Contents

CHAPTER # 1

Introduction to Geant4

Geant4 is the successor of Geant3, the world-standard toolkit for the simulations for HEP detector simulations. It is used for the tracking and geometry for the detectors and experiments. It was prepared by CERN (The European Organization for Nuclear Research). It is used the platform of C++. For our own easiness, we used the G4 instead of Geant4.

1.1 Flexibility of Geant4:

Geant4 provides a wide variety of the functionality and flexibility:

- It provides a wide variety of geometrical descriptions which are most complicated and realistic geometries
- CGS, BREP, Boolean
- Placement, replica, parametrized, reflected, grouped
- XML interface
- Choice of physics processes / model
- Choice of GUI / Visualization/ Persistency / Histogramming technologies

1.2 Physics in Geant4:

There are following physics process and models:

- EM processes
- Hadronic processes
- Photon / Lepton-hadron processes
- Optical photon processes
- Decay processes

- Shower parametrization
- Event biasing techniques

1.3 Purpose of Uses:

It is used for the simulation of passage of particles from the matter. It provides us to the complete set of tools for the study of simulation:

- Geometry and Tracking
- Physics Processes and models
- Graphics and User Interfaces
- Propagation in the relevant fields
- G4 physics processes describe electromagnetic and nuclear interactions of particles with matter, at energies from eV to TeV

1.4 Advantages

- - Simulation of the Geometry of complex setups with efficiently
- - Provide configuration of physical process for application areas
- - Provide the facility for Data Analysis

1.5 Applications

I. **High energy and nuclear physics detectors**
   - ATLAS, CMS, HARP and LHCb at CERN and BaBar at SLAC

II. **Accelerator and shielding**
   - Linacs for medical use

III. **Medicine Radiotherapy**
   - photon, proton and light ion beams
   - brachytherapy
   - boron and gadolinium neutron capture therapy

IV. **Simulation of Scanners**
   - PET and SPECT with GATE (Geant4 Application for Tomographic Emission

V. **Space**
   - Satellites
   - effect of space environment on components (especially electronics

- shielding of instruments
- charging effects

VI. Space Environment

- cosmic ray cut-offs

VII. Astronauts

- dose estimates

CHAPTER # 2

Installation of G4

For the installation of G4 we can use the following operating systems:

- -Unix/Linux on PC with g++\\
- -Mac with g++\\
- -Windows with Visual Studio with c++ compiler

2.1 Installation requirements

- --Visual Studio 2008, 2009, 2010 (With c++ compiler). We used the Visual Studio 2010.
- -CMAKE version 2.8 (cmake gui) or another latest version like 3.11. We used the both CMAKE version 2.8 and 3.11. Either you can use CMAKE versions or dependent own your choice.
- -Operating systems: Windows 7, Windows 8, Windows 10. We used the Windows 10 Operating system.
- -Geant4 source code is: Geant4.9.5. p01

2.2 Installation Process

There are following steps for the installation of the Geant4.

Step1

Install the visual studio 2010 in the windows operating system Windows 10. You can download the Visual Studio from internet, which is easily available on the internet.

Step2

Install the cmake version 2.8 on the windows.

Step3

Now place the source code of Geant4.9.5 in the desired location or drive.

Note

Place your source code Geant4.9.5 in other drive rather than Drive C, sometimes there is error occur like administrator errors. So, to avoid this please you can choose another location for your Geant4 source code.

Here the source code 4.9.5 is an older version so, it's not available on the CERN website you can download this older version from the following link:

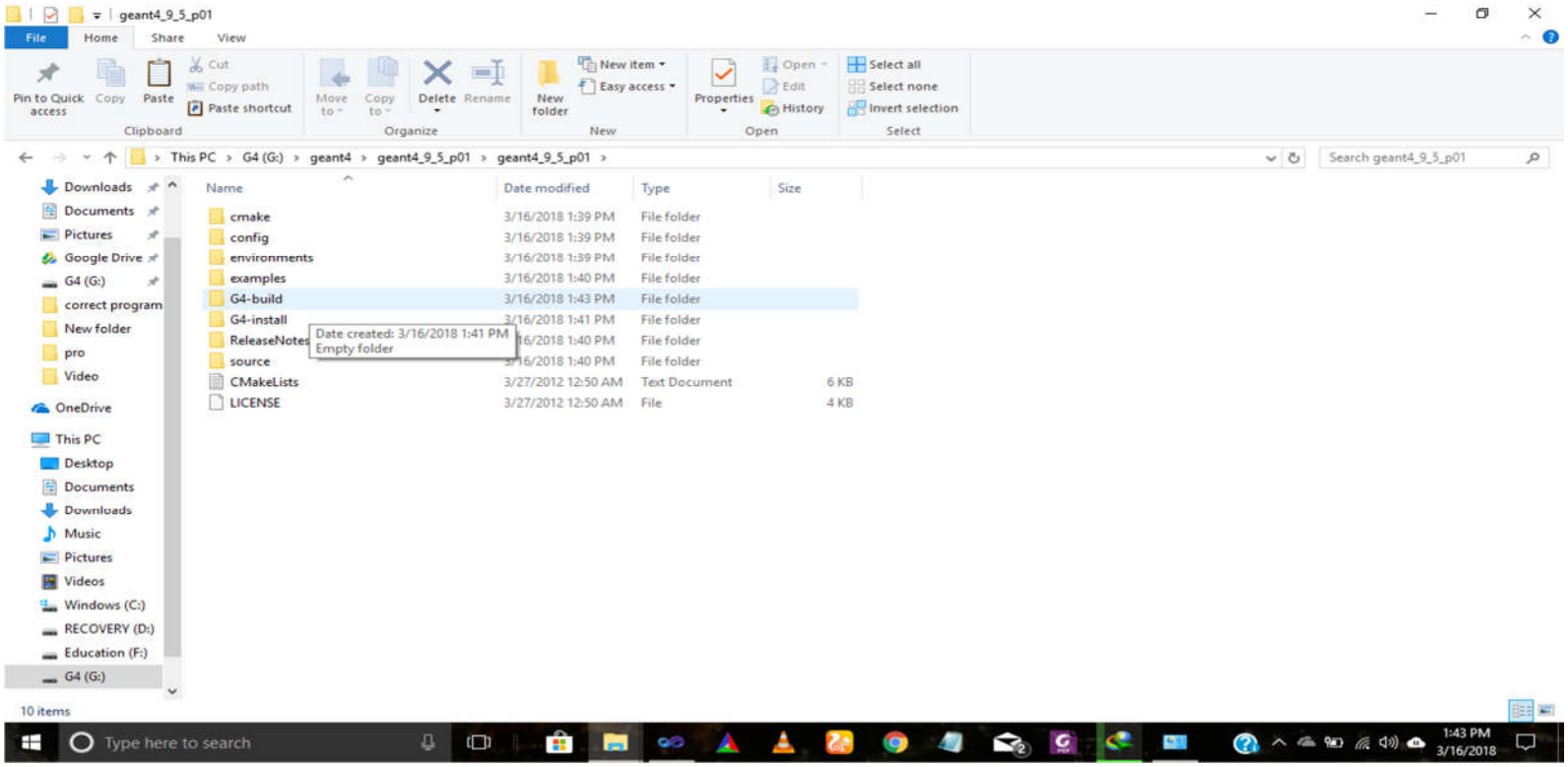
https://www.jlab.org/12gev_phys/packages/sources/geant4/?C=D;O=A

The **CMAKE version 2.8** is also note available on their official website, either you can download this version from the following link or you can download the latest version of **CMAKE 3.11** from the official website:

https://cmake.org/files/v2.8

After downloading the **Geant4.9.5** source code which is in zip file, now unpack or extract this zip file in your desired directory or drive.

Make the two folders with name as **G4_build** and **G4_install** like as shown in the fig:

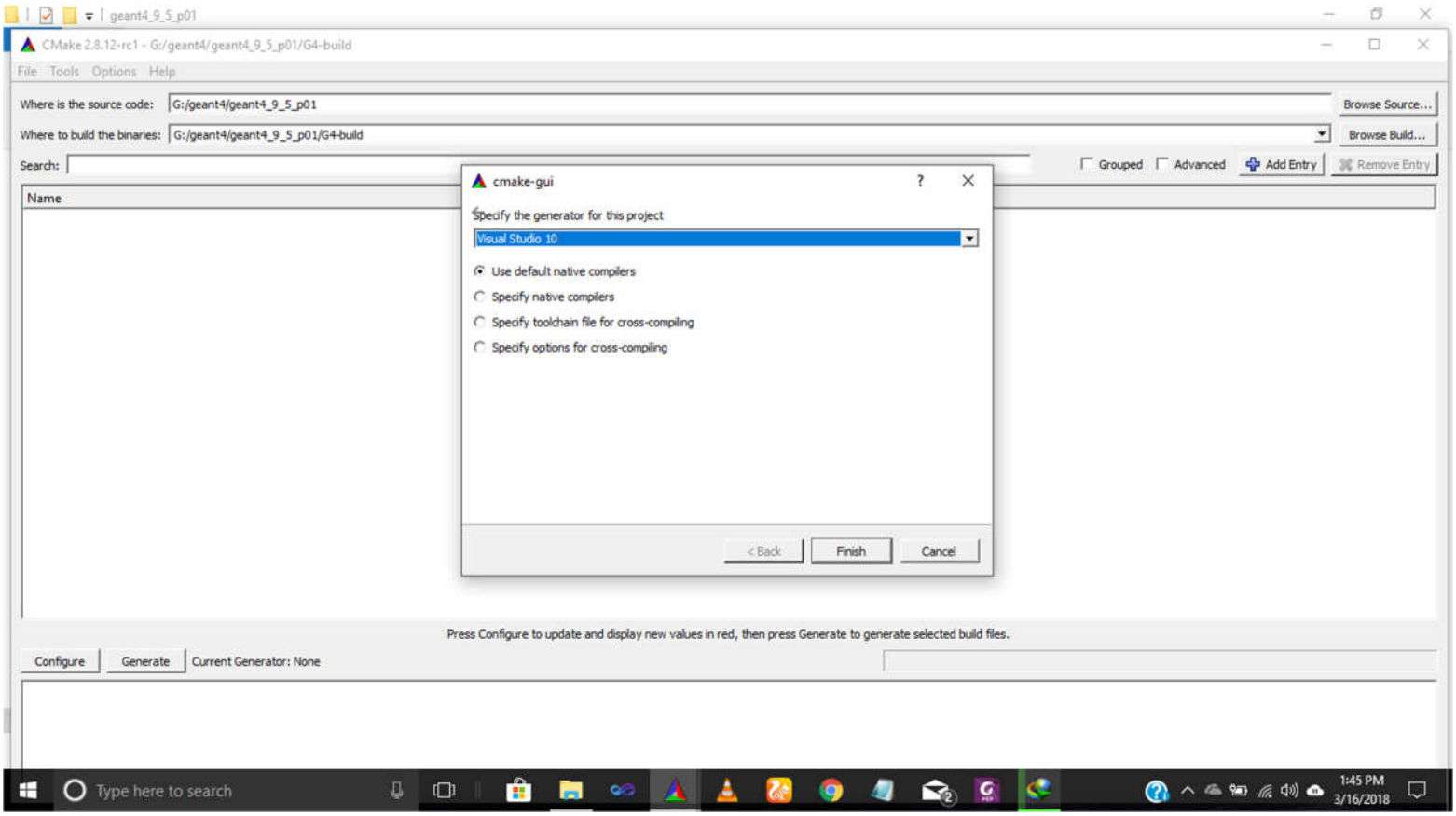


Now, copy the link following for the build data for Geant4.

G:\geant4\geant4_9_5_p01

G:\geant4\geant4_9_5_p01\G4-build

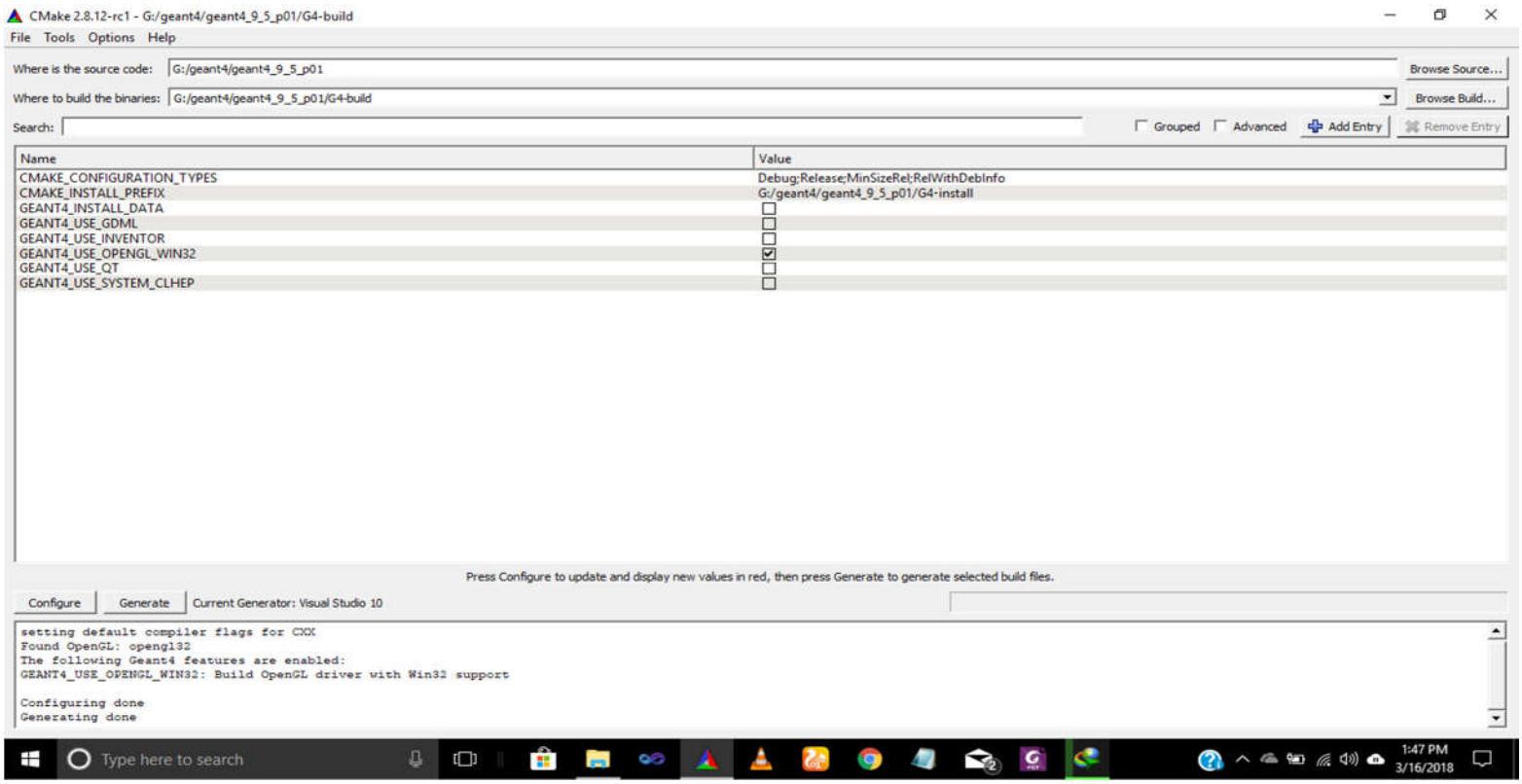
Click the Configuration button in the cmake and the display the windows with **"specify the generator for this project"** like given below:



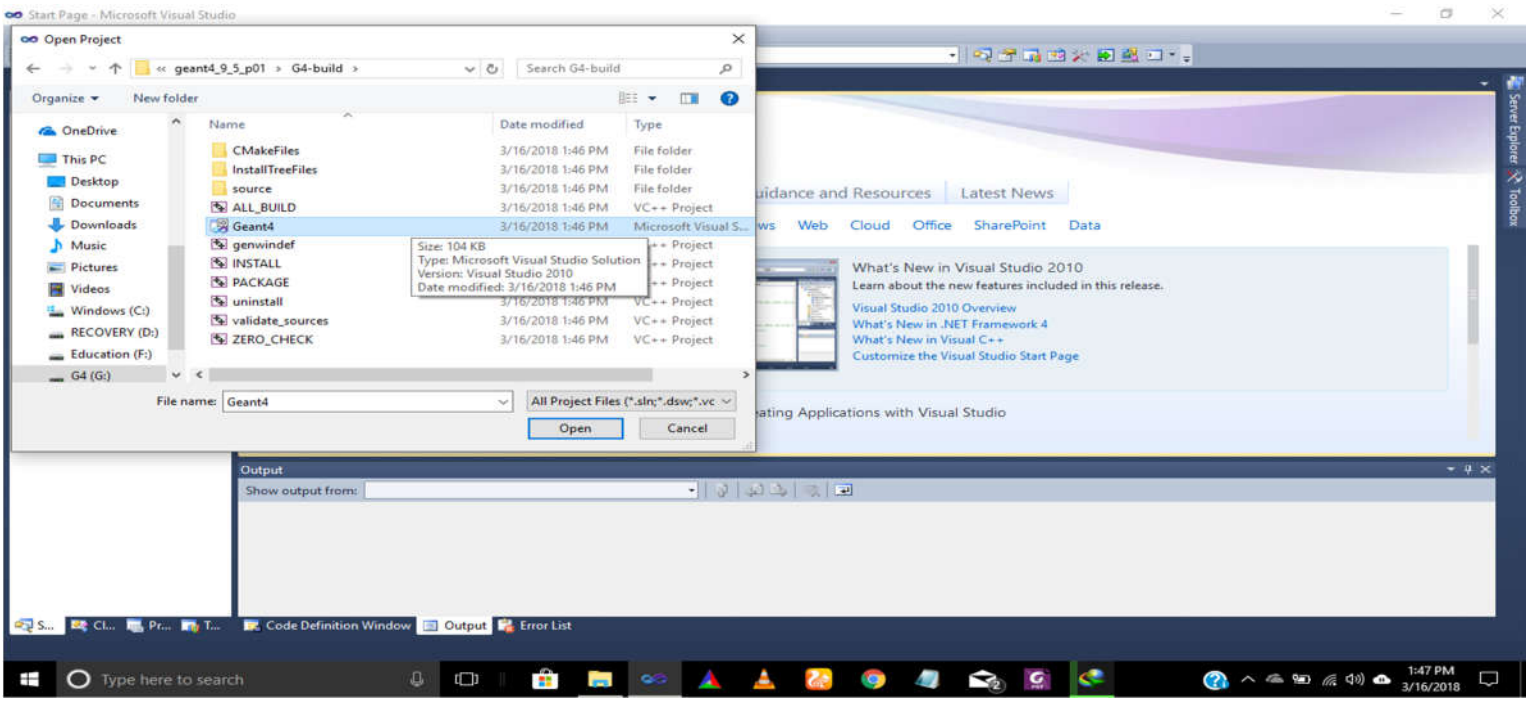
In the above windows **"specify the generator for this project"**, select the **Visual Studio 2010** then **'Finish'** this window. After finishing the windows, again the windows will open and to paste the address in the **"CMAKE_INSTALL-PREFIX"** tab. The address is:

G:\geant4\geant4_9_5_p01\G4-install

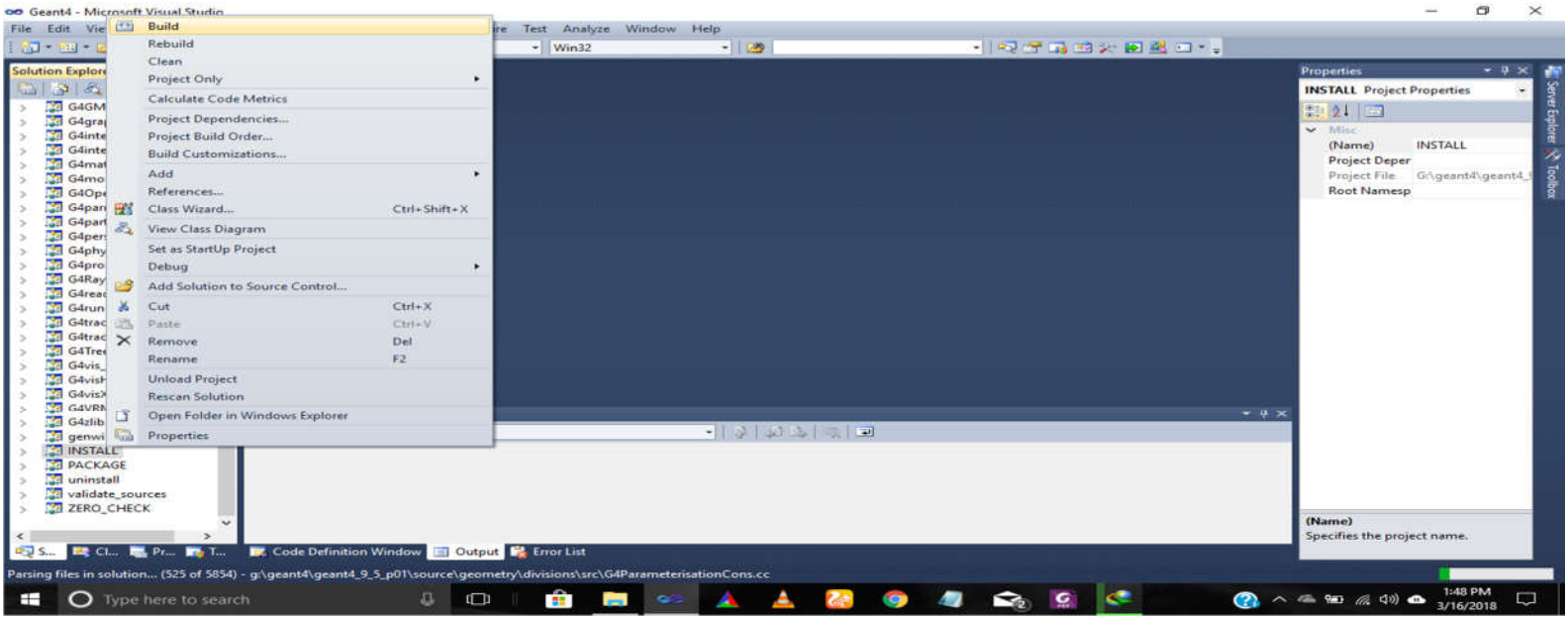
Select (Tick) the tab **"GEANT4_USE_OPENGL-WIN32".** Click the button of **'configure'** as well as to select the button of **'generate'**. After doing this process, there will display a message like **'Configuring done'** and **'Generating done'**. The fig is:



After completing this process. Open the **'Visual Studio 2010'**, click the 'File' tab and open the Project/solution or just press the (Ctrl+Shift+O). There will show windows and click the file Geant4.sln (sln it's data types) and open it, like this:

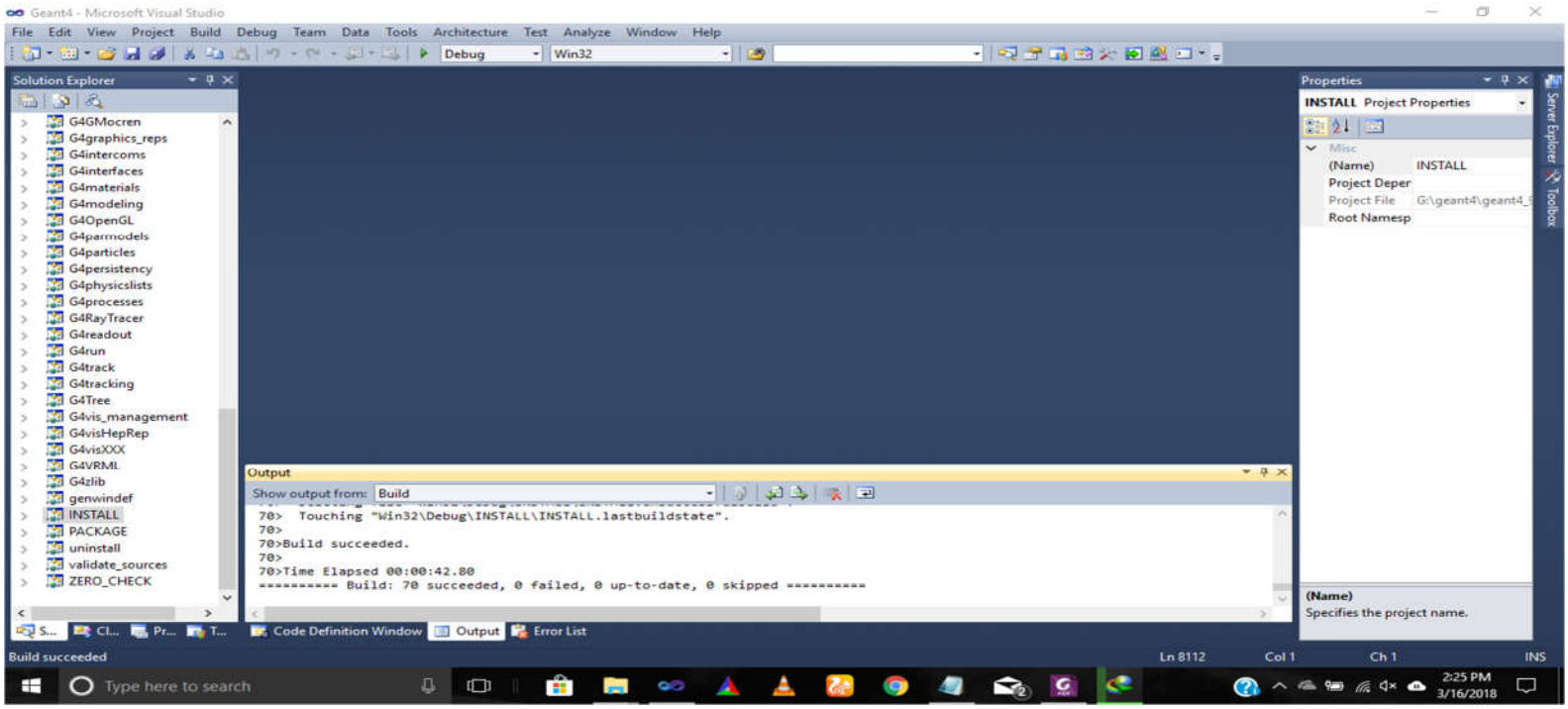


There will show a **“Solution Explorer” and** scroll down the **“Solution Explorer”. And again** there will show a tab like **“INSTALL”** and right click on the **“INSTALL”** tab. There will show windows and click on the **“Build”** tab like below:



There will show a message the given bellow:

******Build: 70 Succeeded, 0 failed, 0 up-to-date, 0 skipped ******



Exit the **‘Visual Studio 2010’.**

2.2 DATA requirement

Now download the DATA required for this:

Note: The Geant4.9.5 DATA is not available again in the CERN website, you can download this from the following link:

http://distfiles.macports.org/geant4/

After downloading the DATA, unpack it in the following link directory:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data

2.3 PATH Setting

After unpacking this data and set the PATH as Environment variables. For this purpose, we set the Variable Name and Variable Values like as that is used this link:

Control Panel\System and Security\System

Here open the **"Advanced System Settings",** click on the **"Environmental Variables"** and use the tabs of **'User Variables'**. Click on the **'New',** write the "**Variable Name"** as **PATH** and write the "**Variable Value"**

G:\geant4\geant4_9_5_p01\G4-install\bin

Variable Name:

G4ABLADATA

Variable Value:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data\G4ABLA3.0

Variable Name:

G4ENSDFSTATE

Variable Value:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data\G4ENSDFSTATE1.2.3

Variable Name:

G4LEDATA

Variable Value:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data\G4EMLOW6.23

Variable Name:

G4LEVELGAMMADATA

Variable Value:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data\PhotonEvaporation2.2

Variable Name:

G4NEUTRONHPDATA

Variable Value:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data\G4NDL4.0

Variable Name:

G4PIIDATA

Variable Value:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data\G4PII1.3

Variable Name:

G4RADIOACTIVEDATA

Variable Value:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data\RadioactiveDecay3.4

Variable Name:

G4REALSURFACEDATA

Variable Value:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data\RadioactiveDecay3.4

Variable Name:

G4SAIDDATA

Variable Value:

G:\geant4\geant4_9_5_p01\G4-install\share\Geant4-9.5.1\Data\G4SAIDDATA1.1

The installation process has been completed.

CHAPTER # 3

Examples in Geant4

There are many examples in G4 which include the source codes. There are following four types of examples which contain the source codes:

- Advanced examples
- Basic examples
- Extended examples
- Novice examples

3.1 Basic Examples

These examples include the user classes, here it's not necessary to include the external libraries like CLHEP and we can work the Geant4 libraries. There are basically four examples in the basic examples B1, B2a, B2b, B3, B4a, B4b, B4c and B4d.

3.1.1 Example B1

Example B1 contains the following features:

- Simple geometry with few solids like boxes, cones and trapezoids
- Geometry with simple placement
- For generating the particles, use particle gun
- There are user action classes
- The Physics List is QBBC

3.1.2 Example B2

Example B2 contains the following features:

- Simplify the tracker geometry with applied uniform magnetic field
- Geometry is simple placement and parameterization
- Scoring within a tracker via sensitive detector and hits
- Physics list is FTFP-BERT
- It also used the example NO:2 in Novice example

3.1.3 Example B3

Example B3 contains the following features:

- Schematic Positron Emission Tomography system
- Geometry with simple rotation placements (G4PVPlacement)
- Radioactive Source
- Scoring within crystals Via G4 Scorers
- Use the Modular Physics provided in G4

3.1.4 Example B4

Example B4 contains the following features:

- The example B4 designed for the Calorimeter with two materials
- The geometry with (G4 Replica)
- There are Scoring in four ways:(a) via User Action, (b) Via user own object, (c) Via G4 Sensitive detector and hits, (d) Via Scorers
- Physics list is FTFP-BERT.
- The UI commands used by G4GenericMessenger
- Starting from the novice example /N0:3 example

3.2 Extended Examples

The purposes of extended examples are:

- Testing and validation of processes and tracking
- Demonstration of Geant4 tools
- Extending the functionality of Geant4

The description of the example is given below:

a) <u>Analysis</u>

- The Histograms through the AIDA interface

b) Biasing

- Examples of event biasing, scoring and reverse-MC-

c) Common

- A set of common classes which can be reused in other examples demonstrating just a feature

d) Electromagnetic

- Specific EM physics simulation with histograms

e) Error propagation

- Use of the error propagation utility (Geant4)

f) Event generator

- Applications using interface to Hep MC

g) Exotic Physics

- Exotic simulation applications (classical magnetic monopole, etc...)

h) Field

- Specific simulation setups in magnetic field

i) G3toG4

- Examples of usage of the G3toG4 converter tool

j) Geometry

- Specific geometry examples and tools: OLAP tool for detection of overlapping geometries

k) Hadronic

- Specific hadronic physics simulation with histograms

l) Medical

- Specific examples for medical physics applications

m) Optical

- Examples of generic optical processes simulation setups

n) Parallel

- Examples of event-level parallelism in Geant4 using the TOP-C distribution, and MPI technique

o) Parameterizations

- Examples for fast shower parameterizations according to specific models

p) Persistency

- Persistency of geometry (GDML or ASCII) and simulation output

q) Radioactive decay

- Examples to simulate the decays of radioactive isotopes and induced radioactivity resulted from nuclear interactions.

r) Run and Event

- Examples to demonstrate how to connect the information between primary particles and hits and utilize user-information classes.

s) Visualization

- Specific visualization features and graphical customizations

3.3 Novice Examples

These Novice examples are 7 in numbers. These examples are hard coded and complex examples provided by the G4. The explanation of the Novice examples is:

3.3.1 Example N01

- Simple example to demonstrate how the GEANT4 basic framework works. No physics involved, only tracking on a simple geometry.

3.3.2 Example N02

- Simulation of a simplified fixed target experiment.
- It includes a simple parameterized geometry. It defines a transverse uniform magnetic field. It includes standard EM physics for gammas, charged leptons and charged hadrons. It includes visualization and detector response.

3.3.3 Example N03

- Simulation of a simple sample Calorimeter setup. It shows the usage of 'replicated' volumes in the geometry.
- It defines user commands for primaries generation (isotropic random distribution). It defines a transverse uniform magnetic field. It includes detector response and statistics on the relevant quantities. It includes a tutorial for visualization, exercising different and visualization drivers.

3.3.4 Example N04

- It defines a simplified collider detector setup. It demonstrates interfacing to the PYTHIA primary generator. It includes the definition of a 'readout' geometry. It defines all ordinary physics processes for leptons and hadrons. Its exercises on event filtering by using the stacking mechanism and It includes visualization.

3.3.5 Example N05

It implements a setup for fast-parameterization. It provides a simple shower parameterization for e+/e-/gamma in an EM calorimeter and a pi+/pi- parameterization by using ghost volumes. It implements the parameterized detector response.

3.3.6 Example N06

The simulation of optical photons generation and transport, define optical surfaces and exercises on optical physics processes (Cerenkov, Scintillation, Absorption, Rayleigh ...). By using stacking mechanism to count the secondary particles generated.

3.3.7 Example N07

- It implements three simplified sandwich calorimeters. It shows how to modify the part of the geometry setup at run-time. It includes detector description parameterization by materials. It demonstrates sharing of a sensitive detector by definition of different sub-detectors.
- It defines different geometrical regions with different production thresholds. It shows customization of the G4Run.

3.4 Advanced Examples

Geant4 provides the advanced examples with realistic applications in the experimental environment. Most of them are used for the analysis tools like (histograms, ntuples and plotting), some other advanced visualization tools and user interfaces facilities in the simulation core for Geant4. The advanced examples are included:

Air shower

It is used for the simulation of Fresnel lens focusing direct or reflected UV light onto a photomultiplier. The object of parameterization and replication of the Geant4 is used to describe the lens geometry. It's used in the configuration of ULTRA experiments.

amsEcal

It's used for the simulation of AMS electromagnetic calorimeter.

brachytherapy

It is used in medical physics as well as applications for the energy deposit in a Phantom filled with soft tissue.

ChargeExchangeMC

It's used for the simulation of real experiments in Petersburg Nuclear Physics Institute (PNPI, Russia).

Composite Calorimeter

It's used for the test beam simulation of the CMS Hadron calorimeter at LHC.

DNA Physics

It provides for the simulation of the Geant4 DNA physics to very low energy for the transport of particles in liquid material.

ERosita

It provides the simplified version for the simulation of the shielding of the eRosita X-ray-mission. It demonstrates the simulation of the PIXE (Particle Induced X-ray Emission).

Gamma Ray Telescope

Its application to typical x-ray telescope with flexible configuration.

Hadron therapy

Its simulation is based on the Monte Carlo studies related to the proton/ion therapy. It provides the facility for the calculations of fundamental quantities like: 3D dose distribution, fluencies and average LET for both primary and secondary particles.

Human Phantom

Its implementing for the Anthropomorphic Phantom body built to import the description from a GDML representation.

Iort Therapy

It's specially developed for the typical needs related to the Intraoperative Radio-Therapy (IORT) technique. It is used for the single dose of radiation directly to the tumor bed, or exposed tumor, during surgery. It is also used for the radiation dosimetry, dose planning and radio-protection studies, etc.

lAr-Calorimeter

It is used for the Forward Liquid Argon Calorimeter (FCAL) of the ATLAS Detector at LHC.

Medical-linac

It’s used for the illustration of simulation of the energy deposit in the Phantom filled with water for a typical linac by using the intensity modulated radiation therapy.

Micro beam

It is used to simulate the irradiation beam line and installed on the AIFIRA electrostatic accelerator facility which is located at CENBG, Bordeaux-Gradignan and France.

Micro dosimetry

The microdosimetry example simulates the track of a 5 MeV proton in liquid water. Geant4 standard EM models are used in the World volume while Geant4-DNA models are used in a Target volume, declared as a region.

Nano beam

It is used to simulate the beam optics of "Nano beam line" and installed on the AIFIRA electrostatic accelerator facility which is located at CENBG, Bordeaux-Gradignan and France.

Purging-magnet

It is used to simulate the electrons travelling in 3D magnetic field as well as in medical environment for simulating the strong purging magnet in the treatment of head.

Radioprotection

It illustrates the response characterization of a novel diamond micro dosimeter for radiation protection in human space missions and aviation.

Underground-physics

It’s used for the underground detector for dark matter searches.

X-ray-fluorescence

It’s illustrates the emission of x-ray fluorescence and PIXE.

X-ray-telescope

Its illustrates the application for the study of the radiation backgrounds in a typical x-ray-telescope.

CHAPTER # 4

Introduction to Geant4 Visualization Driver

4.1 Purposes of G4 Visualization

For quick response to geometries, trajectories and hits.

- High quality outputs for publications
- To zoom, rotate the geometries of detectors
- Flexible camera control to debug the complex geometries
- Interactive picking to get more information about the visualization drivers

4.2 Visualization Driver

There are many visualization drivers in G4:

- OpenGL
- Ray-Tracer
- HepRep/WIRED
- DAWN
- ASCII-Tree

4.2.1 OpenGL

Create a file in the .exe file by typing /vis/open OGLIX. It has the following features:

- View directly from G4.
- Zoom, rotate and translate the detector geometries.
- It gives fast response.
- Limited printing ability (pixel graphics, not vector graphics)

4.2.2 HepRep/ WIRED:

Create a file in the .exe file by typing /vis/open HepRepFile. It has the following features:

- Create a file to view in the WIRED3 HepRep Browser
- Wireframe or simple area fills (not photorealistic)
- There are many interactive features about this Driver:
    - Zoom, rotate, translate, reset
    - Click to show attributes like (momentum, kinetic energy, material, state, PID, density etc.)

    - Special projections like (Fish Eye, Z-Phi, perspective, parallel etc.).
    - Special orientation like (Auto Fly, Wobble, Beam view, Auto Rotate etc.).
- Export too many vector graphic formats like (Postscript, PDF, gif, raw, bmp, jpg, jpeg etc.)

4.2.3 RayTracer:

Create a file in the .exe file by typing /vis/open RayTracer. It has the following features:

- Create a jpeg file.
- It is used to show geometries not show trajectories
- It also supports shadow, transparency, and mirrored surfaces.

4.2.4 ASCIITree:

Create a file in the .exe file by typing /vis/open ATree. It has the following features:

- Text dump of the geometry hierarchy
- Not graphical
- You can calculate the mass and volume of any hierarchy of volumes

4.2.5 DAWN:

Create a file in the .exe file by typing /vis/open DAWNFILE. It has the following features:

- Create a file to view in the DAWN Renderer
- Rendered, photorealistic image
- Runs on Linux/Unix and windows
- No interactive features
- Highest quality technical rendering, and camera focusing

4.3 Installation of Geant4 Visualization Driver

The installation of **WIRED** Visualization driver is given in the following link:

http://conferences.fnal.gov/g4tutorial/g4cd/Documentation/Visualization/G4WIREDTutorial/G4WIREDTutorial.html

The installation of **HepRApp** Visualization driver is given in the following link:

http://www.slac.stanford.edu/~perl/HepRApp/

The installation of **DAWN** Visualization driver is given in the following link:

http://geant4.slac.stanford.edu/Presentations/vis/G4DAWNTutorial/G4DAWNTutorial.html

4.4 Basic concepts and Kernel structure of Geant4 Kernel:

4.4.1 Run in Geant4:

A run of Geant4 starts when "BeamOn", we execute the file of .exe by typing /run/beam-On 10. There are a following characteristics of Run in Geant4.

- Within a run we cannot change the geometry of the detector and physical process
- A run is a collection of events which share the same detector conditions.
- At the beginning of the run we can navigate the geometry of detector, cross-section tables and cut-off values.
- **G4RunManager** class manages processing of run, a run is represented by the **G4Run** class.
- **G4UserRunAction** is the optional user hook.

4.4.2 Event in Geant4:

At the beginning of the processing, an event contains the primary particles. These particles are pushed into stack. When the stack becomes empty, processing of event becomes over.

- **G4EventManager** class manages the processing of event.
- **G4Event** class represent an event, and it contains the following processes at the end of event.
    - List of primary vertexes and particles
    - Hits collections
    - Trajectory collections
    - Digits collections
- **G4UserEventAction** is an optional user hook.

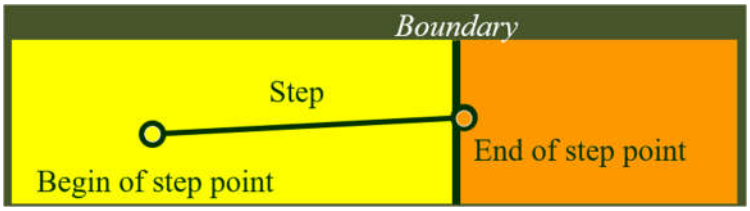
4.4.3 Track in Geant4:

Track is a snapshot of a particle. It contains only the position and physical quantities of current instance.

- Step is a delta information of the track.
- Track is not the collection of steps.
- Track is deleted when:
    - It goes out of the world volume
    - It disappears (e.g. decay)
    - It goes down to zero kinetic energy
    - The user decided to kill it
- No track object persists at the end of event
- **G4TrackingManager** manages the process of a track, the **G4Track** represent the track.
- **G4UserTrackingAction** is the optional user hook.

4.4.4 Step in Geant4:

- Step has two points and it's contain the "delta" information about the particle (like energy of the particle and the time of flight etc.).
- In which of these points each point knows the volume and material. In this case a step is limited to the volume boundary, at the volume boundary, it means the next volume.
    - Because one step knows the information of the material of two volumes, boundary process means the transition radiation or refraction could be simulated.
- **G4SteppingActionManager** class manages process of step, the **G4Step** represents the step.
- **G4UserSteppingAction** is the optional user hook.



4.5 Unit System in Geant4:

Geant4 provides the facility for the unit system which is hard coded to implement in the simulation, we just multiply the proper unit like below:

Length=10.0*cm

kineticEnergy=1.0*GeV or 13.0*TeV

For getting the number we just divide this by unit like below:

G4cout<<Length/cm<<"[cm]"<<G4endl;

G4cout are o-stream objects defined by Geant4. G4endl is also provided.

4.6 Material and Geometry in Geant4:

4.6.1 Construction of Detector in Geant4:

Describe the detector class from the concrete or base class **G4VUserDetectorConstruction**, it has the following features:

- Construct all the necessary materials
- Define the solid/shapes for the geometry
- Construction and placement of volumes for detector
- Instantiate the sensitive detectors and set them for the corresponding volumes (optional)
- Set the magnetic field to detector (optional)
- Set the visualization attributes for the detector (optional)

4.6.2 Defining the Materials:

We can define the material of the detector there are following methods:

- Isotopes, elements, compound, mixtures and molecules.

Here we use only one method, which are very simple like below:

```
G4NistManager* nist = G4NistManager::Instance();
G4Material* env_mat = nist->FindOrBuildMaterial("G4_WATER");
```

4.6.3 Solid and shape:

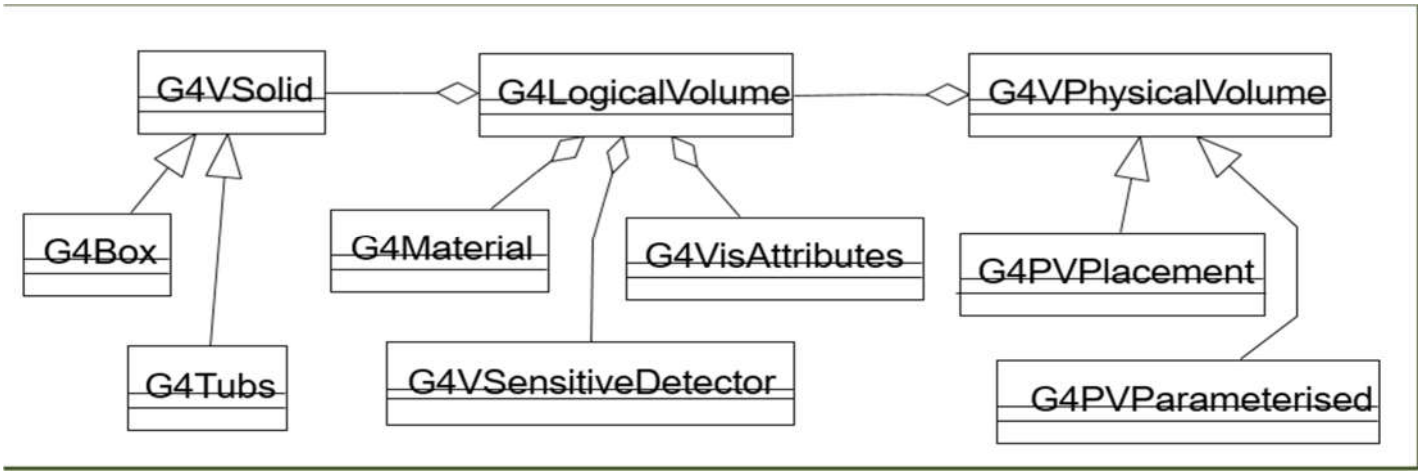
All the shapes of the detectors are abstracted from the **G4VSolid** class, from this class we can derive the other type of solid classes like (**G4Tubs, G4Cons etc.**)

Solids can be defined in **Geant4** by:

- **CSG (Constructed Solid Geometry)** solids like **G4Box, G4Tubs, and G4Cons etc.**
- **BREP (Boundary REPresented)** solids like **G4BREPSolidPolycone, G4BSplineSurface** etc.
- **Boolean Solid** like **G4UnionSolid, G4SubtractionSolid etc.**

4.6.4 Geometry of the Detector:

There are following layers which are used to define the detector geometry:



- A physical volume contains the physical area which must exist and contains the other element like the **World Volume.**

4.6.5 G4LogicalVolume:

It's contains the followings information except of the position and rotation:

- Shapes and dimensions (**G4VSolid**)
- Material, sensitivity, visualization attributes
- Position of the daughter volumes

- Magnetic field

4.6.6 Visualization Attributes:

- Each logical volume can be associated to visualization attributes by using the class **G4VisAttributes** object, which contains the following properties:
    - Visibility of the daughter volumes
    - Invisibility of the daughter volumes

4.6.7 Physical Volume:

There are five types of physical volumes, but we use only three which are given below:

- **G4PVPlacement** in which 1 Placement = One Volume
- **G4PVParameterised** in which 1 Parameterized = many volumes
    - Which are implemented through the concrete class **G4VPVParameterisation**.
- **G4PVReplica** in which 1 Replica = many Volumes
    - Mother volumes is filled by the daughter volumes of the same shape

4.6.8 Magnetic Field:

For the magnetic field we use the class:

- Uniform field:
    - Use object of the **G4UniformMagField** class

G4MagneticField*magField=new G4UniformMagField (G4ThreeVector(1.*Tesla, 0,0));

CHAPTER # 5

Application Development in Geant4:

5.1 Geant4 setup is studying the Proton beam passing through the varied materials:

In this simulation code, we described the Calorimeter of which **detector construction**, **beam generating**, **energy of beam** and **material and shape of Calorimeter** and **the number of layers** which is our target.

In our simulation, we choose the **"Proton Beam"** and **"15 MeV"** energy and the particle like **"Proton"**.

In our simulation code there are seven **"src"** files with **".cc"** data type and eight **"include"** files with **".hh"** header files. We used the **"Basic Example B4a"**.

Now, we will explain the **"src"** files briefly.

5.1.1 B4DetectorConstrction.cc:

a) In this class, we will explain the **detector geometries**, **materials**, **placements of solids** and **visual attributes for the solids** etc. We will just discuss the functions/commands to do so. We can define the material as the given below:

```
G4NistManager* nist = G4NistManager::Instance();
G4Material* env_mat = nist->FindOrBuildMaterial("G4_WATER");
```

We used the **G4Box** constructor which is derived from the **G4VSolid class** to design the Calorimeter. The constructor of this solid is:

It's for the **World volume**:

```
G4VSolid* worldS = new G4Box("World",           // its name
                 worldSizeXY/2, worldSizeXY/2, worldSizeZ/2); // its size
```

It's for the **Calorimeter volume**:

```
G4VSolid* calorimeterS  = new G4Box("Calorimeter",    // its name
        calorSizeXY/2, calorSizeXY/2, calorThickness/2); // its size
```

After this one more constructor is **G4PVPlacement** is defined as:

```
G4LogicalVolume* worldLV= new G4LogicalVolume(
                 worldS,           // its solid
```

```
                wor_mat,  // its material
                "World");         // its name
  G4VPhysicalVolume* worldPV= new G4PVPlacement(
                0,                // no rotation
                G4ThreeVector(),  // at (0,0,0)
                worldLV,          // its logical volume
                "World",          // its name
                0,                // its mother  volume
                false,            // no boolean operation
                0,                // copy number
                fCheckOverlaps);  // checking overlaps
```

After this one more constructor is **G4PVReplica** is defined as:

```
G4LogicalVolume* layerLV= new G4LogicalVolume(
                layerS,           // its solid
                lar_mat,  // its material
                "Layer");         // its name


  new G4PVReplica(
                "Layer",          // its name
                layerLV,          // its logical volume
                calorLV,          // its mother
                kZAxis,           // axis of replication
                nofLayers,        // number of replica
                layerThickness);  // witdth of replica
```

After this one more is **Visualization Attributes** is defined as:

```
G4VisAttributes*VisAtt=new G4VisAttributes( G4Colour::Red());
 G4VisAttributes*VisAtt1=new G4VisAttributes( G4Colour::Yellow ());
 G4VisAttributes*VisAtt2=new G4VisAttributes( G4Colour::White());
 calorLV->SetVisAttributes(VisAtt);
 worldLV->SetVisAttributes(VisAtt1);
```

After this one more is **Magnetic field** is defined as:

```
G4FieldManager* fieldManager
   = G4TransportationManager::GetTransportationManager()->GetFieldManager();
  // Delete the existing magnetic field
  if ( fMagField )  delete fMagField;
  if ( fieldValue != 0. ) {
   // create a new one if not null
   fMagField
    = new G4UniformMagField(G4ThreeVector(fieldValue, 0., 0.));
   fieldManager->SetDetectorField(fMagField);
   fieldManager->CreateChordFinder(fMagField);
  }
  else {
   fMagField = 0;
   fieldManager->SetDetectorField(fMagField);
  }
```

5.1.2 B4aEventAction.cc:

In this class two Virtual Void functions are implemented ***BeginOfEventAction(const G4Event* evt)*** and **EndOfEventAction(const G4Event*evt).** Here we will discuss the second functions. In this function, we will fill the Histograms like below:

```
G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();
  // fill histograms
  analysisManager->FillH1(1, fEnergyAbs);
  analysisManager->FillH1(2, fEnergyGap);
  analysisManager->FillH1(3, fTrackLAbs);
```

```
  analysisManager->FillH1(4, fTrackLGap);
  // fill ntuple
  analysisManager->FillNtupleDColumn(0, fEnergyAbs);
  analysisManager->FillNtupleDColumn(1, fEnergyGap);
  analysisManager->FillNtupleDColumn(2, fTrackLAbs);
  analysisManager->FillNtupleDColumn(3, fTrackLGap);
  analysisManager->AddNtupleRow();
```

And then print per event of this quantity:

```
G4int eventID = evt->GetEventID();
  if ( eventID % fPrintModulo == 0) {
    G4cout << "---> End of event: " << eventID << G4endl;
    G4cout
      << "   Absorber: total energy: " << std::setw(7)
                                << G4BestUnit(fEnergyAbs,"Energy")
      << "       total track length: " << std::setw(7)
                                << G4BestUnit(fTrackLAbs,"Length")
      << G4endl
      << "       Gap: total energy: " << std::setw(7)
                                << G4BestUnit(fEnergyGap,"Energy")
      << "       total track length: " << std::setw(7)
                                << G4BestUnit(fTrackLGap,"Length")
      << G4endl;
  }
```

Here, in the **EndOfEventAction(const G4Event*evt),** we can also find the momentum and position of the particle like below:

```
//Momentum
```

```
G4ThreeVector momentum=B4aSteppingAction::Instance()->GetMomentum();
fmomentum +=momentum;
//G4cout<<"Momentum"<<momentum<<G4endl;
   G4cout << " Momentum " <<  momentum/keV << " [keV] " <<G4endl;
   //Position
G4ThreeVector position=B4aSteppingAction::Instance()->GetPosition();
fposition +=position;
// G4cout<<"Position"<<position<<G4endl;
G4cout << " Position " <<  position/cm <<  " [cm] " <<G4endl;
```

5.1.3 B4PrimaryGeneratorAction.cc:

This class is very important which is used for the initialization of particles by using the particle gun and set the particle and its energy, and position of the particle gun, like below:

```
G4int nofParticles = 1;
 fParticleGun = new G4ParticleGun(nofParticles);
 // default particle kinematic
 G4ParticleDefinition* particleDefinition
= G4ParticleTable::GetParticleTable()->FindParticle("e+");
 fParticleGun->SetParticleDefinition(particleDefinition);
 fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
 fParticleGun->SetParticleEnergy(15.*MeV);
```

Now here set the position of the particle gun on the world volume:

```
 G4double worldZHalfLength = 0;
 G4LogicalVolume* worlLV
   = G4LogicalVolumeStore::GetInstance()->GetVolume("World");
 G4Box* worldBox = 0;
```

```
if ( worlLV) worldBox = dynamic_cast< G4Box*>(worlLV->GetSolid());
if ( worldBox ) {
  worldZHalfLength = worldBox->GetZHalfLength();
}
else  {
  G4cerr << "World volume of box not found." << G4endl;
  G4cerr << "Perhaps you have changed geometry." << G4endl;
  G4cerr << "The gun will be place in the center." << G4endl;
}
// Set gun position
fParticleGun
  ->SetParticlePosition(G4ThreeVector(0., 0., -worldZHalfLength));
fParticleGun->GeneratePrimaryVertex(anEvent);
```

5.1.4 B4RunAction.cc:

In this class, two void virtual functions are implemented, which are:

BeginOfRunAction(const G4Run* run) & EndOfRunAction(const G4Run* aRun)

In the ***BeginOfRunAction(const G4Run* run),*** we will create the Histograms and ntuples like below:

```
G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();
  // Open an output file
  G4String fileName = "B4";
  analysisManager->OpenFile(fileName);
  analysisManager->SetFirstHistoId(1);
  // Creating histograms
  analysisManager->CreateH1("1","Edep in absorber", 100, 0., 800*MeV);
```

```
analysisManager->CreateH1("2","Edep in gap", 100, 0., 100*MeV);
analysisManager->CreateH1("3","trackL in absorber", 100, 0., 1*m);
analysisManager->CreateH1("4","trackL in gap", 100, 0., 50*cm);
// Creating ntuple
analysisManager->CreateNtuple("B4", "Edep and TrackL");
analysisManager->CreateNtupleDColumn("Eabs");
analysisManager->CreateNtupleDColumn("Egap");
analysisManager->CreateNtupleDColumn("Labs");
analysisManager->CreateNtupleDColumn("Lgap");
analysisManager->FinishNtuple();
```

In the ***EndOfRunAction(const G4Run* aRun),*** we will print the Histograms and ntuples like below:

```
G4int nofEvents = aRun->GetNumberOfEvent();
if ( nofEvents == 0 ) return;
// print histogram statistics
G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();
if ( analysisManager->GetH1(1) ) {
  G4cout << "\n ----> print histograms statistic \n" << G4endl;
  G4cout
    << " EAbs : mean = " << G4BestUnit(analysisManager->GetH1(1)->mean(), "Energy")
        << " rms = " << G4BestUnit(analysisManager->GetH1(1)->rms(),  "Energy")
        << G4endl;
  G4cout
    << " EGap : mean = " << G4BestUnit(analysisManager->GetH1(2)->mean(), "Energy")
        << " rms = " << G4BestUnit(analysisManager->GetH1(2)->rms(),  "Energy")
        << G4endl;
```

```
  G4cout
    << " LAbs : mean = " << G4BestUnit(analysisManager->GetH1(3)->mean(), "Length")
          << " rms = " << G4BestUnit(analysisManager->GetH1(3)->rms(),  "Length")
          << G4endl;
  G4cout
    << " LGap : mean = " << G4BestUnit(analysisManager->GetH1(4)->mean(), "Length")
          << " rms = " << G4BestUnit(analysisManager->GetH1(4)->rms(),  "Length")
          << G4endl;
}
 // save histograms
 analysisManager->Write();
 analysisManager->CloseFile();
 // complete cleanup
 delete G4AnalysisManager::Instance();
```

5.1.5 B4aSteppingAction.cc:

In this class, there is one main constructor, which is given below:

Here we can collect energy and track length step by step like below:

```
// get volume of the current step
  G4VPhysicalVolume* volume
    = step->GetPreStepPoint()->GetTouchableHandle()->GetVolume();
  // energy deposit
  G4double edep = step->GetTotalEnergyDeposit();
  // step length
  G4double stepLength = 0.;
  if ( step->GetTrack()->GetDefinition()->GetPDGCharge() != 0. ) {
```

```
  stepLength = step->GetStepLength();
 }
 if ( volume == fDetConstruction->GetAbsorberPV() ) {
  fEventAction->AddAbs(edep,stepLength);
 }
 if ( volume == fDetConstruction->GetGapPV() ) {
  fEventAction->AddGap(edep,stepLength);
 }
```

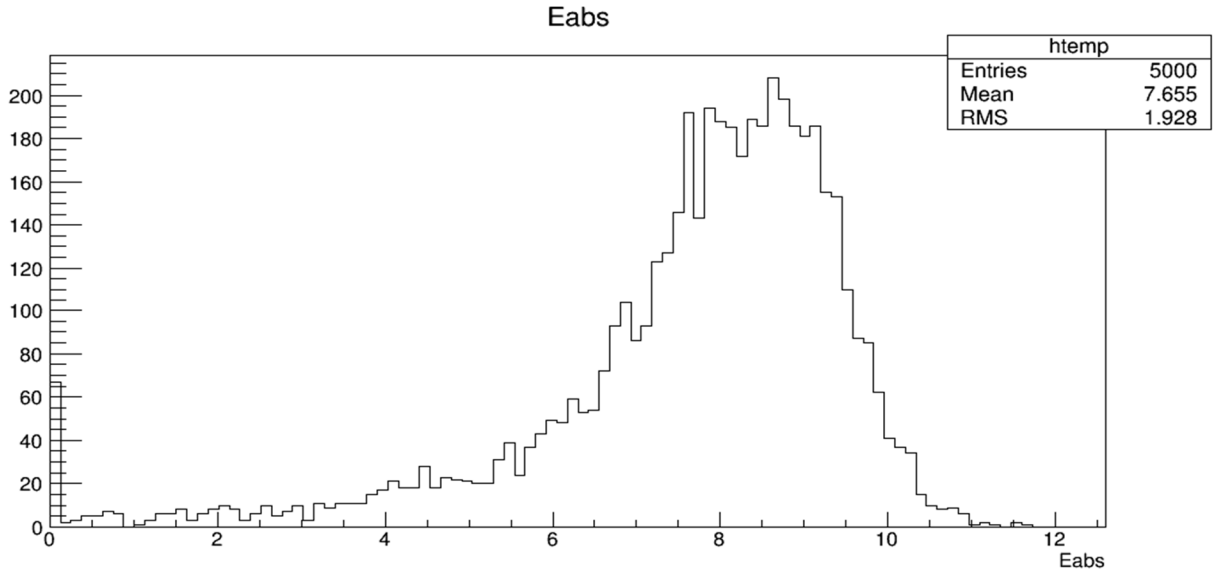
Here we will find the momentum and position of the particle by taking **GetTrack** from the **step** from the **UserSteppingAction**:

```
G4Track* track        = step->GetTrack();
    G4ThreeVector momentum = track->GetMomentum();
    fmomentum +=momentum;
        G4ThreeVector position = track->GetPosition();
    fposition +=position;
```

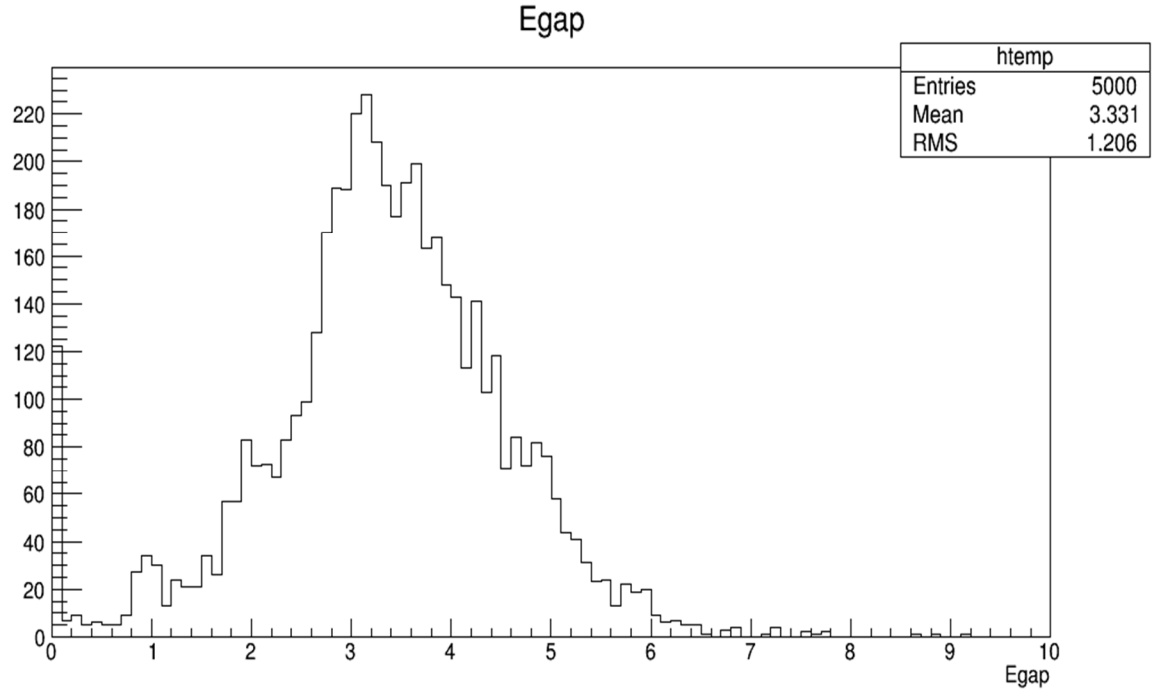
CHAPTER # 5 Results:

When the protons pass through the water material it losses the energy, the histograms it's related are given below:

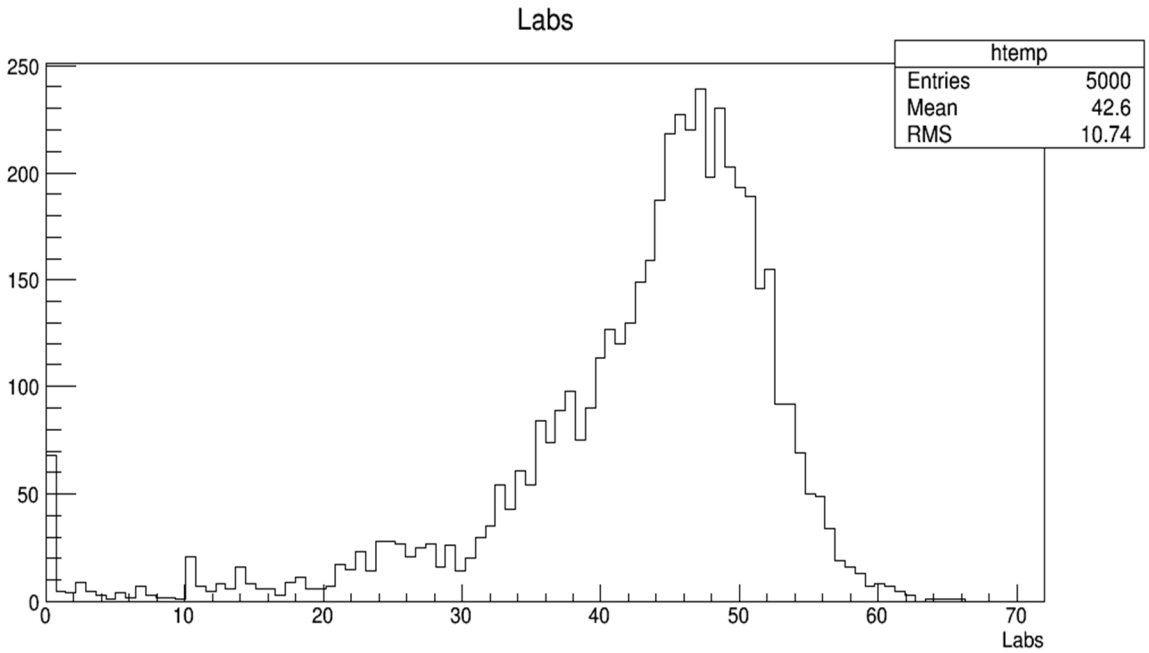
Histogram 1 is related to energy absorption through the water material:



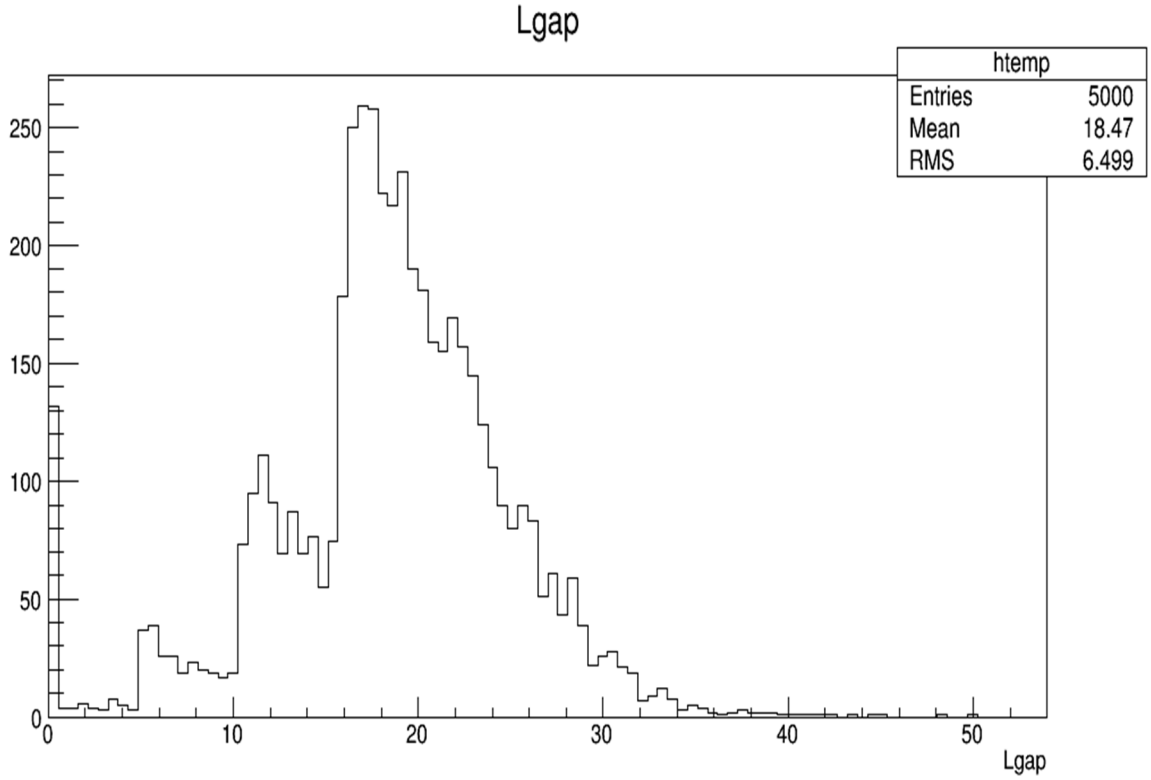
Histogram 2 is related to energy gap through the water material:



Histogram 3 is related to track absorption through the water material:



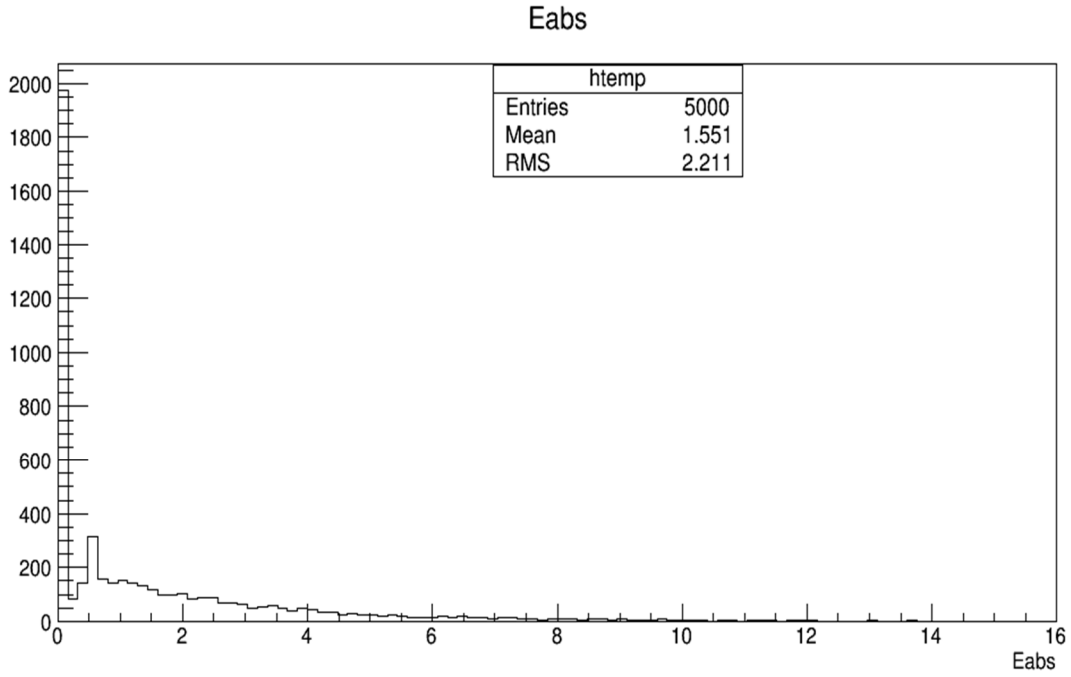
Histogram 4 is related to track gap through the water material:



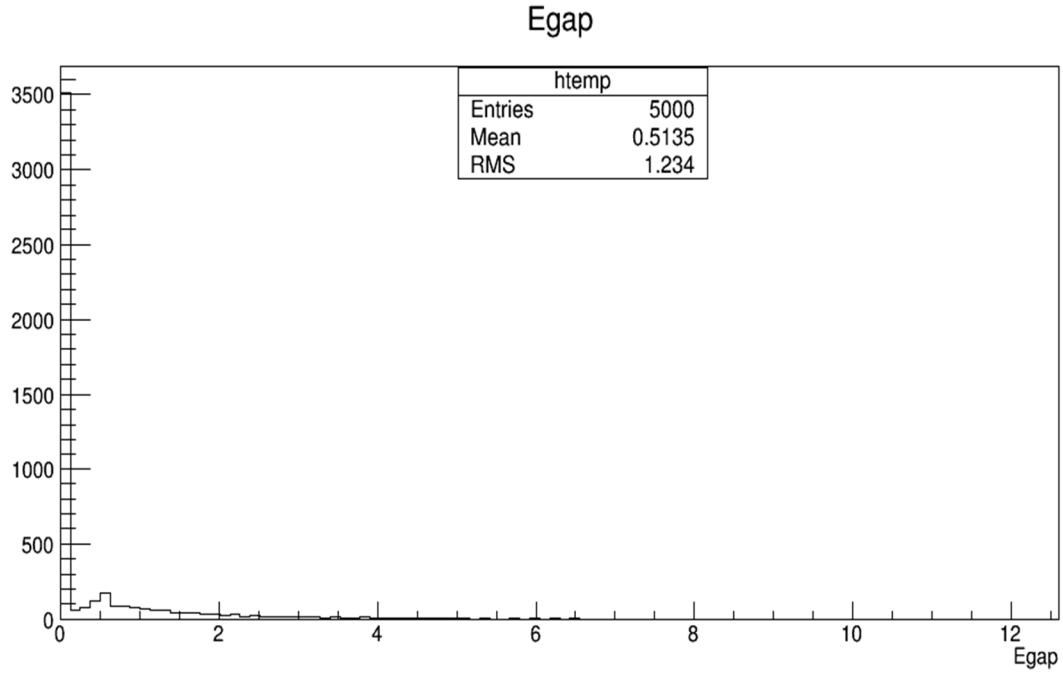
Results:

When the protons pass through the lead (Pb) material it losses the energy, the histograms it's related are given below:

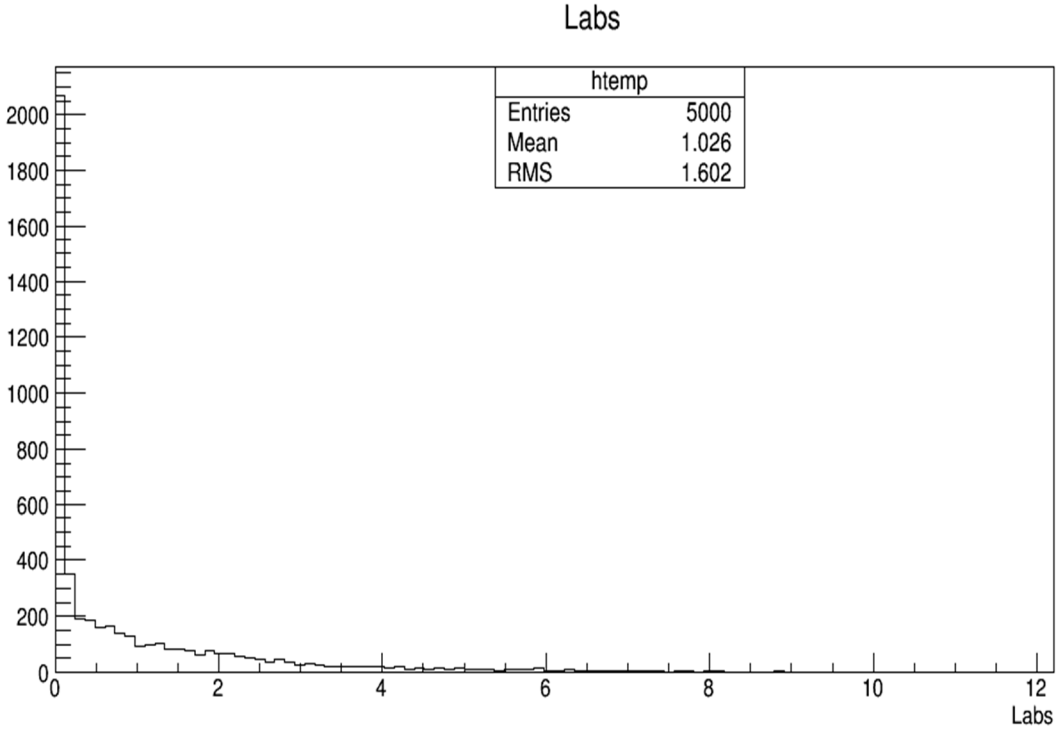
Histogram 1 is related to energy absorption through the lead material:



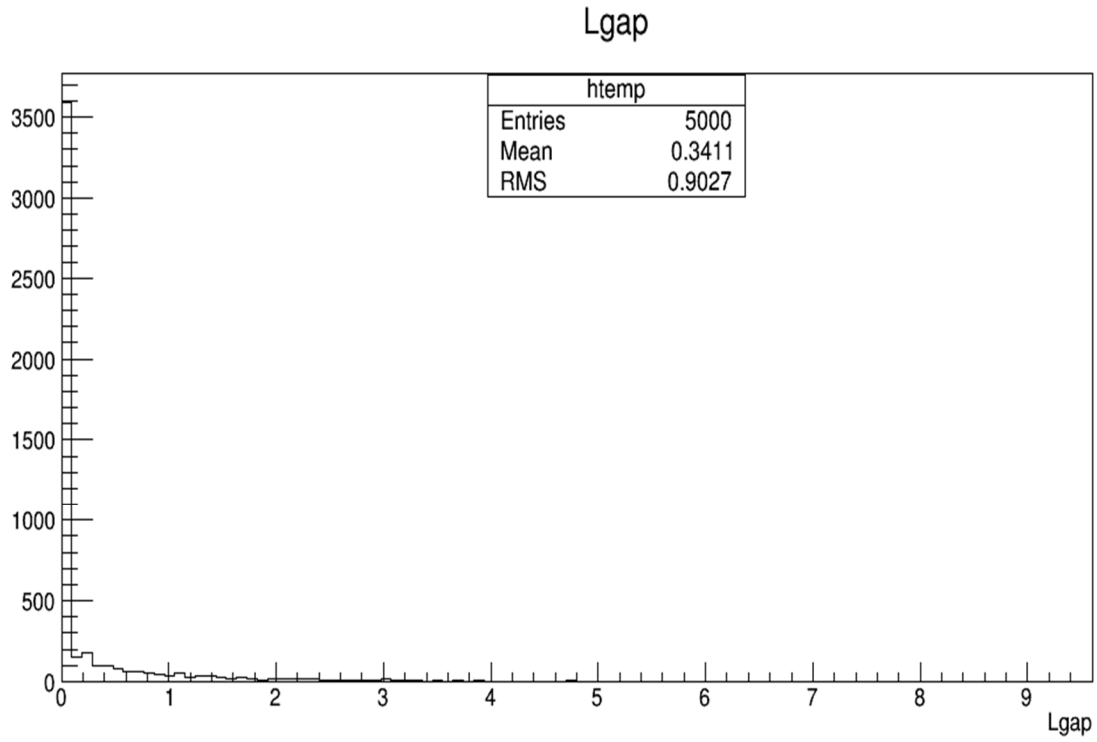
Histogram 2 is related to energy gap through the lead material:



Histogram 3 is related to track absorption through the lead material:



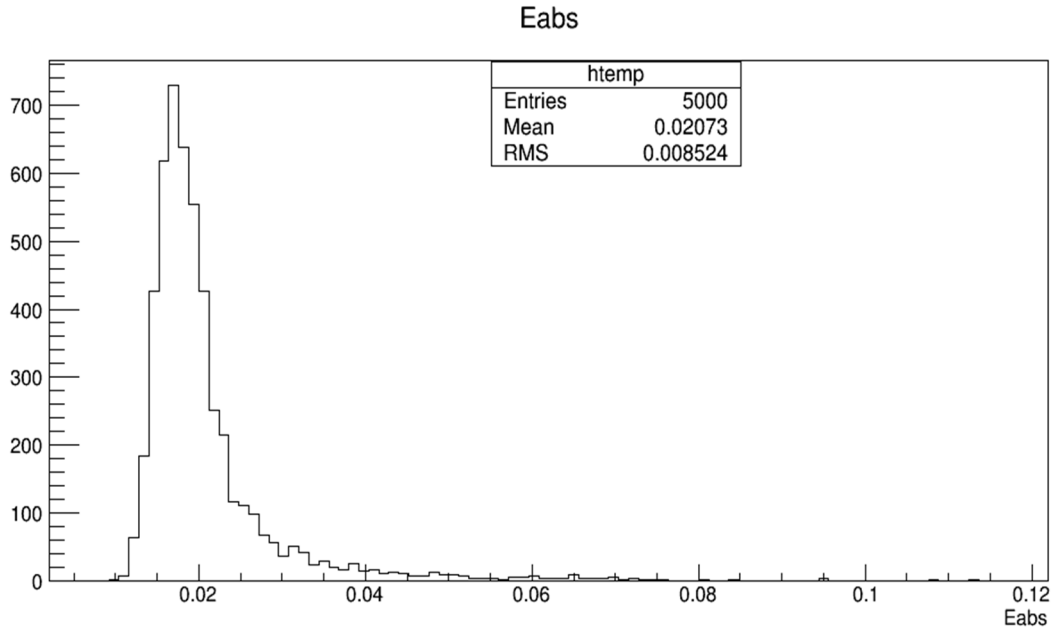
Histogram 4 is related to track gap through the lead material:



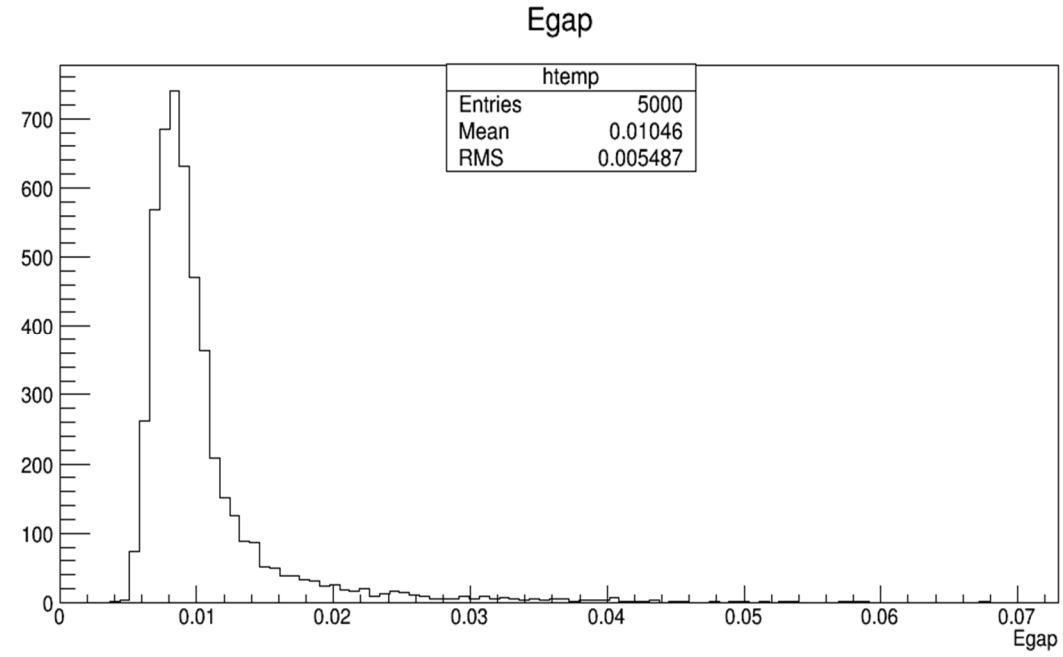
Results:

When the protons pass through the air material it losses the energy, the histograms it's related are given below:

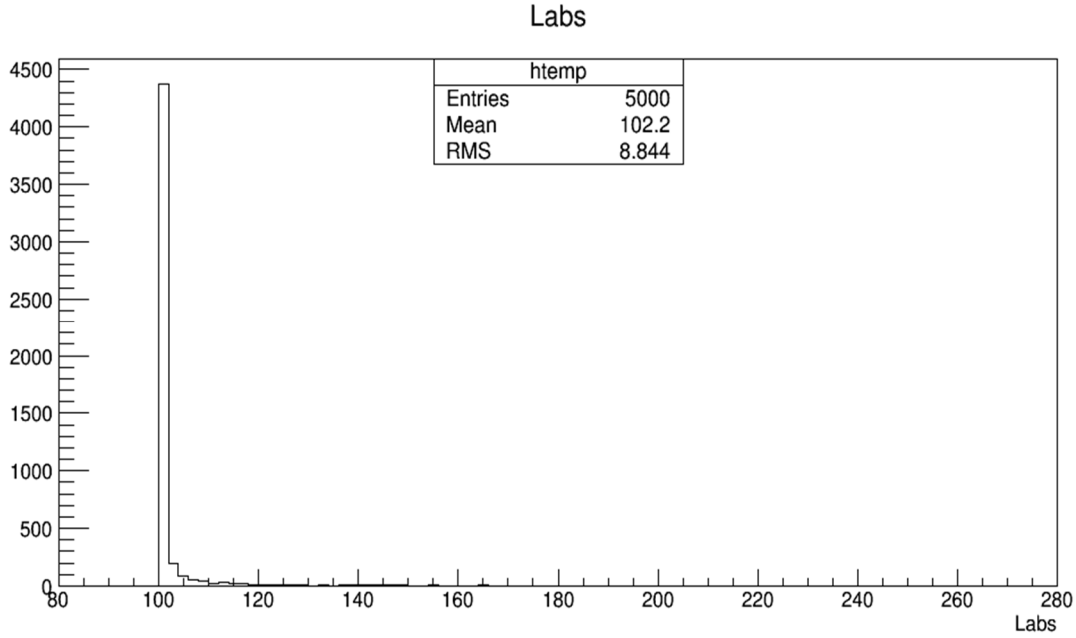
Histogram 1 is related to energy absorption through the air material:



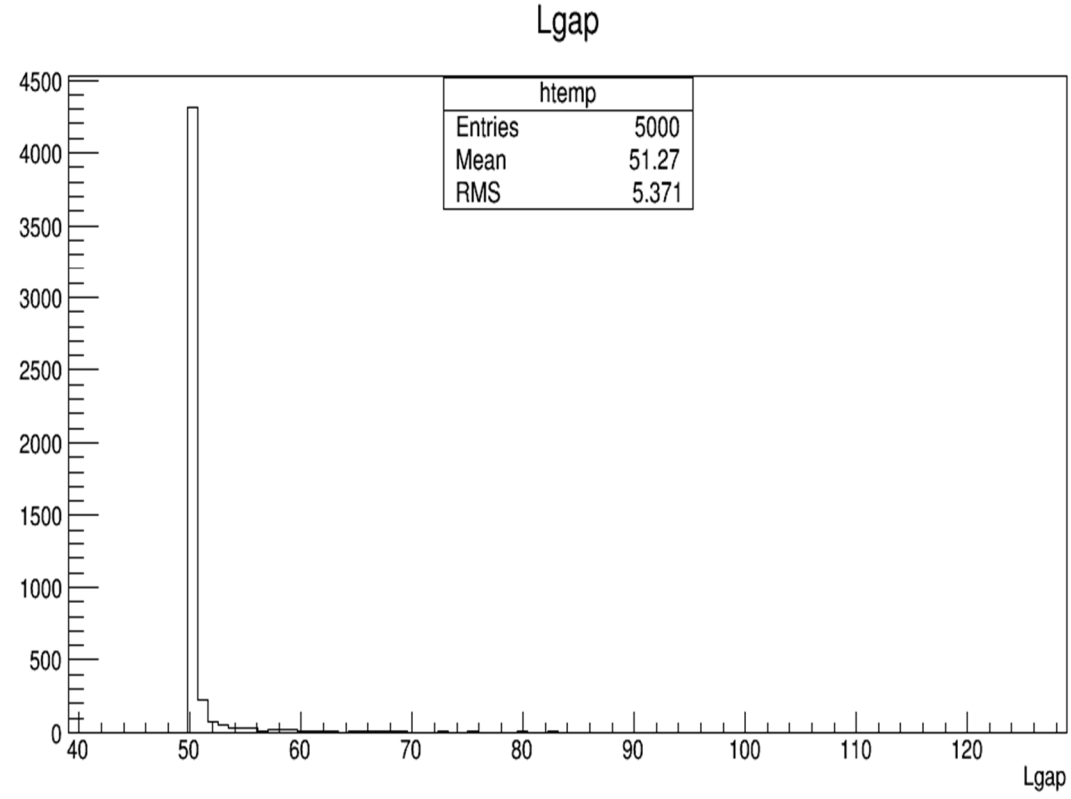
Histogram 2 is related to energy gap through the air material:



Histogram 3 is related to track absorption through the air material:

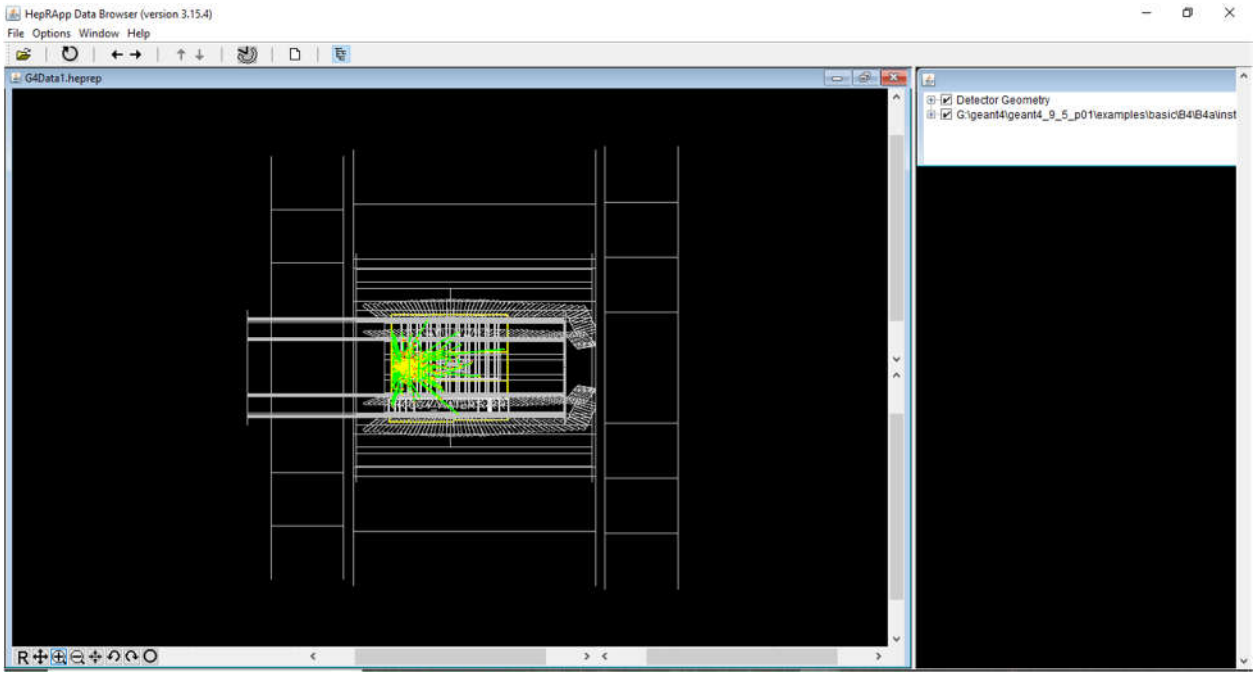


Histogram 4 is related to track gap through the air material:

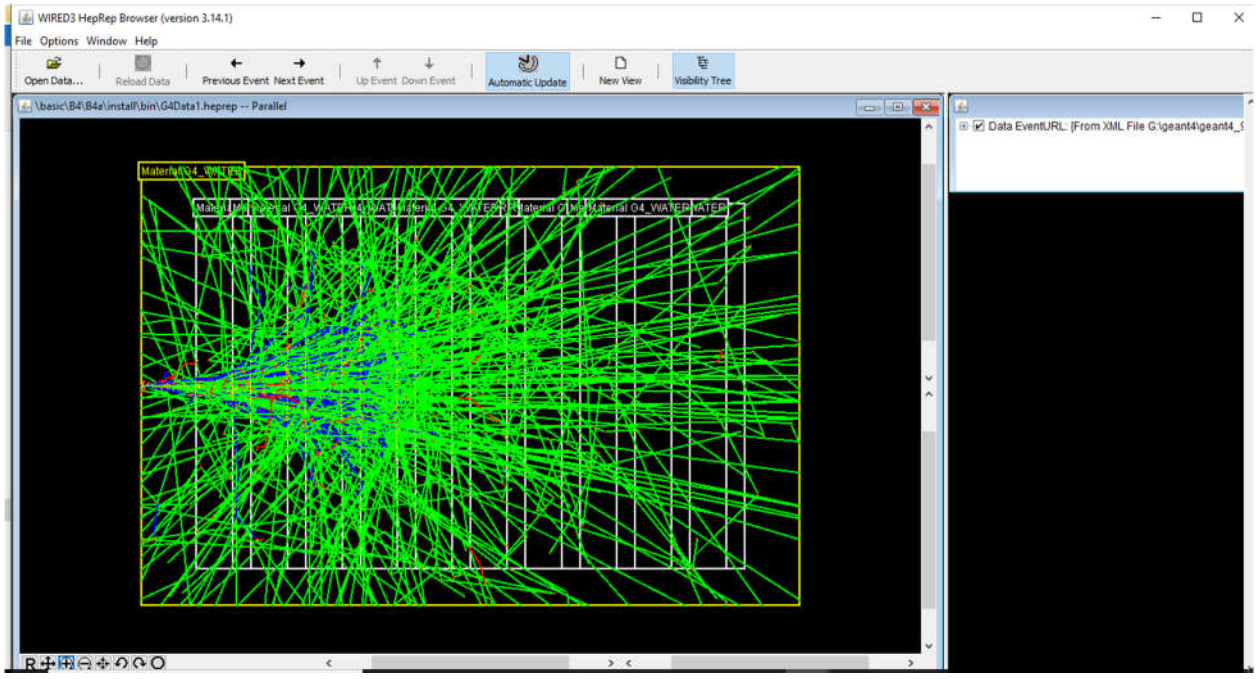


Detector view from the HepRApp Data Visualization Browser:

Babar Detector view from visualization Browser:

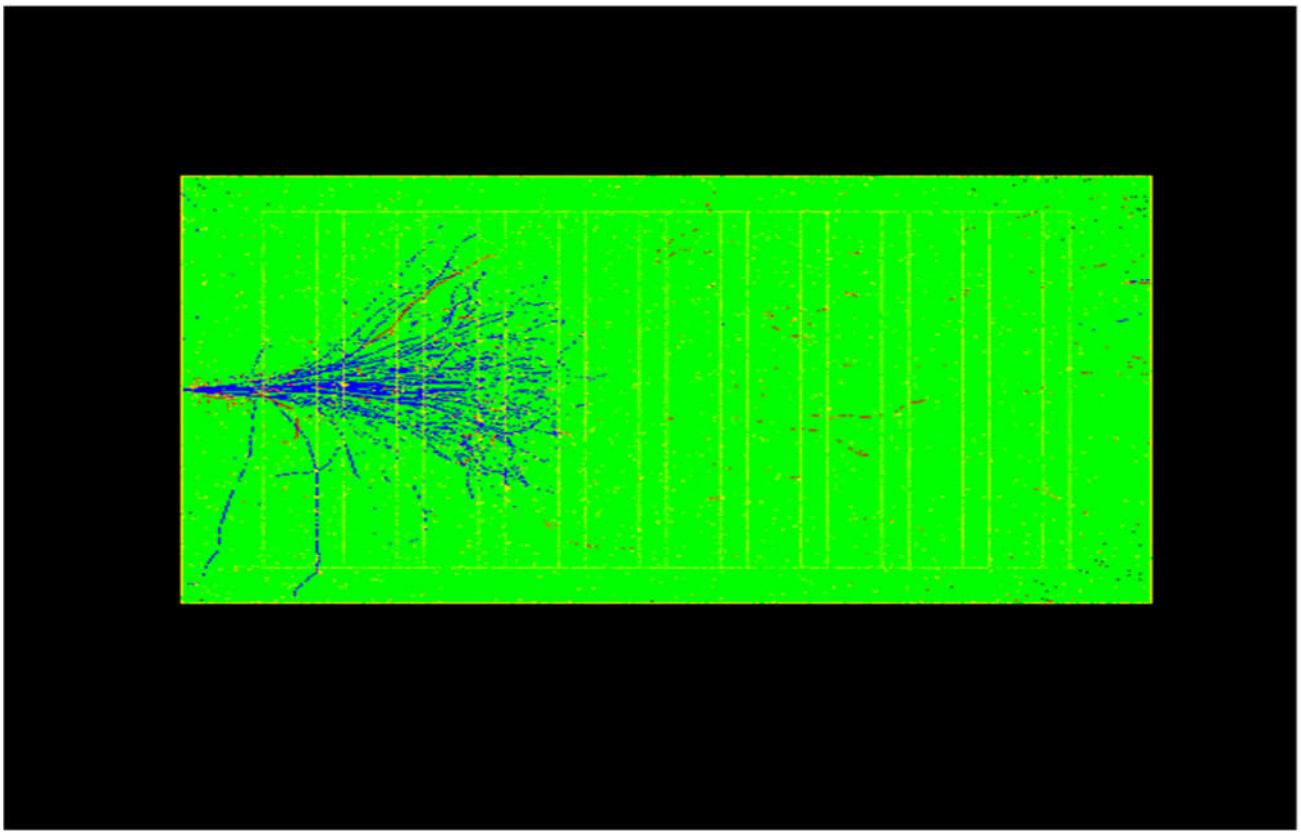


Detector view from the WIRED3 HepRep Visualization Browser:



Detector view from the RayTracer Visulization:

Detector view from the OpenGL Visulizsation:



Refrences

[1] Geant4 collaboration, Geant4 Installation Guide Version: Geant4.9.5, (2011).

[2] Geant4 collaboration, Geant4 User's Guide for Application Developer's Version: Geant4.9.5, (2011).

[3] Geant4 collaboration, Physics Reference Manual Version: Geant4.9.5, (2011).

[4] Geant4 Source Code Version: Geant4.9.5.p01:

[5] *https://www.jlab.org/12gev_phys/packages/sources/geant4/?C=D;O=A*

retrieved on 6 June, 2017.

[6] Cmake Version: 2.8.7:

[7] *https://cmake.org/files/v2.8*

retrieved on 6 June, 2017.

[8] *http://distfiles.macports.org/geant4/*

retrieved on 8 June, 2017.

[9]http://geant4.web.cern.ch/geant4/UserDocumentation/UserGuides/IntroductionToGeant4/html/index.html retrieved on 24 July, 2017.

[10]http://conferences.fnal.gov/g4tutorial/g4cd/Documentation/Visualization/G4WIREDTutorial/G4WIREDTutorial.html

[11] http://www.slac.stanford.edu/~perl/HepRApp/

[12]http://geant4.slac.stanford.edu/Presentations/vis/G4DAWNTutorial/G4DAWNTutorial.html