5-7-2007

Energy Deposition Study of Low-Energy Cosmic Radiation at Sea Level

Pushpa Indumathie Wijesinghe

ENERGY DEPOSITION STUDY OF LOW-ENERGY COSMIC RADIATION AT SEA LEVEL

by

PUSHPA WIJESINGHE

Under the direction of Xiaochun He

ABSTRACT

In this dissertation work, a computer simulation model based on the Geant4 simulation package has been designed and developed to study the energy deposition and track structures of cosmic muons and their secondary electrons in tissue-like materials. The particle interactions in a cubic water volume were first simulated. To analyze the energy deposition and tracks in small structures, with the intention of studying the energy localization in nanometric structures such as DNA, the chamber was sliced in three dimentions. Validation studies have been performed by comparing the results with experimental, theoretical, and other simulation results to test the accuracy of the simulation model. A human body phantom in sea-level muon environment was modeled to measure the yearly dose to a human from cosmic muons. The yearly dose in this phantom is about 22 millirems. This is close to the accepted value for the yearly dose from cosmic radiation at sea level. Shielding cosmic muons with a concrete slab from 0 to 2 meters increased the dose received by the body. This dissertation presents an extensive study on the interactions of secondary electrons created by muons in water.

INDEX WORDS: Radiation Dosimetry Simulation, Track Structures, Sea-Level muon Flux, Energy Deposition

ENERGY DEPOSITION STUDY OF LOW-ENERGY COSMIC RADIATION AT SEA LEVEL

by

PUSHPA WIJESINGHE

A Dissertation Submitted in Partial Fullfilment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2007

ENERGY DEPOSITION STUDY OF LOW-ENERGY COSMIC RADIATION AT SEA LEVEL

by

PUSHPA WIJESINGHE

Major Professor:	Dr. Xiaochun He
Committee:	Dr. Douglas Gies
	Dr. Steven Manson
	Dr. William Nelson
	Dr. Unil Perera

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
May 2007

to my parents and the family, for their guidance, support, love and enthusiasm

Acknowledgments

A journey is easier when you travel together. Interdependence is certainly more valuable than independence. This thesis is the result of six years of work whereby I have been accompanied and supported by many people. It is a pleasure that I now have the opportunity to express my gratitude to all of them. The first person I would like to thank is my advisor, Dr. Xiaochun He. I have been in his project since 2003 when I finished my Master's degree at Georgia State University. His overly enthusiastic and integral view on research and his mission for providing "only high-quality work and not less," has made a deep impression on me. I owe him lots of gratitude for having shown me this way of research. Besides being an excellent advisor, he was as close as a relative and a good friend to me. I am really glad that I have come to get know him in my life. The chain of my gratitude would be definitely incomplete if I forgot to thank the first cause of this chain, Dr. Unil Perera. Using Aristotle's words he was my "Prime Mover" and I give him my deepest and sincere gratitude for recommending me to the graduate program at GSU and for serving as a member of my committee. I would like to express my sincere gratitude to the other members of my dissertation committee who kept an eye on the progress of my work, shared their wealth of knowledge and experience, and always were available when I needed their advice: Dr. William Nelson, Dr. Steven Manson, and Dr. Douglas Gies. I thank you all. I would also like to thank Ms. Carola Butler, Dr. Jun Ying and Ms. Kathryn Mudd who dedicated their valuble time to help in technical problems and who took effort in reading and providing me with valuable comments on earlier versions of this thesis. I would like to express my sincere gratitude to Dr. John Miller, Dr. Leslie Braby, Dr. Noelle Metting, Dr. Mohamed Rinzan,

Dr. Robert Dubois, Dr. Robert Stewarts, and Dr. Susanna Guatelli for valuable comments and instructions given through personal communications. I had the pleasure to work with a very united group of friends who did their graduate work on our project and have been beneficial for the work presented in this thesis. I thank them, too. I feel a deep sense of gratitude for my dear father and mother who formed part of my vision and taught me the good things that really matter in life. The happy memory of my childhood with them still provides a persistent inspiration for my journey in this life. I am glad to be your daughter. Also, I am very grateful for my husband, Janaka, for his encouragement, love, and patience during this long process. One of the best experiences that we lived through in this period was the birth of our son, Thilina, who provided an additional joyful dimension to our life mission. Finally, I dedicate this dissertation to my parents and family for their guidance, support, love, and enthusiasm.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Aim of the Dissertation

Secondary cosmic ray particles are constantly present at the surface of the earth. On average, one secondary particle hits per square centimeter per minute along the vertical direction. These particles have a wide range of energies from a few hundred MeV up to several hundred GeV or more. Most of the particles reaching sea level are highly penetrating muons with an average energy of 4 GeV.[1] These muons are about 200 times heavier than electrons, and they carry a great amount of mass energy in them. One of the important applications of studying cosmic ray particles is to assess the radiobiological effects of this low-dose radiation. In particular, low-energy muons are stopped in large bio systems (for example, in the human body) and very energetic electrons (in several tens of the MeV range) generated from these muons will interact with molecules leading to trails of ionization. When assessing and predicting the biological consequences of cosmic muons , it is essential to understand the extent and the nature of damage that penetrating muons cause to radiation-

sensitive structures of the human body. The radiation dose from cosmic rays received by the human body cannot be measured directly, and experimental and theoretical details of the consequences of low-energy radiation are mostly unknown.[2] Therefore, the only achievable way to learn about the radiation track structures and energy deposition is by simulation. This also infers the properties of a complex problem involving many aspects before actually experimenting with it, which helps to maximize the advantages and minimize the costs and other disadvantages when the problem occurs in the real world. In this work, a computer simulation program based on the Geant4 Simulation Toolkit has been developed to study the energy loss and the track structures of muons and electrons in liquid water, which constitutes more than 80 percent of the soft tissues of the human body.[3]

Although event-by-event Monte Carlo simulations are widely used, these simulations are not only computationally extensive but also are limited with cross sections only available for water .[4–7] A serious limitation of many available simulation systems is the difficulty of adding new or variant physics models. Therefore, the development becomes difficult due to the increasing size, complexity, and interdependency of the procedure-based code.[8] Geant4 was used in this work because it enables researchers to study the track structures and energy deposition in any defined material. In addition, it is not as computationally demanding as other Monte Carlo simulations.

1.2 Ionizing Radiation and Radiation Dose

There are many uses of radiation in medicine, research, and industry such as therapy, radioactive tracing, food irradiation, radiography, process control, sterilizing, and

detecting the age of ancient organic materials. Ionizing radiation is sometimes harmful for life. Therefore the use of radiation should be studied thoroughly by research to analyze its physical, chemical and biological effects. Just after the discovery of x-rays, scientists began to notice the harmful effects of exposure to it. At that time, people did not realize how dangerous x-rays and other forms of radiation can be. Several pioneer radiologists suffered severe injuries and some even died as a result of prolonged exposure to high intensities of x-rays. These early workers in the field had no means of accurately measuring the harm caused by radiation, and they depended on unreliable effects such as the degree of skin reddening caused by exposure, or on timing the exposure from a certain type of x-ray machine to a establish quantity.[9] In order to evaluate the effects of radiation it is necessary to have a measure of the dose. The earlier units of radiation were the *rad* and the *rem*. The exposure or the ability to ionize air is measured by a unit called *roentgen*(R), which is the amount of radiation that will be produced by ionizing one electrostatic unit of charge in 1 cubic centimeter of dry air. The SI unit of exposure is the coulomb per kilogram $C\ kg^{-1}$. However, the concept of exposure is rarely used now.

The absorbed dose is the energy deposited per unit mass of the medium. The SI unit for an absorbed dose is the gray (Gy). It represents an energy absorption of 1 joule per kilogram of the absorbing material ($1Gy = J, kg^{-1}$). However, the same absorbed dose delivered by different types of radiation can result in different degrees of biological damage to body tissues. The total energy deposited is not the only factor that determines the extent of the damage. The biological effect depends on the type of radiation absorbed, which is measured by the equivalent dose. The equivalent dose is a measure of the risk associated

with an exposure to ionizing radiation . The unit of equivalent dose is the sievert (sv).

$$H = D \times W_R \tag{1.1}$$

where H is the equivalent dose (sv), W_R is the radiation weighting factor, and D is the absorbed dose (Gy). The radiation weighting factor is a number that depends on the way the energy of the radiation is distributed along its path through the tissue. The radiation weighting factors for different radiation are given in Table 1.1. The concept of the effective dose is used to express the risk from the exposure of a single organ or tissue in terms of the equivalent risk from the exposure of the whole body, when the radiation exposure to various organs is different. The effective dose is calculated from Equation (1.2). Table 1.2 gives the tissue weighting factors to calculate the effective dose.[10]

$$E = W_T \times H_T \tag{1.2}$$

In Equation (1.2), W_T is the tissue weighting factor, H_T is the equivalent dose in the tissue T, and E is the effective whole body dose.

Table 1.1: Radiation Weighting Factors for Different Kinds of Radiation.

Type	Energy range	Radiation weighting factor, W_R
photons	all	1
electrons/muons	all	1
neutrons	$< 10\ keV$	5
	$10\ keV\ to\ 100\ keV$	10
	$100\ keV\ to\ 2\ MeV$	20
	$2\ MeV\ to\ 20\ MeV$	10
	$> 20\ MeV$	5
protons (not recoil)	$> 2\ MeV$	5
alpha particles	all	20
fission segments	all	20
heavy nuclei	all	20

Table 1.2: Tissue Weighting Factors Used for Calculating the Effective Dose

Organ	Tissue weighting factor, W_T
Gonads	0.20
Red bone marrow	0.12
Colon	0.12
Lung	0.12
Stomach	0.12
Bladder	0.05
Breast	0.05
Liver	0.05
Esophagus	0.05
Thyroid	0.05
Skin	0.01
Bone surfaces	0.01
Remainder	0.05

1.3 Background Radiation

About 80 percent of human exposure to radiation comes from natural sources, while the rest of the radiation originates from manmade sources such as medical X-rays. Figure 1.1 shows the percentages of background radiation coming from different sources.[11]

1.3.1 Cosmic Ray Radiation

Cosmic radiation is composed of stable charged particles and nuclei incident at the top of the terrestrial atmosphere. Cosmic rays were discovered by Victor Hess in 1912. The particles that are accelerated by astrophysical sources are called primary cosmic rays. These particles include protons, electrons and helium, carbon, oxygen, iron and other nuclei synthesized in stars. When those primaries interact with earth's atmosphere, the secondaries are generated.

The most-available secondary particle at sea level is the muon. Muons are produced in the upper atmosphere (typically 15 km above sea level), and they lose about 2 GeV of their energy by ionizations before they reach the ground. Figure 1.2 illustrates the creation process of muons in the atmosphere.[13] Primary cosmic particles (e.g., protons), originating from astrophysical sources such as supernovas, quasars, and black holes, produce pion showers from their encounter with the upper atmosphere nuclei. Pions, which are very short-lived, quickly decay into muons, which continue to move down through the atmosphere. The end products of muon decay include electrons, positrons, and neutrinos. Some muons decay in flight on their way to the ground, while others end up penetrating deep into the earth. The mean energy of ground-level muons is 4 GeV. The overall angular

distribution of muons at the ground is $\propto cos^2\alpha$, where alpha is the polar angle relative to the line from the ground to the zenith. Figure 1.3 shows the muon energy spectrum at sea level.[1]

Cosmic radiation offers both advantages and disadvantages for day-to-day human activities. Cosmic rays can be used to predict potentially damaging solar activity. One example is the massive power blackout in Quebec Hydro company in 1989, which occurred during a period of very intense solar and magnetic activity observed by cosmic ray monitors. Another example of the advantages of cosmic rays is carbon dating. Carbon dating is used to determine the age of carbon-based materials such as plants and animals. Radioactive C_{14}, which help finding the age, is produced from collisions between cosmic rays and carbon atoms. Recent research shows that cosmic muons can even be used to detect terrorist attempts to smuggle uranium or plutonium into a country because of their strong penetrating power.

On the other hand, these particles cannot only damage the human body, but they also can damage the memory cells of computers. They also interfere with the images of the sky made by astronomers and leave spots in them. Muons may also alter the cloud cover and disturb radio communication and navigation. Another possibility is that these particle can be used to prevent pipeline corrosion. Cosmic rays are the cause for aurora australis,–"Southern Lights," which occur near the South Pole, and Aurora borealis,–"Northern Lights," which occur near the North Pole.[14],[15],[16],[17]

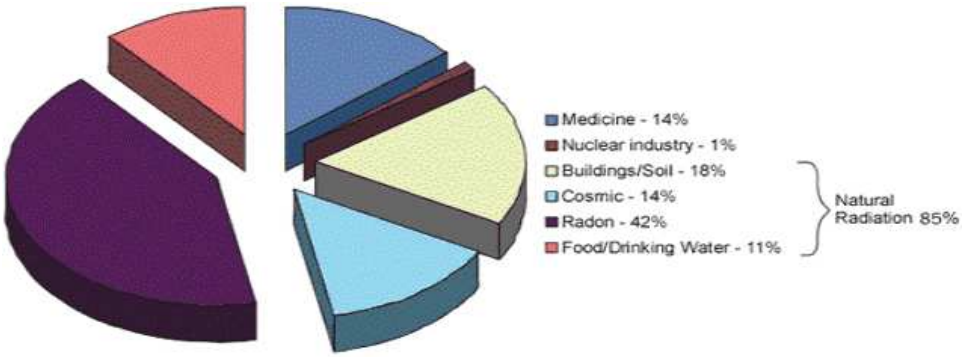

Figure 1.1: Amount of Human Exposure to Background Radiation.

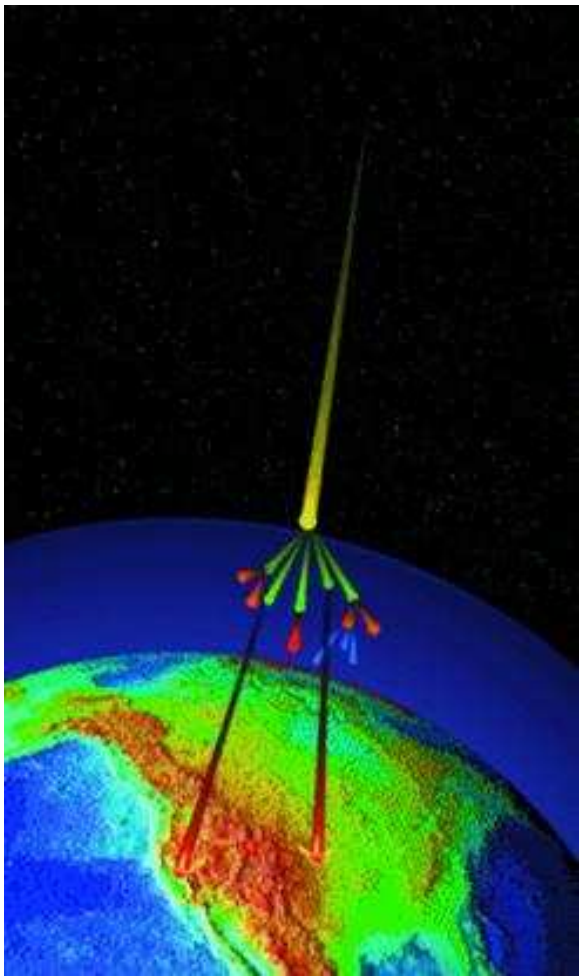

Figure 1.2: Cosmic Ray Muon Creation : As incoming proton (yellow) generates pions (green) which decay into muons (red). Some of the muons decay in flight on their way to the ground and create electrons (blue). Other muons with energies more than 2 GeV reach the ground.

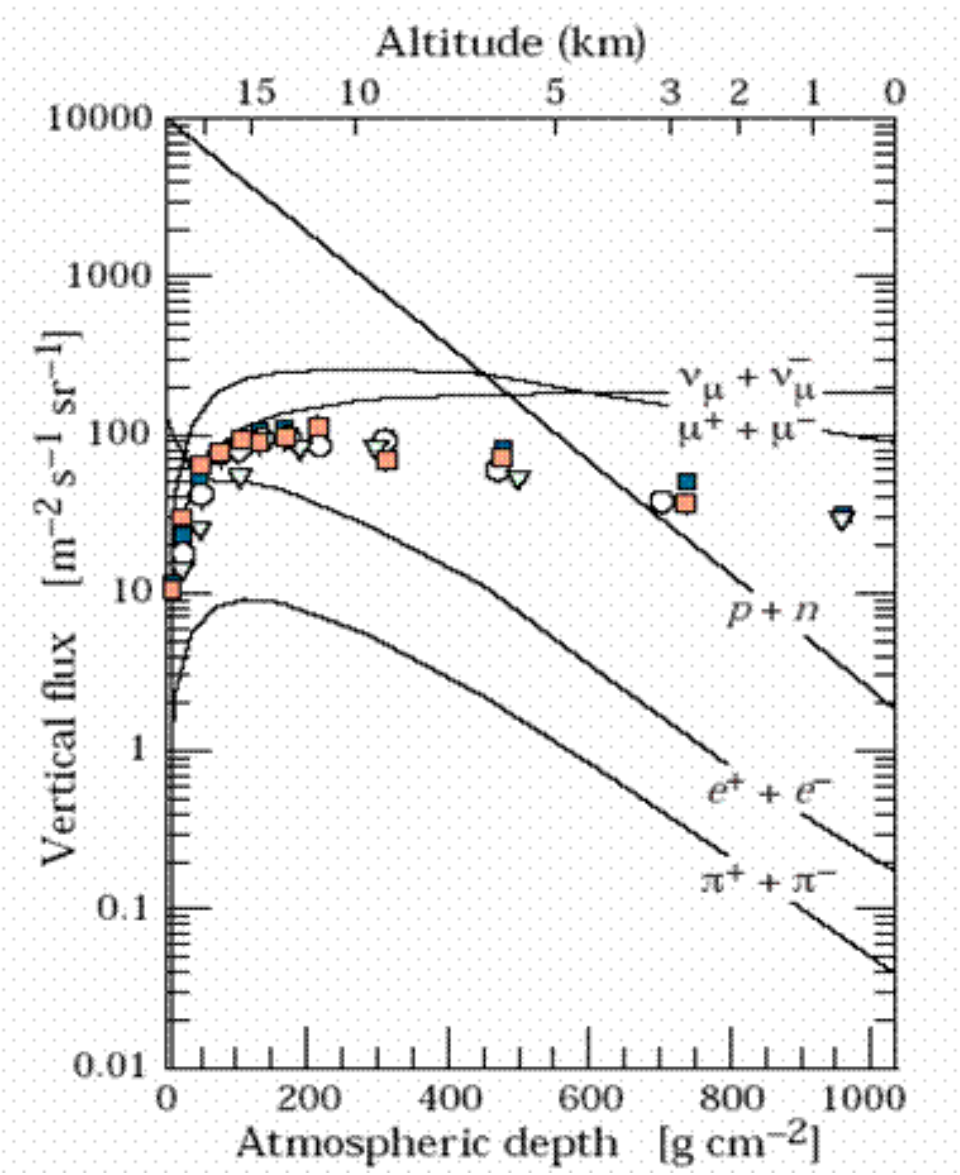

Figure 1.3: Vertical Fluxes of Cosmic Rays in the Atmosphere with E greater than 1 GeV. The points show measurements of negative muons with E greater than 1 GeV.

1.4 Passage of Electrons and Muons Through Water

In order to analyze the radiation effects of muons, it is necessary to understand their track structures and the energy deposition by them in biological matter. The basic concepts required to estimate and model track structures and energy deposition are discussed in the rest of this chapter.

1.4.1 Linear Energy Transfer and Stopping Power

Stopping power is the energy loss of a particle per unit path length in a particular medium. It is specified by the equation $-dE/dx$, where $-dE$ is the energy loss and dx is the increment of the path length. The spatial distribution of energy deposition in the particle track is described by Linear Energy Transfer (LET), or the amount of energy actually deposited per unit length along its path. Although LET and the stopping power are approximately equal for heavily charged particles, they are unequal for electrons.

1.4.2 Gamma Ray Interactions in Water

Interaction processes of gamma rays are very important when assessing biological effects as they impart energy to ionizing electrons. A gamma ray can interact inelastically with a molecule in three ways:

1. Through a photoelectric process where the photon is completely absorbed and an electron is ejected. This process dominates in low-energy photons ($\approx 40\ keV$), especially in materials with small atomic numbers.[18]

2. Through a Compton scattering process where part of the photon's energy is trans-

ferred to an electron. The gamma ray is scattered by an atom and travels in another direction, while an electron is emitted by the parent atom. The scattered gamma ray has lower energy than its initial energy. This is also very important for materials having low atomic numbers (such as water), especially for photon energies in the range of $0.04\ MeV - 10\ MeV$, and

3. When there is a strong nucleus, the gamma ray is transformed into an electron-positron pair. This normally occurs for gamma rays with energies above $1.02\ MeV$.

Figure 1.4 gives the interaction probabilities for different types of gamma ray interactions in water. [19]

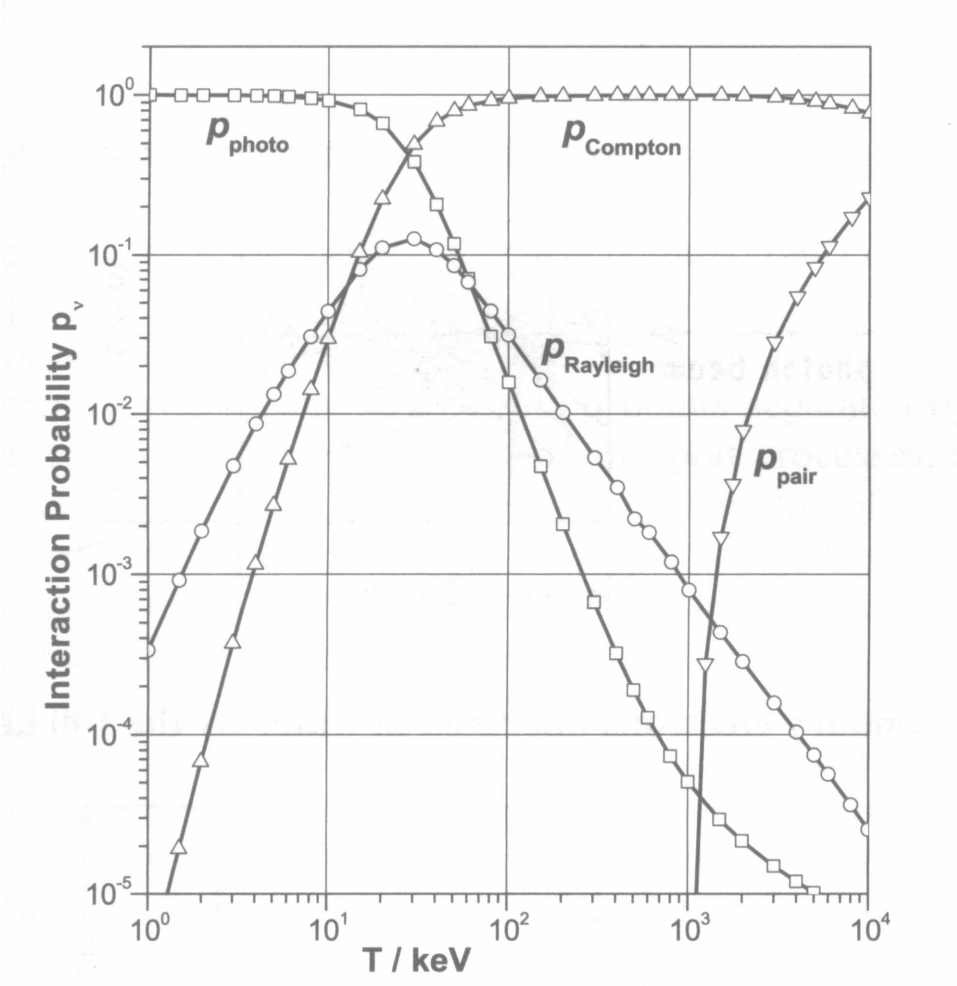

Figure 1.4: The Interaction Probabilities for Different Types of Gamma Ray Interactions in Water.[19] Photo electric effect dominates for the gamma rays in low energy region. Compton scattering is the dominant interaction process for moderate energy gamma rays while the pair production is frequent in higher energy gamma rays.

Gamma rays in a medium will scatter elastically by Rayleigh scattering and transfer some of their energy to secondary electrons. The gamma ray track structures and their energy deposition are characterized by the distances of inelastic collisions and the starting energies of created secondary electrons.

1.4.3 Electron Interactions in Water

Interaction of electrons in water is very important because almost all energies from gamma rays and heavy particles such as muons are finally transferred into secondary and higher generation electrons. Electrons transfer their energy to the medium mainly by the interaction of the electric field. This leads to electronic excitations and ionizations.[18] These electrons generate many low-energy secondary electrons until their energy falls below the ionization threshold. The reason for the preference to produce low-energy secondary electrons in ionization events is that the oscillator strength distribution of molecules composed of low Z atoms peaks in the 30 to 60 eV range. Hence, a fast primary electron tends to lose its energy in 30 to 60 eV increments, which can produce secondary electrons with a few 10s of eV at most. The degradation of low-energy secondary electrons to produce the free radical species that initiate radiation chemistry has also been studied extensively.[20] Electrons below 10 eV tend to cause excitations of rotational, translational, and vibrational modes of the molecules affected. Electron interactions by ionizations cause the ejection of other electrons, leaving the molecules in an excited state that can decay by dissociation, Auger electron emission, and relatively large energy transfers. Usually excitations and ionizations of electrons occur in the outer shells. Elastic collisions contribute to the electron track structures by influencing the location of the next inelastic event, especially

for decreasing electron energy. Their large cross sections (approximately below 200 eV) contribute significantly to the yield of new molecular species.

Bremsstrahlung energy losses rise nearly linearly and dominate for more energetic electrons (approximately higher than 30 MeV). As the spectrum of secondary electrons created by muon interactions with water contains mostly the lower-energy electrons (shown in Figure 4.3 in Chapter 4), more attention here is paid to ionization and excitation energy losses. The stopping power of electrons includes the total energy loss, dE, by collision and radiation for a path length, dl, in matter of density ρ.[21] For energies at which nuclear interactions may be neglected, the total mass stopping power can be separated into two components:

- Collisional stopping power in which all energy losses in particle collisions directly produce secondary electrons and atomic excitations, and
- Radiative stopping power in which all energy losses of primary electrons lead to Bremsstrahlung production.

Collisional and radiative stopping power of the electron is given by equations (1.3) and (1.4), respectively:

$$\left(\frac{S}{\rho}\right)_{col} = 2\pi {r_0}^2 N_e \frac{\mu_0}{\beta^2}\left(ln\frac{E^2(E+2\mu_0)}{2\mu_0 I^2} + \frac{\frac{E^2}{8} - (2E+\mu_0)\mu_0 \ln 2}{(E+\mu_0}\right)^2 + 1 - \beta^2 - \delta) \quad (1.3)$$

$$\left(\frac{S}{\rho}\right)_{rad} = 4{r_0}^2 \frac{N_e Z E}{137}\left(\ln\frac{2(E+\mu_0)}{\mu_0} - \frac{1}{3}\right) \quad (1.4)$$

where r_0 is the electron radius, N_e is the number of electrons per gram, μ_0 is the electron rest energy, E is the electron kinetic energy, β is the electron velocity, I is the mean ionization potential, δ is the density effect correction factor, and Z is the atomic number of the medium. The mean path length for an electron of initial energy E^0 can be defined by integrating the reciprocal of the total stopping power:

$$r_0 = \int_0^{E^0} \left(\frac{S(E)}{\rho}\right)_{tot}^{-1} dE, \tag{1.5}$$

This gives the path length that an electron would travel in the course of slowing down, in an unbounded uniform medium, if its rate of energy loss along the entire track was always equal to the mean rate of energy loss.

1.4.4 Muon Interactions in Water

A cosmic muon is a heavy charged particle, and its mass is about 205 times the mass of an electron. It loses energy primarily through ionizations and excitations of atoms by coulombic interactions until it decays while traversing matter. Its lifetime is approximately 2 μs . Bethe derived an equation for the stopping power of the medium by heavy charged particles. This equation, the Bethe-Bloch formula (1.6), can be used to calculate the stopping power of muons.

$$\frac{dE}{dx}]_{T<T_{cut}} = 2\pi r_e^2 mc^2 n_{el} \frac{(z_p)^2}{\beta^2} \left[ln\left(\frac{2mc^2\beta^2\gamma^2 T_{up}}{I^2}\right) - \beta^2\left(1 + \frac{T_{up}}{T_{max}}\right) - \delta - \frac{2C_e}{Z}\right] \tag{1.6}$$

In this equation, Z is the muon atomic number, m_{mu} is the muon mass, E is the muon energy, v is the muon velocity, z is the target atomic number, e is the charge

of an electron, m is the mass of an electron, c is the velocity of light and β is $\frac{v}{c}$. Several assumptions are made in this theory. The particle is assumed to interact with the target only through electromagnetic forces. Any energy loss due to nuclear reactions between the particle and the target nuclei is ignored.[22] The stopping power increases as the particle energy decreases. Therefore, the number of ions produced in the medium per unit length will increase along the path of the particle. The stopping power reaches a maximum near the end of the path and then it drops to zero as the particles come to rest.

This dissertation is organized as follows. The rest of Chapter 1 explains the theoretical part involved in this dissertation. Chapter 2 explains the basic features of the Geant4 simulation toolkit. Chapter 3 describes the simulation model created to study the track structures and energy deposition using Geant4. Chapter 4 presents all the simulation results performed in detail, and Chapter 5 gives the conclusions and introduces the future work to be initiated from this research.

Chapter 2

Geant4 Simulation

2.1 Introduction

Geant, the acronym for GEometry ANd Tracking, is a new object-oriented simulation software designed using Monte Carlo methods.[23] It provides a wide-ranging set of software components that can be fit to a variety of settings. The functionality of models of Geant4 can be seen and understood easily. The creation and addition of new models is a well-defined procedure with little modification to the existing code. Geant was first developed in the 1970s by an international collaboration formed by individuals from a number of cooperating institutes, HEP (High Energy Particle) experiments, and universities. The first widely used version of this code, Geant3, which was written in FORTRAN, was mainly used to model the physics of particle interactions. The new version of development of the code, Geant4, was first released in 1998 in the form of a toolkit allowing the user to easily extend the components of all domains. It was developed in CERN (European Council for Nuclear Research) to simulate particle interactions with detectors in high energy, nuclear,

and particle physics experiments. Recently this software package has been applied to simulation studies of radiation exposure in space travel and medical applications. It also allows a user to construct simulation codes to study particle trajectories in the interactions of radiation at the cellular and DNA scales. Geant4 includes a complete range of functions including specifying the particles to be used, specifying the processes that the particles will undergo, choosing the model describing each process, defining the materials to be used in the run, defining the geometry of the system, assigning the materials to the components of the geometry, specifying the sensitive detector components, modeling the response of the detectors, generating the primary events, transporting the primary particles through the system simulations, the production of secondary particles as primary particles interact with the matter, and storing event data for further analysis. Geant4 includes the following key domains of simulating the passage of particles through matter:

- Geometry and materials,
- Particle interaction in matter,
- Tracking management,
- Digitization and hit management,
- Event and track management,
- Visualization and visualization framework, and
- User interface.

The early design of Geant4 was based on an analysis of initial user requirements.[24] The physics processes, geometry, and visualization implemented in the Geant4 simulation

toolkit have been tested in many applications.[8, 25] The hierarchical structure of the toolkit is shown in Figure 2.1.[8]

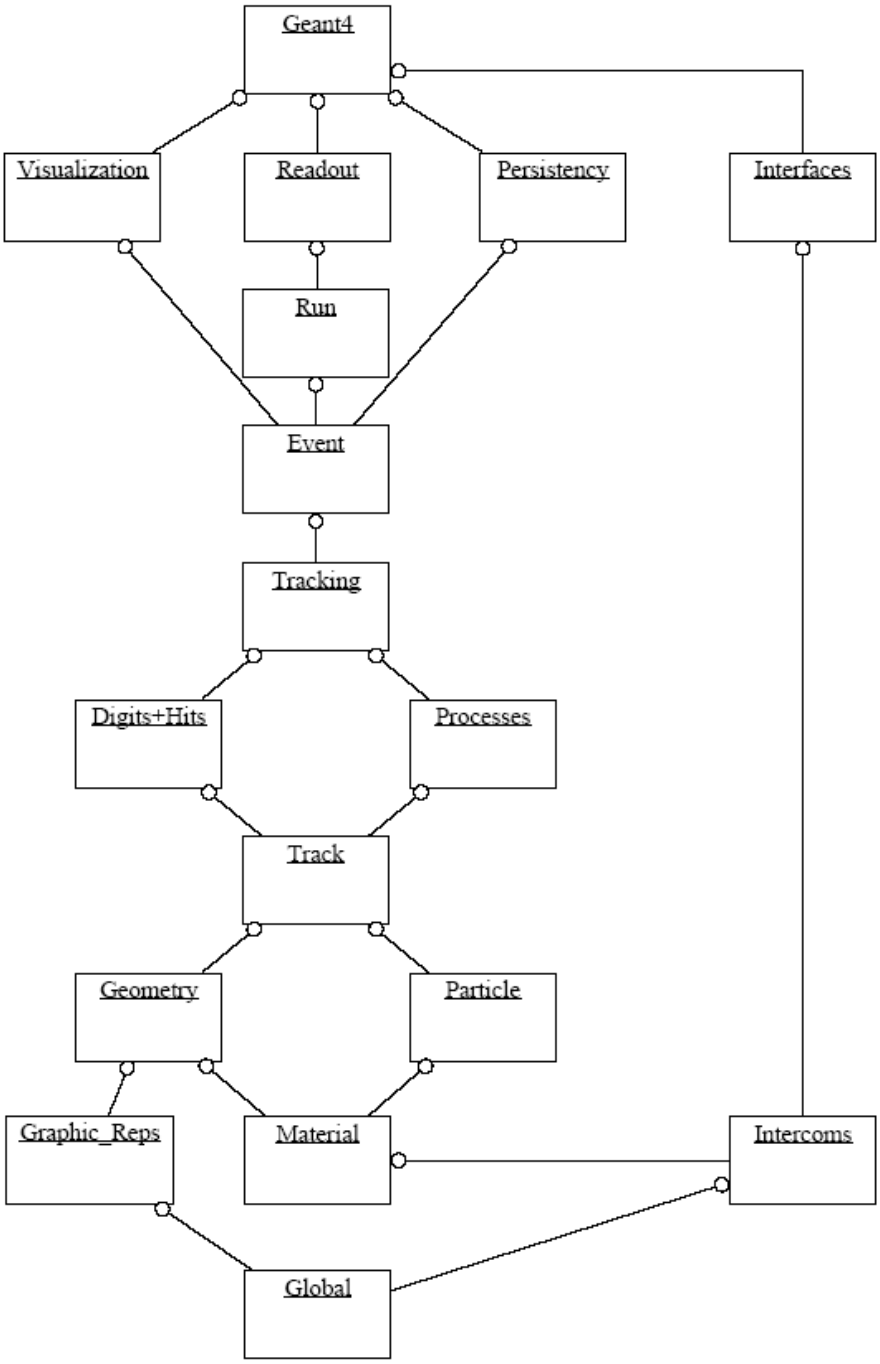

Figure 2.1: Hierarchical Structure of Geant4 Toolkit. The open circle on the joining lines represent the "using" relationship.

2.2 Geant4 Geometry

In Geant4, the spatial positioning, its logical relation and solid modeling of the detector element are managed by the concepts "Logical volume," "Physical volume," and "Solid," respectively. The Geant4 solid model is STEP compliant which is an ISO standard defining the protocol for exchanging geometrical data between CAD systems. This supports multiple solid representations such as Constructive Solid Geometry (CSG) and Boundary REPresented solid (BREP). CSG solids are 3-D primitives and are described by a set of necessary parameters. As an example, the following constructor can be used to create the cylindrical section in Figure 2.2.[26]

```
G4Tubs (const G4String pName,\newline
             G4double  pRMin,\newline
             G4double  pRMax,\newline
             G4double  pDz,\newline
             G4double  pSPhi,\newline
             G4double  pDPhi)\newline
Where pRMin = 10, pRMax = 15, pDz = 20, pSPhi = 0 degrees, and pDPhi
= 90 degrees
```

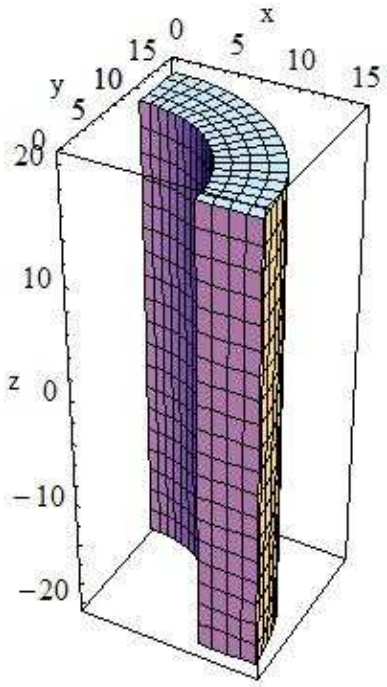

Figure 2.2: An Example of Creating a Geometry of a Cylindrical Portion in Geant4. The minimum radius of the cylinder is 10 *cm*. The maximum radius is 15 *cm*.

The logical volume manages the information associated with the detector elements represented by a given solid and material. It is defined as the mother volume of physical volumes. The information relative to the visualization attributes and user-defined parameters that are relevant to tracking, the electro-magnetic field, or cuts are also managed by the logical volume. Spatial positioning of the volumes that describe the detector elements are represented by physical volumes. A single physical volume as well as repeated volumes can be placed. When the multiple copies of a repeated volume are different in size, they are called parameterized volumes. Figure 2.3 is an example of a geometry developed for a Geant4 example. This is the graphic output of the proton therapy beam line in the Geant4 Hadrontherapy advanced example. The left picture is the real proton therapy beam line used to develop the Geant4 example. It has been installed at the Laboratori Nazionali del Sud (INFN) in Catania, Italy.

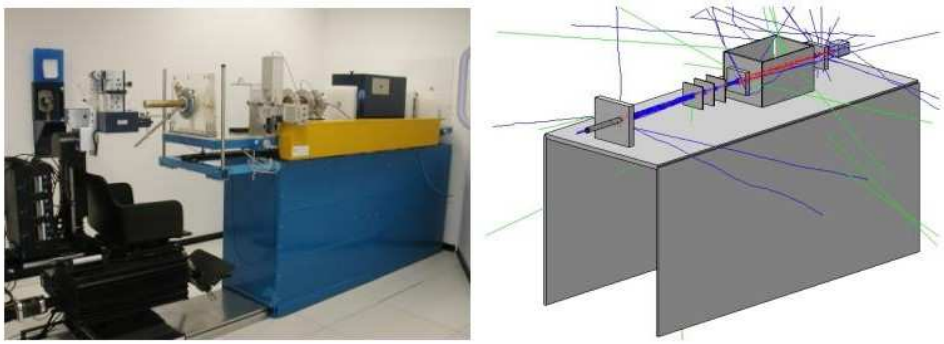

Figure 2.3: An Example of a Complex Geometry Developed in Geant4. This is a graphic output of a proton therapy beam line. The left picture is the real proton therapy beam line installed at the Laboratori Nazionali del Sud (INFN) in Catania, Italy.

2.3 Geant4 Physics Processes

A detailed description of the physics used principles is given in the Geant4 Physics Reference manual.[3] Validation and the details of physics can be found in reference.[25] Energy loss is continuous below a given energy threshold. Above this threshold the energy loss is determined by the production of secondary particles such as gammas, electrons, and positrons. The mean rate of energy loss for all charged particles is given by

$$\frac{dE_{soft}(E, T_{cut})}{dx} = n_{at}. \int_0^{T_{cut}} \frac{d\sigma(Z, E, T)}{dT} T dT, \tag{2.1}$$

The total cross section per atom for the ejection of a secondary particle of energy higher than the threshold is given by Equation (2.2):

$$\sigma(Z, E, T_{cut}) = \int_{T_{cut}}^{T_{max}} \frac{d\sigma(Z, E, T)}{dT} dT, \tag{2.2}$$

where Z is the atomic number, T is the kinetic energy of the ejected secondary particle, E is the total energy of the incident particle, T_{cut} is the kinetic energy cutoff, T_{max} is the maximum energy transferable to the secondary particle, and n_{at} is the number of atoms per volume in the material. When several processes are responsible for the energy loss for a particular particle, the total continuous part of the energy loss is the sum

$$\frac{dE_{soft}(E, T_{cut})}{dx} = \sum \frac{dE_{soft,i}(E, T_{cut})}{dx} \tag{2.3}$$

Electron Incident

Ionization

The integration of equation 2.1 leads to the formula

$$\frac{dE}{dx}_{T<Tcut} = 2\pi r_e^2 mc^2 n_{el} \frac{1}{\beta^2} \left[ln \frac{2(\gamma+1)}{(\frac{I}{mc^2})^2} + F^{\pm}(\tau, \tau_{up}) - \delta \right] \tag{2.4}$$

where r_e is the classical electron radius $\frac{e^2}{4\pi\epsilon_0 mc^2}$, mc^2 is the mass energy of the electron, n_{el} is the electron density in the material, I is the mean excitation energy in the material, $\gamma = \frac{E}{mc^2}$, $\beta^2 = 1 - \frac{1}{\gamma^2}$, $\tau = \gamma - 1$, and δ is density effect function. δ ray production has been implemented using the concept of Möllr and Bhabha scattering.

Bremsstrahlung

This process provides the energy loss of electrons and positrons due to the radiation of photons in the field of a nucleus. Above a threshold, energy loss is simulated by created photons. The mean value of the energy lost by an electron by Bremsstrahlung is

$$E_{Loss}^{brem}(Z, T, k_c) = \int_0^{k_c} k \frac{d\sigma(Z, T, k)}{dk} dk, \tag{2.5}$$

and the total cross section for the emission of a photon of energy larger than k_c is

$$\sigma_{brem}(Z, T, k_c) = \int_{k_c}^{T} \frac{d\sigma(Z, T, k)}{dk} dk, \tag{2.6}$$

e^+e^- Annihilation

The atomic electron is assumed to be free and at rest. The annihilation cross section is given by the equation

$$\sigma(Z,E) = \frac{Z\pi r_e^2}{\gamma+1}\left[\frac{\gamma^{2}+4\gamma+1}{\gamma^2-1} ln\left(\gamma+\sqrt{\gamma^2-1}-\frac{\gamma+3}{\sqrt{\gamma^2-1}}\right)\right] \tag{2.7}$$

where E is the total energy of the incident positron, $\gamma = \frac{E}{mc^2}$, and r_e is the classical electron radius.

Muon Incident

Ionization

The continuous energy loss of muons is calculated using the Bethe-Bloch restricted energy loss formula as discussed in Chapter 2:

$$\frac{dE}{dx}]_{T<T_{cut}} = 2\pi r_e^2 mc^2 n_{el}\frac{(z_p)^2}{\beta^2}\left[ln\left(\frac{2mc^2\beta^2\gamma^2 T_{up}}{I^2}\right)-\beta^2\left(1+\frac{T_{up}}{T_{max}}\right)-\delta-\frac{2C_e}{Z}\right] \tag{2.8}$$

The total cross section per atom is given by

$$\sigma(Z,E,T_{cut}) = \frac{2\pi r_e^2 Z z_p^2 mc^2}{\beta^2}\left[\left(\frac{1}{T_{cut}}-\frac{1}{T_{max}}\right)-\left(\frac{\beta^2}{T_{max}}ln\frac{T_{max}}{T_{cut}}+\frac{T_{max}-T_{cut}}{2E^2}\right)\right] \tag{2.9}$$

The mean free path in a given material is

$$\lambda = (n_{at}.\sigma)^{-1} \tag{2.10}$$

Bremsstrahlung

At moderate muon energies, bremsstrahlung demonstrates the other muon interaction processes (Chapter 2). The differential cross section for muon bremsstrahlung is given by

$$\frac{d\sigma(E,\epsilon,Z,A}{d\epsilon} = \frac{16}{3}\alpha N_A \left(\frac{m}{\mu}r_e\right)^2 \frac{1}{\epsilon A} Z(Z\phi_n + \phi_e)\left(1 - v + \frac{3}{4}v^2\right) \tag{2.11}$$

The e^+e^- pair production by muons is not been included in the simulation model because this process dominates only in the TeV energy range. The upper limit of the muon energy used in the simulation model is 120 GeV (Figure 3.3).

Compton Scattering

The empirical cross section formula is as follows:

$$\sigma(Z,E_r) = \left[P_1(Z)\frac{log(1+2X)}{X} + \frac{P_2(Z) + P_3(Z)X + P_4(Z)X^2}{1 + aX + bX^2 + cX^3}\right] \tag{2.12}$$

where Z is the atomic number of the medium, E_r is the energy of the photon, $X = \frac{E_r}{mc^2}$, m is the electron mass, and $P_i(Z) = Z(d_i + e_i Z + f_i Z^2)$. This equation is used to simulate the Compton scattering of a photon from an atomic electron. The mean free path of a photon undergoing Compton scattering is given by

$$\lambda(E_r) = (\sum i n_{ati}.\sigma_i(E_r))^{-1} \tag{2.13}$$

where n_{ati} is the number of atoms per volume of the i^{th} element of the material.

2.4 Tracking Management

Key consideration in the object design of the *tracking* category of the simulation model is the fact that the performance of the simulation critically depends on the CPU time spent moving the particle in its steps. Geant4 tracks physics processes in a very generic way so that it is independent of the particle type or the specific physics process.[28] Physics processes are characterized by three actions handled by tracking:

- *At rest*, for particles at rest (e.g., decay at rest),
- *Along step*, implements a behavior that happens continuously along a step (e.g., Cherenkov radiation), and
- *Post step*, happens at the end of the step (e.g., secondary particle production by a decay).

Transactions between the event, the track, and the tracking categories are brokered by the interface class, *G4TrackingManager*. This class handles the passing of the required message between the upper hierarchical object and the lower hierarchical object in the tracking category. The tracking manager takes necessary actions to complete tracking the track received from the event manager.

For optimization of the tracking, a new technique has been derived from the voxel-based method. Here the space is subdivided into cubic volume elements (voxels), and a tree-based map is created by recursively dividing the detector into octants. In this *smart voxels* technique, for each mother volume, a 1-D virtual division is performed for each mother volume. The best axis for the virtual division is chosen by using a heuristic. Slices

that contain the same volumes are gathered into one slice in order to optimize memory and performance.

2.5 Visualization

Visualization is one of the most important part of a simulations. There are different requirements for Geant4 visualizations. As an example, the visualization type well-suited for preparing high-quality Post Script outputs will not be good to use for 3-D presentations. It is difficult to respond to various requirements with only one visualizer. Therefore, Geant4 has different visualization drivers for different purposes. Visualization procedures are controlled by the visualization manager. This visualization manager accepts a user's requests for visualization, processes them, and passes the processed requirements to the currently selected visualization driver. In Geant4 visualization, simulated data can include the following components:

- Detector components (Physical volumes, logical volumes, and solids),
- Particle trajectories,
- Hits of particles in detector components,
- Polylines,
- Markers,
- Texts, and
- Coordinate axes.

Attributes such as color can be defined using G4VisAttributes. Table 2.1 list available visualization drivers with their required graphic systems and available platforms.

Table 2.1: Available Visualization Drivers in Geant4.

Driver	Required 3-D graphic system	Platform
DAWNFILE	Fukui Renderer DAWN	UNIX.Windows
DAWN-Network	Fukui Renderer DAWN	Unix
HepRepFile	WIRED event display	UNIX, Windows
OpenGL-Xlib	OpenGL	UNIX with Xlib
OpenGL-Motif	OpenGL	UNIX with Motif
OpenGL-Win32	OpenGL	Windows
OpenInventor-X	OpenInventor, OpenGL	UNIX with Xlib or Motif
Openinventor-Win32	OpenInventor, OpenGL	Windows
RayTracer	(JEPG viewer)	UNIX,Windows
VRMLFILE	(VRML viewer)	UNIX,Windows
VRML-Network	(VRML viewers)	UNIX

Chapter 3

The Simulation Model

3.1 Introduction

This chapter the simulation model that was developed to study the sea-level cosmic ray muon particles and energetic electrons created from them in order to assess the radiobiological effects of this low-dose radiation will be discussd. More than 80 percent of human the tissue consists of water.[29] Because the particular interest is about the particles that are stopped in large bio systems such as the human body, modeling of the simulation started with defining a cubic water volume that represents human tissue. Later, this volume was sliced in 3-D using the Geant4 parameterized volume feature. This enabled the study of nanometric structures of water (about the size of a DNA molecule). In further simulations, in order to determine the dose to a human body, the size of the water volume was changed so that was approximately equal to the size of an average human. With the intention of studying the shielding effect, a concrete block was defined to shield the cosmic rays coming to the water volume. Results obtained by this study are discussed in Chapter 4. Finally the

geometry was developed to represent a simple human body with major body composition such as different types of tissue and bones.

3.2 Structure of the Simulation Model

A schematic modular diagram of the simulation is shown in Figure 3.1. Detector geometry, physics processes, event action, run action, stepping action and tracking action were defined in different independent modules. Information about the incoming particles was fed by an input macro file. An output data file was obtained with detailed information such as energy deposition and also step and track information of the primary as well as secondary particles. Three types of visualization drivers were included in the program to optimize different types of requirements. The major key domains used in modeling the above simulation are discussed in the rest of this chapter.

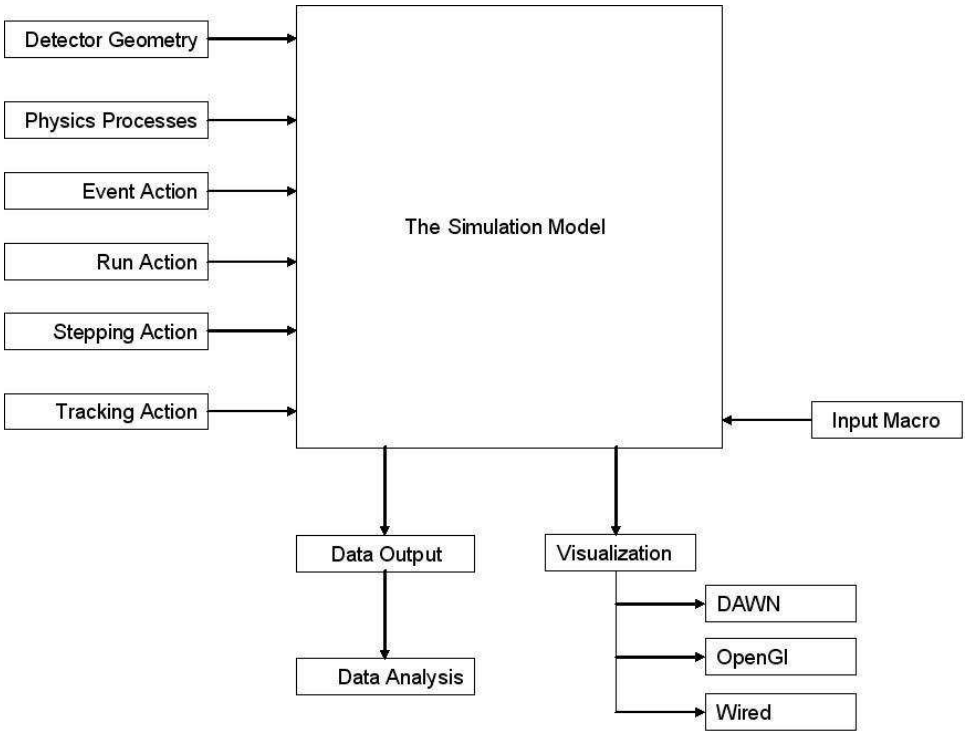

Figure 3.1: Structure of the Simulation Model

3.3 Geometry and Materials

The simulation process was two-fold. First, this process was carried out on various sizes of water volumes by changing the incoming energy and the orientation of muons. The geometry was modified by slicing the cubic water chamber in three dimensionally so that the energy deposition and the particle interactions in small cubes could be simulated very closely.[30] This enabled the study of small structures up to 2 nm cubics (about the size of a DNA molecule). Visualization of the sliced water cube is shown in Figure 3.2. The dose in a volume of a rectangular solid with dimensions of 170, 60, 40 cm in the Z, Y, and X directions, respectively,was simulated. This is the approximate size of a human body. The simulation model was ready to simulate the energy deposition in small water volumes. In order to measure the dose given to a human body, it was inappropriate to launch muons in only one direction. In the Earth's atmosphere, most of the sea-level muons come from all directions with a vertical angle of $0-70$ degrees and have energies in the range from 0 to 120 GeV. Therefore, a similar muon flux environment was inroduced in the chamber.[33, 32, 31]

Muon momentum data to model sea-level flux was taken from reference [32]. These data consist of the actual momenta from 0.2 GeV/c to 120 GeV/c measured using the NMSU-WIZARD/CAPRICE magnetic spectrometer. The measurements were carried out at Lynn lake, Manitoba, Canada (56 N, 101 W) at sea level during July 1994.

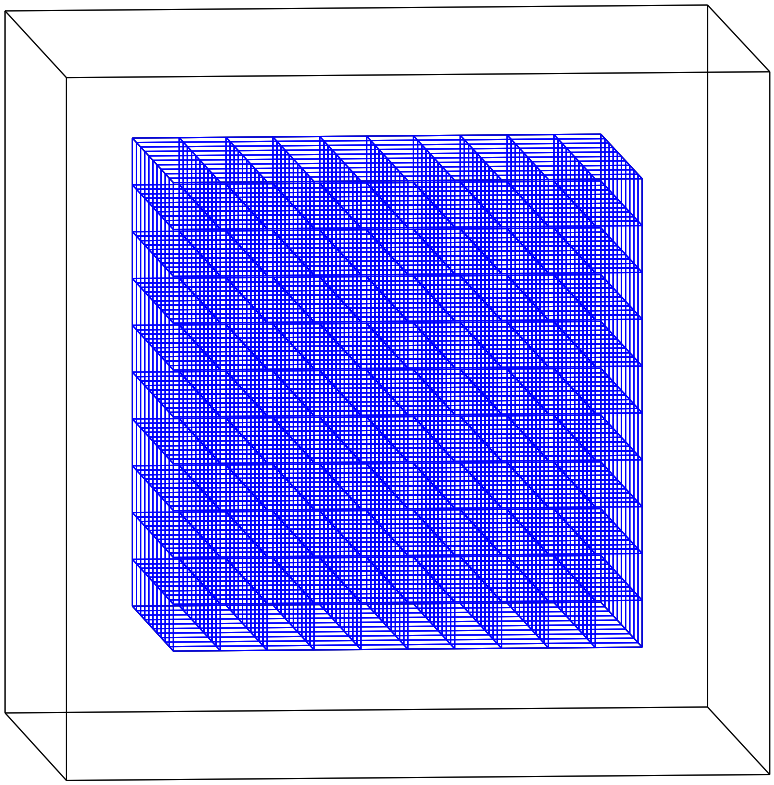

Figure 3.2: 3-D Sliced Water Volume. Size of a small volume is 1 cubic centimeters

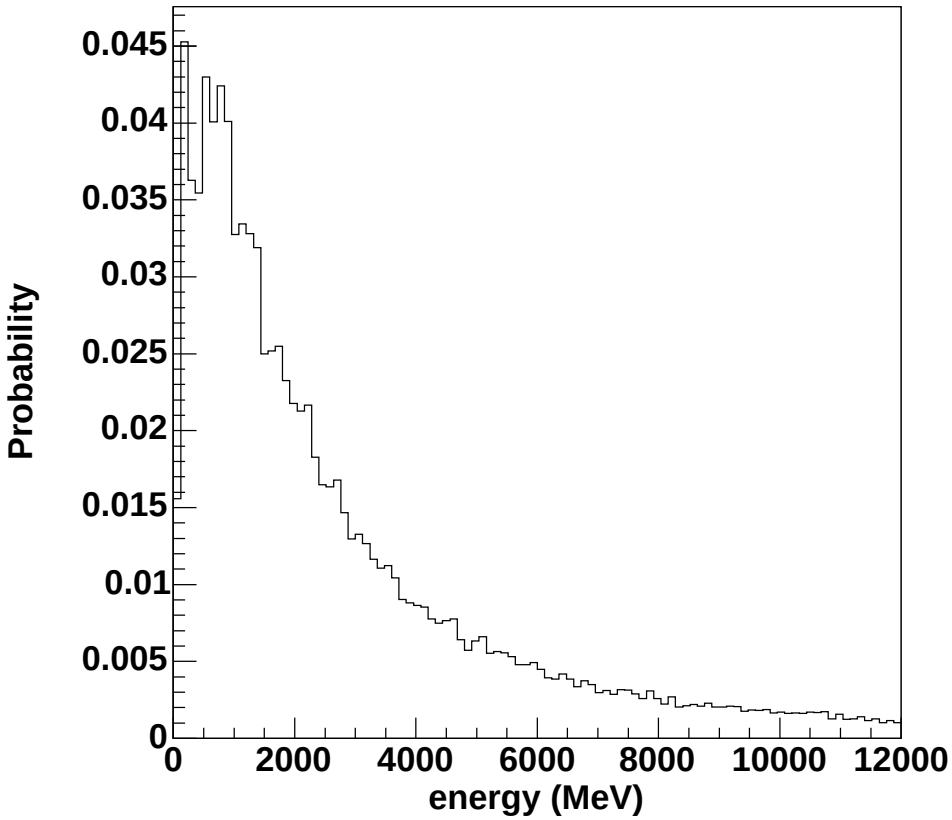

Figure 3.3: Muon Energy Distribution Applied to the Water Chamber Simulation

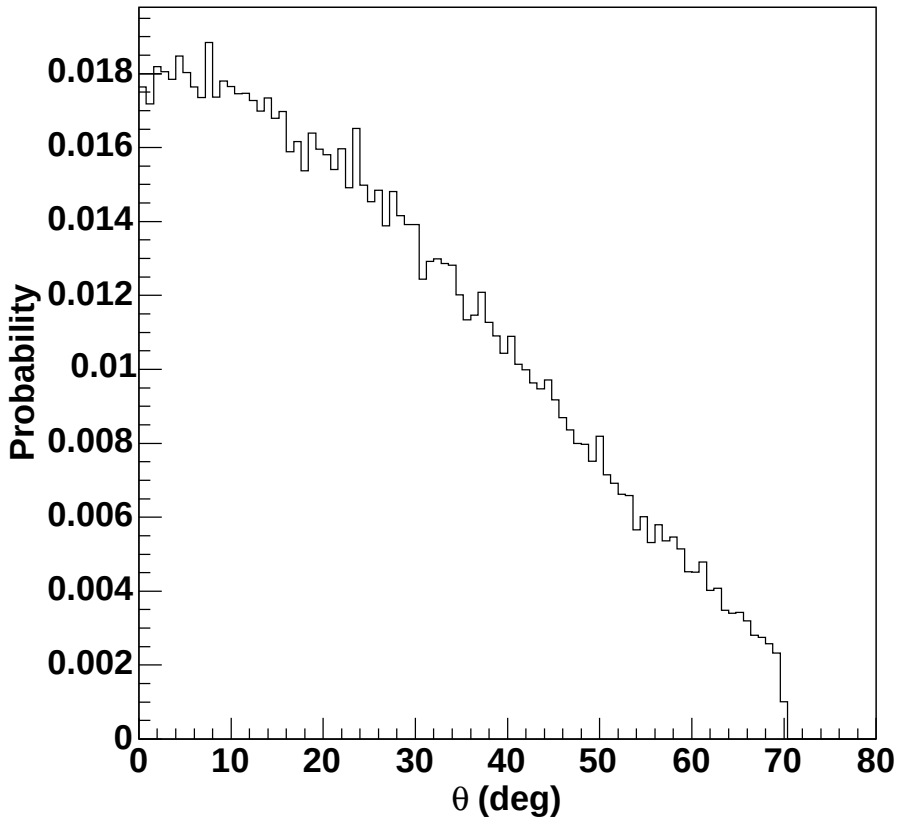

Figure 3.4: Muon Angular Distribution in Chamber

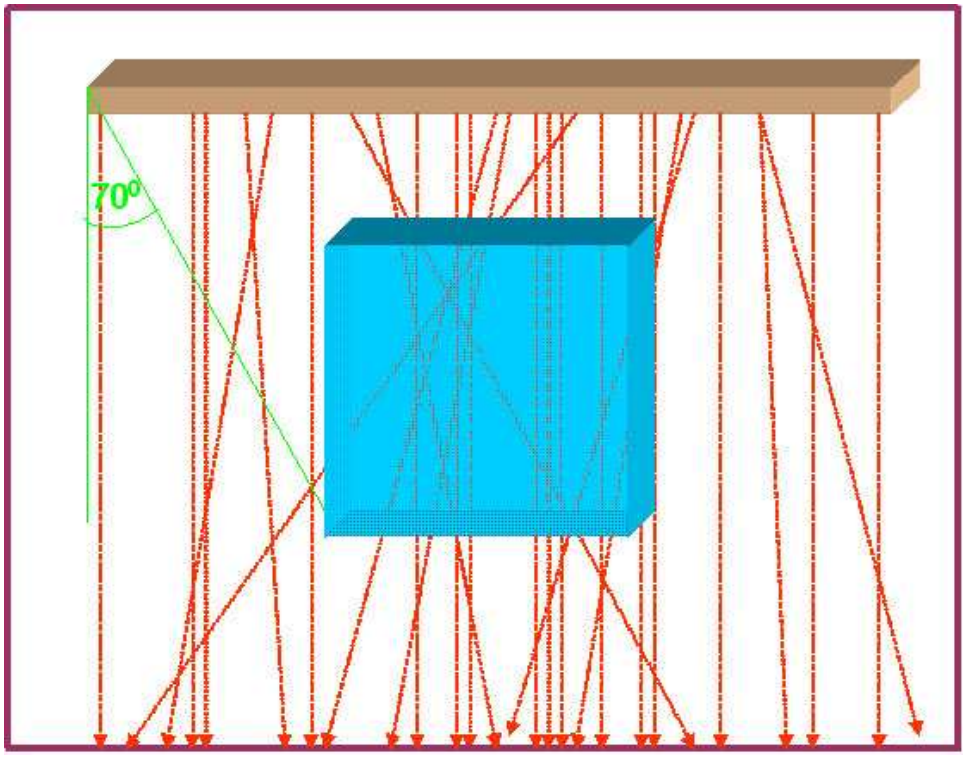

Figure 3.5: Sea-level Muon Flux Implementation. The upper plane is the muon source. The cube in the center is the water chamber.

3.3.1 Sea-Level Muon Flux Generation

Bacause the main objective of this research was to measure the dose of muon and secondary electron radiation received by a human body, the dimensions of the water chamber were changed so that they were approximately equal to the size of a human. The new dimensions of the chamber were $40 \times 60 \times 170\ mm^3$. Previously the muons were launched one by one along the Z direction as mentioned earlier. The next step was to introduce the sea-level cosmic muon flux in this chamber so that it experienced the actual sea-level environment.

Muon Momentum

Muon momentum data to introduce flux were extracted from reference [32]. These data consist of the momentum from 0.2 GeV/c to 120 GeV/c measured using WIZARD/CAPRICE magnet spectrometer owned by New Mexico State University. The experiments to collect these data were carried out at Lynn Lake, Manitoba, Canada ($56.5^0\ N$, $101.0^0\ W$) at the sea level during 1994.

Incident Angle of the Muons

The incident angle of the muons varied from 0 degrees to 70 degrees from the vertical axis. These data were extracted from "Review of Particle Physics", Particle Data Group.[33] The angular distribution of muons is shown in Figure 3.4.

Table 3.1: Measurement of Ground-level Muons.

Momentum Interval($\frac{GeV}{c}$)	$(\mu-)CAPRICE94$
$0.2 - 0.3$	$(1.1 \pm 0.1) \times 10^1$
$0.3 - 0.4$	$(1.36 \pm 0.07) \times 10^1$
$0.40 - 0.55$	$(1.44 \pm 0.04) \times 10^1$
$0.55 - 0.70$	$(1.35 \pm 0.03) \times 10^1$
$0.70 - 0.85$	$(1.33 \pm 0.03) \times 10^1$
$0.85 - 1.0$	$(1.21 - 0.03) \times 10^1$
$1.0 - 1.2$	$(1.10 \pm 0.03) \times 10^1$
$1.2 - 1.4$	$(1.01 \pm 0.02) \times 10^1$
$1.4 - 1.6$	8.7 ± 0.2
$1.6 - 2.1$	7.3 ± 0.1
$2.1 - 2.94$	5.20 ± 0.09
$2.94 - 4.12$	3.38 ± 0.06
$4.12 - 5.5$	1.98 ± 0.04
$5.5 - 7.0$	1.25 ± 0.03
$7.0 - 10.0$	$(6.9 \pm 0.1) \times 10^{-1}$
$15.5 - 23.0$	$(1.08 \pm 0.03) \times 10^{-1}$
$23.0 - 31.1$	$(4.6 \pm 0.2) \times 10^{-2}$
$31.1 - 43.6$	$(1.9 \pm 0.1) \times 10^{-2}$
$43.6 - 61.1$	$(7.1 - 0.6) \times 10^{-3}$
$61.1 - 85.6$	$(3.0 \pm 0.1) \times 10^{-3}$
$85.6 - 120.0$	$(1.2 \pm 0.2) \times 10^{-3}$

The overall angular distribution of muons at the ground was proportional to $cos^2\Theta$. The mean energy of the muons at the ground was 4 GeV.[34] Table 3.1 gives the flux of the muons in each momentum range. The energy spectrum was almost flat below 1 GeV, it steepened gradually to reflect the primary spectrum in the $10 - 100$ GeV range, and then it steepened further at higher energies as shown in Figure 3.3.

Implementation of Muon Flux

Data for the muon momentum were fed into a macro file "momentum.mac" in "DNAPrimarygeneratorAction" class in the function "DNAPrimaryGeneratorAction:: cosmicRayMuonMomentum()." Data for the muon angle was fed into another macro file called "angle.mac" in the same class in the function "DNAPrimarygeneratorAction:: cosmicRayMuonAngle()."

The Geant4 General Particle Source (GPS) was used to generate muons. In Geant4 GPS, one can select different shapes of sources such as planar, disk, sphere, cubic, linear, and point. In this work, the muon momentum and the angle were randomly fed into the chamber by introducing a rectangular planar source. Figure 3.5 shows how the cosmic muon flux has been implemented. The virtual muon source is a plane of length 24 cm and width 65 cm. The length and the width of the source has been determined by the dimensions of the water chamber. The maximum possible angle of incident muons is 70^0 with the vertical axis and there is always a vertical angle of 70^0 from the edge of the source to the lowest edge of the chamber. One particle was created in each square centimeter of the source-plane per second.[13] Once the flux implementation was successful, a concrete shield was created above the human-sized water chamber in order to study the shielding effect. The thickness

of this concrete shield can be changed. The calculated yearly dose and other results are discussed in Chapter 4.

Gradually the human-sized rectangular cube was replaced by a simple human body phantom as shown in Figure 3.6. This phantom consists of most of the parts in an adult human body. Its height is 170 cm. The dimensions to construct the human body were extracted from[35]. The materials used to define each part of the phantom are given in Table 3.2.[36]

Table 3.2: Element Compositionsin Percentage for Defining an average Body Material

bBody Part	Ca	Mg	H	O	N	P	S	Cl	K	Fe	Si	Na	Al	C	Density kgm^{-3}
Head	-	-	11	28	0.7	-	0.1	0.1	-	-	-	0.1	-	59	967
Skull	4	0.1	9	37	6	3	0.2	0.2	0.1	0.1	-	0.1	-	40	1159
Neck	-	-	11	28	0.7	-	0.1	0.1	-	-	-	0.1	-	59	967
Shoulder	-	-	10	71	4	0.2	0.3	0.1	0.4	-	-	0.1	-	14	1061
Torso	-	-	10	71	4	0.2	0.3	0.1	0.4	-	-	0.1	-	14	1061
Skeleton	15	0.1	5	43	5	7	0.3	0.1	0.1	-	-	0.1	-	23.5	1575
Right Hand	-	-	10	71	4	0.2	0.3	0.1	0.4	-	-	0.1	-	14	1061
Left Hand	-	-	10	71	4	0.2	0.3	0.1	0.4	-	-	0.1	-	14	1061
Right Hand Bone	15	0.1	5	43	5	7	0.3	0.1	0.1	-	-	0.1	-	23.5	1575
Left Hand Bone	15	0.1	5	43	5	7	0.3	0.1	0.1	-	-	0.1	-	23.5	1575
Right Leg	-	-	10	71	4	0.2	0.3	0.1	0.4	-	-	0.1	-	14	1061
Left Leg	-	-	10	71	4	0.2	0.3	0.1	0.4	-	-	0.1	-	14	1061
Right Leg Bone	15	0.1	5	43	5	7	0.3	0.1	0.1	-	-	0.1	-	23.5	1575
Left Leg Bone	15	0.1	5	43	5	7	0.3	0.1	0.1	-	-	0.1	-	23.5	1575
Lung Inhale	-	-	10	75	3	0.2	0.3	0.2	0.3	-	-	0.2	-	10	217
Lung Exhale	-	-	10	75	3	0.2	0.3	0.2	0.3	-	-	0.2	-	10	508
Heart	-	-	11	36	2	0.1	0.1	0.1	-	-	-	0.1	-	51	990
Stomach	-	-	66	33	-	-	-	-	-	-	-	-	-	-	1000
Liver	-	12	-	-	-	30	16	17	19	-	-	-	-	6	1071
Concrete	6	-	-	52	-	-	-	-	-	4	32	1	4	-	2500

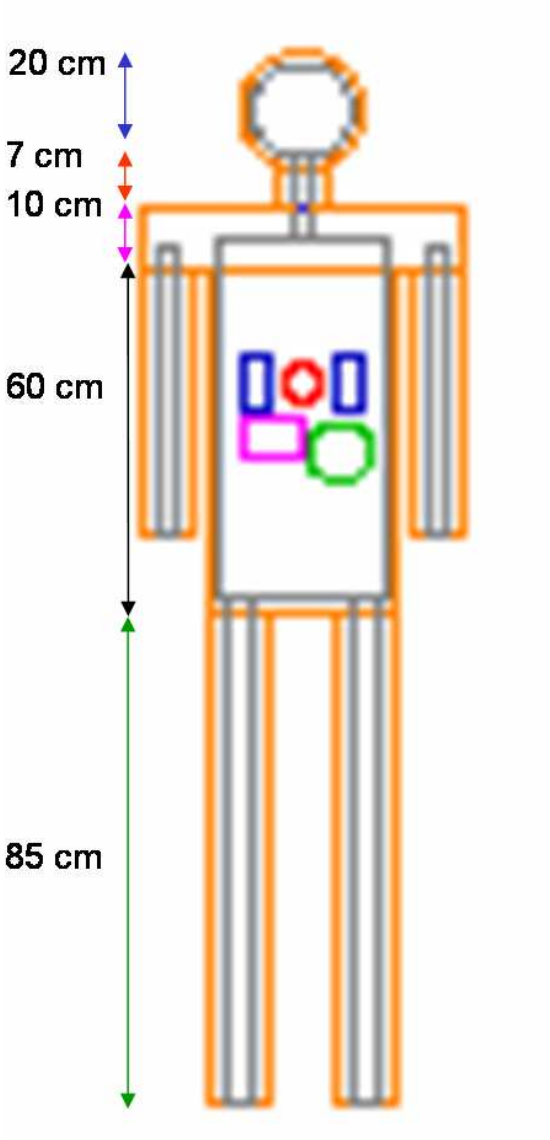

Figure 3.6: Simulated Simple Human Body.

3.4 Physics Processes

Transportation

The transportation process determines the geometrical limits of a step. Since electrons and muons are charged, they have an electromagnetic field. This electromagnetic field is responsible for propagating the particle in the field. This is done using equations of motion originally included in Geant4. These equations of motion of particles are solved utilizing Runge Kutta methods.

Electromagnetic Interaction

As mentioned earlier, the simulation proceeds by steps. The implemented physics decides where should the steps be taken place and what interactions occur at each step. In each interaction the "mean free path" or "interaction length" is calculated and the interaction that produces the shortest mean free path is chosen. Although Geant4 allows researches to define a maximum allowed step length, this facility was not utilized in the simulation. Particles used in this model cannot induce nuclear reactions. Therefore, only electromagnetic physics is taken into consideration. Geant4 consists of "standard electromagnetic interactions" as well as "Low-energy extensions.". Both modules are applied in the current application when and where necessary.

Digitization and Hit Management

A *hit* is a snapshot of the physical interaction of interactions of a track in a sensitive detector. A *digit* represents a detector output. A digit is created from one or more hits and other digits. In the simulation model *chamber*, the sensitive detector is an object derived from the G4VSensitiveDetector class. The *chamber* creates hits using the information given in the current step. At the tracking time, when the step is inside of a volume that has a pointer to a *chamber*, a *chamber* is invoked with the correct step information. The LCDCSPrimaryGeneratorAction module feeds the momentum and angle data for flux implementation using the implemented data. The LCDCSTrackerHit tracks the information of each hit. The energy cuts for each particles are manually defined in the LCDCSStackingAction module. Information such as the particle name, track identification number, initial kinetic energy of the particle, and the generating positions are extracted this class. Also, this module can switch off unnecessary particles such as neutrinoes.

3.5 Tracking Management

The LCDCSTrackingAction module feeds the momentum spectra and angular distribution of the input particles to implement the sea-level flux. The information of the primary particles as well as all generations of secondaries are extracted to the output file using the LCDCSSteppingAction module. This information includes the positions of each step of the particles, energy deposition, replica number, and the name of the particle. The total energy deposited in a particular event and the track length of each particle are extracted in this class. It is possible to flag the output file so that each event can be differentiated.

The trajectories of the particles are extracted and drawn using this class. The formation of parameterized volumes is defined in the LCDCSChamberParameterization module. The coordinates of each replica are defined, and an identification (CopyNo) for each replica is assigned here.

Chapter 4

Energy Deposition Study

4.1 Introduction

Simulation studies have been carried out to study particle interactions in water and other tissue-like materials using the model described in Chapter 3. This chapter presents the simulation results and their significance in low-energy dosimetric studies, and it is organized as follows. The first part of this chapter shows the particle interactions and visualization of tracks originated from particles with different energies in a small water volume. Before using the model for applications, validation studies are carried out in two different aspects. First, the results of a validation study are compared with experimental and other simulation work. Second, the Bethe-Bloch formula is used to verify simulating results of the stopping power of muons in water. The simulated interactions of the spectrum of sea-level muons are presented next. This leads to very interesting results including the yearly dose of a rectangular cubical chamber of the size of an average human body. The next section gives a picture of the muon shielding studies performed with concrete slabs of different thicknesses.

The last part of the chapter describes the detailed study about the spectrum of electrons of all generations created by sea-level muons. This includes the studies of the energy spectrum of secondary and higher generations of electrons and their penetration lengths.

4.2 Particles in a Small Cubic Volume

A cubic volume of $10 \times 10 \times 10\ cm^3$ filled with liquid water was introduced for the initial study of energy deposition. The use of liquid water in this study is motivated by the fact that liquid water constitutes more than 80 percent of human tissue.[29] This water cube is irradiated by electrons and muons with different initial energies. The particles are sent off along the $-Z$ direction at the center of the XY plane of the chamber. Figures 3.1 and 3.2 are two examples for the hits distribution for an electron with initial energy of 50 MeV and a muon with initial energy of 40 MeV in the water volume. A closer look at the interactions of these particles shows that it undergoes several different processes as discussed in Chapter 1. For instance, the 30 MeV electron in Figure 4.1 has undergone ionization and bremsstrahlung processes, and it has created 10 secondary electrons and seven gamma rays. These gamma rays also produce Compton scattering. This electron imparts 27.5 MeV of energy before it exits the water volume. The muon in Figure 4.2 creates four secondary electrons and four gamma rays. The interaction processes are muon ionization, electron ionization, bremsstrahlung, Compton scattering, muon decay, and photo-ionization. This muon imparted more than 29.5 MeV of energy to the secondaries and decayed when it owned 0.45 MeV of energy. The electron created from this decay initially carries 30 MeV of kinetic energy.

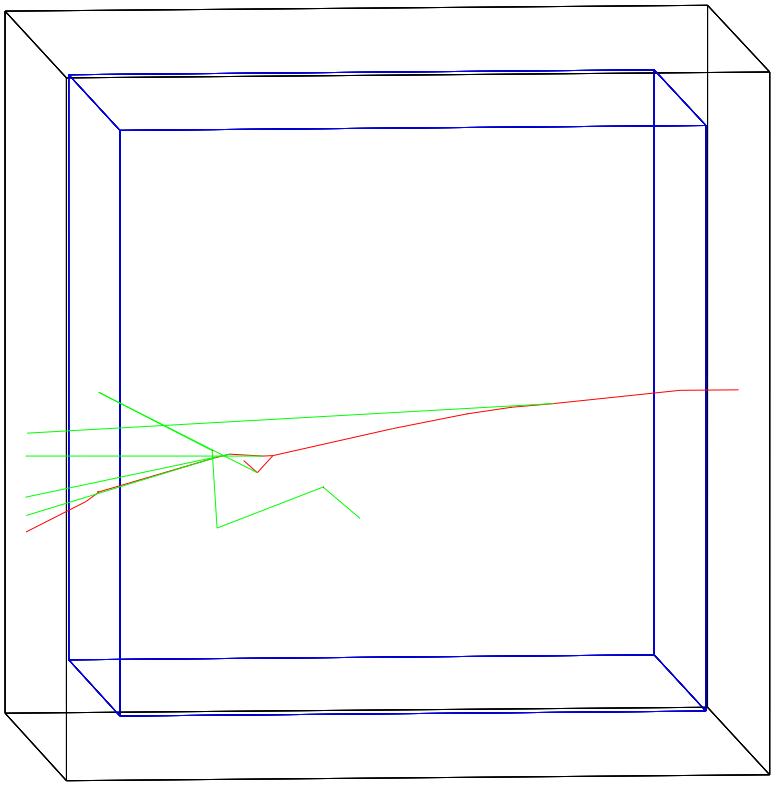

Figure 4.1: Track of a 30 MeV Electron and its Secondaries in $10 \times 10 \times 10$ cm^3 Water Volume. The red-colored particles are electrons and the green-colored particles are created gamma rays. The small red dots at the end of the green tracks represent low-energy electron tracks created in the chamber.

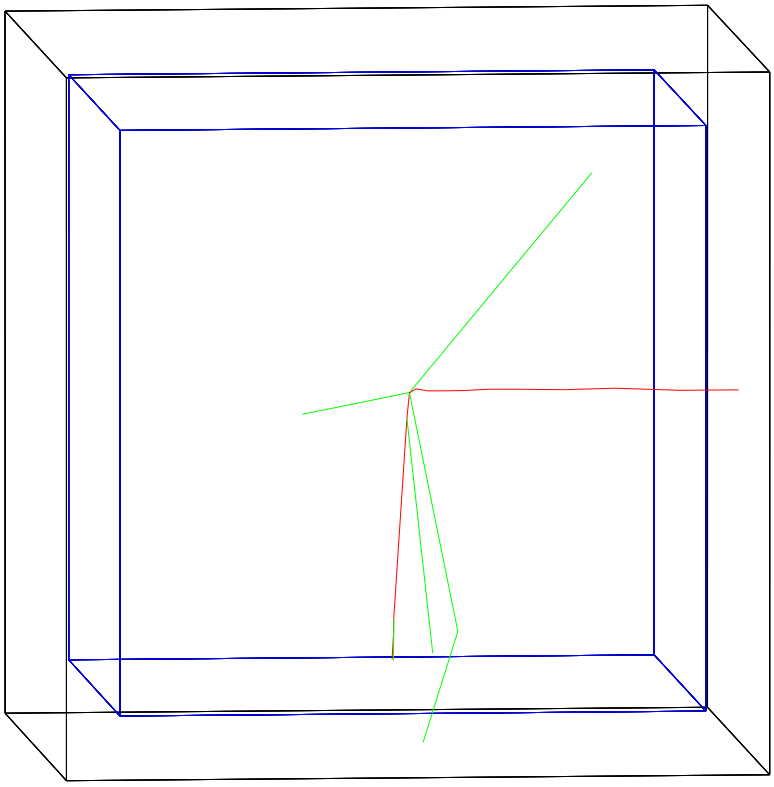

Figure 4.2: Track of a 30 MeV Muon and its Secondaries in $10 \times 10 \times 10$ cm^3 Water Volume. The red-colored particles are negatively charged muons and electrons and the green-colored particles are created gamma rays. The small red dots at the end of the green tracks represent low-energy electron tracks created in the chamber.

In order to analyze the particle interactions in small structures of water, with the intention of studying the energy localization in nanometric structures such as DNA, the water chamber was sliced 3-D (Figure 3.2). Electrons and muons (10^6 of each with different initial energies) were launched through the chamber, and the average energy depositions along the Z axis was calculated. Figure 4.3 shows the average energy deposition per one electron of each initial energy as a function of penetration depth. The highest amount of energy at a particular depth is deposited by 5 MeV electron. This is because the electron completely imparts its energy in the water. A 100 MeV electron deposits an equal amount of energy around 0.2 MeV with its depth and left the water volume. Figure 4.4 shows the average energy deposition per muon at each initial energies along its depth. The muons with 5 MeV and 15 MeV initial energy have imparted all of their energy to the water cube, have stopped inside and decayed while the muon with 100 MeV initial energy deposits an equal amount of energy around 0.2 MeV with depth and then exits the water volume. If the depth of the water chamber is changed, then the energy deposition will also vary. If the depth is large enough, then the 100 MeV electron will also stop and decay inside the volume. This plot is sometimes important in theraputic studies. For example, if a cancer cell needs to be killed, then the position of the cancer cell can be tracked and radiation can be sent to deposit more energy at that position.

Figures 4.5 and 4.6 show the total energy deposition by each electron and muon vs their initial energy, respectively. The curve of electron energy deposition is flat after 30 MeV. This reveals that the electrons with an initial energy less than 30 MeV stop inside the chamber, while the more energetic electrons exit the chamber after losing some

of their energy. Using Figure 4.3, one can calculate the energy deposited by the 100 MeV electron. As this electron deposits 0.2 MeV throughout 100 mm, it deposits a total of 20 MeV of energy, which represents the flat part of the curve in Figure 4.5. However, the curve for total energy deposition by a muon (Figure 4.6) has a different shape. It shows maximum energy deposition for muons in the range of $30 - 35$ MeV. The reason for a 100 MeV muon to deposit a lower of its energy is that it does not decay inside the chamber, while the 35 MeV muon does decay inside the chamber.

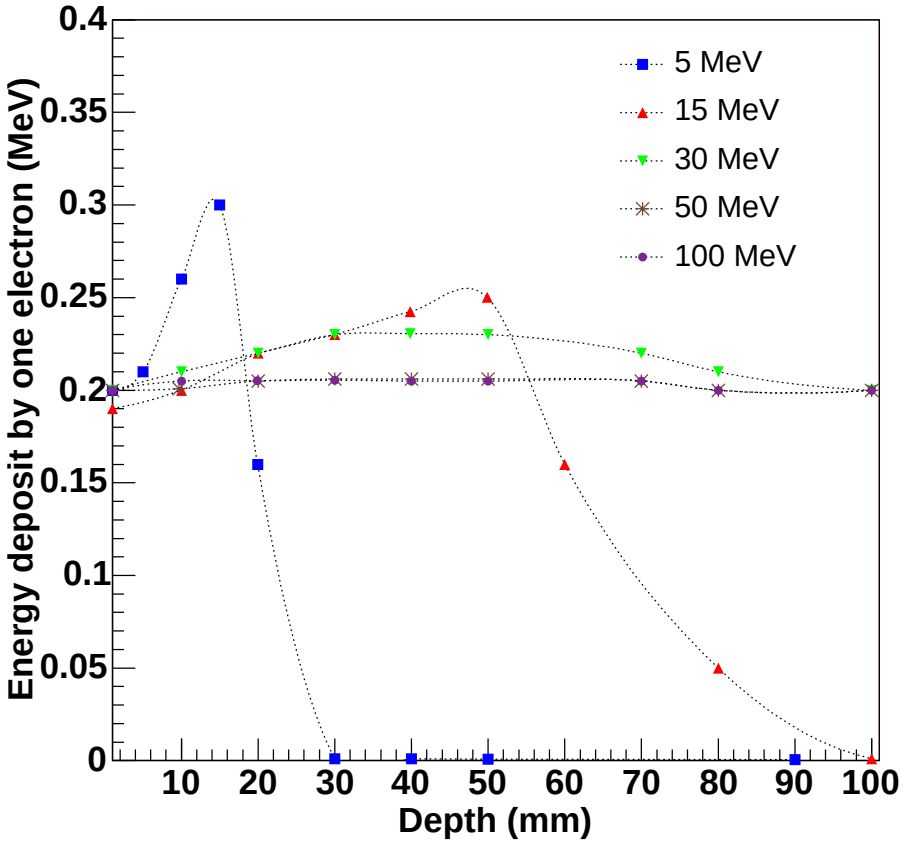

Figure 4.3: Energy Deposition of Electrons with Different Initial Energies as a Function of Penetration Depth.

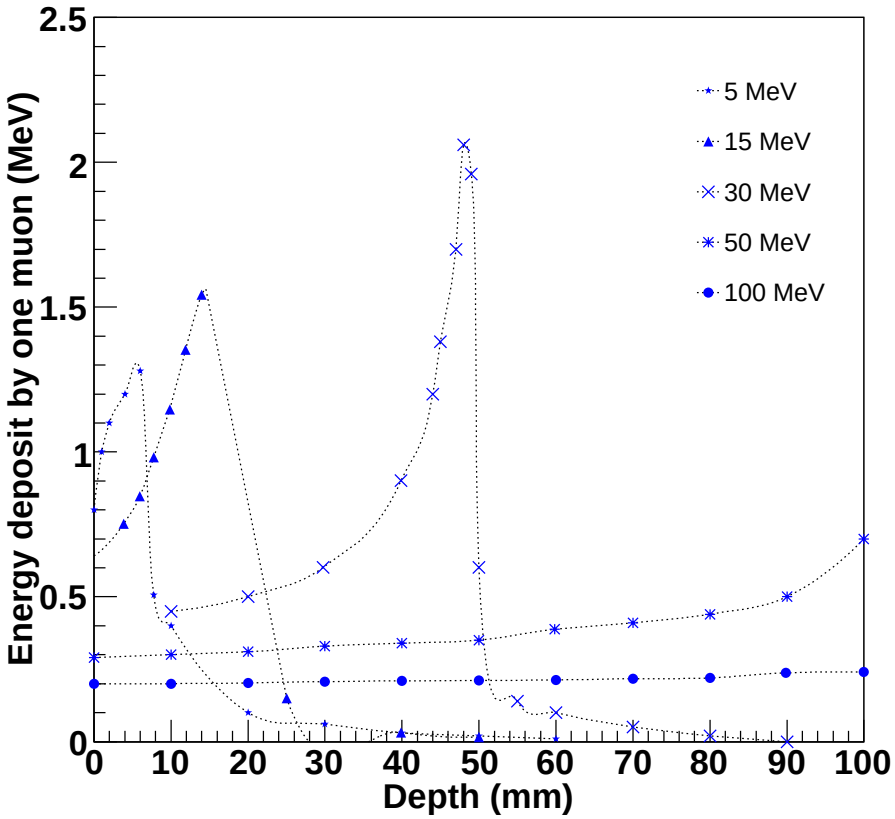

Figure 4.4: Energy Deposition of Muons with Different Initial Energies as a Function of Penetration.

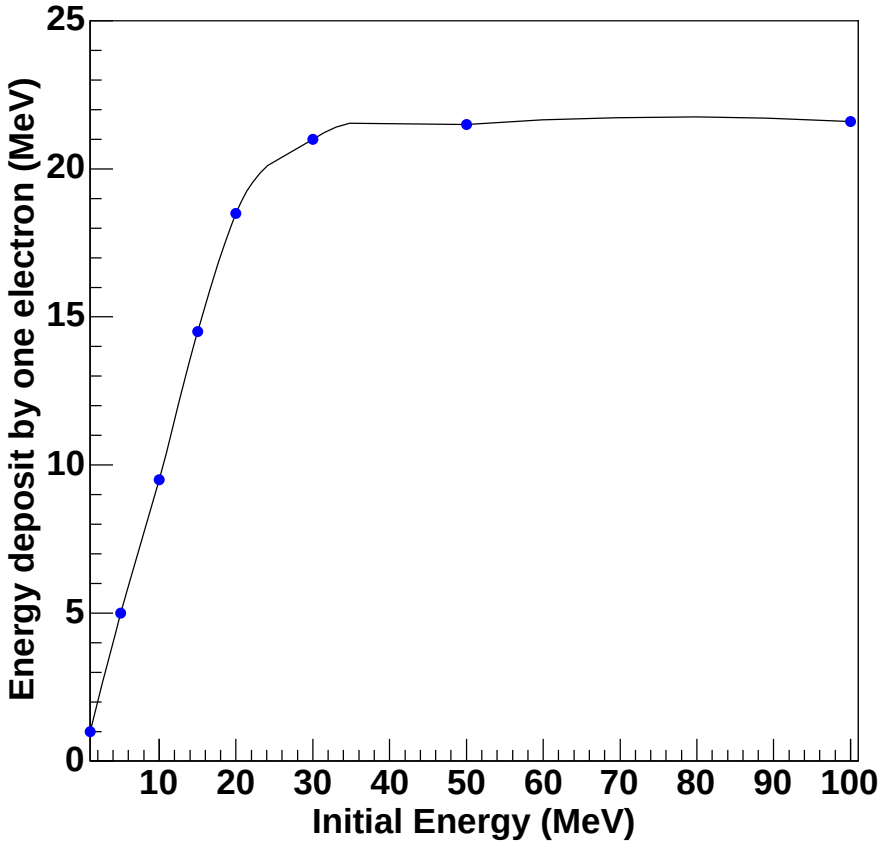

Figure 4.5: Total Energy Deposition in the Chamber by Electrons with Different Initial Energies. The curve of electron energy deposition is flat after 30 MeV. This reveals that the electrons with an initial energy less than 30 MeV stop inside the chamber, while the higher energetic electrons exit the chamber after imparting some of their energies.

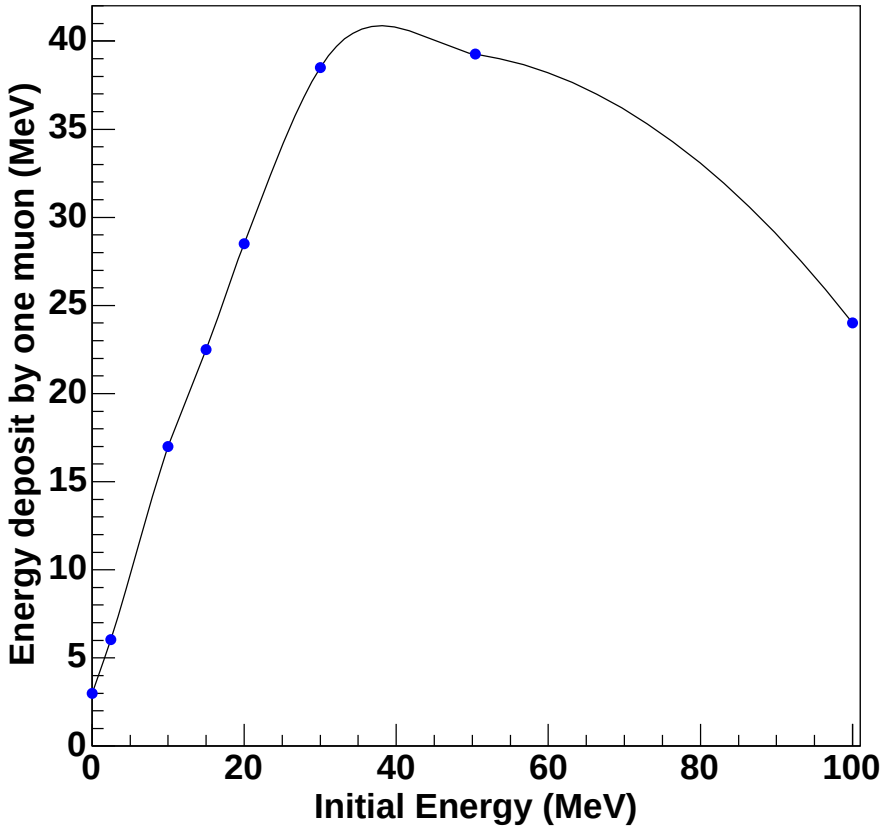

Figure 4.6: Total Energy Deposition in the Chamber by the Muons with Different Initial Energies. It shows a maximum energy deposition for muons in the range of $30 - 35\ MeV$.

4.3 Simulation Validation

This section presents the results of an inter-comparison study of the simulation model[37] with experimental work performed at the Pacific Northwest National Laboratory (PNNL) and event-by-event Monte Carlo simulations conducted at Washington State University. (WSU)[38] As the experimental results of particle interactions in liquid water are unavailable, the initial work has been done to characterize the spatial variation of the energy deposited by slowing and stopping the electrons in air.[39]

An air cube of $10 \times 10 \times 10\ mm^3$, divided into 1 mm slices along the z direction, was irradiated by mono-energetic electrons at the energies of 30, 50, and 80 keV, respectively. The density of the air was set to 1.29 mg/cm^3. Figure 4.7 shows the spread of a 30 keV electron beam through air. The output of the simulation contains the x, y, and z coordinates of energy deposition points, the amount of energy deposited in each tracking volume and the secondary particle name. Figure 4.8 shows the radial distribution of the energy deposition points from the z axis. A Gaussian fit was used to obtain FWHM (Full Width at Half Maximum) values of the scattered electron beam. Experimental data and WSU simulation results are extracted from figures 3, 4 and 5 referenced in.[38] These data, together with the Geant4 simulation results, are discussed in the following section.

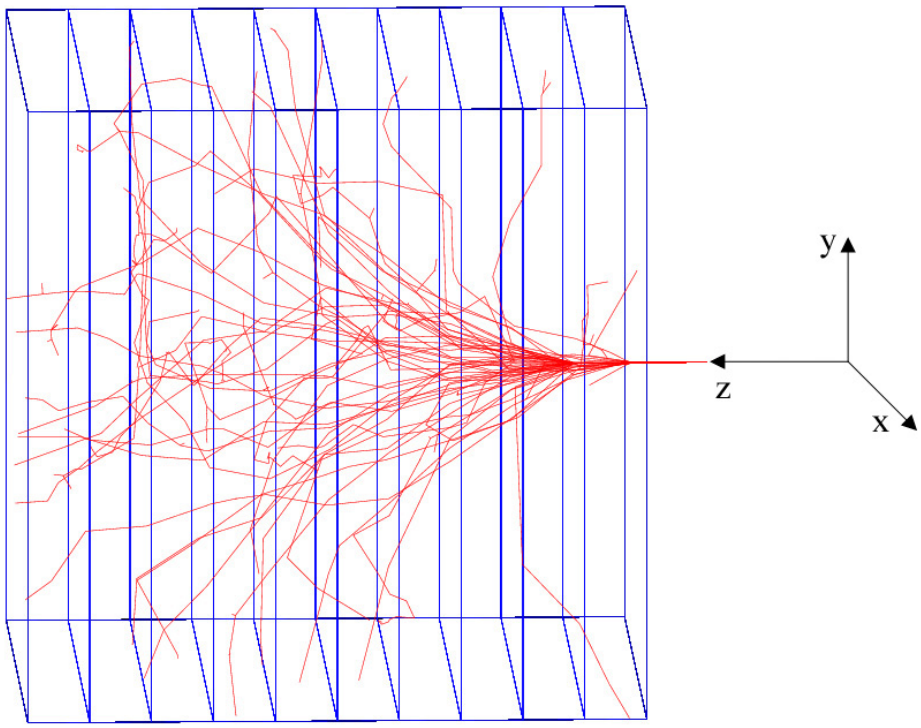

Figure 4.7: Illustration of the Simulation Setup and the Event Display of 50 Electron Tracks at 30 keV Through Air from Geant4 Visualization Toolkit.

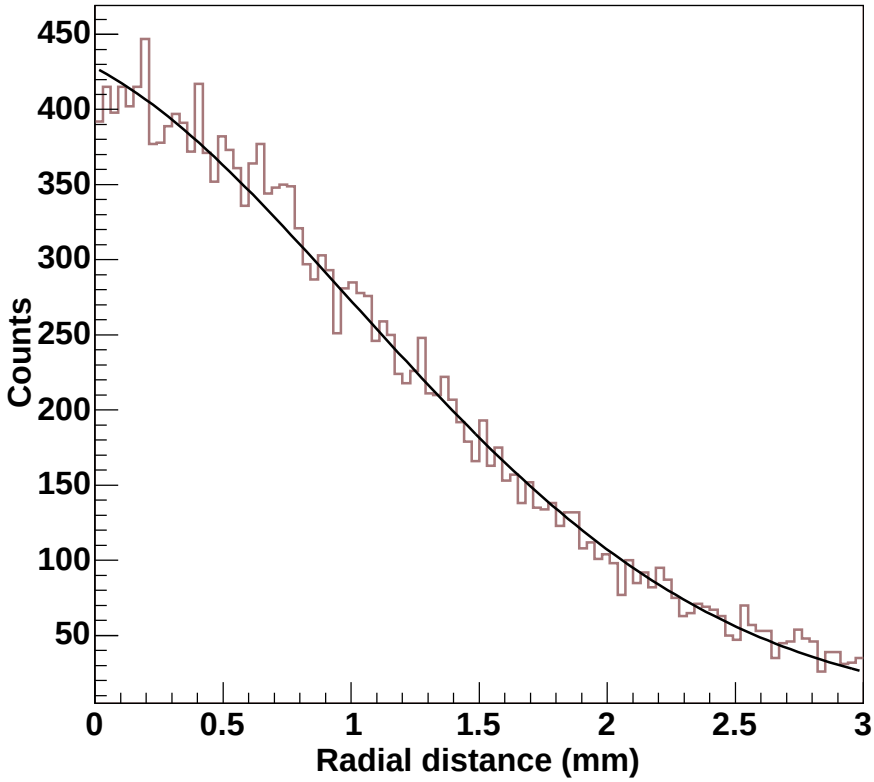

Figure 4.8: The Histogram of the Radial Distribution of the Energy Deposition for 80 keV Electrons. The Solid Curve is from a Gaussian fit. This Gaussian fit is used to calculate FWHM values that are used to plot figures 4.9, 4.10, and 4.11.

Figures 4.9, 4.10, and 4.11 show a comparison of the Geant4 simulation results with PNNL experimental data and WSU simulation results of FWHM values as a function of the penetration distances for electron energies of 30, 50 and 80 keV, respectively. At all energies, FWHM values increase with z and are consistent with the track structures as shown in Figure 4.7. Geant4 results show a good agreement with the experimental data as well as the WSU simulation results. The simulation results are slightly better matched with experimental data than the WSU simulation results. The reason may be that in the WSU method, water vapor is used as the medium, while both the experiment and the Geant4 simulation have been performed in air. At higher penetrations, the experimental values are closer to Geant4 results than at shallow penetrations. Simulation statistical errors do not show up in the plot because they are smaller than the marker size. At very low penetration, both the WSU and the Geant4 simulation results are lower than the experimental data. This discrepancy with the experimental results could be due to the outscatter of the incident electrons in the experiment.[40] Figure 6 shows the average energy depositions for electrons through the air at 30, 50, and 80 keV, respectively, with the depth of the air cube. Electrons with an initial energy of 30 keV deposit more energy than electrons with 50 keV and 80 keV initial energies. This result is in agreement with the stopping power values of electrons in air as shown in Table 4.1.[41] These results are also consistent with the energy deposition model for electrons in air given by reference.[42]

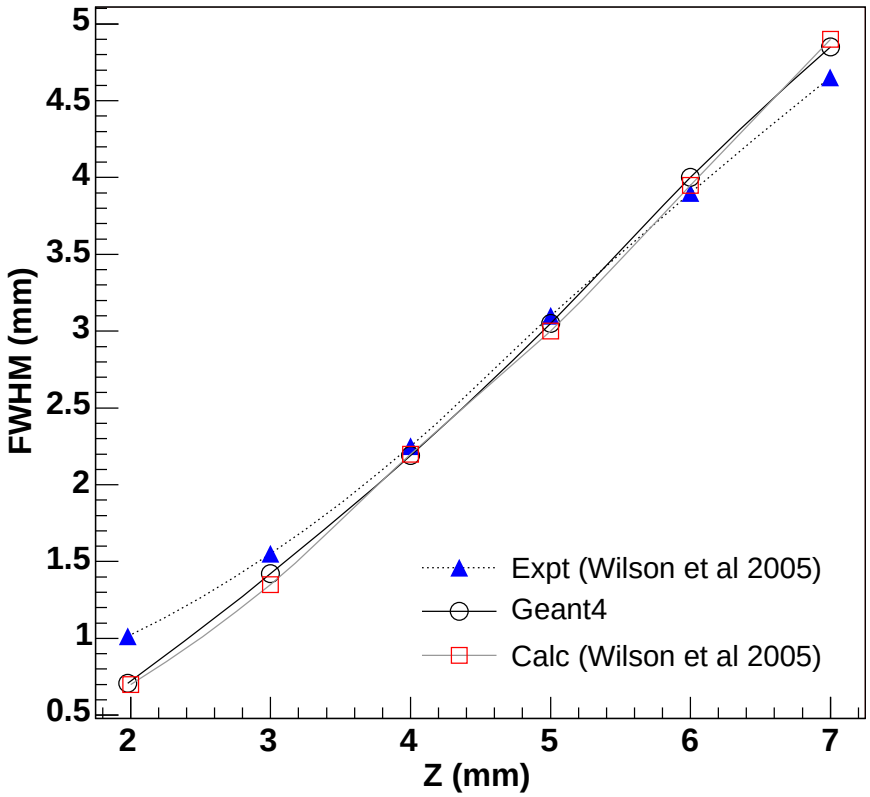

Figure 4.9: Comparison of the FWHM Values Obtained from Geant4 Simulation with Experimental and Simulation Results for 30 keV electrons. The results from Geant4 are shown in the solid curve.

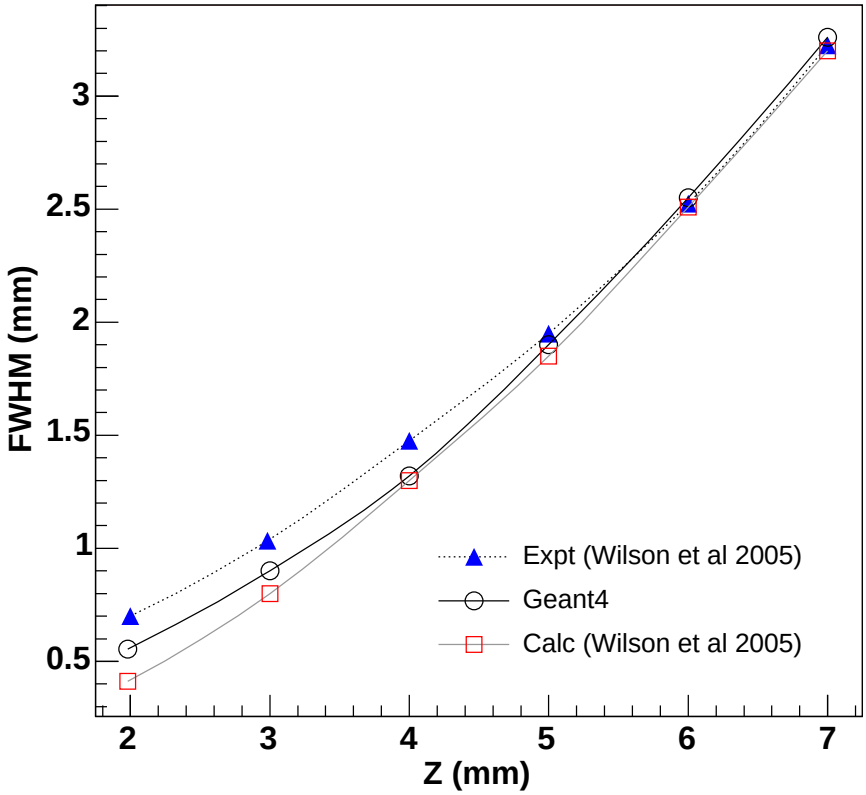

Figure 4.10: Comparison of the FWHM Values Obtained from the Geant4 Simulation with Experimental and Simulation Results for 50 keV Electrons. The results from Geant4 are shown in the solid curve.

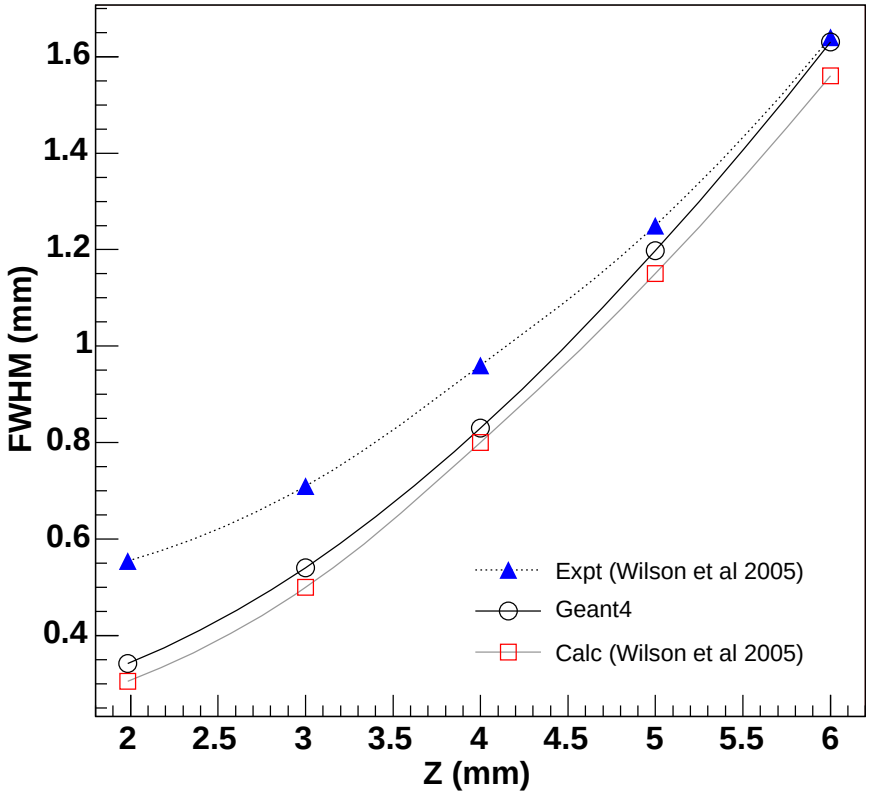

Figure 4.11: Comparison of the FWHM Values Obtained by the Geant4 Simulation with Experimental and Simulation Results for 80 keV electrons. The results from Geant4 are shown in the solid curve.

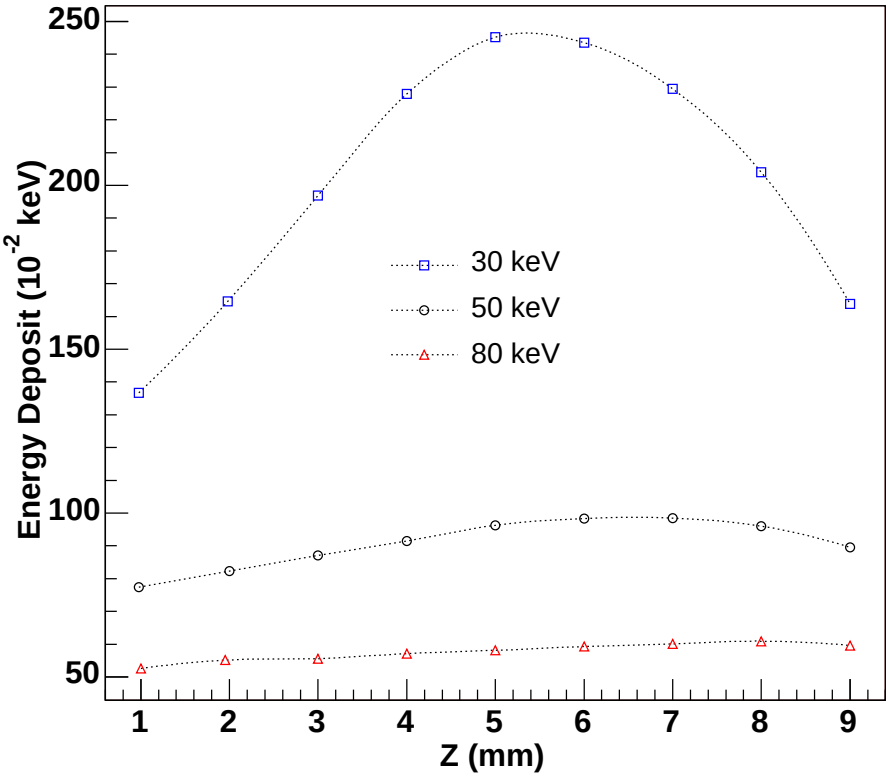

Figure 4.12: Energy Deposition of 30, 50 and 80 keV Electrons in the $10 \times 10 \times 10$ mm^3 Air Cube. This graph shows the total energy deposited by one electron on average as a function of its penetration depth.

Table 4.1: Stopping Powers of Electrons in Air.

Electron energy(keV)	Geant4($MeV\ \frac{cm^2}{g}$)	ICRU($MeV\ \frac{cm^2}{g}$)
30	8.78	8.69
50	6.02	5.82
80	4.26	4.20

Comparison of the Geant4 Results with Bethe-Bloch Formula Calculations

In the earlier section, the results from the Geant4 simulation model were compared with experimental data and the WSU simulation results. In this section, calculations have been done to compare the results with the Bethe-Bloch formula (Equation 1.6). The stopping powers of muons in the range of 0-100 GeV were calculated using the simulation results for water and concrete. These results were compared with the calculations of the Bethe-Bloch formula.

Calculation of Mean Excitation Energy

The atomic number (Z=$\sum Z_i f_i$) and the mass number (A =$\sum A_i f_i$) of concrete were determined by Equations **??** and **??**. The fraction of each element (f_i) in concrete is given in Table 4.2. The Bethe-Bloch formula was used to determine the mean excitation energy for concrete.[41] The calculated value for the mean excitation energy of concrete (135 eV) is consistent with the available literature.[43] Figures 4.13 and 4.14 show the comparison of the Bethe Bloch calculation and the Geant4 simulation results for the stoping power in concrete and water, respectively, of muons in the range of 0 $MeV - 100\ GeV$. Geant4 results are in very good agreement with the Bethe calculations for both concrete and water.

Figure 4.15 is a plot of the stopping power of muons in different media.[33] The dips in both Figure 4.13 and Figure 4.14 can also be seen in this figure for muons of about $2 - 4\ GeV$.

$$\ln I = [\sum \omega_j (Z_j/A_j) \ln I_j]/ < Z/A > \tag{4.1}$$

Table 4.2: Composition of Different Elements in Concrete.

Element	Fraction of element(f_i)
Ca	0.06
Si	0.325
O	0.529
Na	0.01
Fe	0.04
Al	0.033
H	0.01
C	0.001
Mg	0.002
K	0.013

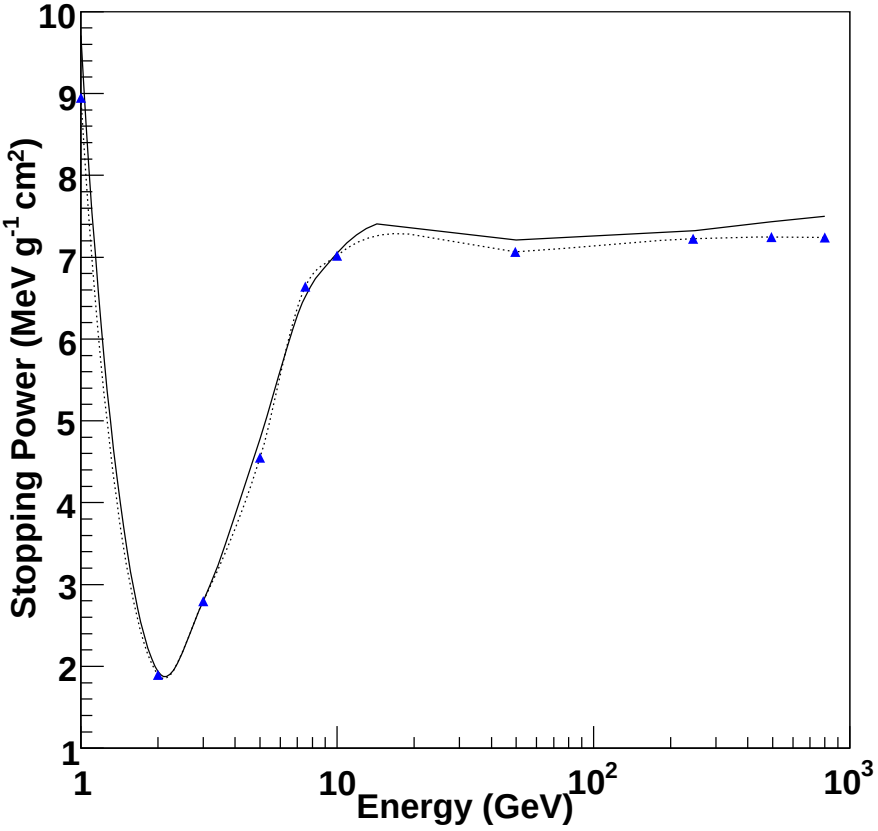

Figure 4.13: Bethe-Bloch Calculation (solid line) and Geant4 Simulation Results for the Mean Energy Loss Rate in Concrete by Muons with energies ranging from 0 MeV to 100 GeV.

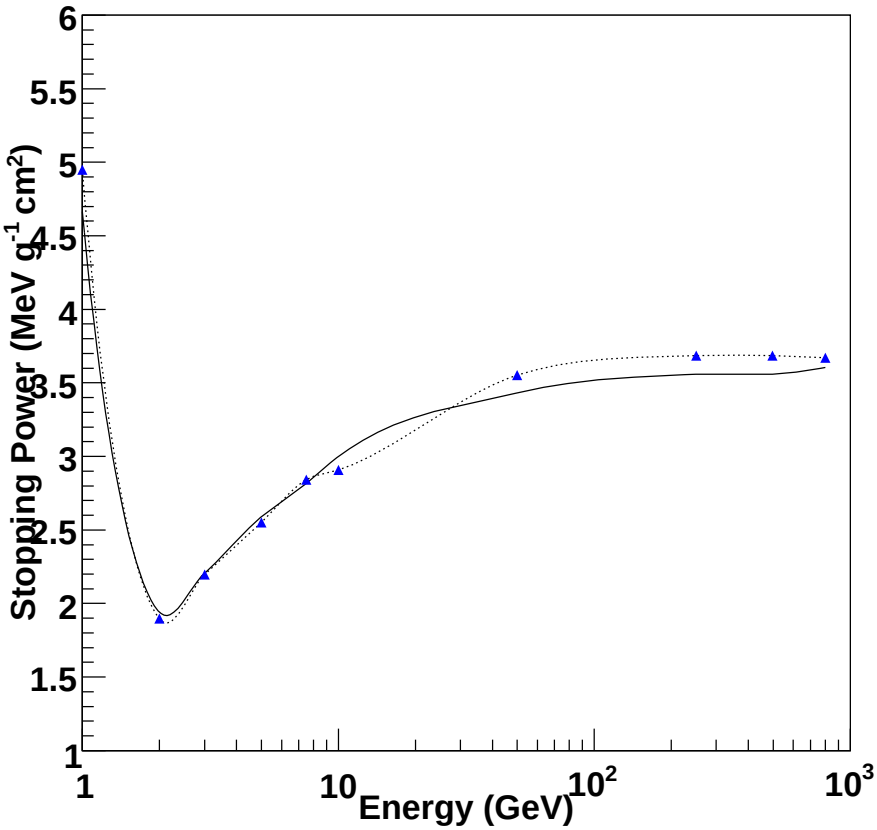

Figure 4.14: Bethe-Bloch Calculation (solid line) and the Geant4 Simulation Results for the Mean Energy Loss Rate in Water by Muons with energies ranging from 0 MeV to 100 GeV.

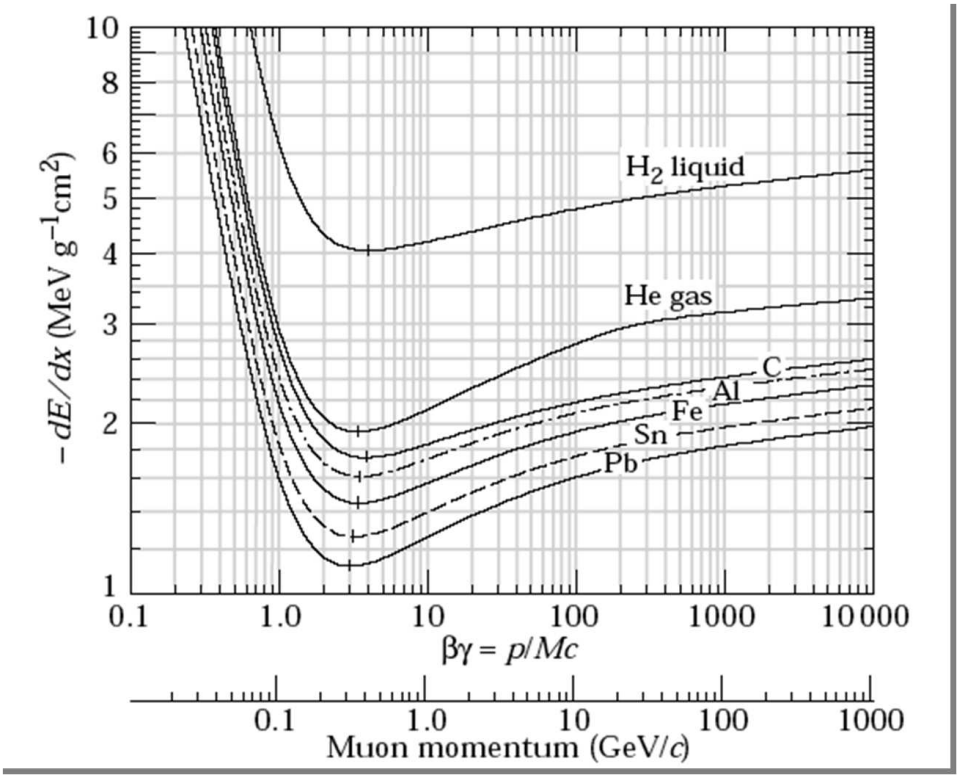

Figure 4.15: Mean Energy Loss Rate for Muons in Liquid Hydrogen, Gaseous Helium, Carbon, Aluminum, Iron, Tin, and Lead.

4.4 Dose in the Human-Sized Water Chamber

A rectangular cube of the size of an average human body ($40 \times 60 \times 170\ cm^3$) was simulated. Muons were launched through this volume, and the interactions were studied. Figure 4.16 shows one example of the tracks of two 5 GeV muons through this cube . Both of these higher energetic muons exit from the other side of the chamber after creating several gamma rays (green-colored tracks) and low-energy electrons (small red dots) by interactions. As shown in Figure 4.17, the sea-level flux was introduced in to the chamber, where the yearly dose was calculated by equation 4.2,

$$Dose = \frac{dE}{\rho dx}. \tag{4.2}$$

This simulation assumes that one muon particle hits one square centimeter per second.[44] The obtained value for the total dose per year was 30.4 millirems. It is approximately equal to the total sea-level dose for an average human body according to the literature.[44]

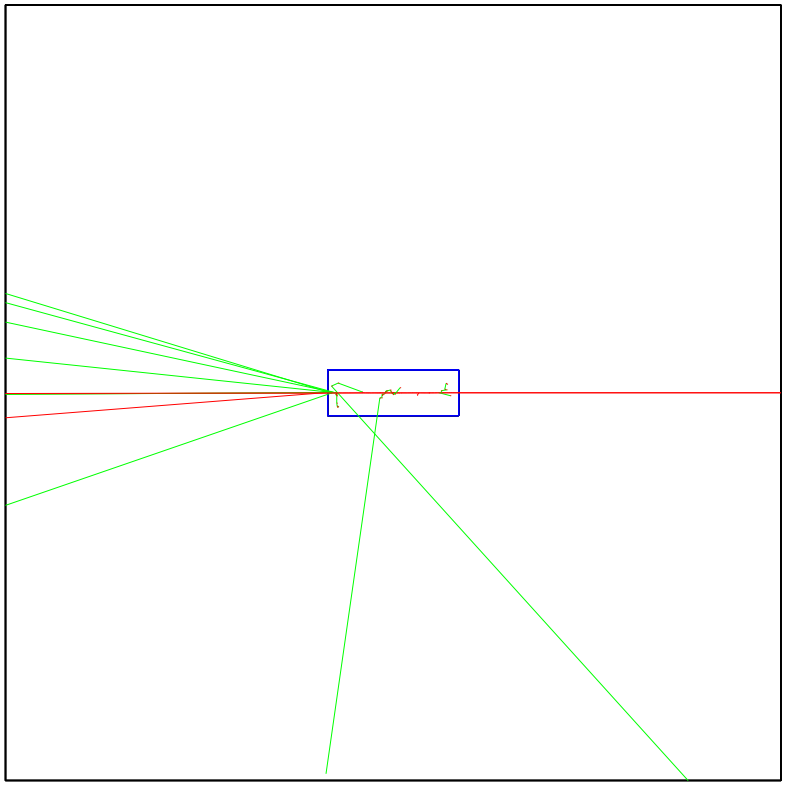

Figure 4.16: Tracks of two Muons (5 GeV) Through a Human-Sized Water Chamber. Both of these highly energetic muons exit from the other side of the chamber after creating several gamma rays (green-colored tracks) and low-energy electrons (small red dots).

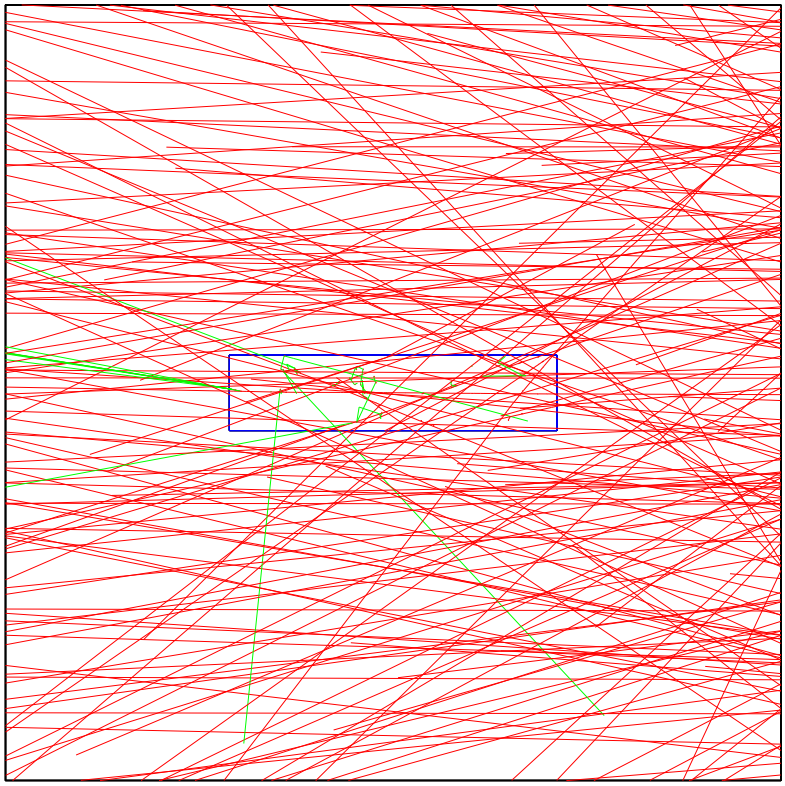

Figure 4.17: The Hits Distribution in Rectangular Cube of the Size of an Average-sized Human. The muons with different initial energies are from different angles. The tracks of the gamma rays created by interactions inside the chamber (green) are clearly visible.

Figure 4.17 shows the hits distribution in the rectangular cube of the size of an average human. The muons with different initial energies come from different angles as described in Chapter 3. The tracks of the gamma rays created by interactions inside the chamber (green) are clearly visible. It is impossible to distinguish the short electron tracks.

4.5 Concrete Shielding

In order to study the effect of shielding, the water chamber was shielded by a concrete slab. The composition of the concrete described in Section 4.3 in this chapter. The total energy deposition in the chamber was calculated for different thicknesses of the concrete slab by different groups of mono-energetic muons with different initial energies. Figures 4.18 and 4.19 are two examples of the hits distributions in the 2-meter concrete shield, and the 1-meter cube water volume respectively for muons with 1 GeV and 80 GeV initial energies. Looking closer at the particle tracks in concrete reveals a huge number of short electron tracks. More tracks are left by 80 GeV muons. These muons have even created a positive Na ion track (blue). The muons have slowed down by imparting most of their energies to the concrete. As a result, almost all of the 1 GeV muons decay inside the water. If the concrete was absent, then those muons will penetrate the water volume without passing most of their energy into water. This concept will be clearly verified later in this chapter. Figure 4.20 represents the energy deposited in the 1-meter water volume by different groups of mono-energetic muons. Muons with higher initial energy ($E > 10$ GeV) penetrate the volume without decaying inside the chamber, while the other muons ($E < 10$ GeV) lose most of their energy inside the chamber. Once a 4-meter concrete slab was

inserted into the system as shown in Figure 4.16, most of the particles slow down creating more short electron tracks inside the volume (Figure 4.21). The sea-level muon flux was applied to the human-sized chamber of the size of a human with a 2-meter concrete slab present. The simulated dose in the chamber was about 40 millirems, which is higher than the simulated dose without shielding. When the thickness of the concrete was gradually changed, the dose in the chamber was also changed. This demonstrated a maximum value (42 millirems) of about 1 meter in thickness. Then the dose started decreasing rapidly. At about 8 meters, most of the muons are blocked by the concrete. Figure 4.22 shows the yearly dose given to the water volume with the presence of a concrete shield of different thicknesses. Thus, the dose in a human body was higher when it was shielded by 1 or 2 meters of concrete than when it was exposed to the muons while walking on the street.

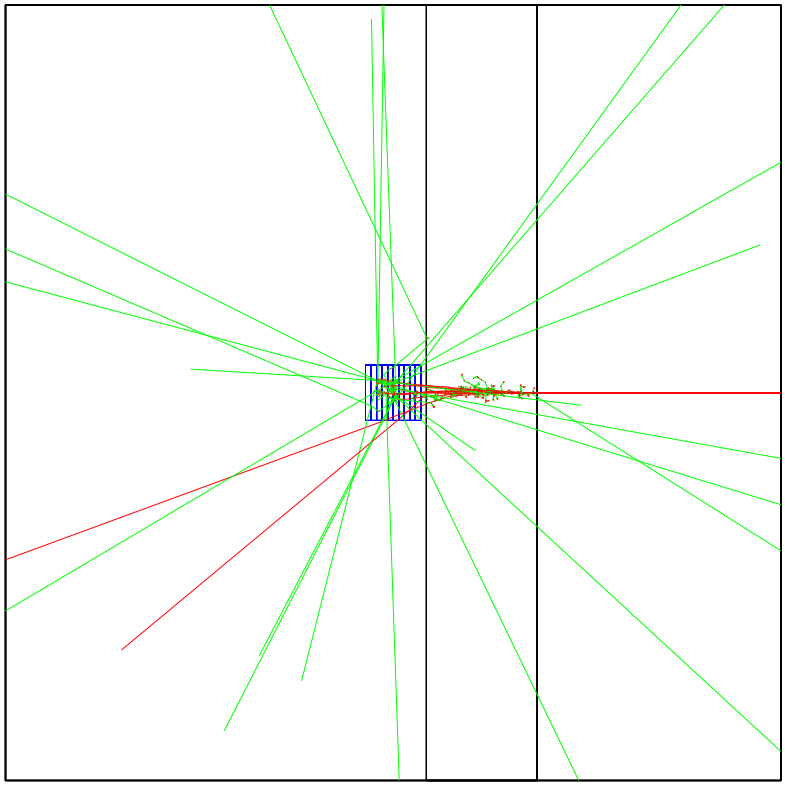

Figure 4.18: Tracks of 1 GeV Muons in 2 Meter Concrete Slab and 1 Meter Water Cube. Almost all the muons have been slowed down by the concrete. As a result, almost all the muons decay inside the water.

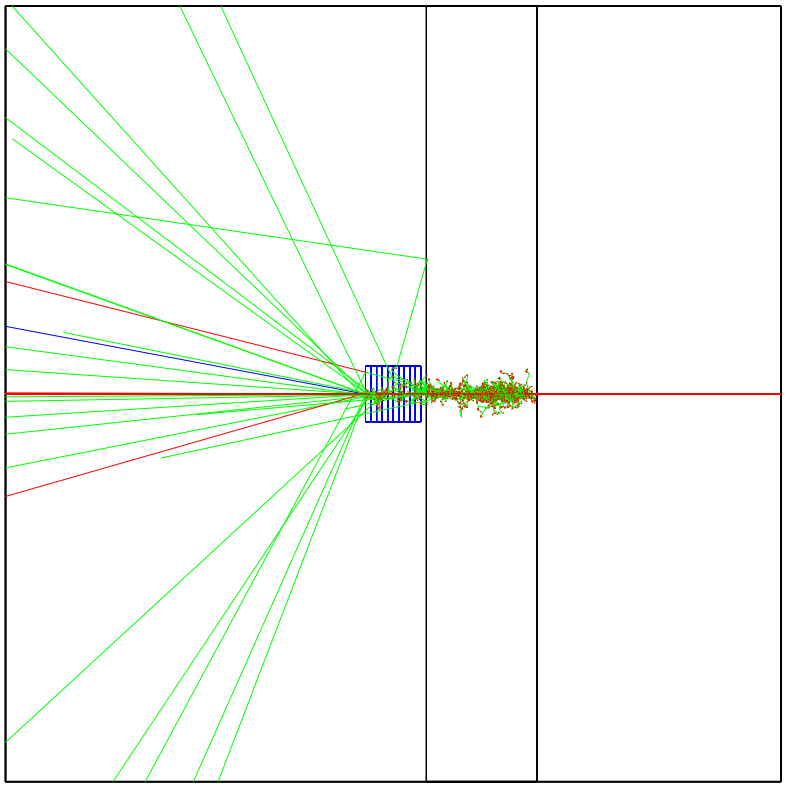

Figure 4.19: Tracks of 80 GeV Muons Through a 1 Meter Water Cube and 2 Meter Concrete Slab.

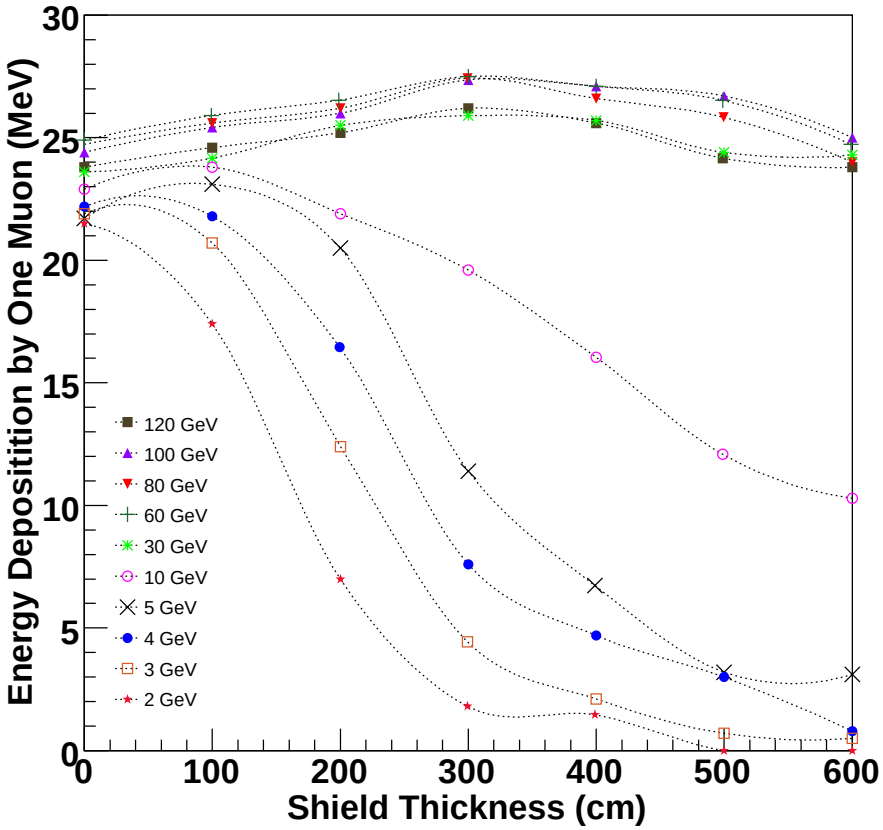

Figure 4.20: Average Energy Deposition in the 1 Cubic Meter Water Volume by Muons with Different Energies for Different Thicknesses of the Concrete. Muons with higher initial energies ($E > 10$ GeV) penetrate the volume without decaying inside the chamber. The other muons ($E < 10$ GeV) deposit most of their energy inside the chamber.

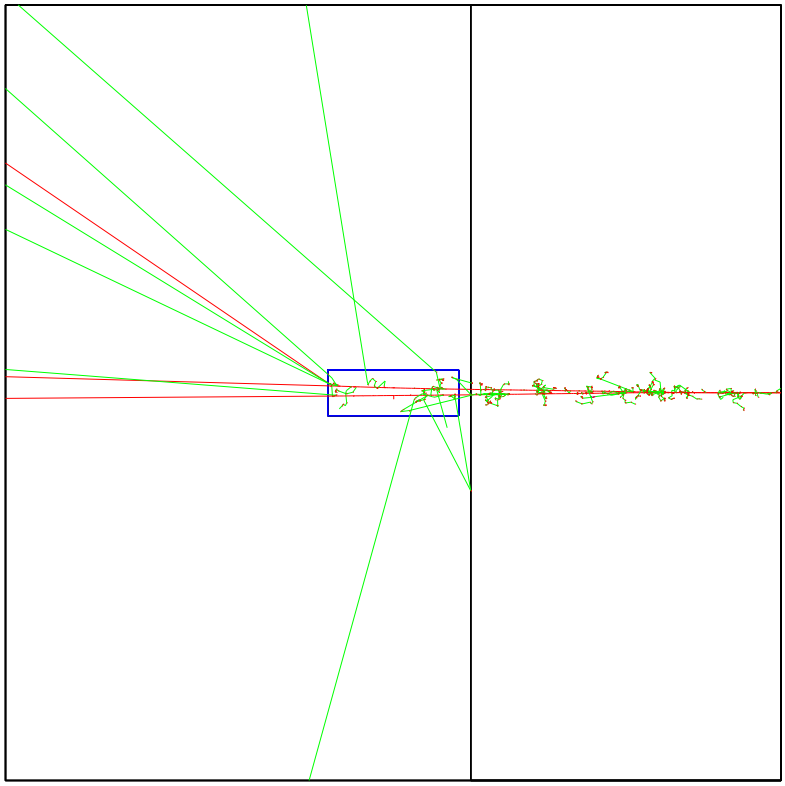

Figure 4.21: Tracks of Two 5 GeV Muons Through a 4 Meter Concrete Slab and Human-Sized Water Chamber. Compared with Figure 4.16, most of the muon particles have been slowed down creating more short electron tracks inside the volume.

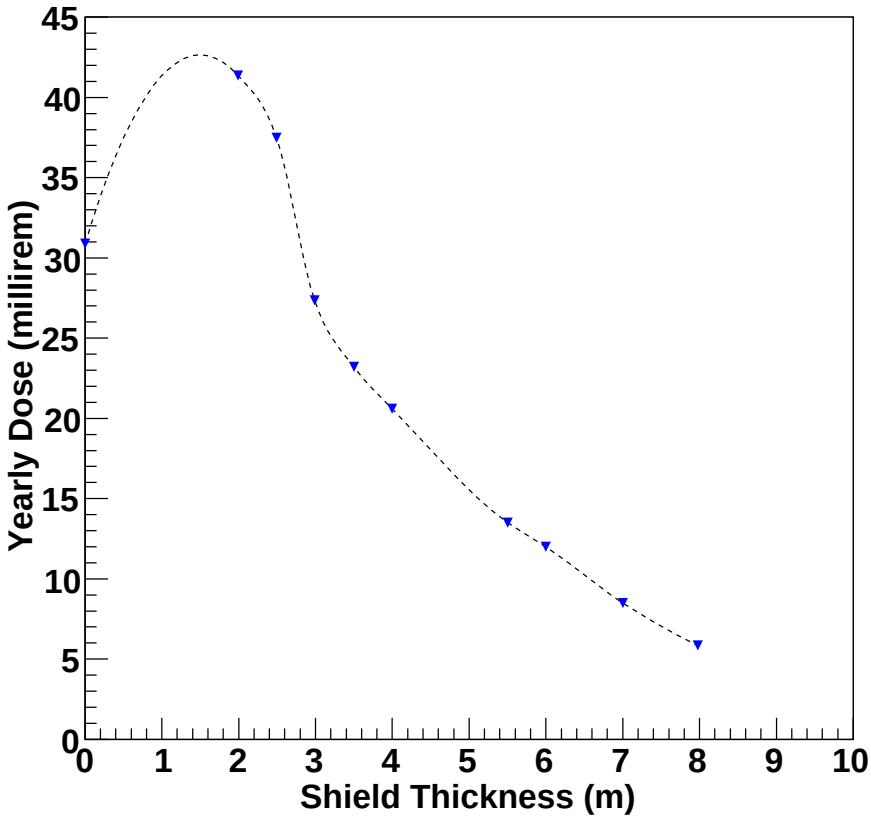

Figure 4.22: The Yearly Dose Given to the Water Volume with the Presence of a Concrete Shield of Different Thicknesses. The curve reaches a maximum value (42 millirems) around 1 meter of thickness. At about 8 meters, most of the muons are blocked by the concrete.

4.6 Human body

The simple human body phantom created by Geant4 includes the head, skull, neck, shoulder, hands, legs, heart, lungs, stomach, liver, and bones. Each part has been modeled by bringing in the appropriate composition as described in Table 3.2. Figure 4.23 shows the muon particles passing through the body. These muons come straight above the head and most of them decay at about the stomach area. Most of the electron tracks are seen just below the waist. Thus the vertical muons could be most harmful to the embryo of a pregnant woman. Figure 4.24 the muon tracks in the human body phantom when the flux is present. The probability that a muon will stop inside the body is very rare because most sea-level muons are higher energetic particles. On the other hand, if a low-energy muon decays inside the body, then it imparts a high amount of energy to the created electron because the muon carries a huge mass energy. The simulated total yearly dose of radiation in the body phantom was 22 millirems. Once the concrete shielding was inserted, the dose increased. The highest dose of about 33.5 millirems, was simulated with the shielding of $1 - 2$ meters. After about 2 meters, it started decreasing (Figure 4.25). Table 4.3 gives the total simulated yearly dose in different body parts. It concludes that most of the dose is given to the torso.

Table 4.3: Average Yearly Dose Received in Each Part of the Body from Sea-Level Cosmic Ray Muons.

Element	Fraction
Head	1.5
Neck	0.7
Torso	9.4
Left hand	1.0
Right hand	1.1
Left leg	1.9
Right leg	2.0
Total bones	4.4
Total body	22.0

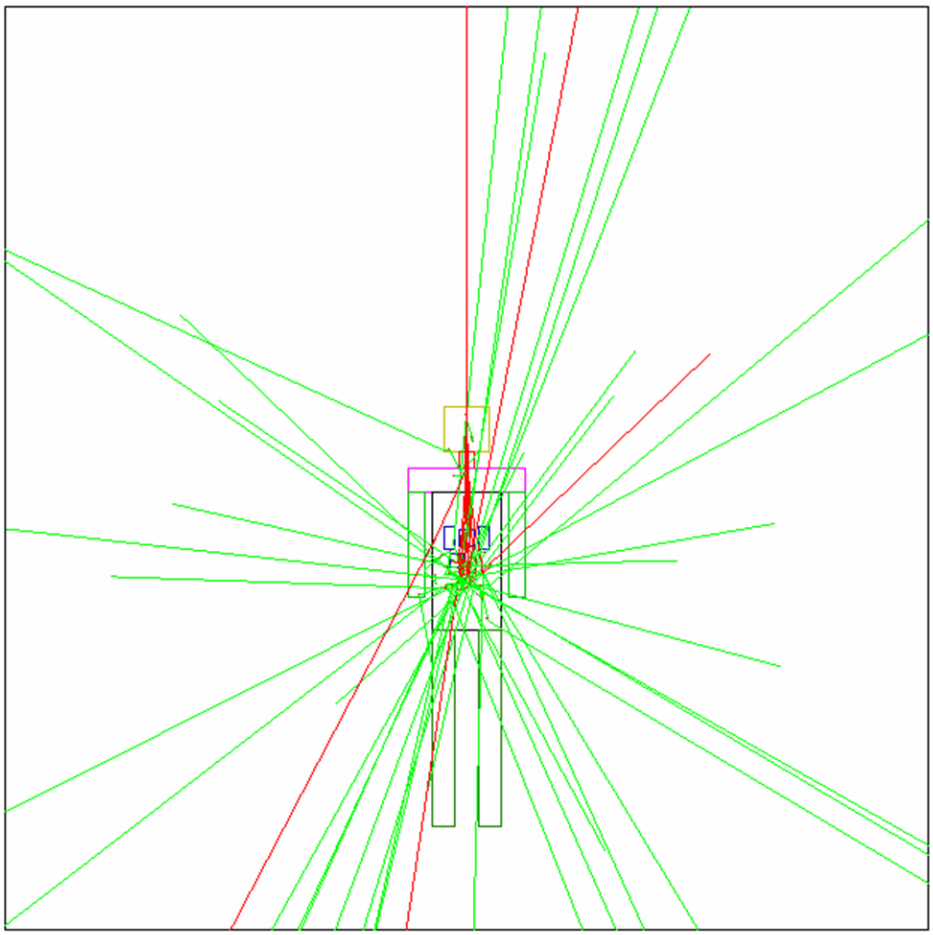

Figure 4.23: Low-Energy Muon Tracks in the Human Body. The muons come straight above the head and most of them decay at about the stomach area. Most of the electron tracks are seen just below the waist.

Figure 4.24: Human Body Phantom exposed to Sea-level Muon Flux for one minute.

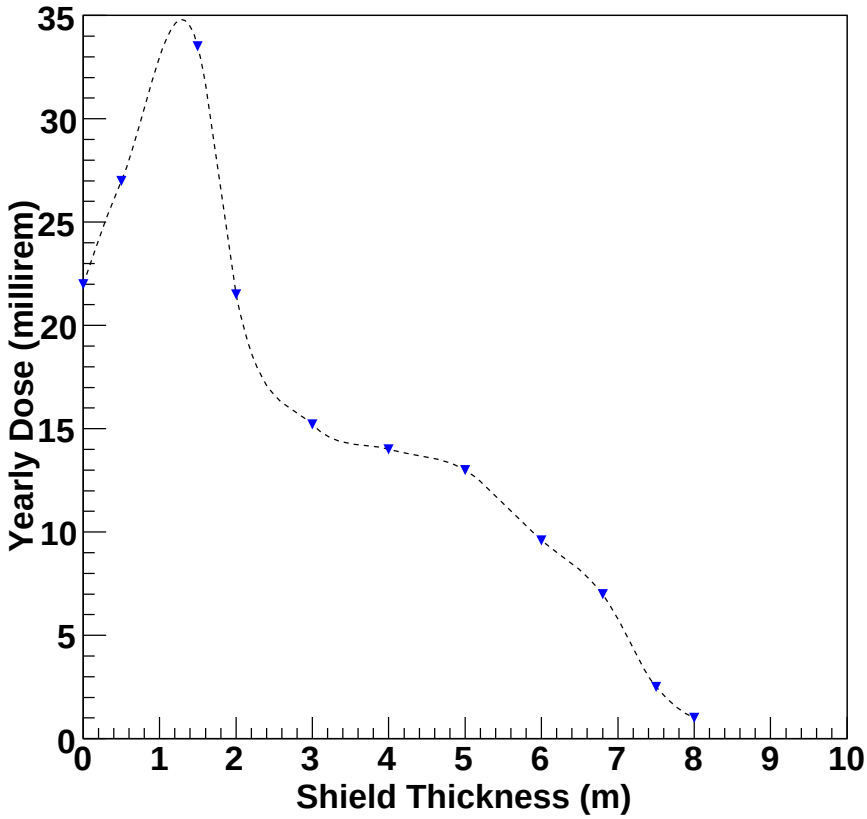

Figure 4.25: Yearly Dose in a Human Body for Different Concrete shieldings. The highest dose of about 33.5 millirems occurs with shielding of 1 to 2 meters.

4.7 Secondary Electrons Created by Muons

Secondary electrons formed in biological tissues by high-energy particles significantly contribute to the fragmentation of small molecules and to single and double strand breaks in DNA.[47] The yield of double strand breaks induced by electrons in the DNA shows a maximum at energies of a few hundred electron volts, as the fast electrons tend to segment the DNA molecule.[48] This is the energy range of most electrons created inside the body by muon decay.[49] This section presents an extensive study conducted about the secondary electrons created by muons. As discussed in Chapter 1, most of the cosmic muons pass completely through the body. Throughout this process, these muons release considerable numbers of low-energy electrons into the body. A small portion of the muons going through the body decay inside and release very energetic electrons. These electrons create a large number of secondary and higher generations of electrons undergoing processes such as ionization, bremsstrahlung, and photoionization from gamma rays. In order to have an understanding of the effects of these particles on nanometric structures such as DNA, extensive knowledge about the distribution of these electrons must be gained. This section addresses questions such as how many secondary and higher generations of electrons were created by muon decay at sea level, and what are the properties (initial energy spectrum, penetration range) of these electrons. The current Geant4 implementation of low-energy processes is only valid for energies down to $250eV$ (and can be used up to approximately $100eV$). Therefore the number of created electrons below this cutoff cannot be determined by Geant4. The total number of electrons was, therefore, calculated partially by applying an approximation. The probability of generating an electron (p_n) of each initial energy

was obtained by the Geant4 simulation. Figure 4.26 represents the Geant4 results of the probability of creating electrons by each energetic electron. The number of further possible electrons by each of these electrons (N) was calculated using the available literature. [18] Figure 4.27 shows the number of possible electrons created by primary electrons with different energies. Finally, $P_n \times N$ was calculated for each electron energy (Figure 4.28). These combined results are in fairly good agreement with the predicted results. [20, 50] The penetration lengths of the low-energy electrons produced in liquid water by muons are shown in Figure 4.29. The energy of the electrons is indicated on the X axis. The solid (black) curve represents the Geant4 results, the stars (blue) represent the experimental data.[51] and the triangles (red) represent the ICRU data.[41] Geant4 results show quite good agreement with the experimental as well as ICRU data. Figure 4.30 shows the track of one secondary electron created by a decaying a 10 MeV muon. Its initial energy is 128 keV, and the track length is about 0.1 mm. A small dot represents one step. It deposits an average energy of about 3 keV in each step and imparts all of its energy to water at point $(11.07, 7.13)$. Figure 4.31 shows the track of another secondary electron created by a 100 keV muon that is created by ionization. Its initial energy is 146 eV, and the track length is about 10 nm. It deposits an average energy of about 3 eV in each step and imparts all of its energy to water at point $(-27.2, -30)$.

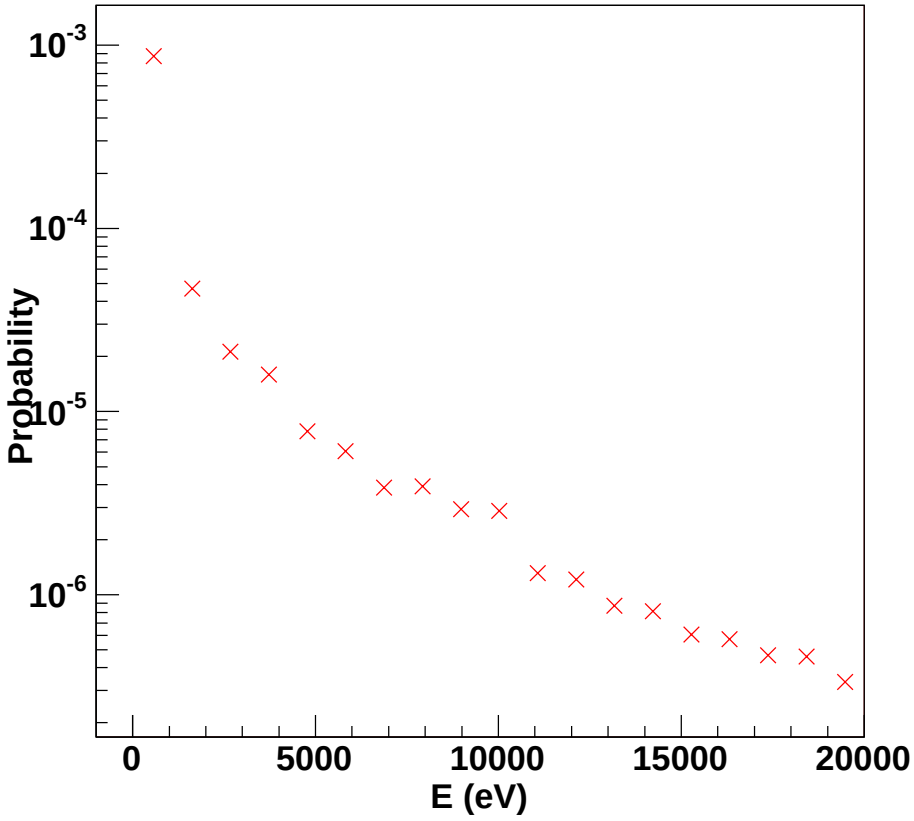

Figure 4.26: The Probability of Generating an Electron (p_n) by Secondaries Created by a 10 MeV Muon. This probability has been calculated by simulating 10^7 muons at 10 MeV.

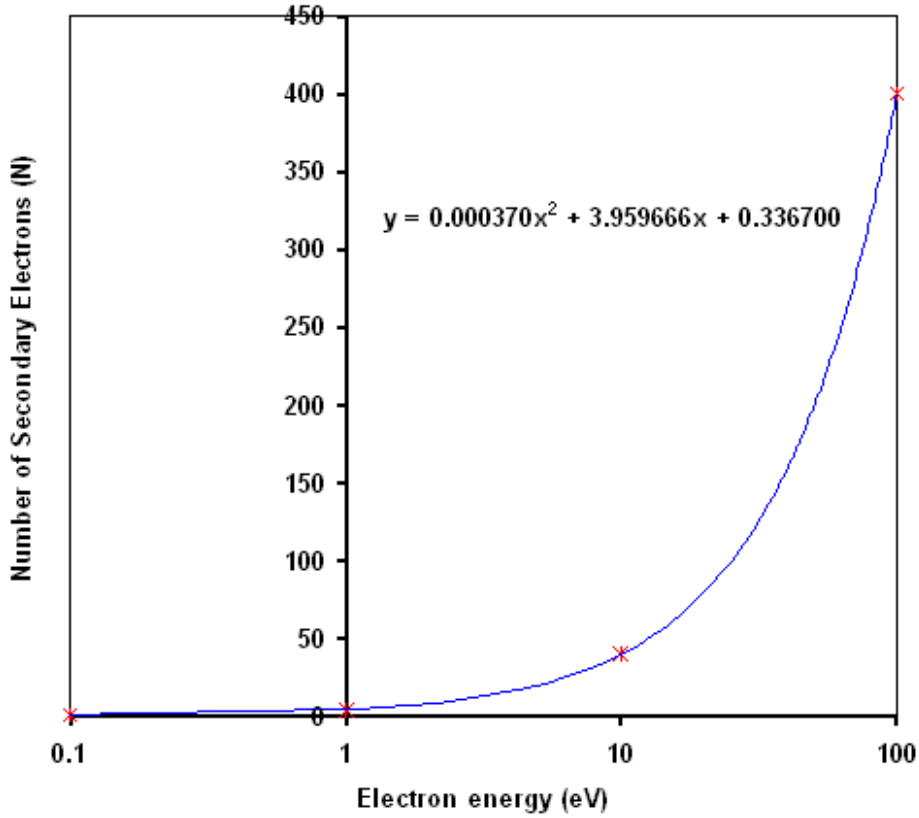

Figure 4.27: Number of Electrons (N) Created by Electrons with Different energies.

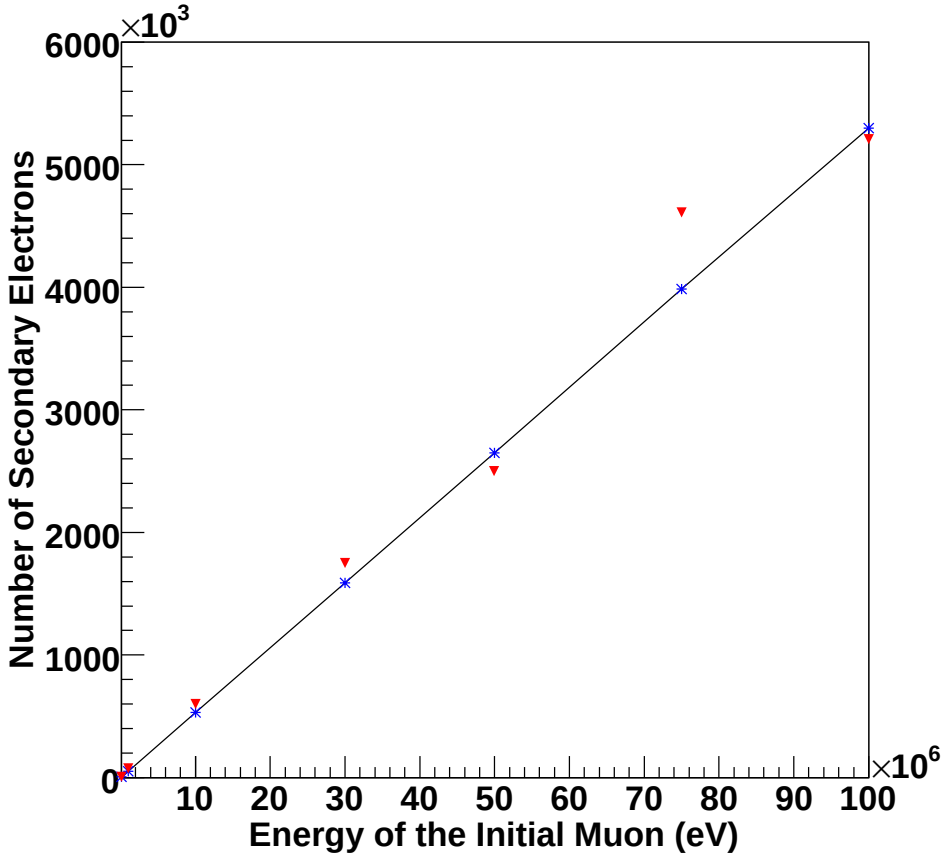

Figure 4.28: Total Electrons Generated by Muons with Different Energies in Water. The solid curve represents the theoretical value. The simulation results are shown by triangles.

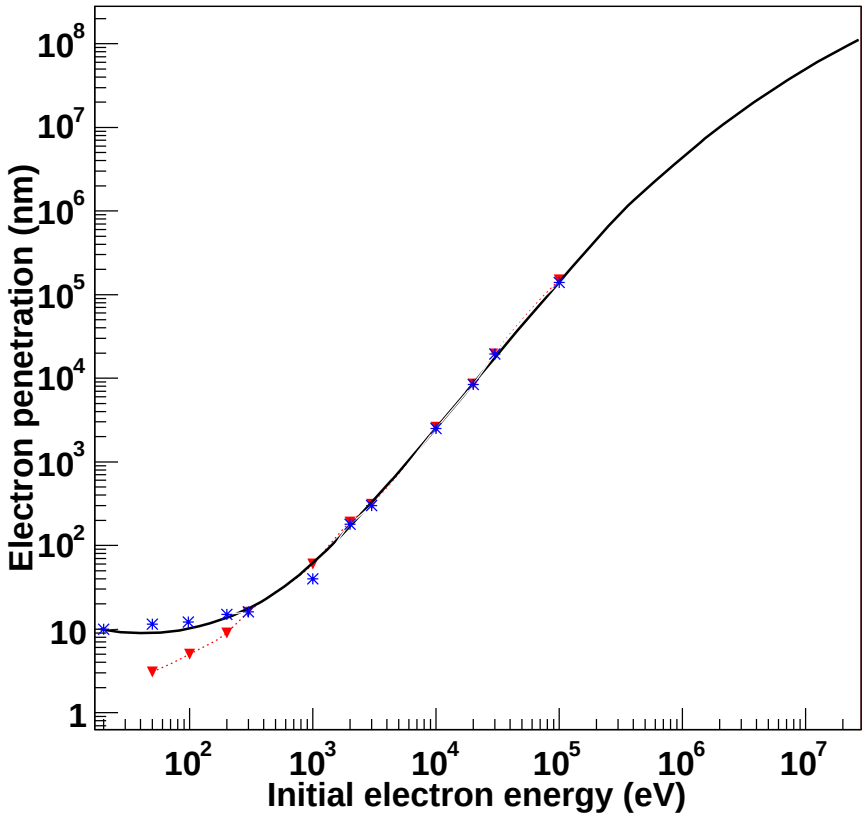

Figure 4.29: The Penetration Lengths of the Low-Energy Electrons Produced in Liquid Water by Muons. The solid curve represents the Geant4 results. The stars represent the experimental data and the triangles represent the ICRU data.

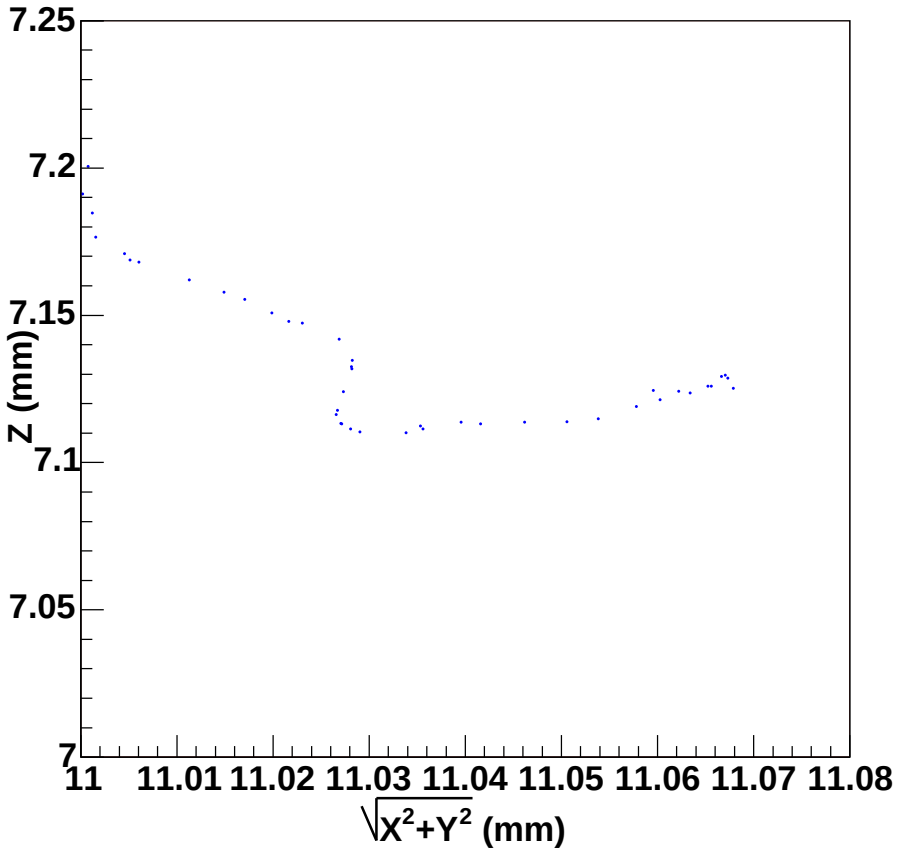

Figure 4.30: Track of a Secondary Electron Created by a 10 MeV Muon. The initial energy of the electron is 128 keV, and the track length is about 0.1 mm. A small dot represents one step. It deposits an average energy of about 3 keV in each step and imparts all of its energy to water.

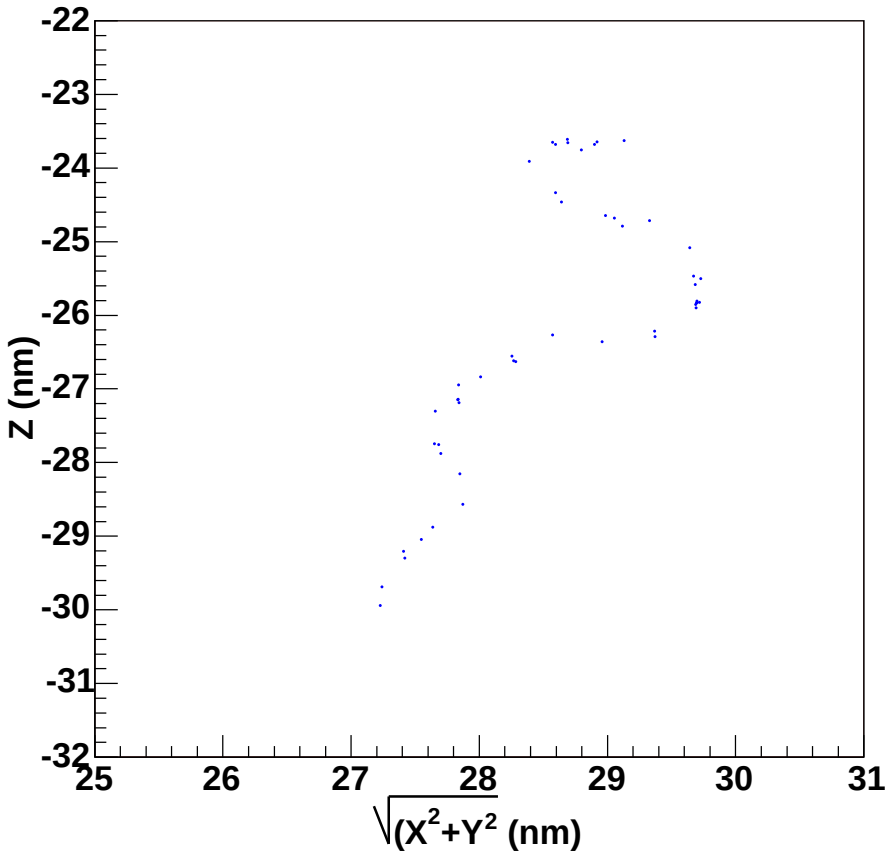

Figure 4.31: Track of a Secondary Electron Created by a 100 keV Muon. The initial energy of the electron is 146 eV, and the track length is about 10 nm. It deposits an average energy of about 3 eV in each step and imparts all of its energy to water.

Chapter 5

Conclusion and Future work

The use of simulations has enormous advantages in radiation protection dosimetry. Geant4 is an appropriate toolkit to simulate radiation track structures and energy deposition because it eases adding variant complex physics models. In this research a simulation model, which is a human body phantom was designed and developed using Geant4 to study the interactions of sea-level cosmic ray muons and their secondary particles in tissue-like materials. The physics processes that the particles undergo such as ionization, bremsstrahlung, Compton scattering, and decay have been explicitly implemented in the model. The simulation model can be used to study the behavior of any particle in any medium. A validation study has been performed to test the applicability of the simulation model. The results have been compared with available data in literature, and the reliability of the predictions has been assessed.

Energy deposition of muons and secondary electrons in a volume of water was studied because 80 percent of the human body consists of water.[29] Tracks created by muons

and electrons were studied in detail. Muons and electrons generated many short electron tracks in water by ionization. To analyze the particle interactions in small structures, with the intention of studying the energy localization in nanometric structures such as DNA, the chamber was sliced in 3-D. Energy deposited by slow muons (30 MeV) in the small water volume was found to be higher than the energy deposited by fast muons (100 MeV). The dose simulated per year in the human-sized water volume was approximately 30 millirems. This is close to the accepted value for the yearly dose from cosmic radiation at sea-level.

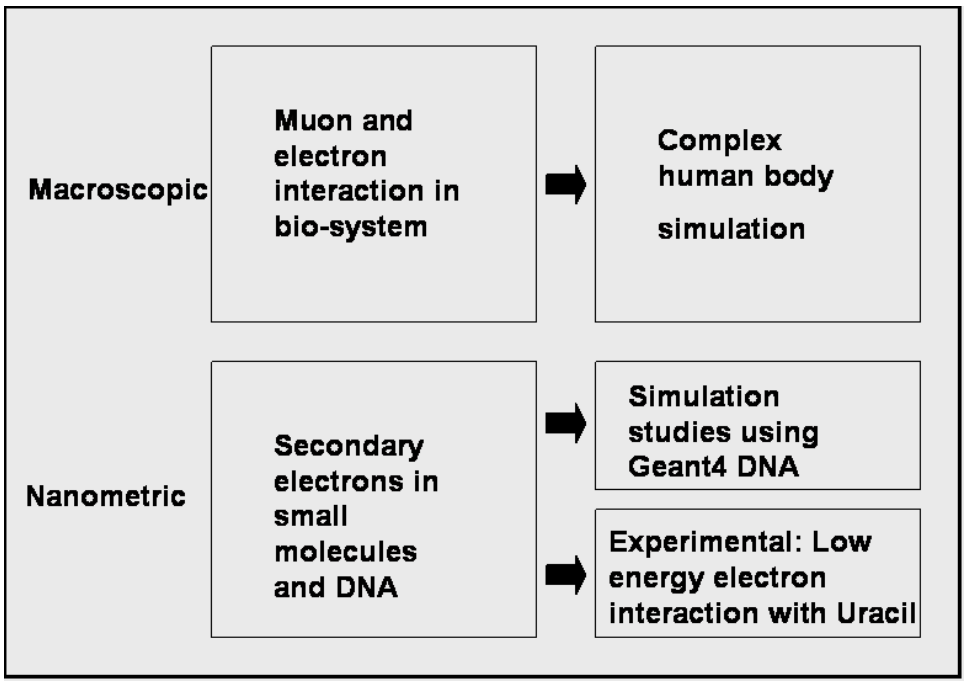

Figure 5.1: The Work to be Continued in the Future. Upper : The macroscopic work focuses on developing a more complex human model. Lower: The nanometric work aims for two tasks.

The yearly dose obtained in the human body phantom was 22 millirems. Shielding cosmic muons with a concrete slab ($0-2$ meters) increase the dose imparted to an average sized human.[53] When the thickness of the slab is increased, the dose is decreased after two meters. Therefore, the thickness of a shield is very important as it determines the dose to the human body, especially for people who work in the basements. The results presented in the thesis work reveal that the secondary electrons created by muons contribute a considerable amount to the dose in the human body. The electrons created by muon decay are mostly in the range of few hundred electron volts. These electrons contribute considerably to DNA double strand breaks.[52] Also, they create a large number of low-energy electrons, which contribute to the fragmentation of small molecules as well as DNA strand breaks. Therefore it is vital to have an accurate knowledge about the interactions of these fast- and low-energy secondary electrons created by muons. The new Geant4 DNA tools will be used to do extensive simulations of electron interactions in nanometric structures of tissue-like materials. This new Geant4 DNA expands the capability of simulating electrons of energies from the MeV range down to the final energy of the secondary electrons of 10 eV.[52] It treats the transport of secondary electrons beyond the Born approximation by newly implemented interaction processes for the low-energy electron collisions. The experimental study which will begin to observe strand breaks of Uracil by low-energy electrons. The summary of the work done and the future research intentions are shown in the Figure 5.1.

Bibliography

[1] W M Yao et. al., Journal of Physics G 33, 1 (2006)

[2] B Grosswendt Formation of ionization clusters in nanometric structures of propane-based tissue-equivalent gas or liquid water by electrons and alpha particles *Radiat. Environ. Biophys.* **41** 103-112 (2002)

[3] Geant4 Simulation Toolkit *http : //geant4.web.cern.ch/geant4/*

[4] J E Turner, H G Paretzke, R N Hamm, H A Wright and R H Ritchie Comparative study of electron energy deposition and yields in water in the liquid and vapor phases *Radiation Research* **92** 47-60 (1982)

[5] H Nikjoo and D T Goodhead Track structure analysis illustrating the prominent role of low-energy electrons in radiobiological effects of low-LET radiations *Phys.Med.Biol.* **36** 229-238 (1991)

[6] V A Semenenko, J E Turner and T B Borak NOREC, a monte carlo code for simulating electron tracks *Radiat. Environ. Biophys.* **42** 213-217 (2003)

[7] R D Stewart, W E Wilson, J C McDonald and J Storm Microdosimetric properties of

ionizing electrons in water: a test of the PENELOPE code system *Phys.Med.Biol.* **47** 79-88 (2001)

[8] J Agostinelli et al Geant4 - A simulation toolkit *Nuclear Instruments and Methods* **A 506** 250-303 (2006)

[9] Radiation Protection $http://trshare.triumf.ca/\ safety/EHS/rpt/rpt_3/node1.html$

[10] T Henriksen and H D Maillie Radiation and Health *Taylor and Francis*

[11] Project SUSTAIN $http://192.107.108.56/portfolios/c/conklin_k/sustainwebquest/radlessonplansgeneral4stules1.htm$

[12] R Frischknecht Human health damages due to ionising radiation in life cycle impact assessment *Environmental Impact Assessment Review20*159-189 (2000)

[13] Stanford Linear Accelerator Center $http://www2.slac.stanford.edu/vvc/cosmicrays/cratmos.html$

[14] R Bellotti et. al. *Phys. Rev. D60*052002 (1999)

[15] R Bellotti et. al. *Phys. Rev. D53*32 (1996)

[16] M Boezio et. al. *Phys. Rev. D62*032001 (2000)

[17] S Coutu et. al. *Phys. Rev. D62*032001 (1999)

[18] H G Paretzke *Radiation track structure theory. In: Kinetics of Nonhomogeneous Processes. Freeman GR Ed. New York. John Wiley and Sons 89-170*(1987)

[19] B Grosswendt Track structure simulation: A basic tool for molecular radiation biology and nanodosimetry *Computing radiation dosimetry* 161-178 workshop proceedings,

Sacavm, Portugal, 22 - 23 June 2002, (2004), Paris : OECD Publications. ISBN 92-64-10823-8

[20] Miller, J.H., Wilson, W.E., Lynch, D.J., Sowa Resat, M., and Trease, H.E. Computational dosimetry for electron microbeams: Monte Carlo track simulation combined with confocal microscopy. Extended abstract. Radiation Research 156(4):*438-439* (2001)

[21] International Commission on Radiation Units and Measurements (ICRU) *Radiation Dosimetry: Electron Beams with Energies Between 1 and 50 MeV* Report 35 International Commission on Radiation Units and Measurements,Bethesda, MD (1984)

[22] J F Ziegler The stopping of energetic light ions in elemental matter *J.Appl. Phys/ Rev. Appl. Phys***85** 1249-1272 (1999)

[23] R Y Rubinstein Simulation and the Monte Carlo Method *John Wiley and Sons, Inc.* New York, NY, USA (1981)

[24] S. Agostinelli G4-A toolkit *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 506* 250-303 (2003)

[25] J Allison et al Geant4 developments and applications *IEEE Transactions on Nuclear Science* **53** 270-278 (2006)

[26] Geant4 user manuel for application developers *http* : *//geant4.web.cern.ch/geant4/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/index.html*

[27] V.M.Grichine *Physics Letters B525* 225-239 (2002)

[28] K Murakami et. al Developemtn of an interface for using EGS4 physics rocesses in Geant4 *Computing in High Energy and Nuclear Phtsics* La Jolla, Ca, USA. (2003)

[29] S B Heymsfield et. al. Chemical determination of human body density in vivo: relevance to hydrodensitometry *American Journal of Clinical Nutrition 50* 1282-1289

[30] Pushpa Wijesinghe and Xiaochun He Geant4 simulation of charged particle interaction with water *International Workshop on Radiation Therepy* Perdue University, IN (2005)

[31] Pushpa Wijesinghe and Xiaochun He Geant4 simulations of muon tracks and energy deposition in tissue like materials *ASA 2006 Radiation Conference* Monterey, CA (2006)

[32] J Kremer et al *Phys. Rev. Lett* **83** 241 (1999)

[33] S Eidelman et. al. Passage of particles through matter *Phy. Lett. B 592* 1 (2004)

[34] L. N. Bogdanova et. al. Cosmic muon flux at shallow depths underground *Physics of Atomic Nuclei 69* 1293-1298 (2006)

[35] W R Lutz and L M Chin Design and characteristics of a 4MeV total body irradiator *Dosimetry in Radiotherapy* 211-220 (1988)

[36] Pushpa Wijesinghe and Xiaochun He Simulation Study of Sea-Level Cosmic Ray Radiation in a Human Body Phantom and Shielding Effects Submitted to International Commission of Radiation Research San Francisco , CA (2007)

[37] Pushpa Wijesinghe and Xiaochun He Simulations of low energy interaction: Validation study of LCDCS model Submitted to Physics in Medicine and Biology.

[38] D J Lynch, W E Wilson, M T Batdorf, M B Sowa Resat, G A Kimmel and J H Miller

Monte carlo simulation of the spatial distribution of energy deposition for an electron microbeam *Radiation Research* **163** 468-472 (2005)

[39] W R Lutz and L M Chin Design and characteristics of a 4MeV total body irradiator *Dosimetry in Radiotherapy* 211-220 (1988)

[40] W E Wilson, D J Lynch, K Wei and L A Barby Microdosimetry of a 25 keV electron microbeam *Radiation Research* **155** 89-94 (2001)

[41] International Commission on Radiation Units and Measurements (ICRU) *Stopping power for electrons and positrons* Report 37 International Commission on Radiation Units and Measurements,Bethesda, MD (1984)

[42] A M Roldan, A Williart, J M Perez and G Garcia Energy Deposition model for electrons in air at incident energies from 0.1 to 100 keV *J.Appl. Phys* **95** 5865 (2004)

[43] National Institute of Standards and Technology $http://www.nist.gov$

[44] J Kremer et al *Phys. Rev. Lett* **83** 241 (1999)

[45] http://www2.slac.stanford.edu/vvc/cosmicrays/cratmos.html

[46] Geant4 User Guide for application developers $http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/index.html$

[47] B Suliki et al Fast electrons from accelerating multiple electron scattering by ion impact: contribution to radiation damages Radiation Damage in Biomolecular Systems Groningen, June 6th-9th Institute of Nuclear Research of the Hung. acad. of Sciences, H-4001, Debrecen, Hungary Department of Atomic Physics, Stockholm University, 10405 Sweden. (2006)

[48] W Friedland P Jacob H G Paretzke T Stork 1998 Radiat. Res. 150 170

[49] Pushpa Wijesinghe and Xiaochun He Georgia Graduate Student Interdesciplinary Conference University of Georgia april 01 (2006)

[50] V Cobut et. al. Monte carlo simulation of fast electron and proton tracks in liquid water: Physical and physiochemical aspects *Radiat. Phys. Chem. 51* 229-243 (1997)

[51] M Michaud, A Wen and L Sanche Elastic and Inelastic cross sections for low-energy electron scattering in amorphous ice *Radiation Research 159* 3-22 (2003)

[52] Maria Grazia Pia Simulation of low energy electron tracks in media of radiological interest Radiation Damage in Biomolecular Systems Groningen, June 6th-9th (2006)

[53] R Bellotti et. al. *Phys. Rev. D53*2 (1996)

Appendix A

Geant4 Codes

```
    //HBODY.cc
// This program is for simulating energy loss of cosmic muons through small DNA cells
// DNA.cc
// 2004.10.08
//

#include "fileIOSingleton.hh" #include "DNADetectorConstruction.hh"
#include "DNAPhysicsList.hh" #include "DNAPrimaryGeneratorAction.hh"
#include "DNARunAction.hh" #include "DNAEventAction.hh" #include
"DNATrackingAction.hh" #include "DNASteppingAction.hh" #include
"DNAStackingAction.hh"
//#include "DNASteppingVerbose.hh"

#include "G4RunManager.hh" #include "G4UImanager.hh" #include
"G4UIterminal.hh" #include "G4UItcsh.hh"

#ifdef G4VIS_USE #include "DNAVisManager.hh" #endif

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
int main(int argc,char** argv) {

  fileIOSingleton* myOut = fileIOSingleton::instance();
  char Ofilename[100];
  G4int seed_index=10;
  std::ifstream myParfile("EnergyLoss.par");
  myParfile >> seed_index ;  // Any old positive integer number will do...
  myParfile >> Ofilename;
```

```
  myParfile.close();
  myOut->Fopen(Ofilename);

 //my Verbose output class
 // G4VSteppingVerbose::SetInstance(new DNASteppingVerbose);

  //Construct the default Run manager
  G4RunManager * runManager = new G4RunManager;

  // UserInitialization classes (mandatory)
  DNADetectorConstruction* DNAdetector = new DNADetectorConstruction;

  runManager->SetUserInitialization(DNAdetector);
  runManager->SetUserInitialization(new DNAPhysicsList);

#ifdef G4VIS_USE
  // Visualization
  G4VisManager* visManager = new DNAVisManager;
  visManager->Initialize();
#endif

  // UserAction classes
  runManager->SetUserAction(new DNAPrimaryGeneratorAction(DNAdetector));
  runManager->SetUserAction(new DNARunAction());
  runManager->SetUserAction(new DNAEventAction());
  runManager->SetUserAction(new DNATrackingAction());
  runManager->SetUserAction(new DNASteppingAction());
  runManager->SetUserAction(new DNAStackingAction());
  //Initialize G4 kernel
  runManager->Initialize();

  //get the pointer to the User Interface manager
  G4UImanager * UI = G4UImanager::GetUIpointer();

  if(argc==1)
  // Define (G)UI terminal for interactive mode
  {
    // G4UIterminal is a (dumb) terminal.
    G4UIsession * session = 0;
#ifdef G4UI_USE_TCSH
      session = new G4UIterminal(new G4UItcsh);
#else
      session = new G4UIterminal();
#endif
```

```
     UI->ApplyCommand("/control/execute ");
      //UI->ApplyCommand("/control/execute cubic.mac");
    session->SessionStart();
    delete session;
  }
  else
  // Batch mode
  {
    G4String command = "/control/execute ";
    G4String fileName = argv[1];
    UI->ApplyCommand(command+fileName);
  }

#ifdef G4VIS_USE
  delete visManager;
#endif
  delete runManager;

  myOut->Fclose();

  return 0;
}

//DNADetectorConstruction.cc
//DNADetectorConstruction.cc
// construction of the detector for small cells for the comparision of DNA molecules
// 2004.10.08

#include "G4Material.hh" #include "G4Box.hh" #include
"G4LogicalVolume.hh" #include "G4PVPlacement.hh" #include
"G4PVParameterised.hh" #include "G4SDManager.hh" #include
"G4RunManager.hh" #include "G4UserLimits.hh" #include
"G4VisAttributes.hh" #include "G4Colour.hh" #include "G4ios.hh"

#include "DNADetectorConstruction.hh" #include "DNATrackerSD.hh"
#include "DNADetectorMessenger.hh"


#include "DNAChamberParameterisation.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNADetectorConstruction::DNADetectorConstruction() :solidWorld(0),
logicWorld(0),  physiWorld(0),
 solidChamber(0),logicChamber(0),physiChamber(0),
```

```
 ChamberMater(0), fWorldLength(0.),  ChamberWidth(0.),
 solidshield(0),logicshield(0), shieldMater(0)
// conSlab("off")
{
   detectorMessenger = new DNADetectorMessenger(this);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNADetectorConstruction::~DNADetectorConstruction() {
  delete detectorMessenger;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
G4VPhysicalVolume* DNADetectorConstruction::Construct() {
//--------- Material definition ---------
  G4double a, iz, z, density;
  G4String name, symbol;
  G4int nel;

   //Vacum
    z=1.;
    a = 1*g/mole;
    density = 1e-25*g/cm3;
    G4Material* dummyMat  = new G4Material(name="dummyMat", z,a,density);

   //Air
    a = 14.01*g/mole;
    G4Element* elN = new G4Element(name="Nitrogen", symbol="N", iz=7., a);
    a = 16.00*g/mole;
    G4Element* elO = new G4Element(name="Oxigen", symbol="O", iz=8., a);
    density = 1.29*mg/cm3;
    G4Material* Air = new G4Material(name="Air", density, nel=2);
    Air->AddElement(elN, .7);
    Air->AddElement(elO, .3);

   //H2O
    a = 1.00794*g/mole;
    G4Element* elH = new G4Element(name="Hydrogen", symbol="H", iz=1., a);
    density = 1.0 *g/cm3;
    G4Material* H2O = new G4Material(name="Water", density, 2 );
    H2O->AddElement(elH, 0.1118);
    H2O->AddElement(elO, 0.8881);

    // Liver
```

```
a = 12.011*g/mole;
G4Element* elC = new G4Element(name="Carbon", symbol="C", iz=6., a);
a = 32.065*g/mole;
G4Element* elS = new G4Element(name="Sulfur", symbol="S", iz=16., a);
a = 35.453*g/mole;
G4Element* elCl = new G4Element(name="Chlorine", symbol="Cl", iz=17., a);
a = 30.0983*g/mole;
G4Element* elK = new G4Element(name="Potassium", symbol="K", iz=19., a);
a = 30.973976*g/mole;
G4Element* elP = new G4Element(name="Phosphorus", symbol="P", iz=30., a);
a = 24.3050*g/mole;
G4Element* elMg = new G4Element(name="Magnesium", symbol="Mg", iz=12., a);
density = 1071*kg/m3;
G4Material* liver = new G4Material(name="Liver", density, 9 );
liver->AddElement(elH,0.102);
liver->AddElement(elC,0.139);
liver->AddElement(elN,0.030);
liver->AddElement(elO,0.716);
//liver->AddElement(elNa,0.002);
liver->AddElement(elP,0.003);
liver->AddElement(elS,0.003);
liver->AddElement(elCl,0.002);
liver->AddElement(elK,0.003);


//Al
a = 26.98*g/mole;
density = 2.7*g/cm3;
G4Material* Aluminium = new G4Material(name="Al", z=13., a, density);

//Pb
// a = 207.19*g/mole;
//density = 11.35*g/cm3;
//G4Material* Pb = new G4Material(name="Pb", z=82., a, density);

// Concrete
a = 40.078*g/mole;
G4Element* elCa = new G4Element(name="Calcium", symbol="Ca", iz=20., a);
a = 28.0855*g/mole;
G4Element* elSi = new G4Element(name="Sillcon", symbol="Si", iz=14., a);
a = 22.989770*g/mole;
G4Element* elNa = new G4Element(name="Sodium", symbol="Na", iz=11., a);
a = 55.845*g/mole;
G4Element* elFe = new G4Element(name="Iron", symbol="Fe", iz=26., a);
```

```
    a = 26.9815*g/mole;
    G4Element* elAl = new G4Element(name="Aluminium", symbol="Al", iz=13., a);
    density = 2.3*g/cm3;
    G4Material* Con = new G4Material(name="Concrete", density, 10 );
    Con->AddElement(elCa, .06);
    Con->AddElement(elSi, .325);
    Con->AddElement(elO, .529);
    Con->AddElement(elNa, .01);
    Con->AddElement(elFe, .04);
    Con->AddElement(elAl, .033);
    Con->AddElement(elH, .01);
    Con->AddElement(elC, .001);
    Con->AddElement(elMg, .002);
    Con->AddElement(elK, .013);

 /*
    // Liver
    a = 12.011*g/mole;
    G4Element* elC = new G4Element(name="Carbon", symbol="C", iz=6., a);
    a = 32.065*g/mole;
    G4Element* elS = new G4Element(name="Sulfur", symbol="S", iz=16., a);
    a = 35.453*g/mole;
    G4Element* elCl = new G4Element(name="Chlorine", symbol="Cl", iz=17., a);
    a = 30.0983*g/mole;
    G4Element* elK = new G4Element(name="Potassium", symbol="K", iz=19., a);
    a = 30.973976*g/mole;
    G4Element* elP = new G4Element(name="Phosphorus", symbol="P", iz=30., a);
    a = 24.3050*g/mole;
    G4Element* elMg = new G4Element(name="Magnesium", symbol="Mg", iz=12., a);
    density = 1071*kg/m3;
    G4Material* liver = new G4Material(name="Liver", density, 9 );
    liver->AddElement(elH,0.102);
    liver->AddElement(elC,0.139);
    liver->AddElement(elN,0.030);
    liver->AddElement(elO,0.716);
    liver->AddElement(elNa,0.002);
    liver->AddElement(elP,0.003);
    liver->AddElement(elS,0.003);
    liver->AddElement(elCl,0.002);
    liver->AddElement(elK,0.003);

*/
    // Trabecular Bone
    density = 1159*kg/m3;
```

```
G4Material* trabecularBone = new G4Material(name="SkeletonSpongiosa", density, 12
trabecularBone->AddElement(elH,0.085);
trabecularBone->AddElement(elC,0.404);
trabecularBone->AddElement(elN,0.058);
trabecularBone->AddElement(elO,0.367);
trabecularBone->AddElement(elNa,0.001);
trabecularBone->AddElement(elMg,0.001);
trabecularBone->AddElement(elP,0.034);
trabecularBone->AddElement(elS,0.002);
trabecularBone->AddElement(elCl,0.002);
trabecularBone->AddElement(elK,0.001);
trabecularBone->AddElement(elCa,0.044);
trabecularBone->AddElement(elFe,0.001);

// dense Bone
density = 1575*kg/m3;
G4Material* denseBone = new G4Material(name="SkeletonRibs", density, 11 );
denseBone->AddElement(elH,0.056);
denseBone->AddElement(elC,0.235);
denseBone->AddElement(elN,0.050);
denseBone->AddElement(elO,0.434);
denseBone->AddElement(elNa,0.001);
denseBone->AddElement(elMg,0.001);
denseBone->AddElement(elP,0.072);
denseBone->AddElement(elS,0.003);
denseBone->AddElement(elCl,0.001);
denseBone->AddElement(elK,0.001);
denseBone->AddElement(elCa,0.146);

// Muscle
density = 1061*kg/m3;
G4Material* muscle = new G4Material(name="Muscle", density, 9 );
muscle->AddElement(elH,0.102);
muscle->AddElement(elC,0.143);
muscle->AddElement(elN,0.034);
muscle->AddElement(elO,0.710);
muscle->AddElement(elNa,0.001);
muscle->AddElement(elP,0.002);
muscle->AddElement(elS,0.003);
muscle->AddElement(elCl,0.001);
muscle->AddElement(elK,0.004);

// Breast
density = 990*kg/m3;
```

```
G4Material* breast = new G4Material(name="Breast", density, 8 );
breast->AddElement(elH,0.109);
breast->AddElement(elC,0.506);
breast->AddElement(elN,0.023);
breast->AddElement(elO,0.358);
breast->AddElement(elNa,0.001);
breast->AddElement(elP,0.001);
breast->AddElement(elS,0.001);
breast->AddElement(elCl,0.001);

// Adipose tissue
density = 967*kg/m3;
G4Material* adiposeTissue = new G4Material(name="adiposeTissue", density, 7 );
adiposeTissue->AddElement(elH,0.114);
adiposeTissue->AddElement(elC,0.598);
adiposeTissue->AddElement(elN,0.007);
adiposeTissue->AddElement(elO,0.278);
adiposeTissue->AddElement(elNa,0.001);
adiposeTissue->AddElement(elS,0.001);
adiposeTissue->AddElement(elCl,0.001);


// lungExhale
density = 508*kg/m3;
G4Material* lungexhale = new G4Material(name="lungExhale", density, 9 );
lungexhale->AddElement(elH,0.103);
lungexhale->AddElement(elC,0.105);
lungexhale->AddElement(elN,0.031);
lungexhale->AddElement(elO,0.749);
lungexhale->AddElement(elNa,0.002);
lungexhale->AddElement(elP,0.002);
lungexhale->AddElement(elS,0.003);
lungexhale->AddElement(elCl,0.002);
lungexhale->AddElement(elK,0.003);

//  LungINhale
density = 217*kg/m3;
G4Material* lunginhale = new G4Material(name="lungInhale", density, 9 );
lunginhale->AddElement(elH,0.103);
lunginhale->AddElement(elC,0.105);
lunginhale->AddElement(elN,0.031);
lunginhale->AddElement(elO,0.749);
lunginhale->AddElement(elNa,0.002);
lunginhale->AddElement(elP,0.002);
```

```
    lunginhale->AddElement(elS,0.003);
    lunginhale->AddElement(elCl,0.002);
    lunginhale->AddElement(elK,0.003);
/*
    // Concrete
    a = 40.078*g/mole;
    G4Element* elCa = new G4Element(name="Calcium", symbol="Ca", iz=20., a);
    a = 28.0855*g/mole;
    G4Element* elSi = new G4Element(name="Sillcon", symbol="Si", iz=14., a);
    a = 22.989770*g/mole;
    G4Element* elNa = new G4Element(name="Sodium", symbol="Na", iz=11., a);
    a = 55.845*g/mole;
    G4Element* elFe = new G4Element(name="Iron", symbol="Fe", iz=26., a);
    a = 26.9815*g/mole;
    G4Element* elAl = new G4Element(name="Aluminium", symbol="Al", iz=13., a);
    density = 2.3*g/cm3;
    G4Material* Con = new G4Material(name="Concrete", density, 10 );

    Con->AddElement(elCa, .04);
    Con->AddElement(elSi, .33);
    Con->AddElement(elO, .53);
    Con->AddElement(elNa, .01);
    Con->AddElement(elFe, .01);
    Con->AddElement(elAl, .03);
    Con->AddElement(elH, .01);
    Con->AddElement(elC, .001);
    Con->AddElement(elMg, .002);
    Con->AddElement(elK, .013);
*/


    G4cout << G4endl << "The materials defined are : " << G4endl << G4endl;
    G4cout << *(G4Material::GetMaterialTable()) << G4endl;

//--------- Sizes of the principal geometrical components (solids)  ---------

    //NbOfCells = 10;
    G4int Noxslices =1;
    G4int Noyslices =1;
    G4int Nozslices= 1;
    NbOfCells = Noxslices*Noyslices*Nozslices;

   // CellWidth = 5*cm;
    //fWorldLength= 4200*cm;
```

```
  fWorldLength= 200*cm;
 //     fWorldLength= .41*mm;
    //CellSpacing = 5*cm;

//--------- Definitions of Solids, Logical Volumes, Physical Volumes ---------

  //------------------------------
  // World
  //------------------------------

  G4double HalfWorldLength = 0.5*fWorldLength;

  solidWorld= new G4Box("world",HalfWorldLength,HalfWorldLength,HalfWorldLength);
  logicWorld= new G4LogicalVolume( solidWorld, dummyMat, "World", 0, 0, 0);

  //  Must place the World Physical volume unrotated at (0,0,0).
  //
  physiWorld = new G4PVPlacement(0,               // no rotation
                                 G4ThreeVector(), // at (0,0,0)
                                 "World",         // its name
                                 logicWorld,      // its logical volume
                                 0,               // its mother  volume
                                 false,           // no boolean operations
                                 0);              // no field specific to volume
/*
//------------------------------
  // shilding material
  //------------------------------
// G4cout << "conslab" << conSlab << G4endl;

// if (conSlab == "on")
//{
  ShieldLenX = 1000.*cm;//0.000012  *cm;
  ShieldLenY = 1000.*cm;//0.000012  *cm;
  ShieldWidth = 400.0 *cm;//0.000012 *cm;


  // Define material for shield
  shieldMater = Con;


  solidshield = new G4Box("shield", ShieldLenX/2., ShieldLenY/2., ShieldWidth/2.);
  logicshield = new G4LogicalVolume(solidshield,shieldMater,"shield",0,0,0);
```

```
 ShieldXpos = 0.0 *cm;
 ShieldYpos = 0.0 *cm;
 ShieldZpos = 300.0 *cm;

physishield = new G4PVPlacement(0,               // no rotation
                                G4ThreeVector(ShieldXpos,ShieldYpos,ShieldZpos), //
                                "shield",         // its name
                                logicshield,      // its logical volume
                                //logicWorld,        // its mother  volume
                                physiWorld,        // its mother  volume
                                 false,           // no boolean operations
                                0);              // no field specific to volume

 //G4cout << "I am inside the soncrete slab " << G4endl;
 //G4cout << "conslab" << conSlab << G4endl;
//}

 */

 //G4cout << "I am outside the concrete slab " << G4endl;

 //------------------------------
 // Water world
 //------------------------------

 //WaterWorldLenX = 39.*cm;
 //WaterWorldLenY = 59.*cm;
 //WaterWorldLenZ = 169.*cm;

   //WaterWorldLenX = .4*mm;
   //WaterWorldLenY = .4*mm;
   //WaterWorldLenZ = .4*mm;

 WaterWorldLenX = 180.*cm;
 WaterWorldLenY = 180.*cm;
 WaterWorldLenZ = 180.*cm;

 solidWaterWorld= new G4Box("waterworld",WaterWorldLenX/2., WaterWorldLenY/2., Water
 logicWaterWorld= new G4LogicalVolume( solidWaterWorld, dummyMat, "WaterWorld", 0, 0

 //  Must place the World Physical volume unrotated at (0,0,0).
 //
 physiWaterWorld = new G4PVPlacement(0,               // no rotation
                                G4ThreeVector(), // at (0,0,0)
```

```
                                "WaterWorld",         // its name
                                logicWaterWorld,      // its logical volume
                                physiWorld,              // its mother  volume
                                false,           // no boolean operations
                                0);              // no field specific to volume


 //------------------------------
 // water chamber
 //------------------------------

 //ChamberLenX = 39.*cm;
 //ChamberLenY = 59.*cm;
 //ChamberWidth = 169.*cm;

     ChamberLenX = 180.*cm;
     ChamberLenY = 180.*cm;
     ChamberWidth = 180.*cm;

    // ChamberLenX = .4*mm;
    // ChamberLenY = .4*mm;
    // ChamberWidth = .4*mm;

 //G4int Noxslices = 10;
 //G4int Noyslices = 10;
 //G4int Nozslices = 10;


 // Define material for the water chamber
  ChamberMater = H2O;
  // ChamberMater = Con;
  //ChamberMater = muscle;
 Zpos = 0.0;

 solidChamber = new G4Box("chamber", ChamberLenX/2., ChamberLenY/2., ChamberWidth/2.
 logicChamber = new G4LogicalVolume( solidChamber, ChamberMater, "Chamber", 0, 0, 0)
// G4double firstPosition = 0.0;//-0.5*ChamberWidth + 0.5*CellWidth;

 G4VPVParameterisation* chamberParam = new DNAChamberParameterisation(
                                  NbOfCells,          // NoCellss
                                  Noxslices,
                                  Noyslices,
                                  Nozslices,
                                  ChamberLenX,
```

```
                                    ChamberLenY,
                                    ChamberWidth);
                                    //firstPosition,         // Z of center of first
                                    //CellSpacing,        // Z spacing of centers
                                    //CellWidth,          // Width Chamber

 physiChamber = new G4PVParameterised(
                                    "Cell",       // their name
                                    logicChamber,    // their logical volume
                                    physiWaterWorld,    // Mother physical volume
                                    kUndefined,          // Are placed along this axi
                                    NbOfCells,    // Number of chambers
                                    chamberParam);   // The parametrisation


// physiChamber = new G4PVPlacement(0,               // no rotation
  //                               G4ThreeVector(Xpos,Ypos,Zpos), // at (0,0,0)
    //                             "Chamber",         // its name
      //                           logicChamber,      // its logical volume
        //                         physiWorld,        // its mother  volume
          //                       false,           // no boolean operations
            //                     0);              // no field specific to volume
 /*
//------------------------------
  // Liver
  //------------------------------

  WaterWorldLenX = 20.*cm;
  WaterWorldLenY = 20.*cm;
  WaterWorldLenZ = 20.*cm;

  solidLiver= new G4Box("liver",20., 20., 20.);
  logicLiver= new G4LogicalVolume( solidLiver, liver, "Liver", 0, 0, 0);

  //  Must place the World Physical volume unrotated at (0,0,0).
  //
  physiLiver = new G4PVPlacement(0,               // no rotation
                                 G4ThreeVector(), // at (0,0,0)
                                 "Liver",         // its name
                                 logicLiver,      // its logical volume
                                 physiChamber,               // its mother  volume
                                 false,           // no boolean operations
                                 0);              // no field specific to volume
```

```
   */
  //-------------------------------------------------
  // Sensitive detectors
  //-------------------------------------------------


  G4SDManager* SDman = G4SDManager::GetSDMpointer();

  G4String chamberSensorSDname = "DNA/ChamberSensor";
  DNATrackerSD* sensorSD = new DNATrackerSD( chamberSensorSDname,this );
  SDman->AddNewDetector( sensorSD );
  logicChamber->SetSensitiveDetector( sensorSD );


//--------- Visualization attributes -------------------------------

  G4VisAttributes* BoxVisAtt= new G4VisAttributes(G4Colour(1.0,1.0,1.0));
  logicWorld  ->SetVisAttributes(BoxVisAtt);

  G4VisAttributes* ChamberVisAtt = new G4VisAttributes(G4Colour(0.0,0.0,1.0));
  logicChamber->SetVisAttributes(ChamberVisAtt);

  return physiWorld;
}

void DNADetectorConstruction::SetChamberMaterial(G4String
materialName) {
  // search the material by its name
  G4Material* pttoMaterial = G4Material::GetMaterial(materialName);
  if (pttoMaterial)
    {ChamberMater = pttoMaterial;
    logicChamber->SetMaterial(pttoMaterial);

    G4cout << "\n----> The chamber is " << ChamberWidth/cm << " cm of "
           << materialName << G4endl;
    }
}

void DNADetectorConstruction::SetChamberThick(G4double newValue) {
  G4double ChWidth = newValue;
  solidChamber->SetZHalfLength(ChWidth/2.);

  G4cout << "\n----> The chamber thick is " << ChWidth/cm << " cm " <<G4endl;
  // G4cout << "You must use the command: /geometry/update before beamOn";
```

```
  // G4cout << G4endl;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNADetectorConstruction::UpdateGeometry() {
  G4RunManager::GetRunManager()->DefineWorldVolume(Construct());
}


void DNADetectorConstruction::UpdateMaterials() {
  G4RunManager::GetRunManager()->DefineWorldVolume(Construct());
}


//DNAPhysicsLists.cc
#include "DNAPhysicsList.hh"

#include "GeneralPhysics.hh" #include "EM_GNPhysics.hh" #include
"MuonPhysics.hh" #include "IonPhysics.hh" #include
"G4DataQuestionaire.hh" #include "globals.hh" #include
"G4ProcessManager.hh" #include "G4ParticleTypes.hh" #include
"HadronPhysicsLHEP_GN.hh"

DNAPhysicsList::DNAPhysicsList():G4VModularPhysicsList() {
 //defaultCutValue = 1.0*cm;
 //defaultCutValue = 500*eV;
 // defaultCutValue = 100*eV;
  G4DataQuestionaire(photon);
  SetVerboseLevel(1);

  // General Physics
  RegisterPhysics(new GeneralPhysics("general"));

  // EM Physics
  RegisterPhysics(new EM_GNPhysics("standard EM plus gamma nuclear"));

  // Muon Physics
  RegisterPhysics(new MuonPhysics("muon"));

  // Hadron Physics
  RegisterPhysics(new HadronPhysicsLHEP_GN("hadron"));

  // Ion Physics
  RegisterPhysics(new IonPhysics("ion"));
```

```
}

DNAPhysicsList::~DNAPhysicsList() {} /* void
DNAPhysicsList::ConstructParticle() {
  // In this method, static member functions should be called
  // for all particles which you want to use.
  // This ensures that objects of these particle types will be
  // created in the program.

  ConstructBosons();
  ConstructLeptons();
  ConstructMesons();
  ConstructBaryons();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void DNAPhysicsList::ConstructBosons() {
  // pseudo-particles
  G4Geantino::GeantinoDefinition();
  G4ChargedGeantino::ChargedGeantinoDefinition();

  // gamma
  G4Gamma::GammaDefinition();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void DNAPhysicsList::ConstructLeptons() {
  // leptons
  //  e+/-
  G4Electron::ElectronDefinition();
  G4Positron::PositronDefinition();
  // mu+/-
  G4MuonPlus::MuonPlusDefinition();
  G4MuonMinus::MuonMinusDefinition();
  // nu_e
  G4NeutrinoE::NeutrinoEDefinition();
  G4AntiNeutrinoE::AntiNeutrinoEDefinition();
  // nu_mu
  G4NeutrinoMu::NeutrinoMuDefinition();
  G4AntiNeutrinoMu::AntiNeutrinoMuDefinition();
}
```

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void ExN02PhysicsList::ConstructBaryons() {
  //  barions
  G4Proton::ProtonDefinition();
  G4AntiProton::AntiProtonDefinition();

  G4Neutron::NeutronDefinition();
  G4AntiNeutron::AntiNeutronDefinition();
}

*/
void DNAPhysicsList::SetCuts() {
  if (verboseLevel > 1)
  {
    G4cout << "PhenixPhysicsList::SetCuts:";
  }

  SetCutsWithDefault();
//G4ProductionCutsTable::GetProductionCutsTable()
  //                                  ->SetEnergyRange(250*eV, 100*GeV);


  G4VUserPhysicsList::DumpCutValuesTable();
}
//DNAChamberParameterisation.cc
//to compare with DNA
//July 31,2004
//
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......//...
#include "DNAChamberParameterisation.hh"

#include "G4VPhysicalVolume.hh" #include "G4ThreeVector.hh" #include
"G4Box.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNAChamberParameterisation::DNAChamberParameterisation(
        G4int    NoChambers,
        G4int    Noxslices,
        G4int    Noyslices,
        G4int    Nozslices,
        G4double xlength,
```

```
        G4double ylength,
        G4double zlength)

{
   fNoChambers =  NoChambers;
   fNoxslices =  Noxslices;
   fNoyslices =  Noyslices;
   fNozslices =  Nozslices;
   fHalfxWidth  = 0.5*xlength / (G4double)Noxslices;
   fHalfyWidth  = 0.5*ylength / (G4double)Noyslices;
   fHalfzWidth  = 0.5*zlength / (G4double)Nozslices;
   fStartX     =  -0.5*xlength+fHalfxWidth;
   fStartY     =  -0.5*ylength+fHalfyWidth;
   fStartZ     =  -0.5*zlength+fHalfzWidth;
   fxSpacing    =  2.0*fHalfxWidth;
   fySpacing    =  2.0*fHalfyWidth;
   fzSpacing    =  2.0*fHalfzWidth;

   fHalfXLength = fHalfxWidth;
   fHalfYLength = fHalfyWidth;
   fHalfZLength = fHalfzWidth;

  G4cout << "---------------------------------" << fHalfXLength << G4endl;
  G4cout << "---------------------------------" << fHalfXLength << G4endl;
  G4cout << "---------------------------------" << fHalfXLength << G4endl;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNAChamberParameterisation::~DNAChamberParameterisation() {}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNAChamberParameterisation::ComputeTransformation (const G4int
copyNo, G4VPhysicalVolume* physVol) const {
  G4int zc = (G4int)floor((G4double)copyNo/((G4double)fNoxslices*fNoyslices));
  G4int remz = copyNo%(fNoxslices*fNoyslices);
  G4int yc = (G4int)floor((double)remz/(G4double)fNoxslices);
  G4int xc = remz%fNoxslices;

  G4double      Xposition= fStartX + xc * fxSpacing;
  G4double      Yposition= fStartY + yc * fySpacing;
  G4double      Zposition= fStartZ + zc * fzSpacing;
  G4ThreeVector origin(Xposition,Yposition,Zposition);
  physVol->SetTranslation(origin);
  physVol->SetRotation(0);
```

```
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNAChamberParameterisation::ComputeDimensions (G4Box&
trackerChamber, const G4int copyNo,
 const G4VPhysicalVolume* physVol) const
{
  //G4cout << "--------------------------------" << fHalfXLength << G4endl;
  trackerChamber.SetXHalfLength(fHalfXLength);
  trackerChamber.SetYHalfLength(fHalfYLength);
  trackerChamber.SetZHalfLength(fHalfZLength);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

//DNADetectorMessenger.cc
//2004/10/08
//
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#include "DNADetectorMessenger.hh"

#include "DNADetectorConstruction.hh" #include "G4UIdirectory.hh"
#include "G4UIcmdWithAString.hh" #include
"G4UIcmdWithADoubleAndUnit.hh" #include "G4UIcmdWithoutParameter.hh"
#include "globals.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

DNADetectorMessenger::DNADetectorMessenger(DNADetectorConstruction*
myDet) :myDetector(myDet) {

  mydetDir = new G4UIdirectory("/mydet/");
  mydetDir->SetGuidance("DNA detector control.");

  ChamThCmd = new G4UIcmdWithADoubleAndUnit("/mydet/setChamberThick",this);
  ChamThCmd->SetGuidance("Set the thickness of the Chamber.");
  ChamThCmd->SetParameterName("Thick from 2cm to 2m",false);
  ChamThCmd->SetDefaultUnit("cm");
  ChamThCmd->SetUnitCategory("Length");
  ChamThCmd->AvailableForStates(G4State_PreInit,G4State_Idle);
```

```
  ChamMatCmd = new G4UIcmdWithAString("/mydet/setChamberMate",this);
  ChamMatCmd->SetGuidance("Select Material of the Target.");
  ChamMatCmd->SetParameterName("Air Water Al Pb Concrete",false);
  ChamMatCmd->AvailableForStates(G4State_PreInit,G4State_Idle);

  UpdateCmd = new G4UIcmdWithoutParameter("/mydet/update/updateGeometry",this);
  UpdateCmd->SetGuidance("Update geometry.");
  UpdateCmd->SetGuidance("This command must be applied before \"beamOn\" ");
  UpdateCmd->SetGuidance("if you used the setChamberThick command.");
  UpdateCmd->AvailableForStates(G4State_Idle);

  DefineCmd = new G4UIcmdWithoutParameter("/mydet/update/updateMaterials",this);
  DefineCmd->SetGuidance("Update the materials.");
  DefineCmd->AvailableForStates(G4State_Idle);


  /*ConSlabCmd = new G4UIcmdWithAString("/mydet/conSlab",this);
  ConSlabCmd->SetGuidance("Flag Concrete Slab on off.");
  ConSlabCmd->SetParameterName("ConSlab",true);
  ConSlabCmd->SetDefaultValue("off");
  ConSlabCmd->SetCandidates("on off");*/

//...oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
} DNADetectorMessenger::~DNADetectorMessenger() {
  delete ChamThCmd;
  delete ChamMatCmd;
  delete mydetDir;
  //delete ConSlabCmd;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void DNADetectorMessenger::SetNewValue(G4UIcommand* command,G4String
newValue) {
   if( command == ChamThCmd )
   { myDetector->SetChamberThick(ChamThCmd->GetNewDoubleValue(newValue));}

  if( command == ChamMatCmd )
   { myDetector->SetChamberMaterial(newValue);}

  if (command == UpdateCmd)
    { myDetector->UpdateGeometry();}
```

```
  if (command == DefineCmd)
    { myDetector->UpdateMaterials();}

 // if (command == ConSlabCmd)
   //  { myDetector-> SetConSlab(newValue);}
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
//DNAEventAction.cc
//2004.10.08
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......//...
#include "DNAEventAction.hh" #include "DNATrackerHit.hh" #include
"G4Event.hh" #include "G4EventManager.hh" #include
"G4TrajectoryContainer.hh" #include "G4Trajectory.hh" #include
"G4VVisManager.hh" #include "G4ios.hh" #include "G4UnitsTable.hh"
#include "G4SDManager.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNAEventAction::DNAEventAction() :trackerCollID(-1) {
fileOut=fileIOSingleton::instance(); }

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNAEventAction::~DNAEventAction() { }

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNAEventAction::BeginOfEventAction(const G4Event*) {
//  G4int event_id = evt->GetEventID();
 if (trackerCollID==-1)
    {
       G4SDManager * SDman = G4SDManager::GetSDMpointer();
    trackerCollID = SDman->GetCollectionID("trackerCollection");
    //G4cout << ">>> tracker ID ========= " << trackerCollID << G4endl;
   }

  // fileOut->fout << "The Launch particle is:" << G4endl;
  // fileOut->fout << "chambernumber        " << " energy deposit             "
  //              << " position " << G4endl;

}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNAEventAction::EndOfEventAction(const G4Event* evt) {
  G4int event_id = evt->GetEventID();
  G4double totECham=0, totLCham=0;
```

```
  G4double totEdep = 0;
// get number of stored trajectories
  G4HCofThisEvent* HCE = evt->GetHCofThisEvent();
  DNATrackerHitsCollection* CHC = 0;
 G4int n_hit = 0;

 if (HCE) CHC = (DNATrackerHitsCollection*)(HCE->GetHC(trackerCollID));
 //G4cout << "before Hits loop" << G4endl;
  if (CHC)
 {
    n_hit = CHC->entries();
 //G4cout << "number of hits:   " << n_hit << G4endl;
    for (G4int i=0;i<n_hit;i++)
       {
       totECham += (*CHC)[i]->GetEdepCham();
 //G4cout << "energy loss :   " << totECham << G4endl;
       totEdep = totEdep + totECham;
       totLCham += (*CHC)[i]->GetTrakCham();
 //G4cout << "total length:   " << totLCham << G4endl;
       //nStep += (*CHC)[ii]->GetNStep();
       }
       //G4cout << "Total energy loss :   " << totEdep << G4endl;
 }
 // G4cout << "Total energy loss :   " << totEdep << G4endl;
  G4TrajectoryContainer* trajectoryContainer = evt->GetTrajectoryContainer();
  G4int n_trajectories = 0;
  if (trajectoryContainer) n_trajectories = trajectoryContainer->entries();

  // periodic printing
  //
 // if (event_id < 100 || event_id%100 == 0) {
   // fileOut->fout << ">>> Event " << evt->GetEventID() << " **********************
   // fileOut->fout << "999999" << "999999"<< G4endl;
    fileOut->fout << std::setw(15) << "9999" <<std::setw(15)<< "9999" <<std::setw(15)

//   fileOut->fout << std::setw(15) << "9999" <<std::setw(15)<< "9999" <<std::setw(15

   //fileOut->fout << std::setw(15) << totEdep <<std::setw(15)<< totEdep << G4endl;
          // G4cout << std::setw(15) << "Total energy dep" <<std::setw(15)<< totEdep
// G4cout << ">>> Event " << evt->GetEventID() << G4endl;
  //  G4cout << "    " << n_trajectories
    //       << " trajectories stored in this event." << G4endl;
   // G4cout << "       total energy: " << std::setw(7)
     //      << G4BestUnit(totECham,"Energy")
```

```
       //    << "       total track length: " << std::setw(7)
         //  << G4BestUnit(totLCham,"Length")
           //<< "       LET: " << totECham/totLCham <<G4endl;
     //G4cout << "  number of steps of e+/e- : " << nStep << G4endl;

   // }

  // extract the trajectories and draw them
  //
  if (G4VVisManager::GetConcreteInstance())
    {
     for (G4int i=0; i<n_trajectories; i++)
        { G4Trajectory* trj = (G4Trajectory*)
                                    ((*(evt->GetTrajectoryContainer()))[i]);
          trj->DrawTrajectory(50);
        }
    }
}  //DNAPrimaryGeneratorAction.cc
//2004.23.04
//
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......//...
#include "DNAPrimaryGeneratorAction.hh" #include
"DNADetectorConstruction.hh" #include
"DNAPrimaryGeneratorMessenger.hh"


#include "G4Event.hh" #include "G4ParticleGun.hh" #include
"G4GeneralParticleSource.hh" #include "G4ParticleTable.hh" #include
"G4ParticleDefinition.hh" #include "globals.hh" #include
"Randomize.hh"



//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNAPrimaryGeneratorAction::DNAPrimaryGeneratorAction(
                                                 DNADetectorConstruction* myDC)
  :myDetector(myDC),rndmFlag("off"),Energy(10.0)
{
  G4int n_particle = 1;
   particleGun = new G4ParticleGun(n_particle);
  gpsGun = new G4GeneralParticleSource();

 //create a messenger for this class
  gunMessenger = new DNAPrimaryGeneratorMessenger(this);
```

```
// default particle

  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition* particle = particleTable->FindParticle("e-");

  particleGun->SetParticleDefinition(particle);
  particleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
  particleGun->SetParticleEnergy(Energy*MeV);

  hitsPerM2PerSecPerSterrad = 0.0;
  sterradians = 0.0;

  fout.open("momentumdist.dat");
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNAPrimaryGeneratorAction::~DNAPrimaryGeneratorAction() {
  delete particleGun;
  delete gpsGun;
  delete gunMessenger;
  fout.close();
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNAPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  //this function is called at the begining of event
  //
  // G4double z0 = 0.*cm;
  G4double z0 = -0.5*(myDetector->GetWorldFullLength());
  G4double y0 = 0.*cm, x0 = 0.*cm;

  particleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));

 /*  G4double z0 = -0.5*(myDetector->GetWorldFullLength());
 G4double  x0 = (myDetector->GetTrackerFullLength())*(G4UniformRand()-0.5);
 G4double  y0 = (myDetector->GetTrackerFullLength())*(G4UniformRand()-0.5);
 G4double  y1 = (myDetector->GetTrackerFullLength())*(G4UniformRand()-0.5);

 //particleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));
 particleGun->SetParticlePosition(G4ThreeVector(0.*cm,0.*cm,0.*cm));
 particleGun->SetParticleMomentumDirection(G4ThreeVector(1,1,100));
 */
```

```
  G4double pi = 4.0*atan(1.0);
  G4double momentum = cosmicRayMuonMomentum();
  G4double momangle = cosmicRayMuonAngle();
  fout << momentum << "  " << momangle << std::endl;
  G4double phi = 2*pi*G4UniformRand();
  G4double px = momentum*cos(phi)*sin(momangle);
  G4double py = momentum*sin(phi)*sin(momangle);
  G4double pz = momentum*cos(momangle);
  //particleGun->SetParticleMomentum(G4ThreeVector(px, py, pz));
  //particleGun->SetParticleMomentumDirection(G4ThreeVector(px, py, pz));
  //particleGun->GeneratePrimaryVertex(anEvent);
  gpsGun->GeneratePrimaryVertex(anEvent);
}

void DNAPrimaryGeneratorAction::SetParticle(G4String particleName) {
  G4String newparticle =  particleName;
  G4ParticleTable* newparticleTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition* pttoparticle = newparticleTable->FindParticle(newparticle);

  if (pttoparticle)
     {
       particleGun->SetParticleDefinition(pttoparticle);
       G4cout << "\n----> The input particle is " << newparticle << G4endl;
     }
 else
   G4cout << "The input particle dose not exist " << G4endl;

}

void DNAPrimaryGeneratorAction::SetEnergy(G4double newValue) {
   G4double NewEnergy = newValue;
   particleGun->SetParticleEnergy(NewEnergy*MeV);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
G4double DNAPrimaryGeneratorAction::cosmicRayMuonMomentum() {
    // Data from Kremer et al, Phys. Rev. Lett., Vol 83 no 21, p4241 (1999).
    // values are lower bin edge, bin average, mu+ rate, mu- rate
    // (laid out this silly way so verification with the paper is easy)
    // NOTE: units are GeV/c, and counts/(GeV/c m^2 sr s)
    static double vals[] = {
        0.0,    0.0,    0.0,    0.0,
        0.2,    0.25,   14.0,   11.0,
```

```
    0.3,    0.35,   16.8,   13.6,
    0.4,    0.47,   17.2,   14.4,
    0.55,   0.62,   16.6,   13.5,
    0.70,   0.78,   15.6,   13.3,
    0.85,   0.92,   14.8,   12.1,
    1.0,    1.1,    13.0,   11.0,
    1.2,    1.3,    12.0,   10.1,
    1.4,    1.5,    10.2,   8.7,
    1.6,    1.84,   9.1,    7.3,
    2.1,    2.49,   6.6,    5.2,
    2.94,   3.49,   4.12,   3.38,
    4.12,   4.78,   2.53,   1.98,
    5.5,    6.21,   1.61,   1.25,
    7.0,    8.37,   0.90,   0.69,
    10.0,   12.42,  0.389,  0.309,
    15.5,   18.85,  0.138,  0.108,
    23.0,   26.68,  0.063,  0.046,
    31.1,   36.69,  0.028,  0.019,
    43.6,   51.47,  0.0099, 0.0071,
    61.1,   72.08,  0.0036, 0.0030,
    85.6,   100.96, 0.0014, 0.0012,
    120.0,  120.0,  0.0,    0.0}; // cutoff at 120 GeV/c
const int nvals = sizeof(vals)/sizeof(vals[0]);
const int nbins = nvals/4 - 1;
const int npdf=256;
static double pdf[npdf];
static double pmax = vals[4*nbins];
static bool init=true;

if(init) {
    // RandGeneral needs equal-sized bins for pdf[]
    // it returns a value in the range [0,1)
    hitsPerM2PerSecPerSterrad = 0.0;
    for(int i=0,ibin=0; i<npdf; ++i) {
        double p = (i+0.5)*pmax/npdf;
        while(p >= vals[4*ibin+5]) ++ibin;
        assert(ibin <= nbins);
        double f = (p - vals[4*ibin+1]) /
                    (vals[4*ibin+5]-vals[4*ibin+1]);
        assert(0.0 <= f && f <= 1.0);
        pdf[i] = (1.0-f)*(vals[4*ibin+2]+vals[4*ibin+3]) +
                f*(vals[4*ibin+6]+vals[4*ibin+7]);
        hitsPerM2PerSecPerSterrad += pdf[i] * pmax/npdf;
        //G4cout << "p" << "     " << p << "     " << "f" << "     " << f << "
```

```
        }
        init = false;
    }

    RandGeneral generator(pdf,npdf); // BUG in RandGeneral - cannot use new
    return generator.shoot() * pmax * GeV;
}

G4double DNAPrimaryGeneratorAction::cosmicRayMuonAngle() {
    const int npdf=128;
    static double pdf[npdf];
    const double thetamax = 70.0*deg;
    static bool init = true;

    if(init) {
        // RandGeneral needs equal-sized bins for pdf[]
        // it returns a value in the range [0,1)
        sterradians = 0.0;
        G4double dtheta = thetamax / npdf;
        for(int i=0; i<npdf; ++i) {
            // Particle Data Group, Review of Particle Properties,
            // 2002. Section 23.3.1.
            double c = cos(dtheta*i);
            pdf[i] = c*c;
            sterradians += 2.0*pi*c*c*sin(dtheta*i)*dtheta;
        }
        init = false;
    }

    RandGeneral generator(pdf,npdf); // BUG in RandGeneral - cannot use new
    return generator.shoot() * thetamax;
}
//DNAPrimaryGeneratorMessenger.cc
//2004/10/08

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#include "DNAPrimaryGeneratorMessenger.hh"

#include "DNAPrimaryGeneratorAction.hh" #include
"G4UIcmdWithAString.hh" #include "G4UIcmdWithADoubleAndUnit.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

```
DNAPrimaryGeneratorMessenger::DNAPrimaryGeneratorMessenger(
                                          DNAPrimaryGeneratorAction* DNAGun)
:DNAAction(DNAGun) {
  ParticleCmd = new G4UIcmdWithAString("/gun/setParticle",this);
  ParticleCmd->SetGuidance("Select particle.");
  ParticleCmd->SetParameterName("e- gammar proton neutron mu+",false);
  ParticleCmd->AvailableForStates(G4State_PreInit,G4State_Idle);


  EngCmd = new G4UIcmdWithADoubleAndUnit("/gun/setEnergy",this);
  EngCmd->SetGuidance("Set the input energy of the input particle.");
  EngCmd->SetParameterName("Energy from 0.1 MeV to 1000MeV",false);
  EngCmd->SetDefaultUnit("MeV");
  EngCmd->SetUnitCategory("Energy");
  EngCmd->AvailableForStates(G4State_PreInit,G4State_Idle);


  RndmCmd = new G4UIcmdWithAString("/gun/random",this);
  RndmCmd->SetGuidance("Shoot the incident particle randomly.");
  RndmCmd->SetGuidance("  Choice : on(default), off");
  RndmCmd->SetParameterName("choice",true);
  RndmCmd->SetDefaultValue("on");
  RndmCmd->SetCandidates("on off");
  RndmCmd->AvailableForStates(G4State_PreInit,G4State_Idle);
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

DNAPrimaryGeneratorMessenger::~DNAPrimaryGeneratorMessenger() {
  delete RndmCmd;
  delete EngCmd;
  delete  ParticleCmd;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void DNAPrimaryGeneratorMessenger::SetNewValue(
                                        G4UIcommand* command, G4String newValue)
{
  if( command ==  ParticleCmd )
   { DNAAction->SetParticle(newValue);}
```

```
   if( command == EngCmd )
   { DNAAction->SetEnergy(EngCmd->GetNewDoubleValue(newValue));}

  if( command == RndmCmd )
   { DNAAction->SetRndmFlag(newValue);}
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
//DNARunAction.cc
//2002.23.04
////
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......//...
#include "DNARunAction.hh"

#include "G4Run.hh" #include "G4UImanager.hh" #include
"G4VVisManager.hh" #include "G4ios.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNARunAction::DNARunAction() {}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNARunAction::~DNARunAction() {}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNARunAction::BeginOfRunAction(const G4Run* aRun) {
  G4cout << "### Run " << aRun->GetRunID() << " start." << G4endl;

  if (G4VVisManager::GetConcreteInstance())
    {
      G4UImanager* UI = G4UImanager::GetUIpointer();
      UI->ApplyCommand("/vis/scene/notifyHandlers");
    }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNARunAction::EndOfRunAction(const G4Run*) {
  if (G4VVisManager::GetConcreteInstance())
    {
     G4UImanager::GetUIpointer()->ApplyCommand("/vis/viewer/update");
    }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

```
//DNAStackingAction.cc
#include "DNAStackingAction.hh"
//#include "DNAStackingMessenger.hh"
#include "G4Track.hh" #include "G4ParticleDefinition.hh" #include
"G4ParticleTypes.hh" #include "G4StepPoint.hh" #include
"G4Material.hh" #include "G4SteppingControl.hh" #include
"G4StepStatus.hh"           // Include from 'track' #include
"G4TouchableHandle.hh"      // Include from 'geometry' #include
"G4LogicalVolume.hh" #include "G4Step.hh" #include "G4Track.hh"
#include "G4ParticleDefinition.hh" #include "G4ThreeVector.hh"
#include "G4VSolid.hh" #include "G4Box.hh" #include
"G4TouchableHistory.hh" #include "G4VTouchable.hh"

#include <iomanip> using namespace std;

DNAStackingAction::DNAStackingAction() {
  minEminusKE = 10*eV;
  //minGammaKE = 100*eV;
// fileOut = fileIOSingleton::instance();

 //StackMessenger = new DNAStackingMessenger(this);
}


DNAStackingAction::~DNAStackingAction() {
  //delete StackMessenger;
}

G4ClassificationOfNewTrack DNAStackingAction::ClassifyNewTrack(const
        G4Track* aTrack)
{
  G4String  name = aTrack->GetDefinition()->GetParticleName();
  G4int  trackid = aTrack->GetTrackID();
  G4double kinen = aTrack->GetKineticEnergy();
  G4ThreeVector pos = aTrack->GetPosition();
  G4double x = pos.x();
  G4double y = pos.y();
  G4double z = pos.z();


  //cout << name <<std::setw(30) << trackid <<std::setw(30) << kinen << endl;
  ofstream fileOut("aaa.dat", ios::app);
  fileOut << std::setw(30) << name <<std::setw(30) << trackid <<std::setw(30) << kine
```

```
  G4ClassificationOfNewTrack classification = fWaiting;
  G4ParticleDefinition* particleType = aTrack->GetDefinition();

  // The following kills the tracking of secondary electrons from the primary
  // and all neutrinos
  if (((aTrack->GetKineticEnergy()<minEminusKE) &&
    (particleType==G4Electron::ElectronDefinition())) ||
    //((aTrack->GetKineticEnergy()<minGammaKE) &&
    // (particleType==G4Gamma::GammaDefinition())) ||
    (particleType==G4NeutrinoE::NeutrinoEDefinition()) ||
//  (particleType==G4AntiNeutrinoE::AntiNeutrinoEDefinition()) ||
//  (particleType==G4NeutrinoMu::NeutrinoMuDefinition()) ||
    (particleType==G4AntiNeutrinoMu::AntiNeutrinoMuDefinition()) ||
    (particleType==G4NeutrinoTau::NeutrinoTauDefinition()) ||
    (particleType==G4AntiNeutrinoTau::AntiNeutrinoTauDefinition()))
    {classification = fKill;}
  else classification = fUrgent;

  return classification;
}

//DNASteppingAction.cc
//2004/05/26
//

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

#include "DNASteppingAction.hh" #include "G4StepPoint.hh" #include
"G4Material.hh" #include "G4SteppingControl.hh" #include
"G4StepStatus.hh"           // Include from 'track' #include
"G4TouchableHandle.hh"      // Include from 'geometry' #include
"G4LogicalVolume.hh" #include "G4Step.hh" #include "G4Track.hh"
#include "G4ParticleDefinition.hh" #include "G4ThreeVector.hh"
#include "G4VSolid.hh" #include "G4Box.hh" #include
"G4TouchableHistory.hh" #include "G4VTouchable.hh"

#include <iomanip>

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

DNASteppingAction::DNASteppingAction() {
```

```
  fileOut = fileIOSingleton::instance();
  G4cout << " I am here " << G4endl;

}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

DNASteppingAction::~DNASteppingAction() { }
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void DNASteppingAction::UserSteppingAction(const G4Step* aStep) {
               G4double edep=aStep->GetTotalEnergyDeposit();

      G4String particleName = aStep->GetTrack()->GetDefinition()->GetParticleName();
      G4ThreeVector position = aStep->GetPreStepPoint()->GetPosition();

 G4TouchableHistory* theTouchable = (G4TouchableHistory*)aStep->GetPostStepPoint()->G

G4int copyID = theTouchable->GetReplicaNumber(0);
      G4double X = position.getX();
      G4double Y = position.getY();
      G4double Z = position.getZ();


       //if(Z>-50.0 && Z<50.0)
     //if(Z == 850.0 && particleName == "mu-")
          //if(Z >= 50)

         fileOut->fout << std::setw(15) << X <<std::setw(15)<< Y <<std::setw(15)<< Z

       //fileOut->fout << std::setw(15) << X <<std::setw(15) << edep << G4endl;
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo...

//DNATrackerSD.cc
//2004.10.08
//
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......//...
#include "DNATrackerSD.hh" #include "G4HCofThisEvent.hh" #include
"G4Step.hh" #include "G4ThreeVector.hh" #include "G4SDManager.hh"
#include "G4ios.hh"

#include "G4ParticleDefinition.hh" #include "G4ParticleWithCuts.hh"
#include "G4ProcessManager.hh" #include "G4ProcessVector.hh"
#include "G4ParticleTypes.hh" #include "G4ParticleTable.hh" #include
```

```
"G4Material.hh" #include "G4ios.hh" #include "iomanip"

#include "DNATrackerHit.hh" #include "DNADetectorConstruction.hh"
#include "G4VPhysicalVolume.hh" #include "G4LogicalVolume.hh"
#include "G4Track.hh" #include "G4Step.hh" #include
"G4ParticleDefinition.hh" #include "G4VTouchable.hh" #include
"G4TouchableHistory.hh" #include "G4ios.hh" #include
"G4UnitsTable.hh"


//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNATrackerSD::DNATrackerSD(G4String name, DNADetectorConstruction*
det) :G4VSensitiveDetector(name),Detector(det) {
  G4String HCname;
  collectionName.insert(HCname="trackerCollection");
 // fileOut=fileIOSingleton::instance();

}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
DNATrackerSD::~DNATrackerSD(){ }

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNATrackerSD::Initialize(G4HCofThisEvent* HCE) {
  trackerCollection = new DNATrackerHitsCollection
    (SensitiveDetectorName,collectionName[0]);

  static G4int HCID = -1;
  if(HCID<0)
    { HCID = G4SDManager::GetSDMpointer()->GetCollectionID(collectionName[0]); }  HCE


  // fileOut-> fout<<G4endl;

}


//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
G4bool DNATrackerSD::ProcessHits(G4Step* aStep, G4TouchableHistory*
ROhist) {
  G4double edep = aStep->GetTotalEnergyDeposit();
  G4double stepl = aStep->GetStepLength();

  if((edep==0.) && (stepl==0.)) return false;
```

```
  G4TouchableHistory* theTouchable
    = (G4TouchableHistory*)(aStep->GetPreStepPoint()->GetTouchable());
  G4VPhysicalVolume* physVol = theTouchable->GetVolume();

  DNATrackerHit* newHit = new DNATrackerHit();
  if (physVol == Detector->GetChamber()) newHit->AddCham(edep,stepl);



  //newHit->SetTrackID(aStep->GetTrack()->GetTrackID());
  //newHit->SetChamberNb(aStep->GetPreStepPoint()->GetTouchable()
  //                     ->GetReplicaNumber());
  // newHit->SetEdep(edep);
  //newHit->SetStepl(stepl);
  //newHit->SetPos(aStep->GetPostStepPoint()->GetPosition());
  trackerCollection->insert( newHit );

  //newHit->Print();
  //newHit->Draw();

  return true;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DNATrackerSD::EndOfEvent(G4HCofThisEvent* HCE) {

  if (verboseLevel>0)
    {
     // G4int NbHits = trackerCollection->entries();
      //     G4cout << "\n-------->Hits Collection: in this event they are " <<
//NbHits
      //          << " hits in the tracker chambers: " << G4endl;
      /* for (G4int i=0;i<NbHits;i++)

      (*trackerCollection)[i]->OutFile();*/

    }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
//DNATrackingAction.cc
//2004.23.04
//
```

```
#include "G4Field.hh" #include "G4TransportationManager.hh" #include
"G4UniformMagField.hh" #include "G4FieldManager.hh" #include
"DNATrackingAction.hh" #include "G4TrackingManager.hh" #include
"G4Track.hh" #include "G4UserTrackingAction.hh" #include
"G4ParticleDefinition.hh" #include "G4UniformMagField.hh" #include
"G4ThreeVector.hh" #include "G4Step.hh" #include
"DNASteppingAction.hh"

DNATrackingAction::DNATrackingAction() {
  fileOut = fileIOSingleton::instance();
  // field->SetFieldValue();
  fout.open("gpsmomentum.dat");
  pi = 4.*atan(1.);
}

DNATrackingAction::~DNATrackingAction() {
  fout.close();
}


void DNATrackingAction::PreUserTrackingAction(const G4Track* aTrack)
{
  using namespace std;
  // Create trajectory only for primaries

  G4int ID=aTrack->GetParentID();
  if(ID==0)
    //  { fpTrackingManager->SetStoreTrajectory(true); }
    // else
    // { fpTrackingManager->SetStoreTrajectory(false); }
    {
      G4ParticleDefinition* particle = aTrack->GetDefinition();
      G4String particleName = particle->GetParticleName();
      G4double particleEnergy = aTrack->GetKineticEnergy();
      G4ThreeVector particleMomentum = aTrack->GetMomentum();
      G4ThreeVector particleMDirection = aTrack->GetMomentumDirection();

      G4double p = particleMomentum.mag();
      G4double theta = particleMomentum.theta();
      G4double phi = particleMomentum.phi();

      fout << p/MeV << "  " << pi - theta << "  " << phi << endl;

      /* fileOut->fout << "The input particle is:" << particleName << G4endl;
```

```
        fileOut->fout << "The input energy is:" << particleEnergy <<" MeV"<< G4endl;
        fileOut->fout << "The input momentum is:" << particleMomentum<< G4endl;
        fileOut->fout << "The input momentum direction is:" << particleMDirection<< G
      */

        InputE =  particleEnergy;
        InputM =  particleMomentum;
       //        fileOut->fout << "chambernumber        " << " energy deposit
       //         << " position " << G4endl;

    }
}

void DNATrackingAction::PostUserTrackingAction(const G4Track*
aTrack) {
    G4int ID=aTrack->GetParentID();
     if(ID==0)
    //  { fpTrackingManager->SetStoreTrajectory(true); }
    // else
    // { fpTrackingManager->SetStoreTrajectory(false); }
             // for(i=0;i<numberofsteps;i++)
             //  G4double edep=aStep->GetTotalEnergyDeposit();
             //  G4double Range=aStep->GetStepLength();
             //  G4double LET = edep/range
             //  }
    {
      //G4ParticleDefinition* particle = aTrack->GetDefinition();
      //G4LogicalVolume* logVol = aTrack->GetLogicalVolumeAtVertex();
      //G4String logVolName = logVol->GetName();
     // G4cout << "Logical volume at vertex: " << logVolName <<G4endl;
     // G4String particleName = particle->GetParticleName();
     // G4double particleEnergy = aTrack->GetKineticEnergy();
     // G4double trackLength = aTrack->GetTrackLength();
     // G4ThreeVector particleMomentum = aTrack->GetMomentum();
      //G4ThreeVector particleMDirection = aTrack->GetMomentumDirection();
      //G4ThreeVector p = particleMDirection;
      //G4double X = p.getX();
      //G4double Y = p.getY();
      //G4double Z = p.getZ();

    /* if ( Z > 0.0)

        {G4double Direction = tan(s
        { G4double Direction = 3.1415926/2.;
```

```
          fileOut->fout <<(InputE - particleEnergy)/trackLength <<"       "  << Direc

      else
         {G4double Direction =3.1415926 - tan(sqrt(X*X + Y*Y)/abs(Z));
          fileOut->fout <<(InputE - particleEnergy)/trackLength << "       "  << Dire
    */
     }
}qrt(X*X + Y*Y)/abs(Z));
         fileOut->fout <<(InputE - particleEnergy)/trackLength << "       "  << Direc

      else if ( Z == 0 )
```

Appendix B

ROOT macros

```
    //Read data from a file and draw 1-D histogram multiplots
{
  gROOT->Reset();

#include <fstream.h>

   ifstream in1;

  in1.open("KEelectrons.dat");

  Float_t x1,x2,x3;

  //Int_t ntheta = 0;

  TH1F *h1 = new TH1F("","",100,0,350.);
  h1->SetLineColor(48);
  h1->SetLineWidth(2);
  h1->SetXTitle("Energy (MeV)");
  h1->SetYTitle("Counts");

 TH1F *h2 = new TH1F("","",100,0,5);
 h2->SetLineColor(45);
 h2->SetLineWidth(2);
 h2->SetXTitle("Energy (MeV)");
 h2->SetYTitle("Counts");

 TH1F *h3 = new TH1F("","",100,0,70);
```

```
 h2->SetLineColor(45);
 h2->SetLineWidth(2);
 h2->SetXTitle("Energy (MeV)");
 h2->SetYTitle("Counts");

 while (1)
  {
    in1 >> x1 >> x2 >> x3;
    if (!in1.good()) break;


  //  h1->Fill(x1);
  //  h2->Fill(x2);
    h3->Fill(x3);

  }

   in1.close();

TCanvas* c1 = new TCanvas("c1","Energy",200,10,800,600);
  // h1->Draw();
  // h2->Draw("same");
   h3->Draw();
   c1->Modified();
   c1->Update();

}

// 2-D graph multi plots
{
   gROOT->Reset();
   gStyle->SetOptFit();
   c1 = new TCanvas("c1","multigraph",200,10,350,350);
   //c1->SetGrid();

      // edeposit per one event
   TMultiGraph *mg = new TMultiGraph();

      // 30 keV create first graph
   Int_t n1 = 9;
   Double_t x1[]  = {1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0};
   Double_t y1[]  = {136.55,163.86,196.91,227.92,245.18,243.56,229.51,204.00,163.86};
   //Double_t ex1[] = {.05,.1,.07,.07,.04};
   //Double_t ey1[] = {.8,.7,.6,.5,.4};
```

```
  // gr1 = new TGraphErrors(n1,x1,y1,ex1,ey1);
   gr1 = new TGraphErrors(n1,x1,y1);
   gr1->SetMarkerColor(kBlue);
   gr1->SetMarkerStyle(21);
   //gr1->Fit("pol6","q");
   mg->Add(gr1);

      // 50 keV create second graph
   Int_t n2 = 9;
   Float_t x2[]  = {1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0};
   Float_t y2[]  = {76.79,82.30,87.08,91.44,96.28,98.35,98.48,96.01,89.577};
   //Float_t y2[]  = {0.06,0.242,0.541,0.92,1.368,1.88,2.4,2.919,3.438};
   //Float_t ex2[] = {.5,.0001,.0001,.0001,.0001,.0001};
   //Float_t ey2[] = {};
   //gr2 = new TGraphErrors(n2,x2,y2,ex2,ey2);
   gr2 = new TGraphErrors(n2,x2,y2);
   gr2->SetMarkerColor(kBlack);
   gr2->SetMarkerStyle(20);
   //gr2->Fit("pol5","q");
   mg->Add(gr2);


   // 80 keV create third graph
   Int_t n3 = 9;
   Float_t x3[]  = {1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0};
   Float_t y3[]  = {52.55,54.43,55.54,57.11,58.11,59.28,60.10,60.91,59.62};
   //Float_t ex3[] = {.04,.12,.08,.06,.05};
   //Float_t ey3[] = {.6,.8,.7,.4,.3,};
   //gr3 = new TGraphErrors(n3,x3,y3,ex3,ey3);
   gr3 = new TGraphErrors(n3,x3,y3);
   gr3->SetMarkerColor(kRed);
   gr3->SetMarkerStyle(20);
   //gr3->Fit("pol5","q");

   mg->Add(gr3);

   mg->Draw("acp");

     //force drawing of canvas to generate the fit TPaveStats
   c1->Update();
/*
   TPaveStats *stats1 = (TPaveStats*)gr1->GetListOfFunctions()->FindObject("stats");
   TPaveStats *stats2 = (TPaveStats*)gr2->GetListOfFunctions()->FindObject("stats");
   //TPaveStats *stats3 = (TPaveStats*)gr3->GetListOfFunctions()->FindObject("stats")
```

```
   stats1->SetTextColor(kBlue);
   stats2->SetTextColor(kRed);
   //stats3->SetTextColor(kBlack);
   stats1->SetX1NDC(0.12); stats1->SetX2NDC(0.32); stats1->SetY1NDC(0.75);
   stats2->SetX1NDC(0.72); stats2->SetX2NDC(0.92); stats2->SetY1NDC(0.78);
   //stats3->SetX1NDC(0.72); stats3->SetX2NDC(0.92); stats3->SetY1NDC(0.78);
  */
   leg = new TLegend(0.2,0.1,0.3,0.3);
   leg->AddEntry(gr1,"30 keV","lp");
   leg->AddEntry(gr2,"50 keV","lp");
   leg->AddEntry(gr3,"80 keV","lp");
   leg->Draw();
   //leg->AddEntry(fun2,"#sqrt{2#pi} P_{T} (#gamma)  latex formula","f");
    // and add a header (or "title") for the legend
   // leg->SetHeader("The Legend Title");
    leg->Draw();

   c1->Modified();
}
 // Plot Bethe Bloch formula
#include<iostream.h> #include<stdlib.h> #include <string.h> #include
<fstream.h> #include <math.h>
//#include <conio.h>

using namespace std;

TH2F *h1 = new TH2F("h1","",100,1,10,100, 0,20.0);
  h1->SetLineColor(4);
  h1->SetLineWidth(15);
  h1->SetXTitle("Electron Energy (MeV)");
  h1->SetYTitle("Stopping Power (MeV cm2/g) ");
  h1->SetFillColor(45);
  h1->SetMarkerSize(1);
  h1->SetMarkerStyle(5);
  h1->SetMarkerColor(5);


void BetheBlochPlotforConcrete()

{
  ofstream out;

  Int_t atonum = 6119;
  Double_t iPot = 135.2.0E-6;
```

```
  Double_t resTene1 = 9.1E-28;
  Double_t massNum = 12165.79;
  Double_t AvNum = 6.022E+23;
  Double_t nBele = (AvNum*atonum)/massNum;
  Double_t pi = 3.1415926;
  Double_t den = 2.3;
  Double_t epsilon = 8.8E-12;
  Double_t Ene = 50000.0;
  Double_t elecharge = 1.6E-19;
  Double_t speedlight = 3E+8;
  Double_t Totdep = 0.0;

  for (Double_t stepleng=6.4; stepleng<200; stepleng+=6.4) {

   Double_t R = (nBele*elecharge*elecharge*elecharge*elecharge*den*(6.24E+12)*(6.24E+
   Double_t Beta2 = 1-((105.6*105.6)/(Ene*Ene));
   Double_t part1 = ((2*0.511*Beta2)/iPot);
   Double_t part2 = 1-(Beta2);
   Double_t Eloss = (((R*stepleng*(1E+4)*(1E-1))/(0.511*Beta2))*(log(part1)-(log(part


   cout << "eloss: " << Eloss  <<  "  Energy "    << Ene << endl;
   h1->Fill(stepleng,Eloss);
   Ene=Ene-Eloss;
   Totdep = Totdep + Eloss ;
 }
   TCanvas *c = new TCanvas("c","Graph2D example",0,0,600,400);
   h1->Draw();
   cout <<  "  Energy of exiting particle "    << Ene << endl;
   cout <<  "  Total energy deposit "    << Totdep << endl;
}
 // Calculate number of electrons in each energy range
{ #include <fstream.h> gROOT->Reset(); ifstream in1;
//in1.open("xxx.dat");
in1.open("../../geant4/bin/Linux-g++/aaa1.dat"); Float_t
x1,x2,x3,x4,x5; string id; int xcount1=0; int xcount2=0; int
xcount3=0; int xcount4=0; int xcount5=0; int xcount6=0; int
xcount7=0; int xcount8=0;

while (in1 >> id >> x1 >> x2 >> x3 >> x4 >> x5) {
  //  cout << "id: " << id << endl;
  if(id=="e-")
    {
      if(x2>=350 && x2<400)
```

```
        xcount1++;
      if(x2>=300 && x2<350)
    xcount2++;
      if(x2>=250 && x2<300)
        xcount3++;
      if(x2>=200 && x2<250)
        xcount4++;
      if(x2>=150 && x2<200)
        xcount5++;
      if(x2>=100 && x2<150)
        xcount6++;
      if(x2>=50 && x2<100)
        xcount7++;
      if(x2>=0 && x2<50)
        xcount8++;

    }
  //cout << x2 << " " << x1 << endl;
} cout<<"xcount:"<< xcount << endl; in1.close(); }

// compare Bethe-bloch formula and simulation results
{ gROOT->Reset();
  c1 = new TCanvas("c1","edep",200,20,700,700);
   c1->SetGrid();

   TH2F *hr = new TH2F("","",1,1,1000,2,1,10);
   hr->SetXTitle("Energy (GeV)");
   hr->SetYTitle("Stopping Power (MeV g^{-1} cm^{2})");
   hr->Draw();
   c1->GetFrame()->SetFillColor(1);
   c1->GetFrame()->SetBorderSize(12);
  /*
  Int_t n1 = 5;

       Float_t x1[]  = {1000.0,5000.0,25000.0,50000.0,80000.0};
      Float_t y1[]  = {148.07,162.79,165.7,166.1,166.2};

   gr1 = new TGraphErrors(n1,x1,y1);
   gr1->SetMarkerColor(kBlue);
   gr1->SetMarkerStyle(21);
   gr1->Draw("LP");
```

```
   Int_t n2 = 5;

      Float_t x2[]  = {1000.0,5000.0,25000.0,50000.0,80000.0};
      Float_t y2[]  = {150.73,165.2,168.599,168.534,167.02};


   gr2 = new TGraphErrors(n2,x2,y2);
   gr2->SetMarkerColor(kBlue);
   gr2->SetMarkerStyle(22);
   gr2->Draw("LP");

*/
  Int_t n3 = 10;

//      Float_t x3[]  = {100.0,200.0,300.0,500.0,750.0,1000.0,5000.0,25000.0,50000.0,
  //    Float_t y3[]  = {970.931,193.802,279.796,477.687,651.45,704.795,721.185,731.2
      Float_t x3[]  = {1.0,2.0,3.0,5.0,7.50,10.0,50.0,250.0,500.0,800.0};
      Float_t y3[]  = {9.70931,1.93802,2.79796,4.77687,6.5145,7.04795,7.21185,7.3128,


   gr3 = new TGraphErrors(n3,x3,y3);
   gr3->SetMarkerColor(kBlue);
   gr3->SetMarkerStyle(22);
   gr3->Draw("CP");


  Int_t n4 = 10;

//      Float_t x4[]  = {100.0,200.0,300.0,500.0,750.0,1000.0,5000.0,25000.0,50000.0,
  //    Float_t y4[]  = {895.0,189.67,279.773,455.147,664.215,702.01,706.106,720.62,7
      Float_t x4[]  = {1.0,2.0,3.0,5.0,7.50,10.0,50.0,250.0,500.0,800.0};
      Float_t y4[]  = {8.950,1.8967,2.79773,4.55147,6.64215,7.0201,7.06106,7.2062,7.2


   gr4 = new TGraphErrors(n4,x4,y4);
   gr4->SetMarkerColor(kBlue);
   gr4->SetMarkerStyle(22);
   gr4->Draw("CP");


   leg = new TLegend(0.2,0.1,0.3,0.3);
    //leg->AddEntry(gr1,"BetheBloch-1mWater","lp");
    //leg->AddEntry(gr2,"LCDCS-1mWater","lp");
    leg->AddEntry(gr3,"BetheBloch-2mConcrete","lp");
```

```
    leg->AddEntry(gr4,"LCDCS-2mConcrete","lp");
    leg->Draw();

}

//Energy depositon read from file
{
  gROOT->Reset();

#include <fstream.h>

  ifstream in1;
  in1.open("Pune_10cm_100slices_30MeV_mu_03-16-05.dat");

  Float_t x1,x2,x3,x4,x5;
  char id[10];

  //Int_t ntheta = 0;

  TH1F *h1 = new TH1F("h1","Energy deposit in each step",100,0.1,10);
  h1->SetLineColor(48);
  h1->SetLineWidth(2);
  h1->SetXTitle("Energy (MeV)");
  h1->SetYTitle("Counts");

 // TH1F *h2 = new TH1F("h2","Scattering angle",100,0,10);
  //h2->SetLineColor(45);
  //h2->SetLineWidth(2);
  //h2->SetXTitle("Angle (radian)");
  //h2->SetYTitle("Counts");


 while (1)
  {
    in1 >> x1 >> x2 >> x3 >> x4 >> x5 >> id;
    if (!in1.good()) break;
    //if ( (x1 == 9999) && (x2 == 9999) && (x3 == 9999) && (x4 == 9999) && (x5 == 999
    //------------
    //else

    h1->Fill(x4);
    //h2->Fill(thetaPb);
 //momentum and energy
{ #include "fstream.h"
```

```
gROOT->Reset(); Int_t n = 0;

Float_t d = 2.7;

TCanvas *c1 = new TCanvas("c1","dE/dx & E",200, 100, 600, 800);

ifstream in1; in1.open("EnergyDep.dat");

ifstream in2; in2.open("EnergyDep.dat");


Float_t energy[100],loss[100],e2[100],loss2[100];

  while (1)
  {
    in1 >> energy[n] >> loss[n];
    in2 >> e2[n] >> loss2[n];
    if (!in1.good()) break;
    n = n + 1;

  }
cout << "n = " << n << endl;

TGraph *Edepend =new TGraph(n,energy,loss); TGraph *Edepend2 =new
TGraph(n,e2,loss2);

//TAxis *t;
//t->GetX()->SetRange(1,2000);
c1->SetLogx(); c1->SetLogy(); Edepend->SetLineColor(6);
Edepend->SetMaximum(1000); Edepend->SetMinimum(1);

Edepend->Draw("ACP");

Edepend2->SetLineColor(4); Edepend2->Draw("CP");


in1.close(); in2.close();

}

//Read mid points of sliced cubes
#include<iostream.h> #include<stdlib.h> #include <string.h> #include
<fstream.h> #include <conio.h>
```

```
using namespace std;



struct Cube{
    double x;
    double y;
    double z;
    int xSlices;
    int ySlices;
    int zSlices;
};

void main()

{

    char FileName[20];
    fstream outfile;
    int i, j, k;
    Cube cube;


    cout<<"\nEnter the Filename:\t";
    cin>>FileName;

    cout<<"\nEnter the side lengths of the cube:\t";
    cin>>cube.x>>cube.y>>cube.z;

    cout<<"\nEnter the number of slices in side x, y, z:\t";
    cin>>cube.xSlices>>cube.ySlices>>cube.zSlices;

    outfile.open(strcat(FileName, ".dat"),ios::out);

    for (i=1;i<=cube.xSlices; i++)
        for (j=1;j<=cube.ySlices; j++)
            for (k=1;k<=cube.zSlices; k++)
                outfile<<i<<","<<j<<","<<k<<"\t"<<(2*i-1)*0.5*cube.x/cube.xSlices<<"\
    outfile.close();
}

//Validation
{
```

```
gROOT->Reset();
gStyle->SetOptFit();
c1 = new TCanvas("c1","multigraph",200,10,350,350);
//c1->SetGrid();

   // draw a frame to define the range
TMultiGraph *mg = new TMultiGraph();

   // create first graph
Int_t n1 = 6;
Double_t x1[]  = {2.0,3.0,4.0,5.0,6.0,7.0};
Double_t y1[]  = {0.7,1.025,1.475,1.95,2.525,3.225};
//Double_t ex1[] = {.05,.1,.07,.07,.04};
//Double_t ey1[] = {.8,.7,.6,.5,.4};
// gr1 = new TGraphErrors(n1,x1,y1,ex1,ey1);
gr1 = new TGraphErrors(n1,x1,y1);
gr1->SetMarkerColor(kBlue);
gr1->SetMarkerStyle(21);
//gr1->Fit("pol6","q");
mg->Add(gr1);

   // create second graph
Int_t n2 = 6;
Float_t x2[]  = {2.0,3.0,4.0,5.0,6.0,7.0};
Float_t y2[]  = {0.55,0.9,1.32,1.9,2.55,3.26};
//Float_t y2[]  = {0.067,0.392,0.885,1.526,2.263,2.95,3.642,4.3681,5.069};
//Float_t ex2[] = {.5,.0001,.0001,.0001,.0001,.0001};
//Float_t ey2[] = {};
//gr2 = new TGraphErrors(n2,x2,y2,ex2,ey2);
gr2 = new TGraphErrors(n2,x2,y2);
gr2->SetMarkerColor(kBlack);
gr2->SetMarkerStyle(20);
//gr2->Fit("pol5","q");
mg->Add(gr2);


// create third graph
Int_t n3 = 6;
Float_t x3[]  = {2.0,3.0,4.0,5.0,6.0,7.0};
Float_t y3[]  = {0.4,0.8,1.3,1.85,2.51,3.2};
//Float_t ex3[] = {.04,.12,.08,.06,.05};
//Float_t ey3[] = {.6,.8,.7,.4,.3,};
//gr3 = new TGraphErrors(n3,x3,y3,ex3,ey3);
gr3 = new TGraphErrors(n3,x3,y3);
```

```
   gr3->SetMarkerColor(kRed);
   gr3->SetMarkerStyle(20);
   //gr3->Fit("pol5","q");

   mg->Add(gr3);

   mg->Draw("acp");

     //force drawing of canvas to generate the fit TPaveStats
   c1->Update();
/*   TPaveStats *stats1 =
(TPaveStats*)gr1->GetListOfFunctions()->FindObject("stats");
   TPaveStats *stats2 = (TPaveStats*)gr2->GetListOfFunctions()->FindObject("stats");
   //TPaveStats *stats3 = (TPaveStats*)gr3->GetListOfFunctions()->FindObject("stats")
   stats1->SetTextColor(kBlue);
   stats2->SetTextColor(kRed);
   //stats3->SetTextColor(kBlack);
   stats1->SetX1NDC(0.12); stats1->SetX2NDC(0.32); stats1->SetY1NDC(0.75);
   stats2->SetX1NDC(0.72); stats2->SetX2NDC(0.92); stats2->SetY1NDC(0.78);
   //stats3->SetX1NDC(0.72); stats3->SetX2NDC(0.92); stats3->SetY1NDC(0.78);
   */

   leg = new TLegend(0.2,0.1,0.3,0.3);
   leg->AddEntry(gr1,"Expt (Wilson et al 2005)","lp");
   leg->AddEntry(gr2,"Geant4","lp");
   leg->AddEntry(gr3,"Calc (Wilson et al 2005)","lp");
   leg->Draw();
   //leg->AddEntry(fun2,"#sqrt{2#pi} P_{T} (#gamma)  latex formula","f");
    // and add a header (or "title") for the legend
   // leg->SetHeader("The Legend Title");
    leg->Draw();

   c1->Modified();
}


//Read from file
#include <iostream> #include <fstream> using namespace std;

int main() {

  ifstream myFile;

  myFile.open("myData.log");
```

```
  float  x,y,z;
  int nlines = 0;
  //TFile *f = new TFile("basic.root","RECREATE");
  //TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
  //TNtuple *ntuple = new TNtuple("ntuple","data from ascii file","x:y:z");

  while (1) {
    myFile >> x >> y >> z;
    if (!myFile.good()) break;
    if (nlines < 20) printf("x=%8f, y=%8f, z=%8f \n",x,y,z);
    //h1->Fill(x);
    //ntuple->Fill(x,y,z);
    nlines++;
  }
  printf(" found %d pointsn \n ",nlines);

  myFile.close();

  return 0;
  //f->Write();
}
```