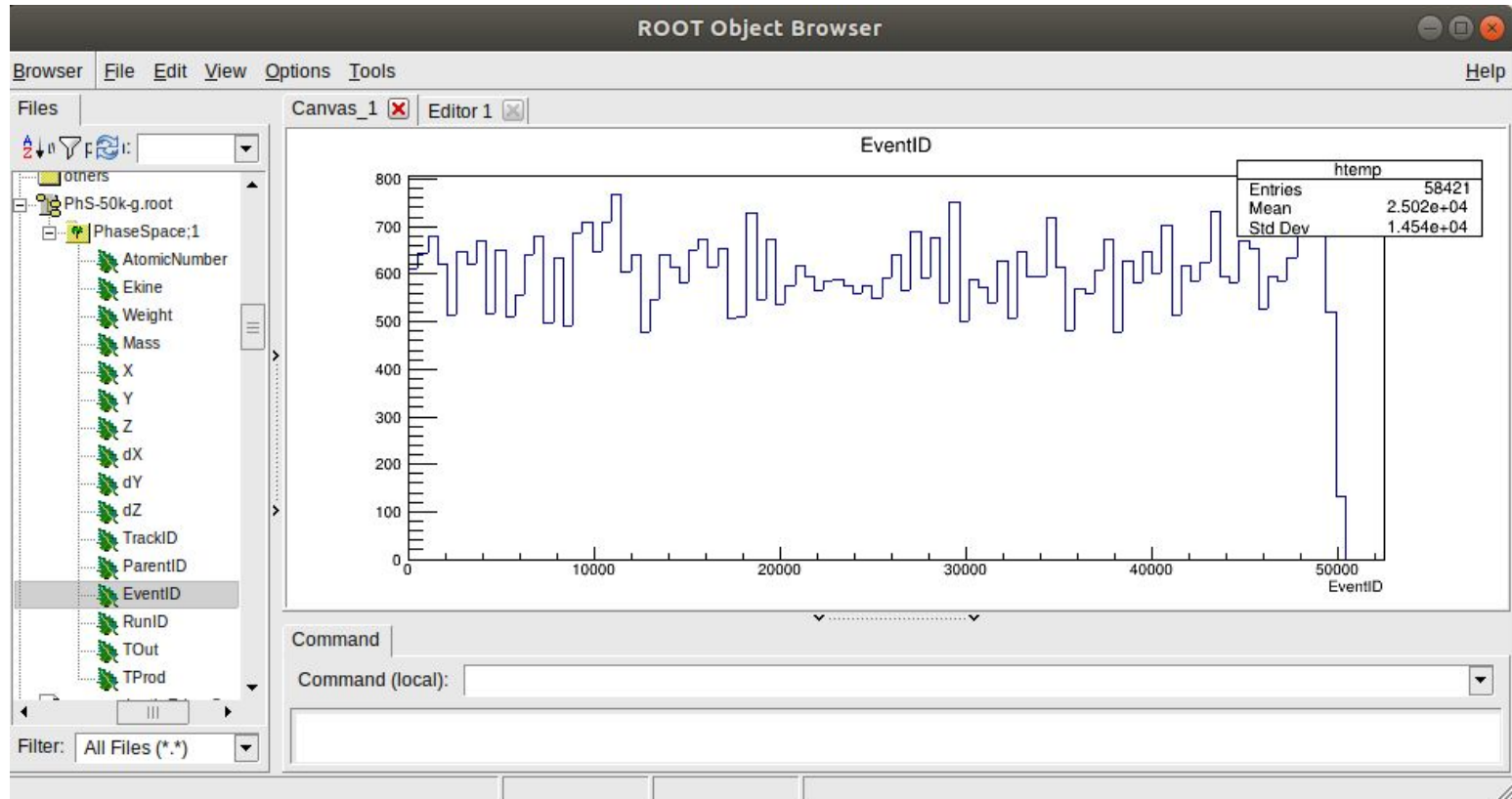
- Contains details of particles that collide with the phase space region
- Stored in root format and can be analyzed using the ROOT software (next slide).
- PhS-Analysis.C shows analysis of the root file through graphs.

Root Installation Source -

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



# OUTPUTS - Phase space file opened in ROOT



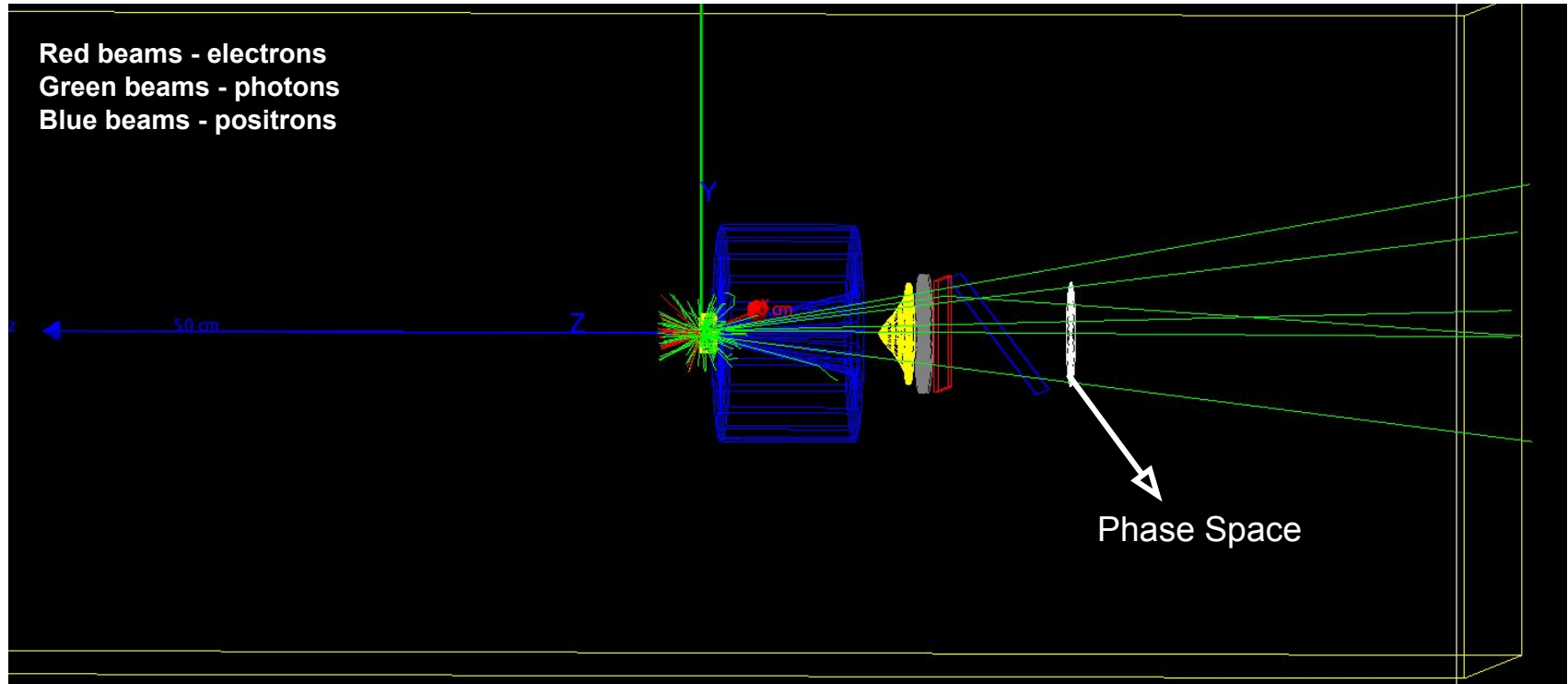
# Use of Monte Carlo Method - Bremsstrahlung splitting

- 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.

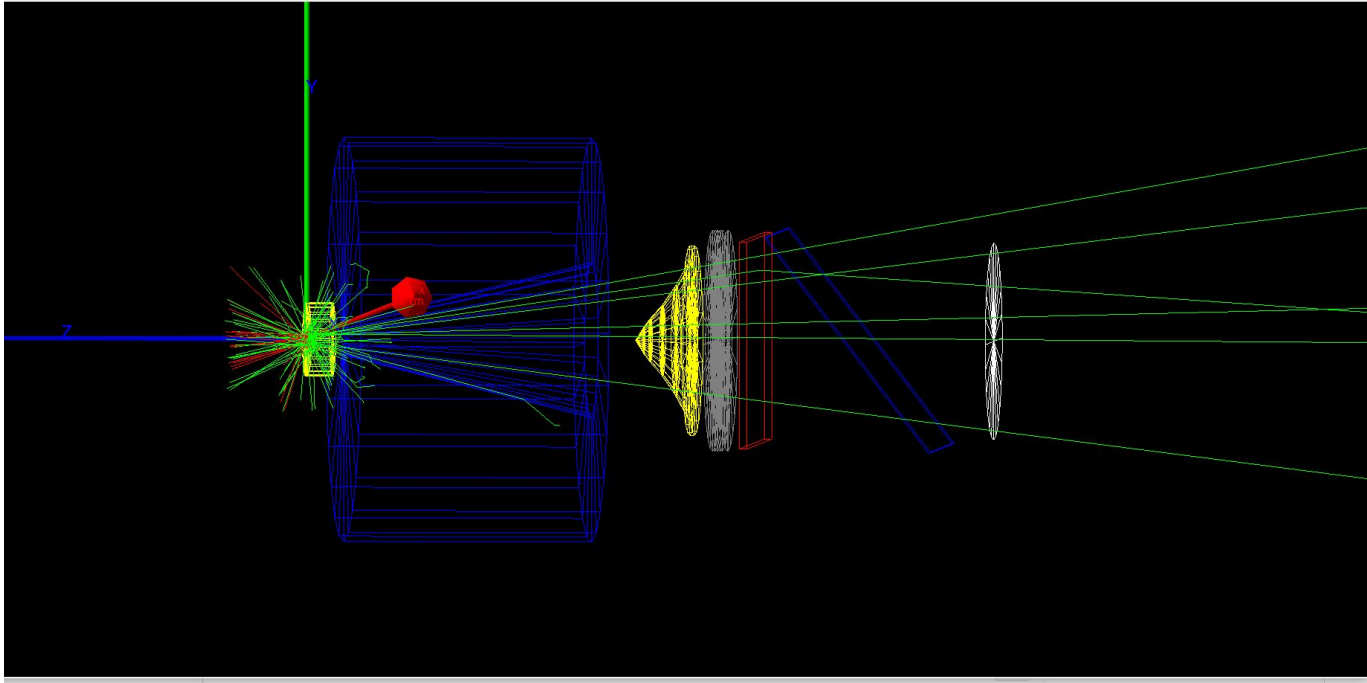
# Russian Roulette and Splitting

- 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.

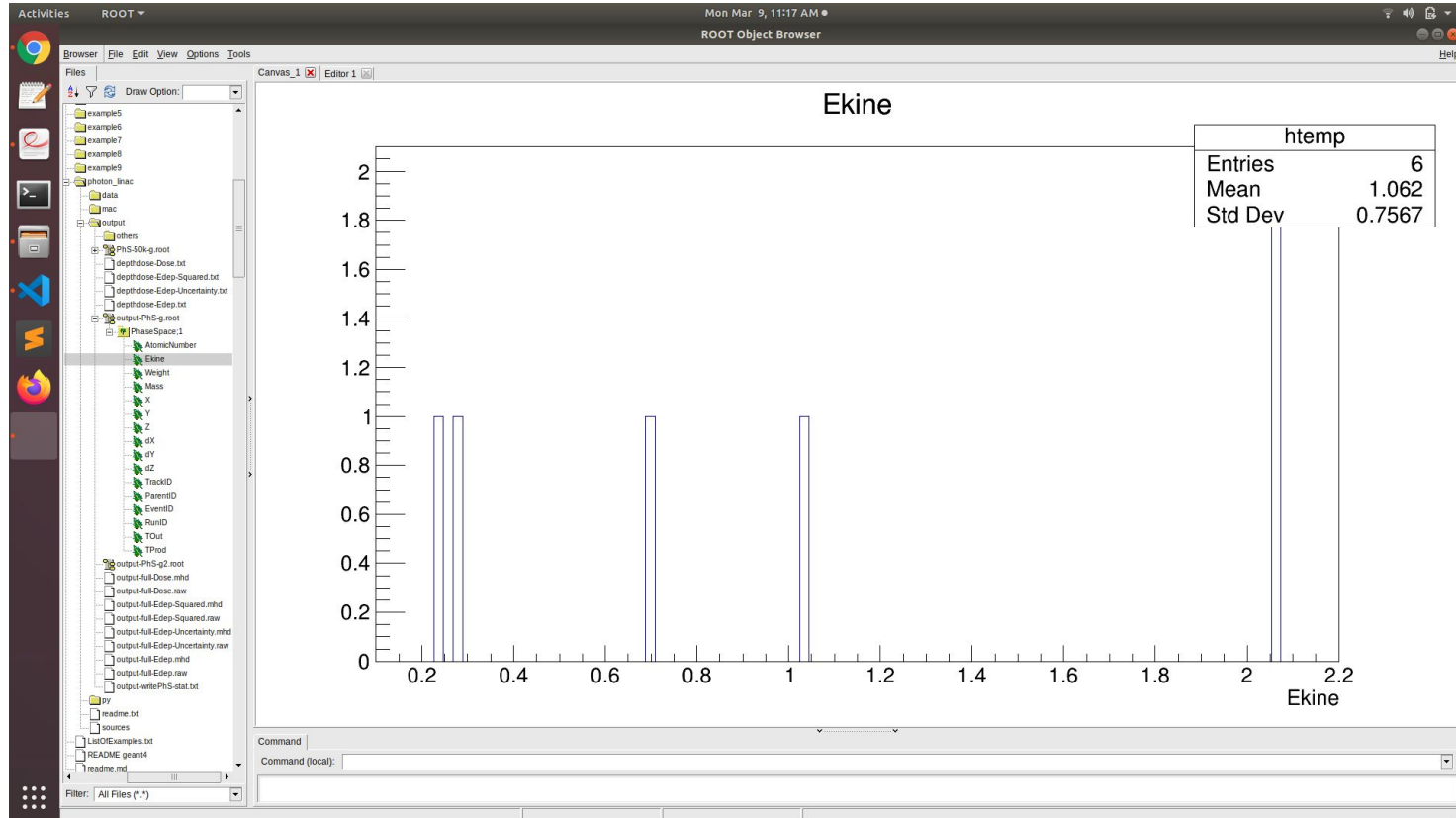
# Without Bremsstrahlung splitting - 500 primaries (electrons)



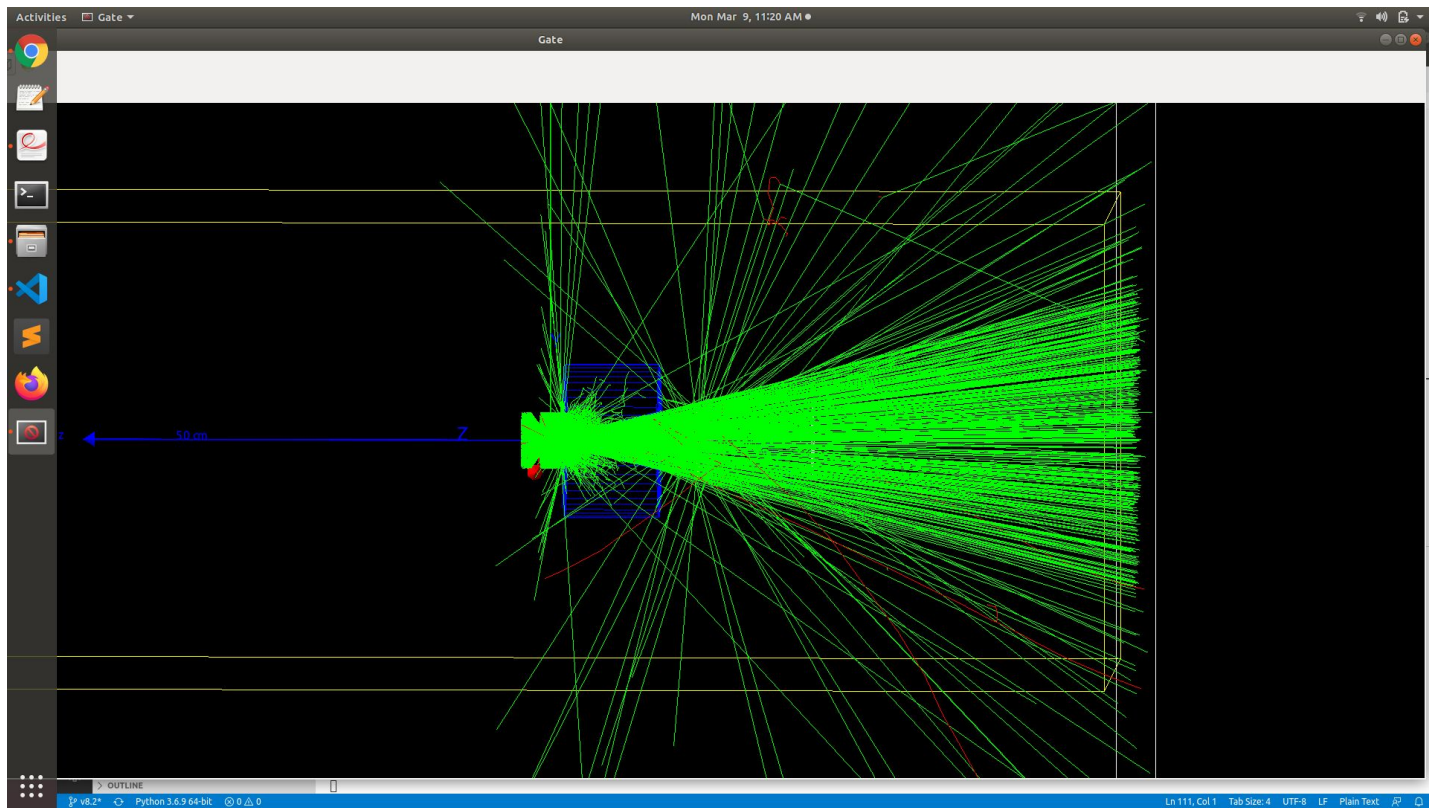
# Closer look



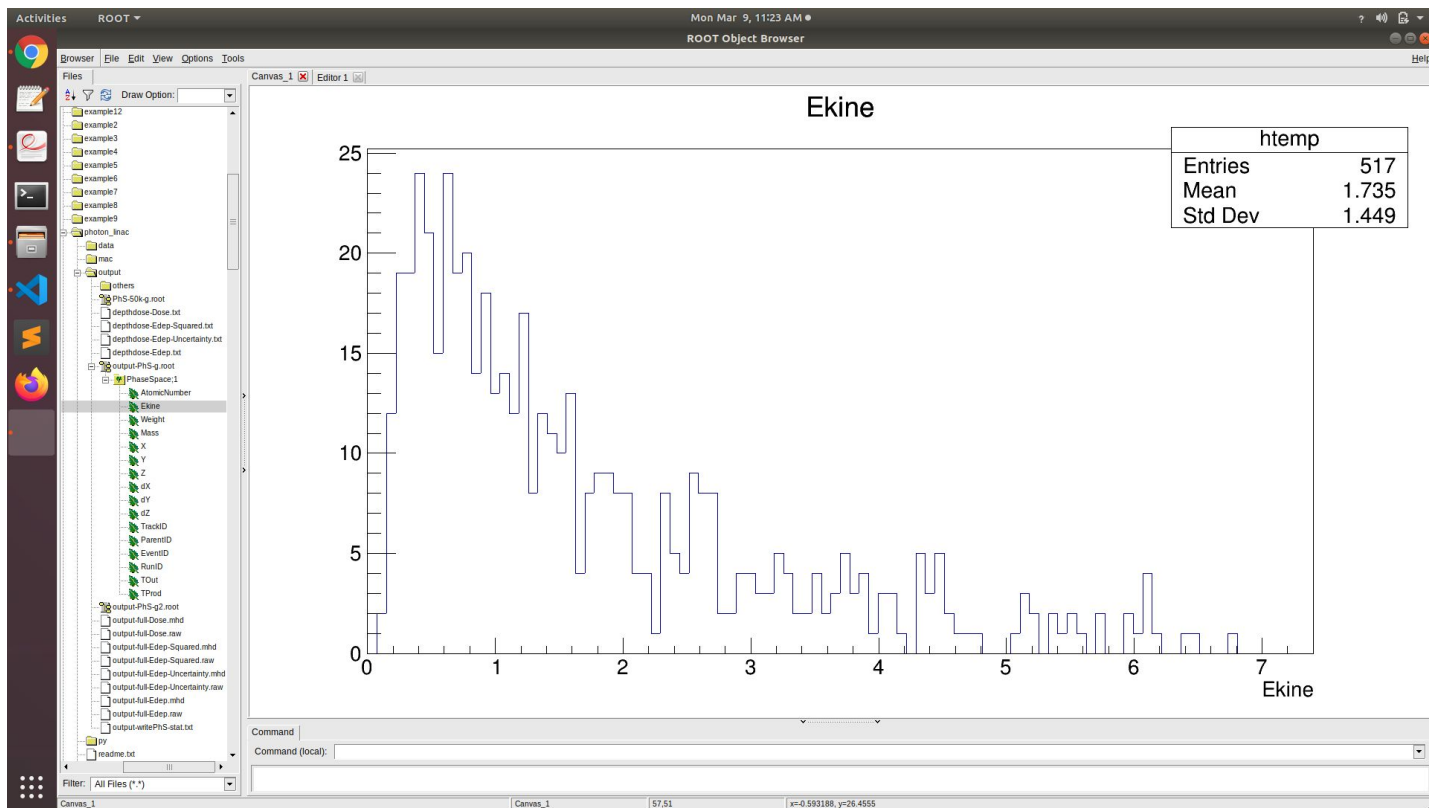
# Phase Space has only 6 entries (Only 6 photons formed)



# 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](#))

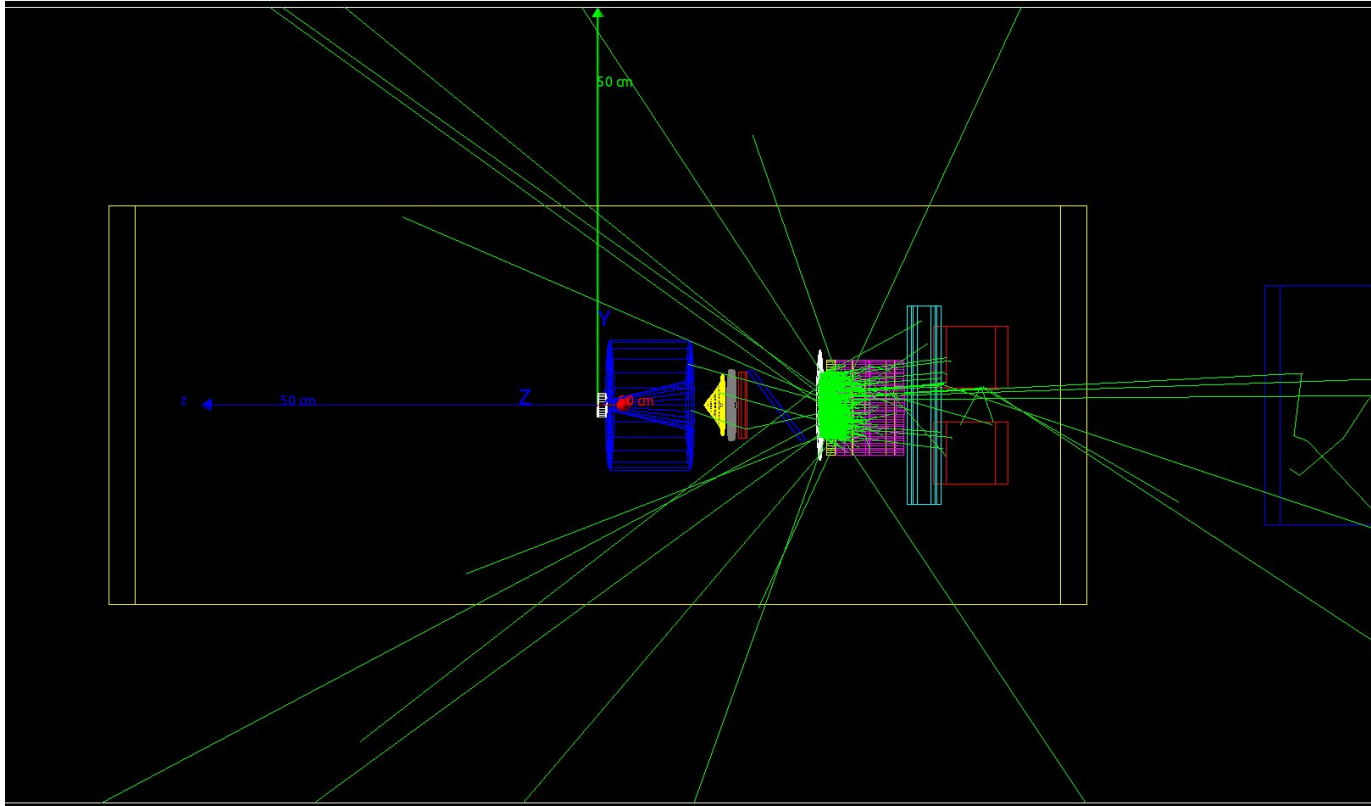
# Kill Actors

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.

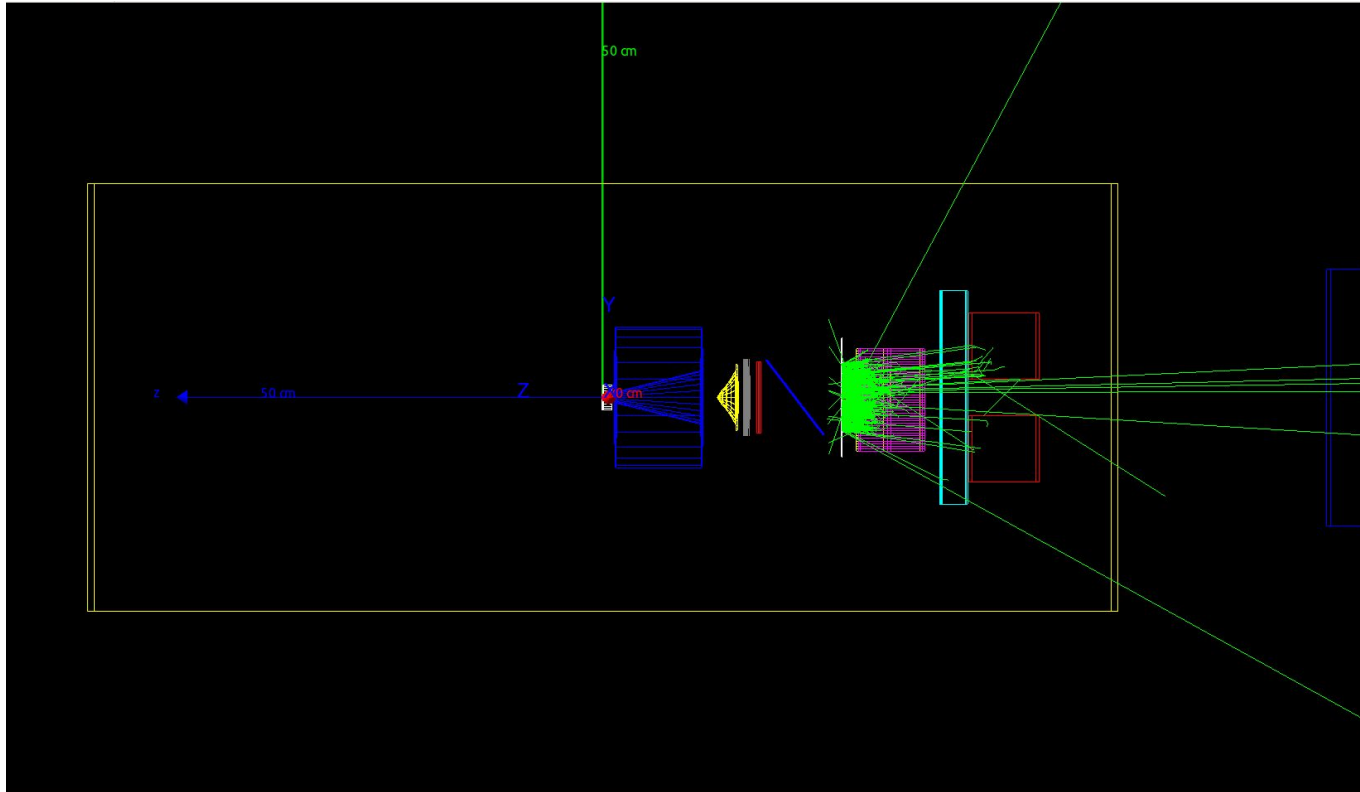
Killing beams in unwanted directions will **save simulation time** as their trajectories need not be followed or recorded any further.

The next slides demonstrate its use.

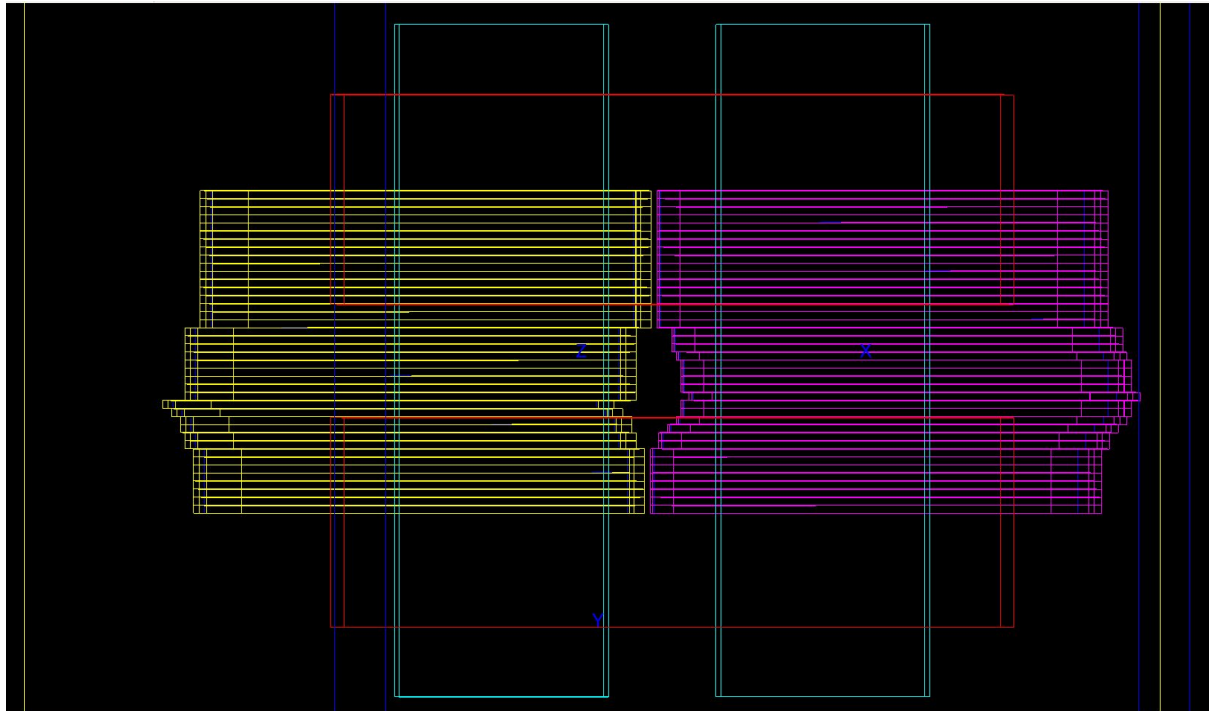
# Scattered particles from PhS - without KillActor



# Scattered particles from PhS - with KillActor

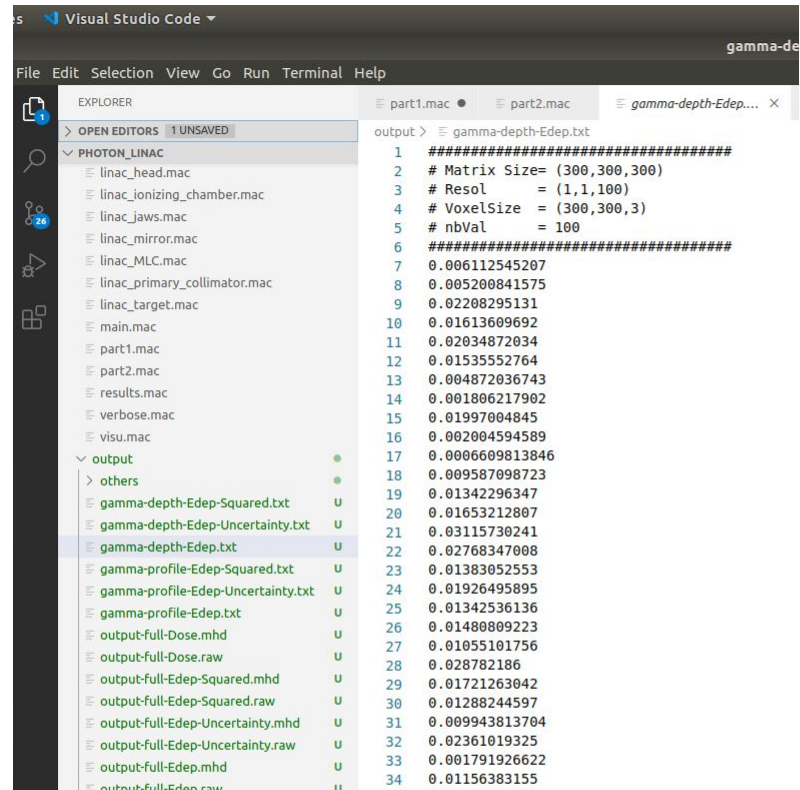


# MLCs placements - a visual



# OUTPUTS

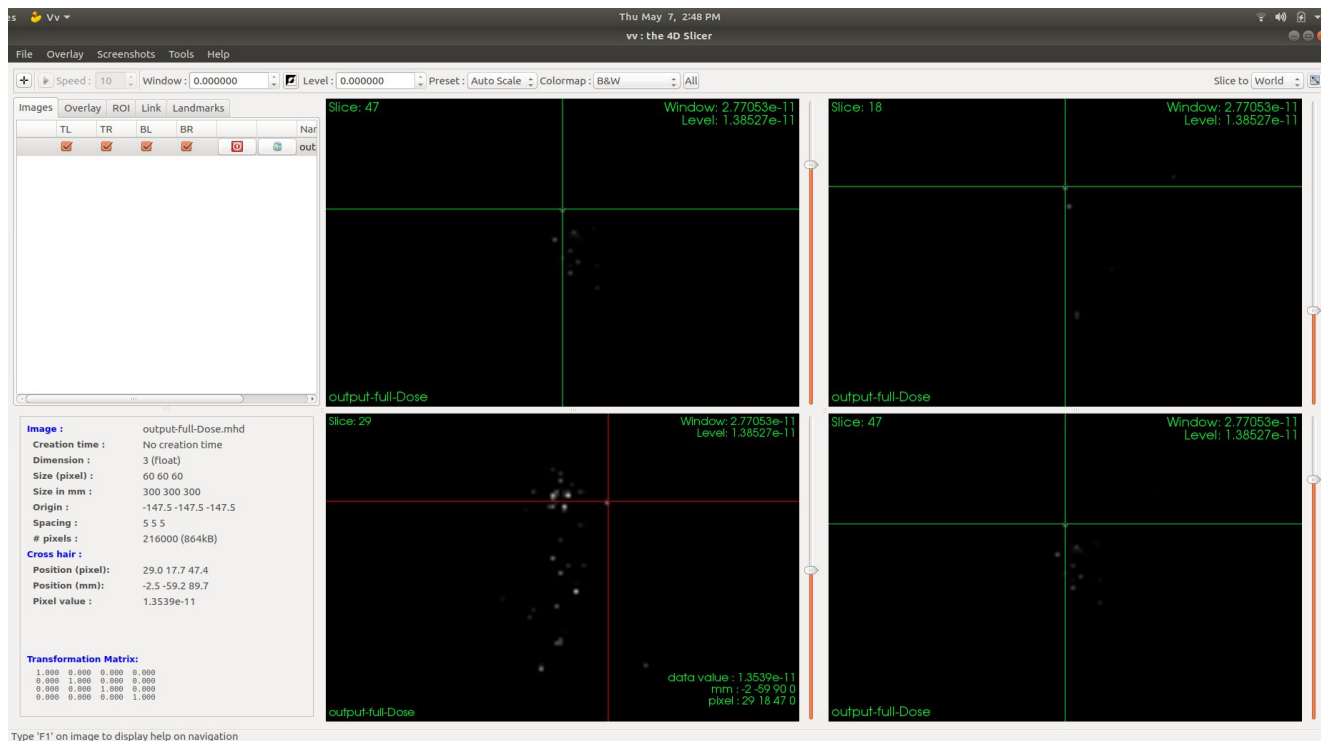
1. The energy/dose deposited on the waterbox (phantom) as per the given specifications
2. The square of the energy/dose depositions.
3. The uncertainty of the results (reduces as the number of trials increases)



```
Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS 1 UNSAVED
PHOTON_LINAC
  linac_head.mac
  linac_ionizing_chamber.mac
  linac_jaws.mac
  linac_mirror.mac
  linac_MLC.mac
  linac_primary_collimator.mac
  linac_target.mac
  main.mac
  part1.mac
  part2.mac
  results.mac
  verbose.mac
  visu.mac
output
  others
  gamma-depth-Edep-Squared.txt
  gamma-depth-Edep-Uncertainty.txt
  gamma-depth-Edep.txt
  gamma-profile-Edep-Squared.txt
  gamma-profile-Edep-Uncertainty.txt
  gamma-profile-Edep.txt
  output-full-Dose.mhd
  output-full-Dose.raw
  output-full-Edep-Squared.mhd
  output-full-Edep-Squared.raw
  output-full-Edep-Uncertainty.mhd
  output-full-Edep-Uncertainty.raw
  output-full-Edep.mhd
  output-full-Edep.raw
output > gamma-depth-Edep.txt
1 #####
2 # Matrix Size= (300,300,300)
3 # Resol      = (1,1,100)
4 # VoxelSize  = (300,300,3)
5 # nbVal      = 100
6 #####
7 0.006112545207
8 0.005200841575
9 0.02208295131
10 0.01613609692
11 0.02034872034
12 0.01535552764
13 0.004872036743
14 0.001806217902
15 0.01997004845
16 0.002004594589
17 0.0006609813846
18 0.009587098723
19 0.01342296347
20 0.01653212807
21 0.03115730241
22 0.02768347008
23 0.01383052553
24 0.01926495895
25 0.01342536136
26 0.01480809223
27 0.01055101756
28 0.028782186
29 0.01721263042
30 0.01288244597
31 0.009943813704
32 0.02361019325
33 0.001791926622
34 0.01156383155
```

# OUTPUTS

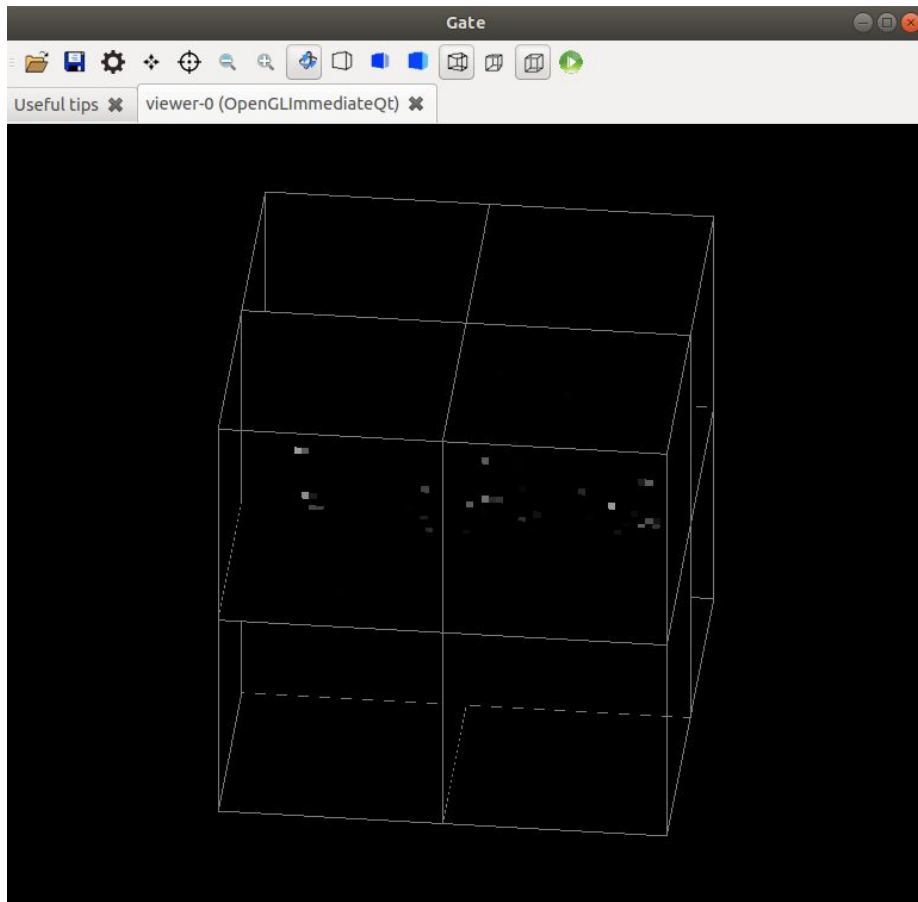
The 3-D figures of the energy depositions are stored in mhd-rw format, which can be viewed using [vv software](#).



# OUTPUTS

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

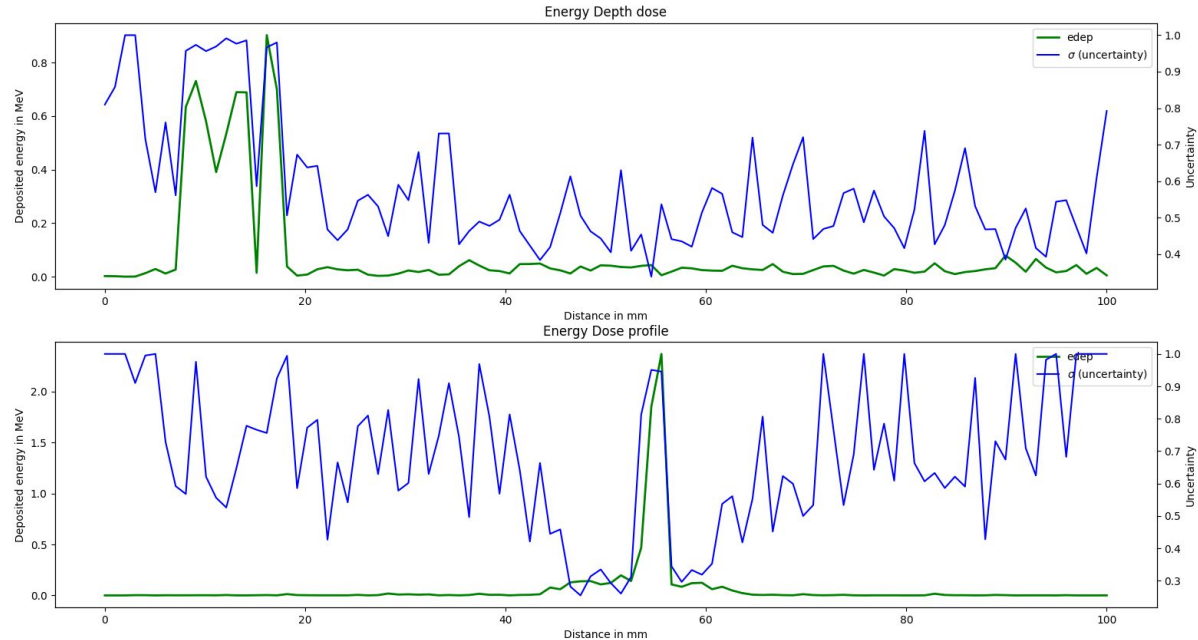
But it is difficult to analyse the result  
from this 3D visual. The greyish white  
spots are the energy/dose  
depositions in the phantom.





# OUTPUTS

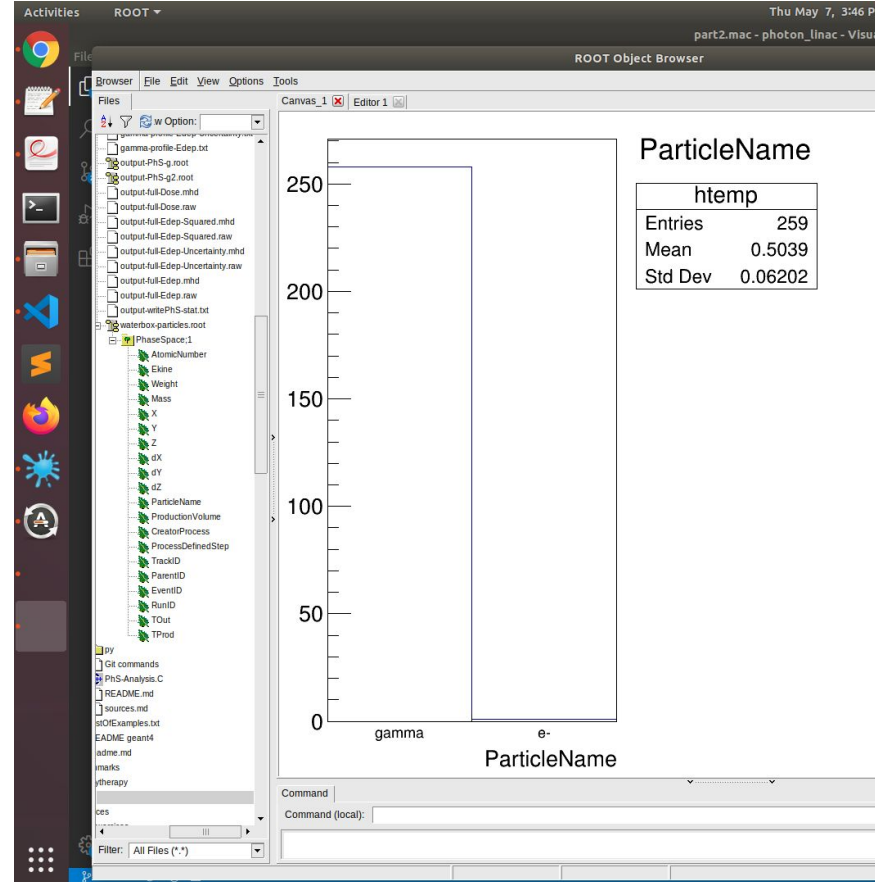
3. The dose outputs can be saved in text files to obtain graphical outputs of depth dose and dose profile in python -



# 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)