A GEANT4 Simulation of the Recoil Proton Stopping Acceptance in the Near-Threshold $\gamma p \rightarrow \pi^0 p$ Reaction on a Scintillator Target

by

Minh Thanh Thái

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1999

©1999 Minh Thanh Thái
All Rights Reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of the thesis document in whole or in part.

Signature of Author

Department of Physics
May 19, 1999

Certified by

Aron M. Bernstein, Prof.
Thesis Supervisor, Department of Physics

Certified by

Marcello M. Pavan, Post Doc.
Thesis Supervisor, Department of Physics

Accepted by

David E. Pritchard, Prof.
Senior Thesis Coordinator, Department of Physics
A GEANT4 Simulation of the Recoil Proton Stopping Acceptance in the Near-Threshold $\gamma p \rightarrow \pi^0 p$ Reaction on a Scintillator Target

by

Minh Thanh Thái

Submitted to the Department of Physics on May 20, 1999, in partial fulfillment of the requirements for the degree of Bachelor of Science in Physics

Abstract

The simulation of the recoil proton stopping acceptance in the near-threshold $\gamma p \rightarrow \pi^0 p$ reaction on a $CH_{1.1}$ scintillator target has been carried out using the new state-of-the-art CERN GEANT4 toolkit. The photo-production cross-section was generated to be uniform in $\cos \theta_{CM}$ and $\phi_{CM}$. Incident photons with energy uniformly distributed in the range of 150MeV to 250MeV were used. The simulation focused on the effects of the size of the incident photon beam. In additions, the hardening efficiency of the bremsstrahlung photon beam using beryllium absorber was studied.

Thesis Supervisor: Aron M. Bernstein
Title: Professor of Physics

Thesis Supervisor: Marcello M. Pavan
Title: Post Doctoral Fellow
Acknowledgments

First and foremost, I would like to thank Prof. Aron Bernstein for his support over the three years I have worked with him. He has always been available to answer questions and to give advice, and he has never been short of new and exciting projects to work on.

Next, I am grateful to Marcello Pavan, Ph.D., who has worked with me side by side all these years. Especially, his help has been invaluable to me finishing this work.

Finally, I am deeply indebted to my parents for providing me with the opportunity to pursue the study of physics. Emotionally, they have always been there to help me through all the struggles I faced in school.
Contents

1 Introduction ........................................... 8
   1.1 The experiment .................................. 9
   1.2 Monte Carlo modeling .......................... 10
   1.3 GEANT4, a simulation tool ....................... 11

2 Elements of the Simulation ............................ 13
   2.1 Geometry ........................................ 13
   2.2 Photon beam ...................................... 15
   2.3 Physics processes ............................... 16
      2.3.1 Neutral pion decay process ................... 17
      2.3.2 Electromagnetic interaction: General .......... 17
      2.3.3 Electromagnetic interaction: Gamma incident .... 18
      2.3.4 Electromagnetic interaction: Electron incident ... 21
   2.4 Neutral pion photo-production on the proton simulation .. 24
      2.4.1 Reaction modeling ........................ 24
      2.4.2 Kinematics ............................... 25

3 Results and Discussion ............................... 28
   3.1 Recoil proton stopping acceptance ................... 28
      3.1.1 Polar angle distribution .................... 28
      3.1.2 Proton kinetic energy and energy deposit ...... 36
      3.1.3 Recoil proton stopping acceptance ............... 39
   3.2 Hardening photon beam .......................... 47
List of Figures

2-1 The geometry setup of the simulation. ................. 14

3-1 Distribution of polar angle when the recoil protons stop or escape the target for 150MeV photons. ......................... 29

3-2 Distributions of polar angle when the recoil protons stop or escape the target for 250MeV photons. ......................... 29

3-3 Distributions of total energy deposit on polar angle for 150MeV. Middle: recoil protons start from the center of the target. Bottom: no energy deposit to the target from photons (from $\pi^0$) interaction. . . . 31

3-4 Surface plots of the distribution of total energy deposit on polar angle for 150MeV photons. Bottom: recoil protons start from the center of the target. ........................................ 32

3-5 Distributions of total energy deposit on polar angle for 250MeV. Middle: recoil protons start from the center of the target. Bottom: no energy deposit to the target from photons. ..................... 34

3-6 Surface plots of the distribution of total energy deposit on polar angle for 250MeV photons. Bottom: recoil protons start from the center of the target. ........................................ 35

3-7 Distribution of total energy deposit on kinetic energy of recoil protons. Bottom: recoil protons start from the center of the target ....... 37

3-8 Surface plot of the distribution of total energy deposit on kinetic energy of recoil protons. Bottom: recoil protons start from the center of the target ................................. 38
3-9 Distributions of photon energy on polar angle. Top: all protons. Bottom: stopped protons only. ............................................. 40

3-10 Surface plots of the distributions of photon energy on polar angle. Top: all protons. Middle: stopped protons only. Bottom: stopping acceptance (ratio stop/all). ............................................. 41

3-11 Distribution of photon energy on the polar angle when recoil protons are stopped in the target, along with theoretical line when proton kinetic energies are 44MeV and 31MeV. Bottom: recoil protons start from the center of the target. ............................................. 43

3-12 Distribution of photon energy on the polar angle when recoil protons escape the target, along with theoretical line when proton kinetic energies are 44MeV and 31MeV. Bottom: recoil protons start from the center of the target. ............................................. 44

3-13 Distribution of the difference between proton kinetic energy and energy deposit, along with theoretical line when proton kinetic energies are 44MeV and 31MeV. Middle: recoil protons start from the center of the target. Bottom: narrow incident photon beam. ............................................. 46

3-14 The photon energy distribution with and without a 20cm(top) and 40cm(bottom) long Beryllium absorber. ............................................. 48
Chapter 1

Introduction

The main goal of this thesis is to initiate a sophisticated simulation of the experimental apparatus to be used in a proposed measurement of polarized target asymmetries in the threshold neutral pion photo-production on the proton, $\gamma p \rightarrow \pi^0 p$. This experiment will be performed at the MAMI laboratory in Mainz, Germany. The Monte Carlo simulation described herein was implemented using the new state-of-the-art GEANT version 4 simulation toolkit from CERN. Using this simulation, a simple study is made of the recoil proton detection acceptance in the active scintillator target, as well as the effect of adding a beryllium absorber to the bremsstrahlung incident photon beam.

This chapter will continue with a brief description of the experiment, the Monte Carlo modeling, and the software.

Chapter two describes of all the elements of the simulation including the geometry of the experiment apparatus, the characteristic of the incident photon beam as well as the physical processes handled by the simulation.

Chapter three presents the results from the simulation and comparison to analytical predictions are made. Some conclusion will me discussed in Chapter four.
1.1 The experiment

In the experiment of threshold neutral pion photo-production on the proton, a beam of bremsstrahlung photons hits a scintillator target, upon which recoil proton and scattered neutral pion are detected in the scintillator and in an external $BaF_2$ crystal spectrometer (TAPS [Pierre94]), respectively. The goal of the experiment is to separate the desired reaction $\gamma p \rightarrow \pi^0 p$ from the reaction $\gamma C \rightarrow \pi^0 p^{11} B$ in the target. This is why is crucial to understand the recoil proton stopping distribution. The incident photon beam is created by bombarding a high energy electron beam onto a thin gold foil. The scattered electrons are swept off the beam by a magnetic field “tagger”, leaving the photon beam in the forward direction. The bremsstrahlung photon beam has the characteristic spectrum in which the majority of the photons are “soft”, having close to zero momentum. Threshold neutral pion photo-production, however, requires incident photons with momentum greater than 144.7MeV. This implies that most of bremsstrahlung photons are not energetic enough to cause the desired photo-production. Nonetheless, the low energy photons can cause unwanted Compton scattering and pair production events in the scintillator target. Therefore, it is necessary to “harden” the photon beam by filtering the photon beam through a beryllium block before hitting the target. The effect is a strongly preferential cutoff in the low energy spectrum with respect to the high energy photons.

The target is made of scintillator material $CH_{1.1}$. In the photo-productions of a neutral pion on the proton, the recoil protons are detected inside the target. The recoil protons traveling inside the target interact with the molecular electrons and create light that can be collected and transported through a light-guide to a photo-multiplier tube on top of the target. The energy loss of the recoil protons is then proportional to the read-out voltage of the photo-multiplier tube. If the proton stops in the target, this energy loss is equal to the proton’s initial kinetic energy.

The neutral pion, having a short lifetime of $8.4 \times 10^{-17}$sec, will decay almost instantaneously into a pair of gammas. These gammas travel in the opposite directions in the pion’s rest frame. As a result, by measuring their momentum, one can obtain
the momentum of the neutral pion produced. In the experiment, this is achieved by coincident detection of the pair of gammas using the TAPS (Two/Three Arms Photon Spectrometer [Pierre94]) placed surrounding the target. For the purpose of our simulation, we assume complete neutral pion acceptance. In other words, all produced pions are detected.

1.2 Monte Carlo modeling

Monte Carlo simulation techniques can be used to help design a new experiment, as well as to verify the results of some experiments. In many cases, the simulations provide physicists with a better understanding of the physical processes taking place in an experimental apparatus.

In a real experiment, many parameters are not measurable or not easily measurable. Monte Carlo simulations enable one to test the effect of various hypotheses on the experimental outcome. The simulations can be designed to provide predictions of the detector responses according to different hypotheses. In terms of designing a new experiment, this helps determining the feasibility of the experiment using certain apparatus. In the case of analyzing experimental results, the simulations can help verifying a hypothesis, filter out bad output resulting from errors or calibrating over some systematic noise.

Software simulations provide a very effective tool for studying the detector response. With proper modeling of the detector, the simulation program can provide outputs that can be compared against real readouts from the experiment. However, in the simulation, one has the flexibility to add or remove the physical processes that govern the detector response. These processes can correspond to the main reaction or some background reaction that would contribute to the output as noise. Being able to evaluate different factors in the experiment individually allows the designer to improve the response by focusing on those factors that affect the output the most. Modification of the apparatus in the search for improvement can first be simulated, allowing lower cost and time required to test out new apparatus. In addition, the
simulation also can help determine unforeseen problems with the design.

With the ability to predict the detector response, Monte Carlo simulations provide an excellent tool for parameter optimization. A simulation program running on a fast computer can carry out the experiment with different parameters in a short amount of time to determine the best response. These parameters can be the dimensions of some components in the apparatus, or different setups on the measuring equipment, etc...

Software simulations are very good at dealing with random fluctuations on various components of the experiment. By repeating the simulations a large number of times, high statistical accuracy can be obtained. In a simulation, the source of fluctuations on the output can be traced back to some random variation in the components by allowing each parameter to vary individually.

In some experiments, the outputs depend on so many factors including the geometry of the apparatus, different physics involved, random fluctuations that it is extremely difficult to predict the results analytically. In a simulation, all elements are folded together, and the outputs are provided as the results of the combination of all the knowledge about the components of the experiment. In other words, the output from a simulation is only as good as our knowledge of an experiment. If the predicted output is significantly different from the obtained results then there is something more about the experiment that is yet to be determined or understood.

1.3 GEANT4, a simulation tool

The simulation in this project was implemented using GEANT version 4. This is a new state-of-the-art Monte Carlo simulation toolkit developed by scientists at CERN [CERN].

GEANT version 4 is big step up from version 3, most noticeably the programming language being switched from FORTRAN to C++. This introduces object oriented programming along with completely new data structures and new methods for modeling experiments. Switching to C++ also allow the codes that execute much faster than before; hence, able to handle more complex problems. It is also less time
consuming to model an experiment in GEANT4, thanks to C++’s reusability.

GEANT4 is capable of performing sophisticated simulation in a short period of time, by using routines optimized for fast simulations. The optimization is achieved by employing data structures that offer high-speed access, as well as efficient data transfering during the simulation. In addition, the modeling of the geometry of the experimental apparatus can be constructed to improve tracking of particle paths.

GEANT4 allows a user to develop simulation programs that are highly customized for the user’s needs, while avoiding interference with the internal handling of the software. In other words, new physics processes can be defined externally by the user, and they would integrate seamlessly with the existing internal calculation. With GEANT4, a simulation program can be designed to carry out different experiments on the same setup, which is a real time-saving feature.

GEANT4 offers a variety of visualization tools. Along with third-party software that support new drawing interface, the task of setting up new three dimensional geometry can be made with ease. GEANT4 can support animation, connect to a CAD program.

GEANT4 provides an extensive setup data input/output format. This allows processing simulated data with more powerful data analysis software.
Chapter 2

Elements of the Simulation

This chapter describes all the components of the experiment as they are modeled in the simulation program in this thesis.

2.1 Geometry

The geometry of the experimental apparatus has been modeled after the proposed experiment setup at the MAMI laboratory in Mainz, Germany. It consists of the scintillator target, the beryllium absorber for hardening the incident photon beam, a sweeping magnet for charged particles produced in the beryllium absorber and an incident photon beam collimator (see Figure 2-1). The gold foil and the tagger magnet, as well as the detailed construction of the target (beside the scintillator) are omitted in this implementation. The incident photon beam is simulated to be uniform in energy with a uniform circular cross section 1.5cm in diameter. The TAPS $\pi^0$ spectrometer is also not included since the detection of the gamma rays escaping from the target is assumed to have 100% acceptance.

The simulation represents the components of the apparatus as volumes with different shapes, sizes, materials and other properties. A volume can contain other volumes, building up a hierarchical relational structure between components in the geometry. There is only one mother volume that contains all the components in the experiment. This volume has no parents and is called the experimental hall. In this
Figure 2-1: The geometry setup of the simulation.
work, the hall is implemented as a rectangular box with sufficient volume to contain all the apparatus. The hall is filled with air at very low pressure. The air is defined here as a gas mixture of 80% oxygen and 20% nitrogen.

The beryllium absorber used for hardening the photon beam is placed 50.0cm after the entrance of the bremsstrahlung photon beam in the experimental hall. It is implemented as a solid cylinder with axis along the incident beam. The main axis of the tube is aligned on the center photon beam. The tube has a diameter of 3.0cm and a length of 20.0cm or 40.0cm, depending on the different experiment setup.

A collimator is position right after the beryllium absorber along the beam to block most of the off-axis particles resulting from the interaction between the photon beam and the absorber, and in order to maintain a narrow photon beam. The collimator is a hollow tube made of lead. The diameter of the opening along the main axis of the tube is 0.5cm. The collimator is 10.0cm long, and 3.0cm in outer diameter.

The target is implemented as a solid cylinder 1.8cm long and 1.8cm in diameter. It is positioned 250cm downstream from the collimator with its main axis oriented perpendicular to the beam direction. The scintillator material used for the target is $CH_{1.1}$, a polymer chain of molecules with 10 carbon atoms and 11 hydrogen atoms.

A sweeping magnet is placed between the collimator and the target. It is used to clear out all the charged particles in the photon beam created in the beryllium absorber that would hit the target. The magnetic field should be strong enough to completely sweep all the charged particles with the momentums in the range specified by the experiments. Since it is safe to assume that no charged particle will pass the sweeping magnet, it is simplified in the simulation as an empty box filled with vacuum. Any charged particle entering this volume will be taken out of the simulation program.

### 2.2 Photon beam

In the real experiment, incident photon beam on the target has the bremsstrahlung characteristic energy spectrum and angular distribution. Since an individual event of a photon hitting the target is not affected by the charateristic probability distri-
bution, the same statistical results can be obtained by generating the photon beam with a uniform energy distribution on the same range, and multiplying the distribution of the results with the bremsstrahlung distribution. This is done since the bremsstrahlung energy spectrum is very strongly peaked at zero energy, with a characteristic $\sim 1/E_\gamma$ falloff, so a simulation of the true spectrum would spend most of its time generating uninteresting, very low energy gammas. Based on this observation, this simulation only generates incident photon beam with uniform energy distribution between $150\text{MeV}$ and $250\text{MeV}$. Because of the setup of the geometry the angular acceptance of the incident beam on the target is very narrow, it is then safe to ignore the Bremsstrahlung angular distribution and assume a divergenceless beam. The beam diameter is 1.5cm, slightly smaller than the target.

2.3 Physics processes

The GEANT4 toolkit provides the user with an extensive set of physics processes. Each particle is associated with a set of processes that it can undergo. There are two major groups of processes that this simulation is concerned with: the decay processes and the electromagnetic processes. The simulation program keeps track of all the particles. It constructs the path of a particle in the simulation geometry by taking steps inside some materials and transitions from one to another. At each step, based on the cross-section of a reaction of a particle in the current conditions (material, different fields, etc...), the program would make the decision whether or not to trigger a process. The calculation involves comparing the path length the particle has traveled in the material to the mean free path in such material. The mean free path can be derived from the cross section of the reaction. If a process is to take place, the simulation would determine the final state of the reaction: the type of secondary particles, and their momentums. GEANT4 has a library of reaction models to help the simulation program accomplished all these calculations. To help speed up the simulation GEANT4 parameterizes most of the physics formula in forms of polynomials. This section will continue on to the presentation of the physical model
that the program uses to perform the calculation internally.

2.3.1 Neutral pion decay process

Having a lifetime of $\tau = 8.4 \times 10^{-17}\text{ sec}$, the neutral pion is the only particle in this simulation that would go through a decay process. Since the pion has such a short lifetime, it decays almost instantaneously after it is produced and only travels only a negligible distance. The mean free path $\lambda$ is calculated as

$$\lambda = \beta c \tau \gamma$$

with $\gamma = \sqrt{1 - \beta^2}$ depends on the velocity of the particle $\beta = v/c$.

There are two decay processes that the neutral pion can undergo. The first process

$$\pi^0 \rightarrow \gamma + \gamma$$

occurs 98.8% of the time. The momentum of the decay products are calculated using the phase-space decay channel.

The second process

$$\pi^0 \rightarrow \gamma + e^+ + e^-$$

happens 1.2% of the time. The calculation for the decay products utilizes the Dalitz decay channel.

2.3.2 Electromagnetic interaction: General

Charged particles in this simulation can undergo the processes of energy loss and multiple scattering.

Energy loss

The mean energy loss is calculated using the range and the inverse range tables:

$$\Delta T = T_0 - f(r_0 - \text{step})$$
where \( T_0 \) is the kinetic energy, \( r_0 \) the range at the beginning of the step step and the function \( f_T(r) \) is the inverse of the range table (i.e. it gives the kinetic energy of the particle for a range value of \( r \)). At each step, the actual loss is derived from the mean loss by adding Landau fluctuation from the model GLANDZ.

**Multiple scattering**

Multiple scattering of charged particles in material is calculated using a multiple scattering model. This model simulates the scattering of the particle after a given step in the medium, then computes the mean path length correction and the mean lateral displacement as well. The mean properties of the multiple scattering process are determined by the transport mean free path, \( \lambda \), which is a function of the energy in a given material. The true path length, \( t \), can be transformed from the geometrical path length, \( z \) as:

\[
t = -\lambda \ln \left(1 - \frac{z}{\lambda}\right)
\]

where the condition \( z < \lambda \) is required.

The mean of the cosine of the scattering angle, \( \theta \), after a true step length \( t \) is given by:

\[
<\cos \theta> = \exp \left(-\frac{t}{\lambda}\right).
\]

This multiple scattering model uses the transport free path length values calculated by Liljequist et al. [Lil87, Lil90] for electrons and positrons in the kinetic energy range \([0.1keV, 20MeV]\) in 15 materials.

**2.3.3 Electromagnetic interaction: Gamma incident**

Gamma traveling inside materials can participate in the following physics processes: photoelectric effect, Compton scattering and gamma conversion (pair production).
**Photoelectric effect**

The photoelectric effect concerns the interaction between an incident gamma and an atomic electron. The model is used for gammas with energy between $10\text{keV}$ and $50\text{MeV}$. Let $E_\gamma$ be the incident gamma energy, and $\epsilon = E_\gamma/m_ec^2$. The photoelectric total cross-section per atom has been parameterised as

$$\sigma(Z, \epsilon) = \frac{Z^\alpha}{\epsilon^\beta} F(Z, \epsilon)$$

where $\alpha$ and $\beta$ are fit parameters and $F(Z, \epsilon)$ has a functional form depending on the range of $E_\gamma$.

The mean free path is given by

$$\lambda(E_\gamma) \equiv \frac{1}{\sum(E_\gamma)} = \frac{1}{\sum_{elm}[n_{elm}\sigma(Z_{elm}, E_\gamma)$$

where $\sum_{elm}$ runs over all elements the material is made of.

In the final state, the photoelectron is emitted with kinetic energy

$$T_{\text{photoelectron}} = E_\gamma - B_{\text{shell}}(Z_{elm})$$

where $B_{\text{shell}}$ is the binding energy of the shell. The scattered electron has the same direction as the incident gamma.

**Compton scattering**

An empirical cross-section formula is used, which reproduces rather well the cross-section data down to $10\text{keV}$:

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

where:

$$Z = \text{atomic number of the medium},$$
$E_\gamma = \text{energy of the photon},$

$m_e = \text{electron mass},$

$X = \frac{E_\gamma}{m_e c^2},$

$P_i(Z) = Z(d_i + c_i Z + f_i Z^2).$

The mean free path for a photon to interact via Compton scattering is given by

$$\lambda(E_\gamma) \equiv \frac{1}{\sum (E_\gamma)} = \frac{1}{\sum_{elm}[n_{elm}\sigma(Z_{elm}, E_\gamma)]}$$

where \(\sum_{elm}\) runs over all elements the material is made of.

In the final state, the scattering angle \(\theta\) is defined by the Compton formula, assuming an elastic collision:

$$E_1 = E_0 \frac{m_e c^2}{m_e c^2 + E_0(1 - \cos \theta)}$$

where \(E_0, E_1\) are energies of the incident and scattered photons respectively. The Monte Carlo program would randomly generate the \(E_1\) according to the quantum mechanical Klein-Nishina [Klein29] differential cross-section:

$$\Phi(E_0, E_1) = \frac{X_0 n \pi r_0^2 m_e c^2}{E_0^2} \left[ \frac{1}{\epsilon + \epsilon} \right] \left[ 1 - \epsilon \sin \theta^2 \right]$$

where:

$$\epsilon = \frac{E_0}{E_1},$$

\(n = \text{number of electron per volume},\)

\(X_0 = \text{radiation length},\)

\(r_0 = \text{classical electron radius}.\)
Gamma conversion

A parameterised formula from L. Urban [Geant4] is used to estimate the total cross-section:

\[ \sigma(Z, E_\gamma) = (Z + 1)[F_1(Z) + F_2(Z)X + F_3(Z)X^2] \]

where:

\[ X = \log \frac{E_\gamma}{m_e c^2} \]

\[ F_i(Z) = a_{i0} + a_{i1}X + a_{i2}X^2 + a_{i3}X^3 + a_{i4}X^4 + a_{i5}X^5. \]

The mean free path is given by

\[ \lambda(E_\gamma) \equiv \frac{1}{\sum(E_\gamma)} = \frac{1}{\sum_{elm}[n_{elm}\sigma(Z_{elm}, E_\gamma)]]} \]

where \( \sum_{elm} \) runs over all elements the material is made of.

The secondary position and electron energies are sampled using the Bethe-Heitler cross-section with Coulomb correction.

2.3.4 Electromagnetic interaction: Electron incident

The physics processes associated with the electron and positron include ionisation, multiple scattering (see section 2.3.2), Bremsstrahlung and annihilation.

Ionisation

The total cross-section [Mess70] for Möller scattering \((e^- e^-)\) is given by:

\[ \sigma(Z, E, T_{cut}) = \frac{2\pi r_0^2 m Z}{\beta^2 (E - m)} \left[ \frac{\gamma - 1}{\gamma^2} \left( \frac{1}{x} - 1 \right) + \frac{1}{x} - \frac{1}{1-x} - \frac{2\gamma - 1}{\gamma^2} \ln \frac{1-x}{x} \right] \]

and for Bhabha scattering \((e^+ e^-)\):

\[ \sigma(Z, E, T_{cut}) = \frac{2\pi r_0^2 m Z}{E - m} \left[ \frac{1}{\beta^2} \left( \frac{1}{x} - 1 \right) + B_1 x + B_2(1-x) - \frac{B_3}{2}(1-x^2) + \frac{B_4}{3}(1-x^3) \right] \]
where
\begin{align*}
\gamma &= \frac{E}{m} \quad \beta^2 = 1 - \frac{1}{\gamma^2} \\
x &= \frac{T_{	ext{kin}}}{E - m} \quad \gamma = \frac{1}{\gamma + 1} \\
B_1 &= 2 - y^2 \quad B_2 = (1 - 2y)(3 + y^2) \\
B_3 &= (1 - 2y)^2 + (1 - 2y)^3 \quad B_4 = (1 - 2y)^3.
\end{align*}

**Bremsstrahlung**

The parameterisation for total cross-section and the energy loss for low energy (< 10 GeV) can be obtained using the tabulated cross-section values of Seltzer and Berger [Sel85] together with the Bethe-Heitler formula as:

\begin{align*}
\sigma(Z, T, k_e) &= \frac{Z(Z + \xi_\sigma)(T + m)^2}{T(T + 2m)}(\ln \frac{T}{k_e})^\alpha F_\sigma(Z, X, Y) \\
E_{\text{loss}}^{\text{Brems}} &= \frac{Z(Z + \xi_l)(T + m)^2}{T(T + 2m)}\left(\frac{k_e C_M}{T}\right)^\beta F_l(Z, X, Y)
\end{align*}

where \( m \) is the mass of the electron,

\begin{align*}
X &= \ln \frac{E}{m} \quad Y = \ln \frac{v_E}{k_e} \quad \text{for the total cross section } \sigma \\
X &= \ln \frac{E}{m} \quad Y = \ln \frac{k_e}{v_E} \quad \text{for the energy loss } E_{\text{loss}}^{\text{Brems}}
\end{align*}

with \( E = T + m \). The constants \( \xi_\sigma, \xi_l, \alpha, \beta, v_\sigma, v_l \) are parameters to be fitted. The Migdal correction factor [Mig56] is:

\[ C_M = \frac{1}{1 + \frac{n r_o \lambda r^2 (T + m)^2}{\pi k_e^2}} \]

with

\begin{align*}
\lambda &= \text{reduced electron Compton wavelength} \\
r_o &= \text{classical electron radius} \\
n &= \text{electron density in the medium}.
\end{align*}
**$e^+e^-$ annihilation**

The cross-section per atom can be derived from the cross-section formula of Heitler [Heit54]:

$$\sigma(Z, E) = \frac{Z \pi r_0^2}{\gamma + 1} \left[ \frac{\gamma^2 + 4 \gamma + 1}{\gamma^2 - 1} \ln \left( \gamma + \sqrt{\gamma^2 - 1} \right) - \frac{\gamma + 3}{\sqrt{\gamma^2 - 1}} \right]$$

- $E$ = total energy of the incident positron,
- $\gamma = \frac{E}{m_e c^2}$,
- $r_0$ = classical electron radius.

The mean free path for a photon to interact via Compton scattering is given by

$$\lambda(E, \gamma) \equiv \frac{1}{\Sigma(E, \gamma)} = \frac{1}{\sum_{elm}[n_{elm}\sigma(Z_{elm}, E, \gamma)$$

where $\sum_{elm}$ runs over all elements the material is made of.

At final state, $e^+e^- \rightarrow 2\gamma$. The gamma energy can be sampled from the differential cross-section of the two-photon positron-electron annihilation:

$$\frac{d\sigma(Z, \epsilon)}{d\epsilon} = \frac{Z \pi r_0^2}{\gamma - 1} \left( \frac{1}{\epsilon} \right) \left[ 1 + \frac{2\gamma}{(\gamma + 1)^2} - \epsilon - \frac{1}{(\gamma + 1)^2} \frac{1}{\epsilon} \right]$$

where $Z$ is the atomic number of the material, $r_0$ the classical electron radius, and $\epsilon \in [\epsilon_{min}, \epsilon_{max}]$:

$$\epsilon_{min} = \frac{1}{2} \left[ 1 - \sqrt{\frac{\gamma - 1}{\gamma + 1}} \right]$$

$$\epsilon_{max} = \frac{1}{2} \left[ 1 + \sqrt{\frac{\gamma - 1}{\gamma + 1}} \right].$$

The azimuthal angle $\phi$ is chosen isotropically, while the angle between the incident $e^+$ and the first gamma $\theta$ can be obtained from energy-momentum conservation:

$$\cos \theta = \frac{1}{P_c} \left[ T + mc^2 \frac{2\epsilon - 1}{\epsilon} \right] = \frac{\epsilon (\gamma + 1) - 1}{\epsilon \sqrt{\gamma^2 - 1}}$$
2.4 Neutral pion photo-production on the proton simulation

The $\gamma p \rightarrow \pi^0 p$ reaction is not a physics process included in the GEANT4 package. As a result, it is necessary to implement this reaction as an add-on process to the simulation program. GEANT4 provides a user with a model to describe his own process.

2.4.1 Reaction modeling

In this model, a newly defined reaction is confined to some particular volumes (components) in the geometry. It is the target in this work where the neutral pion photo-production would take place. The conditions necessary for the reaction to happen is defined in the model. The cross-section for the reaction is usually included in these conditions. In order to speed up the simulation in our work, the reaction is designed to be triggered by every photon with energy above some threshold that enter the target. The threshold photon energy for the $\gamma p \rightarrow \pi^0 p$ reaction is $144.7\,MeV$. The exact location of the interaction is also provided by the user to the program. The position is randomly chosen in the path of the photon between its entrance and exit in the target if the photon were to pass through the target. The random function used for this position is uniform, which means the reaction has equal probability of taking place anywhere on the path of the photon beam inside the target. This is an excellent approximation for thin targets.

After specifying the location of the reaction, the outcome of the reaction is passed to the simulation program by a user routine. This includes terminating the photon that undergoes the reaction and introducing secondary particles with their momentums. For neutral pion photo-production, the secondary particles are the scattered neutral pion and recoil proton. The momentums of these particles are calculated from the energy of the photon that causes the reaction and the polar angle ($\theta$) of the produced pion. The cosine of this polar angle in the center of mass frame is
randomly chosen with a flat probability distribution between \((-1, 1]\). The azimuthal angle \(\phi\) is generated isotropically. In the next step, the program will calculate for directions and kinetic energies in the laboratory frame of the produced pion and the recoil proton according to the kinematics presented in the following section. An event is aborted if the two photons, products of the pion decay, do not escape the target. This is consistent with the real experiment where the readout is discarded if the TAPS spectrometers do not register the pair of photons.

### 2.4.2 Kinematics

Neutral pion photo-production on the proton represents a two-body reaction. The initial conditions include the incident photon with momentum \(k_{\gamma}\) and the proton at rest in the laboratory frame. The energy and momentum of the particle is represented in the four vector form \(P = (E, \vec{p})\), where energy consists of the kinetic energy and the rest mass of the particle \(E = T + m\). The momentum and the energy can be related as \(E^2 - |\vec{p}|^2 = m^2\). The dot product of two four vectors \(P_1 \cdot P_2 = E_1 E_2 - \vec{p}_1 \cdot \vec{p}_2\). The four vector of the incident photon would be \(P_{\gamma} = (k_{\gamma}, \vec{k}_{\gamma})\) since \(m_{\gamma} = 0\).

The total energy \(W\) is frame invariant:

\[
W^2 = (k_{\gamma} + M_p)^2 - |\vec{k}_{\gamma}|^2 = M_p^2 + 2k_{\gamma}M_p.
\]

In the center of mass system, conservation of momentum implies that the produced pion and the recoil proton would have the momentums with same magnitude and opposite directions \(\vec{p}\) and \(-\vec{p}\). The four vectors for the produced pion, recoil proton would then be:

\[
P_{\pi^0}^* = (E_{\pi^0}^*, \vec{p}) \\
P_p^* = (E_p^*, -\vec{p}).
\]
Conservation of total energy and frame invariant implies:

\[ W = P^* + P_p^* \]
\[ \Rightarrow P_p^* = W - P^* \]
\[ \Rightarrow P_{p*}^2 = (W - P^*)^2 \]
\[ \Rightarrow M_p^2 = W^2 + m_{p*}^2 - \frac{2W \cdot P^*}{2W E_{p*}} \]

The energy of the produced pion in the center of mass frame can be derived from the above equations as:

\[ E_{p*}^* = \frac{W^2 + m_{p*}^2 - M_p^2}{2W} \]

The momentum of the pion in the center of mass frame:

\[ |\vec{p}^*| = \sqrt{E_{p*}^*^2 - m_{p*}^2}. \]

The energy of the recoil proton in the center of mass frame is:

\[ E_p^* = \sqrt{M_p^2 + |\vec{p}^*|^2}. \]

Transformation of these values from the center of mass frame will give us the momentums of the secondary particles in the laboratory frame. The transformation between the two frames is governed by the relativistic boost:

\[ \beta_{CM} \rightarrow \frac{\vec{p}}{E} = \frac{k_{\gamma}}{k_{\gamma} + M_p} \]
\[ \gamma_{CM} = \frac{1}{\sqrt{1 - |\beta_{CM}|^2}}. \]

The momentum in the laboratory frame is obtained by transforming components parallel and perpendicular to the direction of the incident photon

\[ \vec{p}' = \vec{p}_{||} + \vec{p}_{\perp} \]
\[ \vec{p} = P_{\pi} + P \]
\[ \vec{p}_{\pi} = \gamma_{CM} \left( \vec{p}_{||} + \beta_{CM} \vec{E}^* \right) \]
\[ \vec{p}_\perp = \vec{p}_{\perp}. \]

The threshold energy for the incident photon can be derived from a special case where the produced pion and the recoil proton stay at rest in the center of mass frame, \( \vec{p}_T = 0 \)

\[ W_T = M_p + m_{\pi^0} \]
\[ k_T = \frac{W_T^2 - M_p^2}{2M_p} \approx 144.7 MeV. \]
Chapter 3

Results and Discussion

A series of Monte Carlo simulations was performed to determine the recoil proton stopping acceptance from the $\gamma p \rightarrow \pi^0 p$. Three simulation runs were performed with incident photon energy set at 150MeV, 250MeV, and randomly generated in the range of 150MeV to 250MeV. Each run generated 50000 photons.

3.1 Recoil proton stopping acceptance

Several plots has been made from the data set generated by the simulation program. The relationship between the distributions of incident photon energy, the scattering angle of the pion, the energy deposited in the target and the proton kinetic energy was analyzed.

3.1.1 Polar angle distribution

Figure 3-1 and 3-2 show the histogram of the scattering angle of the pion when the photon is at 150MeV and 250MeV. At the near threshold energy most recoil protons are stopped inside the target, while at the higher energy, 250MeV, about half of the protons escape.
Figure 3-1: Distribution of polar angle when the recoil protons stop or escape the target for 150MeV photons.

Figure 3-2: Distributions of polar angle when the recoil protons stop or escape the target for 250MeV photons.
Figure 3-3 shows the distribution of the energy deposit on the polar angle, $\theta_{\pi}$ when photon energy is 150MeV. The top figure shows the distribution for a normal setup, while the middle figure the distribution when all the recoil protons start from the center of the target. The bottom figure shows the cases where the energy deposit to the target from the photons arising from $\pi^\circ$ decay is turned off.

The solid line on figure 3-3 corresponds to the events in which the protons are stopped inside the target and deposit all their kinetic energy. The distribution below that line corresponds to the protons that escape the target and deposit only a fraction of their kinetic energy. In the middle figure, when all protons start from the center, the protons’ initial kinetic energy is not high enough for all protons to escape, most protons stop inside the target, and we observe little distribution below the line.

The thin distribution above the solid line shows the contribution to the energy deposit from the photons when escaping target. These are the pair of photons resulting from the $\pi^\circ$ decay inside the target. When traveling in the scintillator these photons create Compton scattering and the electrons loss energy. However, the photon’s energy are still very high, and these photons will be detected by the TAPS. In the bottom figure, the program was set up to ignore the energy deposit from the photons, thus there is no distribution above the line.

Figure 3-4 shows the surface plots from the above distributions.
Figure 3-3: Distributions of total energy deposit on polar angle for 150MeV. Middle: recoil protons start from the center of the target. Bottom: no energy deposit to the target from photons (from $\pi^0$) interaction.
Figure 3-4: Surface plots of the distribution of total energy deposit on polar angle for 150MeV photons. Bottom: recoil protons start from the center of the target.
Figure 3-5, 3-6 present similar cases to the previous paragraph, except incident photons now have 250MeV energy. A linear relation between the proton kinetic energy and scattering angle is observed. As the polar angle increases the kinetic energy increases, the protons can travel further. From the minimum value of 44MeV, all the protons have enough energy to travel the furthest distance possible inside the target; as a result, afterwards they all escape the target. 44MeV corresponds to the amount of energy the the proton losses when traveling from one side of the target to the other. When recoil protons start from the center of the target, the middle figure shows that the minimum value is only 31MeV, approximately the amount of energy they loss to travel the distance of the radius of the target. When energy is above the minimum value, the distribution is not a thin line. This is due to the Landau fluctuation on the energy loss.

Notice there is a faint distribution on the middle figure 3-5 and 3-6. The physics behind this distribution is not yet determined. This could be a bug or an artifact.
Figure 3-5: Distributions of total energy deposit on polar angle for 250MeV. Middle: recoil protons start from the center of the target. Bottom: no energy deposit to the target from photons.
Figure 3-6: Surface plots of the distribution of total energy deposit on polar angle for 250MeV photons. Bottom: recoil protons start from the center of the target.
3.1.2 Proton kinetic energy and energy deposit

Figure 3-7 and 3-8 show the distributions of the energy deposit and the proton kinetic energy. The bottom figures show the distribution when all recoil protons start from the center of the target. In the top figure, a one-to-one relationship is observed when the energy is less than 44MeV. This implies that the recoil protons do not have sufficient kinetic energy to escape the target, thus they deposit all their kinetic energy to the target. Above 44MeV all proton would escape. Again the shift of the minimum proton kinetic energy from 44MeV to 31MeV is observed in the bottom figure.

The top figure show that there are protons escaping the target at all energy when the beam has non-zero size. The bottom figure demonstrates a clear cut using the minimum energy of 31MeV which is the amount of energy a proton would lose traveling the radius of the target. The protons up to ~35MeV correspond to those that traveled the larger distance from the target center to the upper or lower edge of the target.
Figure 3-7: Distribution of total energy deposit on kinetic energy of recoil protons. Bottom: recoil protons start from the center of the target.
Figure 3-8: Surface plot of the distribution of total energy deposit on kinetic energy of recoil protons. Bottom: recoil protons start from the center of the target.
3.1.3 Recoil proton stopping acceptance

Figure 3-9 and 3-10 shows the distributions of incident photon energy on the cosine of the scattering angle in the center of mass frame. This distribution was generated uniformly in $\cos(\theta_{\gamma CM})$. For this run, 150000 events were generated. In figure 3-9 top figure shows the distribution for all protons, while the bottom figure shows the distribution for stopped protons only. In figure 3-10 top figure shows the distribution for all protons, while the middle figure shows the distribution for stopped protons only, and the bottom figure shows the recoil proton stopping acceptance, which is the ratio between the stopped protons over all the recoil protons. This acceptance was obtained by dividing the histogram of the two distributions.
Figure 3-9: Distributions of photon energy on polar angle. Top: all protons. Bottom: stopped protons only.
Figure 3-10: Surface plots of the distributions of photon energy on polar angle. Top: all protons. Middle: stopped protons only. Bottom: stopping acceptance (ratio stop/all).
Figure 3-11 also shows the recoil proton stopping acceptance but with scattering angle in the laboratory frame instead. Also drawn are two theoretical prediction boundaries when the proton has kinetic energy, $T_p$, of 44MeV and 31MeV. The boundary matches the 44MeV line. The cosine of the predicted polar angle is given by:

$$\cos \theta_{\pi^o} = \frac{-2M_pT_p - m^2_{\pi^o} + 2k\gamma(k\gamma - T_p)}{2k\gamma\sqrt{(k\gamma - T_p)^2 - m^2_{\pi^o}}}$$

The bottom figure 3-11 shows similar case but the recoil protons now start from the center of the target. As expected the boundary is now the 31MeV line instead.

Figure 3-12 shows the distribution of incident photon energy and the polar angle, but only for protons that escape. The bottom figure 3-12 again shows the cases when the protons start from the center of the target. The difference between the two cases is quite profound. When the protons start from random location in the target the 44 MeV line is clearly a boundary for the stopped proton as in top figure 3-11, but is not a boundary for the escaped protons as in top figure 3-12. If the protons start from the center of the target, the 31MeV is a boundary for both the stopped and the escaped protons. This implies the effect of the size of the target on the stopping acceptance of the recoil proton.
Figure 3-11: Distribution of photon energy on the polar angle when recoil protons are stopped in the target, along with theoretical line when proton kinetic energies are 44MeV and 31MeV. Bottom: recoil protons start from the center of the target.
Figure 3-12: Distribution of photon energy on the polar angle when recoil protons escape the target, along with theoretical line when proton kinetic energies are $44\,\text{MeV}$ and $31\,\text{MeV}$. Bottom: recoil protons starts from the center of the target.
Figure 3-13 shows the same implication from above, but instead of analyzing the acceptance, the difference between the proton kinetic energy and the energy deposit was histogrammed. In the top figure, placing a cut on neither 44MeV nor 31MeV proton can cleanly separate events with the recoil protons stopped and the events with the recoil protons escaped. In the middle figure, when all recoil protons start from the center of the target, the 31MeV can separate these events deterministically. The bottom figure shows the distribution if a really narrow incident photon beam is used instead. The results show no improvement from the top figure.
Figure 3-13: Distribution of the difference between proton kinetic energy and energy deposit, along with theoretical line when proton kinetic energies are 44\,MeV and 31\,MeV. Middle: recoil protons start from the center of the target. Bottom: narrow incident photon beam.
3.2 Hardening photon beam

Figure 3-14 shows the energy distribution of the photon beam that enters the target using a Beryllium absorber of length 20cm and 40cm respectively. The higher distribution is the uniformly generated spectrum of the beam, and the lower distribution is the spectrum of the beam that makes it through the absorber. An overall attenuation is observed; however, the attenuation is much stronger at the low energy. Comparing the two choices of absorber, the longer absorber provides a cut up to a higher energy, but the overall attenuation is increased as well.
Figure 3-14: The photon energy distribution with and without a 20cm (top) and 40cm (bottom) long Beryllium absorber.
Chapter 4

Conclusion

The simulation program using GEANT4 successfully reproduces the expected results. The recoil proton stopping acceptance was plotted in a range of photon energy from 150MeV to 250MeV. The effect of the size of the target was determined to introduce uncertainty in the determination of whether a recoil proton of a particular energy stops in the target. This error was further determined to persist even with a narrower incident photon beam.

The “hardening” effect of the beryllium absorber was plotted for absorber with two different lengths. It is seen that it performs as expected, attenuating the photon beam much more at low energies compared to high energies.

GEANT4 has proven itself to be a powerful Monte Carlo simulation toolkit. The simulation program developed in this work, while rather simple, has laid out a foundation for developing a much more extensive study in the future.
Bibliography

[CERN] CERN website http://wwwinfo.cern.ch


Appendix A

Source Codes
name := exampleN01
G4TARGET := $(name)
G4EXLIB := true

ifndef G4INSTALL
    G4INSTALL = ../../../
endif

ifdef MAKEHISTO
    CPPFLAGS+=-I../tools/include -DMAKEHISTO
    EXTRALIBS+= -L/home/mtthai/mainz/lib -lG4TestTool
    EXTRALIBS+= -L/usr/local/cern/pro/lib -lpacklib
    EXTRALIBS+= -lf2c -lgcc
endif

.PHONY: all
all: lib bin

include ../config/binmake.gmk
#ifndef GLOBAL_VARS__HH
#define GLOBAL_VARS__HH

// Global variables for hbook:

extern G4double pion0CosThetaCM;
extern G4double pion0ThetaCM;
extern G4double pion0PhiCM;
extern G4double pion0Theta;
extern G4double pion0Phi;
extern G4double protonTheta;
extern G4double protonPhi;
extern G4double incidentGamma;
extern G4double energyDeposit;
extern G4bool protonStopped;
extern G4double openingAngle;
extern G4double protonKinetic;
extern G4double initialGamma;

// Other global variables:

extern G4int pion0TrackID;
extern G4ParticleMomentum aGamma;
extern G4bool firstGamma;
extern G4bool secondGamma;
extern G4RotationMatrix* target_rot;
extern G4bool modeTriggered;
extern G4String center;
extern G4bool targetHit;

#endif //GLOBAL_VARS__HH
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4UIGAG.hh"

#ifdef G4VIS_USE
#include "ExNO1VisManager.hh"
#endif

#include "ExNO1DetectorConstruction.hh"
#include "ExNO1PhysicsList.hh"
#include "ExNO1PrimaryGeneratorAction.hh"
#include "ExNO1EventAction.hh"
#include "ExNO1TrackingAction.hh"
#include "ExNO1SteppingAction.hh"
#include "Randomize.hh"
#include "time.h"

//----------------------------
// Global variables for hbook:
//----------------------------
G4double pionOCosThetaCM = 0.;
G4double pionOThetaCM = 0.;
G4double pionOPhiCM = 0.;
G4double pionOPhi = 0.;
G4double pionOTheta = 0.;
G4double protonTheta = 0.;
G4double protonPhi = 0.;
G4double incidentGamma = 0.;
G4double energyDeposit = 0.;
G4bool protonStopped = false;
G4double openingAngle = 0.;
G4double protonKinetic = 0.;
G4double initialGamma = 0.;
//------------------------
// Other global variables:
//------------------------
G4int pionOTrackID = 0;
G4ParticleMomentum aGamma;
G4bool firstGamma = false;
G4bool secondGamma = false;
G4RotationMatrix* target_rot = NULL;
G4bool modeTriggered = false;
G4String center = "no";
G4bool targetHit=false;
//--------------------------
// Parameterisation manager:
//--------------------------
#include "G4GlobalFastSimulationManager.hh"
#ifdef MAKEHISTO
#include <fstream.h>
#include <stdlib.h>
#include "HbookManager.hh"
ofstream outDataFile("examplen01.dat");
#endif

int main() {
#ifdef MAKEHISTO
if (!outDataFile)
  G4cerr<<"Could not open examplen01.dat\n";

theHbookManager.SetFilename("exampleN01.hbook");
G4cout "HBook filename: exampleN01.hbook" << endl;
#endif

HepRandom::getTheEngine()->setSeed((long)time(NULL), 0);

// Construct the default run manager

G4RunManager* runManager = new G4RunManager;

// set mandatory initialization classes
runManager->SetUserInitialization(new ExN01DetectorConstruction);
runManager->SetUserInitialization(new ExN01PhysicsList);

#ifdef G4VIS_USE
   // visualization manager
   G4VisManager* visManager = new ExN01VisManager;
   visManager->Initialize();
#endif

// set mandatory user action class
runManager->SetUserAction(new ExN01PrimaryGeneratorAction);
runManager->SetUserAction(new ExN01EventAction);
runManager->SetUserAction(new ExN01TrackingAction);
runManager->SetUserAction(new ExN01SteppingAction);

// Initialize G4 kernel
runManager->Initialize();

// get the pointer to the UI manager and set verbosities
G4UImanager* UI = G4UImanager::GetUIpointer();

// G4UITerminal is a (dumb) terminal.
// G4UIsession * session = new G4UITerminal;
G4UIsession * session = new G4UIGAG;

UI->ApplyCommand("/control/execute prerunN01.mac");
session->SessionStart();
delete session;

// job termination
#ifdef G4VIS_USE
   delete visManager;
#endif

delete runManager;
return 0;
}
ExNO1DetectorConstruction::ExNO1DetectorConstruction()
{
}
ExNO1DetectorConstruction::~ExNO1DetectorConstruction()
{
}
G4VPhysicalVolume* ExNO1DetectorConstruction::Construct()
{
  G4double a; // atomic mass
  G4double z; // atomic number
  G4double density;
  G4String name;
  G4String symbol;
  G4double inRad;
  G4double outRad;
  G4double halfLen;
  G4double sPhi;
  G4double dPhi;
  G4double xPos;
  G4double yPos;
  G4double zPos;

  //Basic elements
  a = 1.01*g/mole;
  G4Element* H_element = new G4Element(name="Hydrogen", symbol="H", z= 1., a);
  a = 12.01*g/mole;
  G4Element* C_element = new G4Element(name="Carbon", symbol="C", z= 6., a);
  a = 14.01*g/mole;
  G4Element* N_element = new G4Element(name="Nitrogen", symbol="N", z= 7., a);
  a = 16.00*g/mole;
  G4Element* O_element = new G4Element(name="Oxygen", symbol="O", z= 8., a);

  //Simple material
  a = 39.95*g/mole;
  density = 1.782e-03*g/cm3;
  G4Material* Ar = new G4Material(name="ArgonGas", z=18., a, density);
  a = 26.98*g/mole;
  density = 2.7*g/cm3;
  G4Material* Al = new G4Material(name="Aluminum", z=13., a, density);
  a = 207.19*g/mole;
  density = 11.35*g/cm3;
  G4Material* Pb = new G4Material(name="Lead", z=82., a, density);
  a = 9.00*g/mole;
  density = 1.848*g/cm3;
  G4Material* Be = new G4Material(name="Beryllium", z=4., a, density);
density = universe_mean_density; //from PhysicalConstants.h
G4double pressure = 3.e-18*pascal;
G4double temperature = 2.73*kelvin;
G4Material* vacuum = new G4Material(name="Galactic", z=1., a=1.01*g/mole,
density, kStateGas, temperature, pressure);

//Define material from elements
G4int ncomponents;
G4int natoms;
density = 1.032*g/cm3;
G4Material* Sci = new G4Material(name="Scintillator", density, ncomponents=2);
Sci->AddElement(C_element, natoms=10);
Sci->AddElement(H_element, natoms=11);

//Define material from elements, mixture by fraction of mass
G4double fractionmass;
density = 1.290*mg/cm3;
G4Material* Air = new G4Material(name="Air", density, ncomponents=2);
Air->AddElement(N_element, fractionmass=0.7);
Air->AddElement(O_element, fractionmass=0.3);

// -------------------------------------------------- volumes

// ------------------------------- experimental hall (world volume)
// ------------------------------- beam line along x axis

G4double half_x = 1.0*m;
G4double half_y = 1.0*m;
G4double half_z = 3.0*m;
G4Box* experimentalHall_box
  = new G4Box("expHall_box", half_x, half_y, half_z);

G4LogicalVolume* experimentalHall_log
  = new G4LogicalVolume(experimentalHall_box, Air, "expHall_log", 0, 0, 0);

G4VPhysicalVolume* experimentalHall_phys
  = new G4PVPlacement(0, G4ThreeVector(), "expHall",
    experimentalHall_log, NULL, false, 0);

// Berylium block
inRad = 0.*cm;
outRad = 1.5*cm;
halfLen = 20.*cm;
sPhi = 0.*deg;
dPhi = 360.*deg;
G4Tubs* beBlock_tubs
  = new G4Tubs("beBlock_tubs", inRad, outRad, halfLen, sPhi, dPhi);

G4LogicalVolume* beBlock_log
  = new G4LogicalVolume(beBlock_tubs, Be, "beBlock_log", 0, 0, 0);

xPos = 0.*cm;
yPos = 0.*cm;
zPos = 0.*cm;
G4VPhysicalVolume* beBlock_phys
  = new G4PVPlacement(0, G4ThreeVector(xPos, yPos, zPos), "beBlock",
    beBlock_log, experimentalHall_phys, false, 0);

// Collimator after Berylium block
inRad = 0.25*cm;
outRad = 1.5*cm;
halfLen = 5.*cm;
sPhi = 0.*degree;
dPhi = 360.*degree;
G4Tubs* beCol_tubs
  = new G4Tubs("beCol_tubs", inRad, outRad, halfLen, sPhi, dPhi);

G4LogicalVolume* beCol_log
  = new G4LogicalVolume(beCol_tubs, Pb, "beCol_log", 0, 0, 0);

xPos = 0.*cm;
yPos = 0.*cm;
zPos = 25.1*cm;
G4VPhysicalVolume* beCol_phys
  = new G4PVPlacement(0, G4ThreeVector(xPos, yPos, zPos), "beCol",
                      beCol_log, experimentalHall_phys, false, 0);

//Sweeping Magnet
half_x = 15.*cm;
half_y = 15.*cm;
half_z = 15.*cm;
G4Box* swpMag_box = new G4Box("swpMag_box", half_x, half_y, half_z);

G4LogicalVolume* swpMag_log
  = new G4LogicalVolume(swpMag_box, Air, "swpMag_log", 0, 0, 0);

xPos = 0.*cm;
yPos = 0.*cm;
zPos = 95.*cm;
G4VPhysicalVolume* swpMag_phys
  = new G4PVPlacement(0, G4ThreeVector(xPos, yPos, zPos), "swpMag",
                      swpMag_log, experimentalHall_phys, false, 0);

//Concrete wall
inRad = 5.*cm;
outRad = 1.*m;
halfLen = 5.*cm;
sPhi = 0.*degree;
dPhi = 360.*degree;
G4Tubs* wall_tubs
  = new G4Tubs("wall_tubs", inRad, outRad, halfLen, sPhi, dPhi);

G4LogicalVolume* wall_log
  = new G4LogicalVolume(wall_tubs, Air, "wall_log", 0, 0, 0);

xPos = 0.*cm;
yPos = 0.*cm;
zPos = 120.*cm;
G4VPhysicalVolume* wall_phys
  = new G4PVPlacement(0, G4ThreeVector(xPos, yPos, zPos), "wall",
                      wall_log, experimentalHall_phys, false, 0);

//Scintillator target
inRad = 0.*cm;
outRad = 9.*mm;
halfLen = 9.*mm;
sPhi = 0.*degree;
dPhi = 360.*degree;
G4Tubs* target_tubs
  = new G4Tubs("target_tubs", inRad, outRad, halfLen, sPhi, dPhi);

G4LogicalVolume* target_log
  = new G4LogicalVolume(target_tubs, Sci, "target_log", 0, 0, 0);

target_rot = new G4RotationMatrix();
G4double angle = -90.*deg;
target_rot->rotateX(angle);

xPos = 0.*cm;
yPos = 0.*cm;
zPos = 270.*cm;
G4VPhysicalVolume* target_phys
    = new G4PVPlacement(target_rot, G4ThreeVector(xPos, yPos, zPos), "target",
                        target_log, experimentalHall_phys, false, 0);

  // G4FastSimulationManager doesn’t exist yet: we set it
  // (not needed if we set a G4VFastSimulationModel which
  // takes care of creating one if needed)
  new G4FastSimulationManager(target_log);

  // target_log->GetFastSimulationManager()->
  // AddGhostPlacement(target_rot, G4ThreeVector(xPos, yPos, zPos));

  //------------------ Parameterisation ------------------------------
  // builds a model and sets it to the envelope of the calorimeter:
  ExNOlMyModel* pMyModel = new ExNOlMyModel(target_log);

  //------------------ Visualization attribution ----------------------
  G4VisAttributes* pVisAtt = new G4VisAttributes();
  pVisAtt->SetForceSolid(true);
  // pVisAtt->SetForceWireframe(true);
  // beBlock_log->SetVisAttributes(pVisAtt);
  target_log->SetVisAttributes(pVisAtt);

  //--------------------------
  experimentalHall_log->SetVisAttributes (G4VisAttributes::Invisible);
  return experimentalHall_phys;
}
#ifndef ExN01DetectorConstruction_H
#define ExN01DetectorConstruction_H 1

class G4VPhysicalVolume;

#include "G4VUserDetectorConstruction.hh"

class ExN01DetectorConstruction : public G4VUserDetectorConstruction
{
   public:
      ExN01DetectorConstruction();
      ~ExN01DetectorConstruction();

      public:
         G4VPhysicalVolume* Construct();

};

#endif
#include "ExN01EventAction.hh"
#include "ExN01EventActionMessenger.hh"

#include <rw/tvordvec.h>

#include "G4Event.hh"
#include "G4EventManager.hh"
#include "G4HCofThisEvent.hh"
#include "G4VHitsCollection.hh"
#include "G4TrajectoryContainer.hh"
#include "G4Trajectory.hh"
#include "G4VVisManager.hh"
#include "G4SDManager.hh"
#include "G4UImanager.hh"
#include "G4ios.hh"
#include "G4UnitsTable.hh"
#include "G4VisAttributes.hh"
#include "global_vars.hh"
#include "G4ParticleTypes.hh"

#ifdef MAKEHISTO
#include <fstream.h>
#include <stdlib.h>
#include "HbookHistogram.hh"
extern ofstream outDataFile;
#endif //MAKEHISTO

#include "G4UImanager.hh"

ExN01EventAction::ExN01EventAction()
:drawFlag("all"),eventMessenger(NULL)
{
    eventMessenger = new ExN01EventActionMessenger(this);
    #ifdef MAKEHISTO
    incidentGammaBook=new HbookHistogram("Incident Gamma(MeV)",250,0.,250.);
    initialGammaBook=new HbookHistogram("Initial Gamma(MeV)",250,0.,250.);
    #endif //MAKEHISTO
}

ExN01EventAction::~ExN01EventAction()
{
    delete eventMessenger;
}

void ExN01EventAction::BeginOfEventAction()
{
    modeTriggered = false;
    protonStopped = true;
    energyDeposit = 0.;
    firstGamma = false;
    secondGamma = false;
    targetHit = false;
}

void ExN01EventAction::EndOfEventAction()
{...

void ExN01EventAction::UserSteppingAction(G4Step*)
{...

void ExN01EventAction::UserTrackingAction(G4Track*)
{...

void ExN01EventAction::UserDefinedAction(G4UserAction*)
{...

void ExN01EventAction::ParticleTypeAction(G4Particle*,G4Particle*)
{...

void ExN01EventAction::UserPreRunAction()
{...

void ExN01EventAction::UserPostRunAction()
{...

void ExN01EventAction::EndOfRunAction()
{...

void ExN01EventAction::EndOfParticleAction()
{...

void ExN01EventAction::ConstructParticle(G4TypeInt)
{...

void ExN01EventAction::ConstructWorld(G4TypeInt)
{...

void ExN01EventAction::PreUserTrackingAction()
{...

void ExN01EventAction::PostUserTrackingAction()
{...

void ExN01EventAction::PreUserInitialization()
{...

void ExN01EventAction::PostUserInitialization()
{...

void ExN01EventAction::PreUserEventAction()
{...

void ExN01EventAction::PostUserEventAction()
{...

void ExN01EventAction::PreUserTrackingAction()
{...

void ExN01EventAction::PostUserTrackingAction()
{...

void ExN01EventAction::PreUserDateInitialization()
{...

void ExN01EventAction::PostUserDateInitialization()
{...

void ExN01EventAction::PreUserDateTrackingAction()
{...

void ExN01EventAction::PostUserDateTrackingAction()
{...

void ExN01EventAction::PreUserDateAction()
{...

void ExN01EventAction::PostUserDateAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserTimeAction()
{...

void ExN01EventAction::PostUserTimeAction()
{...

void ExN01EventAction::PreUserTimeEventAction()
{...

void ExN01EventAction::PostUserTimeEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
{...

void ExN01EventAction::PostUserDateEventAction()
{...

void ExN01EventAction::PreUserDateEventAction()
void ExN01EventAction::EndOfEventAction()
{
    const G4Event* evt = fpEventManager->GetConstCurrentEvent();

    #ifdef MAKEHISTO
    if (!protonStopped)
        energyDeposit+=G4Proton::ProtonDefinition()->GetPDGMass();

    initialGammaBook->accumulate(initialGamma/MeV);
    if (targetHit)
        incidentGammaBook->accumulate(incidentGamma/MeV);
    if (modeTriggered&&secondGamma) {
        outDataFile<<pion0CosThetaCM<<' '<<pion0ThetaCM/deg<<' '<<'
        <pion0Theta/deg'<<' '<<protonTheta/deg<<' '<<incidentGamma/MeV<<' '<<energyDeposit/MeV
        <' '<<((protonStopped)?1:0)<<' '<<openingAngle/deg<<'
        <protonKinetic<<endl;
    }
    #endif //MAKEHISTO

    if (evt->GetEventID() < 50)
        G4cout << ">>> Event " << evt->GetEventID() << endl;
    else if (evt->GetEventID()%100 == 0)
        G4cout << ">>> Event " << evt->GetEventID() << endl;

    G4TrajectoryContainer * trajectoryContainer = evt->GetTrajectoryContainer();
    G4int n_trajectories = 0;
    if(trajectoryContainer)
    { n_trajectories = trajectoryContainer->entries(); } 
    if (n_trajectories!=0)
        G4cout << " " << n_trajectories
        " trajectories stored in this event." << endl;

    #ifdef G4VIS_USE
    if(G4VisManager::GetConcreteInstance()) {
        for(G4int i=0; i<n_trajectories; i++) {
            G4Trajectory* trj = (*(evt->GetTrajectoryContainer()))[i];
            if (drawFlag == "all") trj->DrawTrajectory(50);
            else if ((drawFlag == "charged")&&(trj->GetCharge() != 0.))
                trj->DrawTrajectory(50);
        }
    }
    #endif
}

//....ooo000000ooo........ooo000000ooo........ooo000000ooo........ooo000000ooo....
#ifndef ExN01EventAction_h
#define ExN01EventAction_h

#include "G4UserEventAction.hh"
#include "globals.hh"

#ifdef MAKEHISTO
#include "HbookHistogram.hh"
#endif

class ExN01EventAction;

class ExN01EventAction : public G4UserEventAction
{
 public:
   ExN01EventAction();
   ~ExN01EventAction();

   public:
   void BeginOfEventAction();
   void EndOfEventAction();

   void SetDrawFlag(G4String val) {drawFlag = val;};

 private:
   G4String drawFlag; // control the drawing of event
   ExN01EventActionMessenger* eventMessenger;

#ifdef MAKEHISTO
   HbookHistogram* incidentGammaBook;
   HbookHistogram* initialGammaBook;
#endif
};

#endif
```cpp
#include "ExN01EventActionMessenger.hh"
#include "ExN01EventAction.hh"
#include "G4UIcmdWithAString.hh"
#include "globals.hh"

ExN01EventActionMessenger::ExN01EventActionMessenger(ExN01EventAction* EvAct)
: eventAction(EvAct)
{
    DrawCmd = new G4UIcmdWithAString("/event/drawTracks",this);
    DrawCmd->SetGuidance("Draw the tracks in the event");
    DrawCmd->SetGuidance(" Choice : none, charged(default), all");
    DrawCmd->SetParameterName("choice",true);
    DrawCmd->SetDefaultValue("charged");
    DrawCmd->SetCandidates("none charged all");
    DrawCmd->AvailableForStates(Idle);
}

ExN01EventActionMessenger::~ExN01EventActionMessenger()
{
    delete DrawCmd;
}

void ExN01EventActionMessenger::SetNewValue(G4UIcommand * command,G4String newValue
{
    if(command == DrawCmd)
    {
        eventAction->SetDrawFlag(newValue);
    }
}
```
#ifndef ExN01EventActionMessenger_h
#define ExN01EventActionMessenger_h 1

#include "globals.hh"
#include "G4UImessenger.hh"

class ExN01EventAction;
class G4UIcmdWithAString;

class ExN01EventActionMessenger: public G4UImessenger
{
  public:
    ExN01EventActionMessenger(ExN01EventAction*);
    ~ExN01EventActionMessenger();

    void SetNewValue(G4UIcommand*, G4String);

  private:
    ExN01EventAction* eventAction;
    G4UIcmdWithAString* DrawCmd;
};

#endif
#include "ExNO1MyModel.hh"
#include "Randomize.hh"
#include "G4ParticleTypes.hh"
#include "global_vars.hh"

ExNO1MyModel::ExNO1MyModel(G4Envelope *anEnvelope) :
    G4VFastSimulationModel("ExNO1MyModel",anEnvelope)
{
}

ExNO1MyModel::~ExNO1MyModel()
{
}

G4bool ExNO1MyModel::IsApplicable(const G4ParticleDefinition& particleType)
{
    return
        &particleType == G4Gamma::GammaDefinition();
}

#define THRESHOLD 144.7*MeV

G4bool ExNO1MyModel::ModelTrigger(const G4FastTrack& fastTrack) {
    //--------------------------------------
    // UserTrigger() method: method which has to decide if
    // the parameterisation has to be applied.
    // Here ModelTrigger() asks the user (ie you) a 0/1 answer.
    // Note that quantities like the local/global position/direction etc..
    // are available at this level via the fastTrack parameter (allowing
    // to check distance from boundaries, see below to allow the decision)
    //--------------------------------------
    if (modeTriggered)
        return false;
    if (fastTrack.GetPrimaryTrack()->GetKineticEnergy() > THRESHOLD) {
        modeTriggered = true;
        return true;
    }
    return false;
}

void ExNO1MyModel::DoIt(const G4FastTrack& fastTrack,
                        G4FastStep& fastStep)
    //--------------------------------------
    // User method to code the parameterisation properly
    // said.
    //
    //--------------------------------------
{
    // The primary track continues along its direction.
    // One secondary (a photon) is added:
    //
    //-------------------------------
    // Primary:
    //    idem as in "DefaultModel":
    //
    G4ThreeVector position;
    G4double distance;
    distance = fastTrack.GetEnvelopeSolid()->DistanceToOut(fastTrack.GetPrimaryTrackLocalPosition(),
                                                                 fastTrack.GetPrimaryTrackLocalDirection());

position = fastTrack.GetPrimaryTrackLocalPosition() +
    distance*fastTrack.GetPrimaryTrackLocalDirection();

// -- set final position:
fastStep.SetPrimaryTrackFinalPosition(position);

// ---------------------------
// Secondary:
//  Adds one "secondary":
// ---------------------------

// -- Build the secondary particles:
// -------------------------------
// -- position: Uniformly random over the initial gamma track in the target
G4double gammaKinetic = fastTrack.GetPrimaryTrack()->GetKineticEnergy();
G4ParticleMomentum gammaDirection(fastTrack.GetPrimaryTrackLocalDirection());
G4double gammaDist = fastTrack.GetEnvelopeSolid()->DistanceToOut(fastTrack.GetPrimaryTrackLocalPosition(),
    gammaDirection);
G4ThreeVector posi(0., 0., 0.);
if (center=="no") {
    posi = fastTrack.GetPrimaryTrackLocalPosition() +
        gammaDist*G4UniformRand()*gammaDirection;
}

// -- Kinetic energy:
G4ParticleMomentum* pProtonDirection = new G4ParticleMomentum();
G4double pion0Kinetic;
G4ParticleMomentum* pPion0Direction = new G4ParticleMomentum();
if (!MyCalculate(&gammaDirection, gammaKinetic,
    pProtonDirection, &protonKinetic,
    pPion0Direction, &pion0Kinetic)) {
    return;
}
G4DynamicParticle protonDynamique(G4Proton::ProtonDefinition(),
    *pProtonDirection, protonKinetic);
G4DynamicParticle pion0Dynamique(G4PionZero::PionZeroDefinition(),
    *pPion0Direction, pion0Kinetic);

G4double t1
    = gammaKinetic - pion0Kinetic
    - protonKinetic-G4PionZero::PionZeroDefinition()->GetPDGMass()
    - G4PionZero::PionZeroDefinition()->GetPDGMass();

// -------------------------------
//-- Creation of the secondary Track:
// -------------------------------

// -- First, user has to say how many secondaries will be created:
fastStep.SetNumberOfSecondaryTracks(2);

fastStep.CreateSecondaryTrack(protonDynamique, posi,
    fastTrack.GetPrimaryTrack()->GetGlobalTime());
fastStep.CreateSecondaryTrack(pion0Dynamique, posi,
    fastTrack.GetPrimaryTrack()->GetGlobalTime());
fastStep.KillPrimaryTrack();
G4bool ExN01MyModel::MyCalculate(G4ParticleMomentum* pGammaDirection,
G4double gammaKinetic,
G4ParticleMomentum* pProtonDirection,
G4double* pProtonKinetic,
G4ParticleMomentum* pPion0Direction,
G4double* pPion0Kinetic)
{
    const G4double pionOMass = G4PionZero::PionZeroDefinition()->GetPDGMass();
    const G4double protonMass = G4Proton::ProtonDefinition()->GetPDGMass();
    G4double WT = protonMass + pionOMass;
    G4double kT = (WT*WT - protonMass*protonMass)/2/protonMass;

    if (gammaKinetic < kT) {
        G4cerr << "MyModel: gamma less than threshold\n";
        return false;
    }

    incidentGamma = gammaKinetic;
    G4double W = sqrt(protonMass*(protonMass + 2*gammaKinetic));
    G4double betaCM = gammaKinetic/(gammaKinetic+protonMass);
    G4double gammaCM = 1/sqrt(1-betaCM*betaCM);
    G4double pion0EnergyCM =
        (pionOMass*pionOMass + 2*gammaKinetic*protonMass)/W/2;
    G4double momentumCM =
        sqrt(pion0EnergyCM*pion0EnergyCM - pion0Mass*pion0Mass);
    G4double protonEnergyCM =
        sqrt(protonMass*protonMass + momentumCM*momentumCM);

    // -- direction: Uniformly random over dcos(theta) and dphi in CM coordinates
    G4double costhetaCM = 2.*G4UniformRand() - 1.;
    pionOCosThetaCM = costhetaCM;
    G4double thetaCM = acos(costhetaCM)*rad;
    pionOThetaCM = thetaCM;
    G4double phiCM = 360.*deg*G4UniformRand()-180.*deg;
    pionOPhiCM = phiCM;

    G4double pPer = momentumCM*sin(thetaCM);
    G4double pPar = gammaCM*(betaCM*pion0EnergyCM + momentumCM*costhetaCM);
    G4double pion0Momentum = sqrt(pPar*pPar + pPer*pPer);
    pPion0Direction->setX(pPer*sin(phiCM));
    pPion0Direction->setY(pPer*cos(phiCM));
    pPion0Direction->setZ(pPar);
    pion0Theta = pPion0Direction->theta();
    pion0Phi = pPion0Direction->phi();
    pPion0Direction->transform(*target_rot);
    *pPionOKinetic = pionOEnergy - pionOMass;

    G4double protonMomentum = sqrt(pPar*pPar + pPer*pPer);
    pProtonDirection->setX(-pPer*sin(phiCM));
    pProtonDirection->setY(-pPer*cos(phiCM));
    pProtonDirection->setZ(pPar);
    protonTheta = pProtonDirection->theta();
    protonPhi = pProtonDirection->phi();
    pProtonDirection->transform(*target_rot);
    *pProtonKinetic = protonEnergy - protonMass;
}
return true;
}
#ifndef ExN01MyModel_h
#define ExN01MyModel_h

#include "G4VFastSimulationModel.hh"

class ExN01MyModel : public G4VFastSimulationModel
{
  public:
  //-------------------------
  // Constructor, destructor
  //-------------------------
  ExN01MyModel (G4LogicalVolume* anEnvelope);
  ~ExN01MyModel ();

  //----------------------------
  // Virtual methods of the base
  // class to be coded by the user
  //----------------------------

  // -- IsApplicable
  G4bool IsApplicable(const G4ParticleDefinition&);
  // -- ModelTrigger
  G4bool ModelTrigger(const G4FastTrack &);
  // -- User method DoIt
  void DoIt(const G4FastTrack&, G4FastStep&);

  protected:
  G4bool MyCalculate(G4ParticleMomentum*, G4double,
                      G4ParticleMomentum*, G4double*,
                      G4ParticleMomentum*, G4double*);
};
#endif
```cpp
#include "ExNOlPhysicsList.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 "G4FastSimulationManagerProcess.hh"

ExNOlPhysicsList::ExNOlPhysicsList() {
  defaultCutValue = 2.0*mm;
  SetVerboseLevel(1);
}

ExNOlPhysicsList::~ExNOlPhysicsList() {}

void ExNOlPhysicsList::ConstructParticle() {
  ConstructBosons();
  ConstructLeptons();
  ConstructMesons();
  ConstructBarions();
}

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

  // gamma
  G4Gamma::GammaDefinition();

  // optical photon
  G4OpticalPhoton::OpticalPhotonDefinition();
}

void ExNOlPhysicsList::ConstructLeptons() {
  // leptons
  G4Electron::ElectronDefinition();
  G4Positron::PositronDefinition();
  G4MuonPlus::MuonPlusDefinition();
  G4MuonMinus::MuonMinusDefinition();
  G4NeutrinoE::NeutrinoEDefinition();
  G4AntiNeutrinoE::AntiNeutrinoEDefinition();
  G4NeutrinoMu::NeutrinoMuDefinition();
  G4AntiNeutrinoMu::AntiNeutrinoMuDefinition();
}

void ExNOlPhysicsList::ConstructMesons() {
  // mesons
  G4PionPlus::PionPlusDefinition();
  G4PionMinus::PionMinusDefinition();
```
G4PionZero::PionZeroDefinition();
G4Eta::EtaDefinition();
G4EtaPrime::EtaPrimeDefinition();
G4RhoZero::RhoZeroDefinition();
G4KaonPlus::KaonPlusDefinition();
G4KaonMinus::KaonMinusDefinition();
G4KaonZero::KaonZeroDefinition();
G4AntiKaonZero::AntiKaonZeroDefinition();
G4KaonZeroLong::KaonZeroLongDefinition();
G4KaonZeroShort::KaonZeroShortDefinition();

// 000.....oooOOOOooo..............oooOOOOO00000ooo..............oooO0000ooo

void ExNOlPhysicsList::ConstructBarions()
{
    // barions
    G4Proton::ProtonDefinition();
    G4AntiProton::AntiProtonDefinition();
    G4Neutron::NeutronDefinition();
    G4AntiNeutron::AntiNeutronDefinition();
}

void ExNOlPhysicsList::ConstructProcess()
{
    // Define transportation process
    AddTransportation();
    AddParameterisation();
    ConstructEM();
    ConstructGeneral();
}

#include "G4ComptonScattering.hh"
#include "G4GammaConversion.hh"
#include "G4PhotoElectricEffect.hh"
#include "G4MultipleScattering.hh"
#include "G4eIonisation.hh"
#include "G4eBremsstrahlung.hh"
#include "G4eplusAnnihilation.hh"
#include "G4hIonisation.hh"

extern G4String gammaReaction;
void ExNOl1PhysicsList::ConstructEM() {
    theParticleIterator->reset();
    while( (*theParticleIterator)() )
    {
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        G4String particleName = particle->GetParticleName();
        if (particleName == "gamma") {
            // gamma
            pmanager->AddDiscreteProcess(new G4PhotoElectricEffect());
            pmanager->AddDiscreteProcess(new G4ComptonScattering());
        }
    }
}
pmanager->AddDiscreteProcess(new G4GammaConversion());
} else if (particleName == "e-") {
    // electron
    pmanager->AddProcess(new G4MultipleScattering(),-1, 1, 1);
    pmanager->AddProcess(new G4eIonisation(), -1, 2, 2);
    pmanager->AddProcess(new G4eBremsstrahlung(), -1,-1,3);
} else if (particleName == "e+") {
    // positron
    pmanager->AddProcess(new G4MultipleScattering(),-1, 1, 1);
    pmanager->AddProcess(new G4eIonisation(), -1, 2, 2);
    pmanager->AddProcess(new G4eBremsstrahlung(), -1,-1,3);
    pmanager->AddProcess(new G4eplusAnnihilation(), 0,-1,4);
}
}

#include "G4Decay.hh"

void ExN01PhysicsList::ConstructGeneral()
{
    // Add Decay Process
    G4Decay* theDecayProcess = new G4Decay();
    theParticleIterator->reset();
    while( (*theParticleIterator) () ){
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        if (theDecayProcess->IsApplicable(*particle)) {
            pmanager->AddProcess(theDecayProcess);
            // set ordering for PostStepDoIt and AtRestDoIt
            pmanager->SetProcessOrdering(theDecayProcess, idxPostStep);
            pmanager->SetProcessOrdering(theDecayProcess, idxAtRest);
        }
    }
}

void ExN01PhysicsList::AddParameterisation()
{
    G4FastSimulationManagerProcess* theFastSimulationManagerProcess =
        new G4FastSimulationManagerProcess();
    theParticleIterator->reset();
    while( (*theParticleIterator) () ){ 
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        // both postStep and alongStep action are required: because
        // of the use of ghost volumes. If no ghost, the postStep
        // is sufficient.
        pmanager->AddProcess(theFastSimulationManagerProcess, -1, 1, 1);
    }
}

void ExN01PhysicsList::SetCuts(G4double cut) {
    if (verboseLevel >0){
        G4cout << "ExN01PhysicsList::SetCuts:"
        G4cout << "CutLength : " " << cut/mm " " (mm)" " << endl;
    ```
// set cut values for gamma at first and for e- second and next for e+,
// because some processes for e+/e- need cut values for gamma
SetCutValue(cut, "gamma");
SetCutValue(cut*5., "e-" );
SetCutValue(cut*5., "e+");

// set cut values for proton and anti_proton before all other hadrons
// because some processes for hadrons need cut values for proton/anti_proton
SetCutValue(cut, "proton");
SetCutValue(cut, "anti_proton");

// set a cut value to all particles
SetCutValueForOthers(cut);

if (verboseLevel>1) DumpCutValuesTable();
}
#ifndef ExNO1PhysicsList_h
#define ExNO1PhysicsList_h 1

#include "G4VUserPhysicsList.hh"
#include "globals.hh"

class ExNO1PhysicsList: public G4VUserPhysicsList {
public:
    ExNO1PhysicsList();
    ~ExNO1PhysicsList();

protected:
    // Construct particle and physics process
    void ConstructParticle();
    void ConstructProcess();
    void SetCuts(G4double);

    // these methods Construct particles
    void ConstructBosons();
    void ConstructLeptons();
    void ConstructMesons();
    void ConstructBarions();

    // these methods Construct physics processes and register them
    void AddParameterisation();
    void ConstructGeneral();
    void ConstructEM();
};

#endif
#include "ExNOlPrimaryGeneratorAction.hh"
#include "ExNOlPrimaryGeneratorMessenger.hh"

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

ExNOlPrimaryGeneratorAction::ExNOlPrimaryGeneratorAction()
    : energyFlag("all"), sizeFlag("big")
{
    G4int n_particle = 1;
    particleGun = new G4ParticleGun(n_particle);
    gunMessenger = new ExNOlPrimaryGeneratorMessenger(this);

    G4String particleName;
    G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
    G4ParticleDefinition* particle
        = particleTable->FindParticle(particleName="gamma");

    particleGun->SetParticleDefinition(particle);
    particleGun->SetParticlePosition(G4ThreeVector(0., 0., -50.*cm));
    particleGun->SetParticleMomentumDirection(G4ThreeVector(0., 0., 1.));
    particleGun->SetParticleEnergy(150.*MeV);
}

ExNOlPrimaryGeneratorAction::~ExNOlPrimaryGeneratorAction()
{
    delete particleGun;
}

void ExNOlPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
    // Gun position is randomized as a circle of radius = 0.75cm
    G4double r;
    if (sizeFlag="big")
        r = 1.5*cm*(G4UniformRand()-0.5);
    else
        r = 0. ;

    // G4double r = 0.1*mm*(G4UniformRand()-0.5);
    G4double phi = 360.*deg*G4UniformRand();
    G4double xPos = r*cos(phi);
    G4double yPos = r*sin(phi);
    G4double zPos = -50.*cm;
    particleGun->SetParticlePosition(G4ThreeVector(xPos, yPos, zPos));

    // initialGamma = 215.*MeV*G4UniformRand();
    if (energyFlag="150")
        initialGamma = 150.*MeV;
    else if (energyFlag="250")
        initialGamma = 250.*MeV;
    else if (energyFlag="hard")
        initialGamma = 250.*MeV*G4UniformRand();
    else
        initialGamma = 150.*MeV+100.*MeV*G4UniformRand();

    particleGun->SetParticleEnergy(initialGamma);
    particleGun->GeneratePrimaryVertex(anEvent);
}
#ifndef ExNO1PrimaryGeneratorAction_h
#define ExNO1PrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"
#include "globals.hh"

class G4ParticleGun;
class G4Event;
class ExNO1PrimaryGeneratorMessenger;

class ExNO1PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
public:
  ExNO1PrimaryGeneratorAction();
  ~ExNO1PrimaryGeneratorAction();

public:
  void GeneratePrimaries(G4Event* anEvent);
  void SetEnergyFlag(G4String val) {energyFlag=val;)
  void SetSizeFlag(G4String val) {sizeFlag=val;}

private:
  G4ParticleGun* particleGun;
  ExNO1PrimaryGeneratorMessenger* gunMessenger;
  G4String energyFlag;
  G4String sizeFlag;
};

#endif
#include "ExN01PrimaryGeneratorMessenger.hh"
#include "ExN01PrimaryGeneratorAction.hh"
#include "G4UIcmdWithAString.hh"

ExN01PrimaryGeneratorMessenger::ExN01PrimaryGeneratorMessenger(ExN01PrimaryGeneratorAction *ExN01Gun)
{
    EnergyCmd = new G4UIcmdWithAString("/gun/mymode",this);
    EnergyCmd->SetGuidance(" Choice : 150 250 hard all(default)");
    EnergyCmd->SetParameterName("choice",true);
    EnergyCmd->SetDefaultValue("all");
    EnergyCmd->SetCandidates("150 250 hard all");
    EnergyCmd->AvailableForStates(PreInit,Idle);
    SizeCmd = new G4UIcmdWithAString("/gun/size",this);
    SizeCmd->SetGuidance(" Choice : small big(default)");
    SizeCmd->SetParameterName("choice",true);
    SizeCmd->SetDefaultValue("big");
    SizeCmd->SetCandidates("small big");
    SizeCmd->AvailableForStates(PreInit,Idle);
    CenterCmd = new G4UIcmdWithAString("/gun/center",this);
    CenterCmd->SetGuidance(" Choice : yes no(default)");
    CenterCmd->SetParameterName("choice",true);
    CenterCmd->SetDefaultValue("no");
    CenterCmd->SetCandidates("yes no");
    CenterCmd->AvailableForStates(PreInit,Idle);
}

void ExN01PrimaryGeneratorMessenger::SetNewValue(G4UIcommand * command,G4String newValue)
{
    if (command==EnergyCmd)
        ExN01Action->SetEnergyFlag(newValue);
    else if (command==CenterCmd){
        ExN01Action->SetSizeFlag("small");
        center=newValue;
    }
    else if (command==SizeCmd)
        ExN01Action->SetSizeFlag(newValue);
}

extern G4String center;
#ifndef ExNO1PrimaryGeneratorMessenger_h
#define ExNO1PrimaryGeneratorMessenger_h

#include "G4UImessenger.hh"
#include "globals.hh"

class ExNO1PrimaryGeneratorAction;
class G4UIcmdWithAString;

class ExNO1PrimaryGeneratorMessenger: public G4UImessenger
{
 public:
   ExNO1PrimaryGeneratorMessenger(ExNO1PrimaryGeneratorAction*);
   ~ExNO1PrimaryGeneratorMessenger();

   void SetNewValue(G4UIcommand*, G4String);

 private:
   ExNO1PrimaryGeneratorAction* ExNO1Action;
   G4UIcmdWithAString* EnergyCmd;
   G4UIcmdWithAString* SizeCmd;
   G4UIcmdWithAString* CenterCmd;
};

#endif
ExN01SteppingAction::ExN01SteppingAction()
{
}

ExN01SteppingAction::~ExN01SteppingAction()
{
}

void ExN01SteppingAction::UserSteppingAction()
{
    G4SteppingManager* SM = GetOmnipotentSteppingManager();
    G4Step* theStep = SM->GetStep();
    G4Track* theTrack = theStep->GetTrack();
    G4ParticleDefinition* particleType = theTrack->GetDefinition();
    G4StepPoint* thePostPoint = theStep->GetPostStepPoint();
    G4VPhysicalVolume* thePostPV = thePostPoint->GetPhysicalVolume();
    G4String thePostPVname;
    if (thePostPV != NULL) {
        thePostPVname = thePostPV->GetName();
    }
    else {
        return;
    }
    // check if it is entering to the calorimeter volume
    G4StepPoint* thePrePoint = theStep->GetPreStepPoint();
    G4VPhysicalVolume* thePrePV = thePrePoint->GetPhysicalVolume();
    G4String thePrePVname = thePrePV->GetName();
    // if(thePrePVname(0, 4) == "calo") { return; }
    // if(thePostPVname(0, 4) != "calo") { return; }
    // Sweeping magnet cleans up all charged particles
    if (((thePrePVname(0, 6) != "swpMag") && (thePostPVname(0, 6) == "swpMag"))) {
        if (particleType->GetPDGCharge() != 0.0) {
            theTrack->SetTrackStatus(fSuspend);
            // G4cout<<"Minh: Magnet swept " << particleType->GetParticleName() << endl
            return;
        }
    }
    // Concrete wall block everything
    if (((thePrePVname(0, 4) == "wall") && (thePostPVname(0, 4) == "wall"))) {
        theTrack->SetTrackStatus(fSuspend);
        return;
    }
    // Entering the target
    if (((thePrePVname(0, 6) == "target") && (thePostPVname(0, 6) == "target"))) {
        energyDeposit += theTrack->GetTotalEnergy();
        if ((particleType == G4Gamma::GammaDefinition()) && (!targetHit)) {
            incidentGamma = theTrack->GetTotalEnergy();
        }
    }
}
targetHit=true;
}
return;
}

// Leaving the target
if ((thePrePVname(0,6) == "target") && (thePostPVname(0,6) != "target")) {
    energyDeposit -= theTrack->GetTotalEnergy();

    if (particleType == G4Proton::ProtonDefinition()) {
        protonStopped = false;
        return;
    }

    if ((particleType == G4Gamma::GammaDefinition()) &&
        (theTrack->GetParentID() == pionOTrackID)) {
        if (!firstGamma) {
            aGamma = theTrack->GetMomentum();
            firstGamma = true;
            return;
        }

        if (!secondGamma) {
            openingAngle = aGamma.angle(theTrack->GetMomentum());
            secondGamma = true;
            return;
        }
    }
}
}
#ifndef ExN01SteppingAction_H
#define ExN01SteppingAction_H 1

#include "globals.hh"
#include "G4UserSteppingAction.hh"

class ExN01SteppingAction : public G4UserSteppingAction
{
    public:
        ExN01SteppingAction();
        ~ExN01SteppingAction();

        virtual void UserSteppingAction();
};

#endif
void ExN01TrackingAction::PreUserTrackingAction()
{
  G4TrackingManager* trackingManager = GetOmnipotentTrackingManager();
  G4Track* aTrack = trackingManager->GetTrack();

  if (aTrack->GetDefinition()==G4PionZero::PionZeroDefinition()) {
    pionOTrackID = aTrack->GetTrackID();
  }
}
#ifndef ExNO1TrackingAction_h
#define ExNO1TrackingAction_h 1

#include "G4UserTrackingAction.hh"

class ExNO1TrackingAction : public G4UserTrackingAction {

public:
    ExNO1TrackingAction(){};
    ~ExNO1TrackingAction(){};

    void PreUserTrackingAction();

};

#endif
#ifndef ExN01VisManager_h
#define ExN01VisManager_h 1

#include "G4VisManager.hh"

//....ooo000000oooo........ooo0000000oo........ooo0000000oo........ooo0000000oo....

class ExN01VisManager: public G4VisManager {

public:
    ExN01VisManager ();

private:
    void RegisterGraphicsSystems ();

};

#endif
```cpp
#include "ExNO1VisManager.hh"

// Supported drivers...

#if defined G4VIS_USE_DAWN
#include "G4FukuiRenderer.hh"
#endif

#if defined G4VIS_USE_DAWNFILE
#include "G4DAWNFILE.hh"
#endif

#if defined G4VIS_USE_OPACS
#include "G4Wo.hh"
#include "G4Xo.hh"
#endif

#if defined G4VIS_USE_OPENGLX
#include "G4OpenGLImmediateX.hh"
#include "G4OpenGLStoredX.hh"
#endif

#if defined G4VIS_USE_OPENGLWIN32
#include "G4OpenGLImmediateWin32.hh"
#include "G4OpenGLStoredWin32.hh"
#endif

#if defined G4VIS_USE_OPENGLXM
#include "G4OpenGLImmediateXm.hh"
#include "G4OpenGLStoredXm.hh"
#endif

#if defined G4VIS_USE_OIX
#include "G4OpenInventorX.hh"
#endif

#if defined G4VIS_USE_OIWIN32
#include "G4OpenInventorWin32.hh"
#endif

#if defined G4VIS_USE_RAYX
#include "G4RayX.hh"
#endif

#if defined G4VIS_USE_VRML
#include "G4VRML1.hh"
#include "G4VRML2.hh"
#endif

#if defined G4VIS_USE_VRMLFILE
#include "G4VRML1File.hh"
#include "G4VRML2File.hh"
#endif

ExN01VisManager::ExN01VisManager () {}

void ExN01VisManager::RegisterGraphicsSystems () {
```
#ifdef G4VIS_USE_DAWN
   RegisterGraphicsSystem (new G4FukuiRenderer);
#endif

#ifdef G4VIS_USE_DAWNFIL
   RegisterGraphicsSystem (new G4DAWNFIL);
#endif

#ifdef G4VIS_USE_OPACS
   RegisterGraphicsSystem (new G4Wo);
   RegisterGraphicsSystem (new G4Xo);
#endif

#ifdef G4VIS_USE_OPENGLX
   RegisterGraphicsSystem (new G4OpenGLImmediateX);
   RegisterGraphicsSystem (new G4OpenGLStoredX);
#endif

#ifdef G4VIS_USE_OPENGLWIN32
   RegisterGraphicsSystem (new G4OpenGLImmediateWin32);
   RegisterGraphicsSystem (new G4OpenGLStoredWin32);
#endif

#ifdef G4VIS_USE_OPENGLXM
   RegisterGraphicsSystem (new G4OpenGLImmediateXm);
   RegisterGraphicsSystem (new G4OpenGLStoredXm);
#endif

#ifdef G4VIS_USE_OIX
   RegisterGraphicsSystem (new G4OpenInventorX);
#endif

#ifdef G4VIS_USE_OIWIN32
   RegisterGraphicsSystem (new G4OpenInventorWin32);
#endif

#ifdef G4VIS_USE_RAYX
   RegisterGraphicsSystem (new G4RayX);
#endif

#ifdef G4VIS_USE_VRML
   RegisterGraphicsSystem (new G4VRML1);
   RegisterGraphicsSystem (new G4VRML2);
#endif

#ifdef G4VIS_USE_VRMLFILE
   RegisterGraphicsSystem (new G4VRML1File);
   RegisterGraphicsSystem (new G4VRML2File);
#endif

if (fVerbose > 0) {
   G4cout <<
   "\nYou have successfully chosen to use the following graphics systems."
   << endl;
   PrintAvailableGraphicsSystems ();
}

//....0000000000.........000000000000.........000000000000.........000000000000....