Санкт-Петербургский политехнический университет Петра Великого
Физико-механический институт
Высшая школа фундаментальных физических исследований

Курсовой проект

Моделирование гамма-спектрометра в каркасе GEANT4

По дисциплине ”Специальный практикум”

Выполнил
студент гр. 5040302/10301 С. А. Буланова

Научный руководитель:
к.ф.-м.н. Я. А. Бердников

«____» _____ 2021 г.

Санкт-Петербург
2021

ЗАДАНИЕ
НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА

студенту группы 5040303/10301 Булановой Софье Андреевне

1. Тема проекта: «Моделирование гамма-спектрометра в каркасе GEANT4»

2. Срок сдачи студентом законченного проекта: «14» декабря 2021 г.

3. Исходные данные к проекту: вариант 4 (см. приложение)

4. Содержание пояснительной записки введение, основная часть (раскрывается структура основной части), заключение, список используемых источников, приложения. В приложении необходимо привести исходный код модели, **используя при этом моноширинный шрифт, например Courier.**

Примерный объем пояснительной записки 15 страниц печатного текста.

5. Перечень графического материала: чертеж гамма-спектрометра, визуализация геометрии гамма-спектрометра без событий, визуализация геометрии гамма-спектрометра с одним событием, визуализация геометрии гамма-спектрометра со ста событиями, энергетические спектры для каждой энергии гамма-излучения, график: зависимость эффективности регистрации по пику от энергии гамма-излучения.

6. Консультанты: —

7. Дата получения задания: «27» октября 2021 г.

Руководитель Бердников Я. А.

Задание принял к исполнению Буланова С.А.

Исходные данные к курсовому проекту по теме
«Моделирование гамма-спектрометра в каркасе GEANT4»
Вариант 4

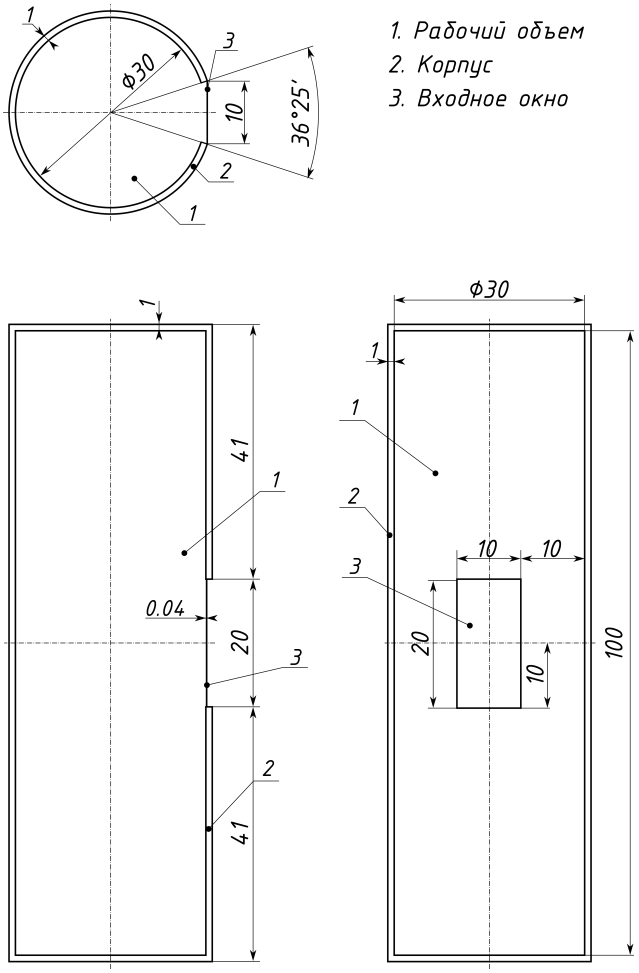
1. Детектор: пропорциональный счетчик

1.1. Материалы детектора:

- Рабочее вещество: Xe(95%) + CO_2(5%)
  - T = 20^O C
  - P = 250 мм. рт. ст.
- Корпус: нержавеющая сталь
- Входное окно: Be

1.2. Геометрия детектора:

- Рабочий объем:
  - длина 100 мм
  - диаметр 30 мм
- Толщина корпуса: 1 мм
- Размеры входного окна:
  - длина: 20 мм
  - ширина: 10 мм
  - толщина: 40 мкм



1.3. Характеристики для расчета энергетического разрешения:

- Фактор Фано: F = 0.36
- Коэффициент внутреннего усиления: G = 10^2
- Относительная дисперсия коэффициента внутреннего усиления: $\delta^2 G = 0.67$
- Энергия образования одной пары: ε = 22 эВ
- Относительная дисперсия однородности детектора: $\delta^2 n = 10^{-4}$
- Шум электроники: $N_ш = 100$

2. Гамма-излучение:

- Изотропное на расстоянии r = 5 см
- Энергетический диапазон: 0.20 кэВ—20 кэВ
- Количество линий в энергетическом диапазоне: 10-20

3. Условие для выбора числа событий: в пике полного поглощения для каждой энергетической линии должно быть не меньше 4×10^4 событий

Содержание

1 Построение модели детектора

Разделим детектор на несколько более простых геометрий. Он состоит из центральной части, содержащей Be-окно, и из двух боковых цилиндров, которые являются объектами класса G4Tubes. Для удобства материал цилиндров – железо. Внутрь помещается цилиндр, отвечающий геометрии газа. Газ состоит из смеси $Xe(95\%) + CO_2(5\%)$. Бериллиевое окно представляет собой объект класса G4Tubes и помещается в отверстие центрального цилиндра. На рис. 1 приведена схема детектора в Geant4. Бериллиевое окно сделано прозрачным, газ внутри – синим.

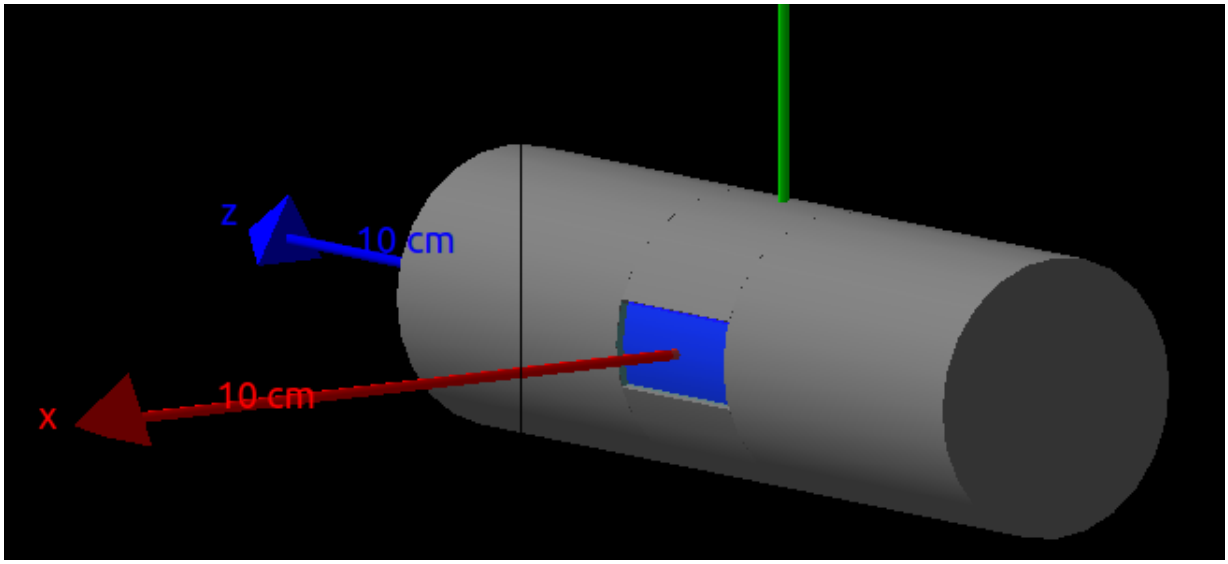


Рис. 1: Визуализация модели детектора в Geant4

Далее создаем источник частиц. Он расположен на расстоянии 5 см от центра координат, в точке (-5., 0., 0.). Знак указывает направление вдоль оси X. События расположены в конусе с углом разлета, рассчитанным из геометрических соображений: частицы, вылетающие из источника, должны максимально задействовать площадь окна, т.е. в основании конуса будет лежать круг с радиусом, равным половине ширины окна. На рис. 2 и рис. 3 приводятся визуализации с одним событием и 100 событиями соответственно.

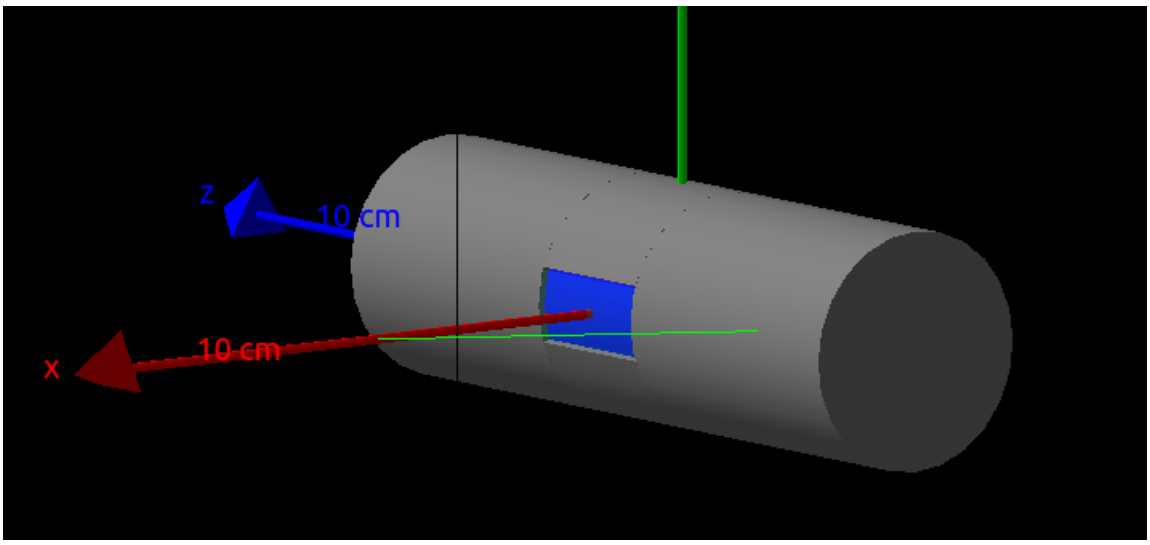


Рис. 2: Визуализация модели детектора с одним событием в Geant4 (энергия $E_\gamma = 20$ кэВ)

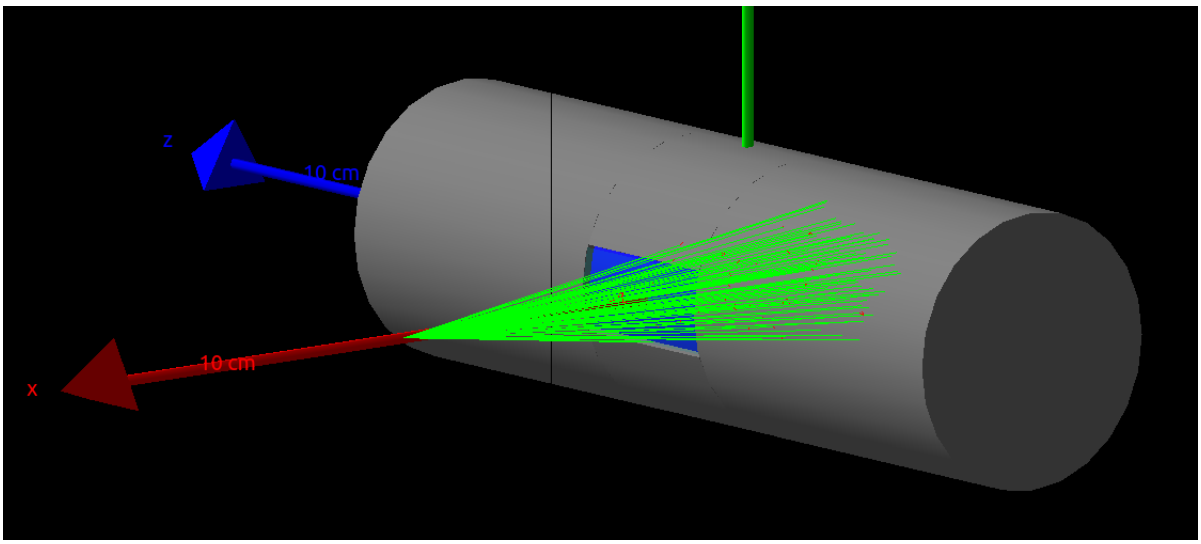


Рис. 3: Визуализация модели детектора с сотней событий в Geant4 (энергия $E_\gamma = 20$ кэВ)

2 Энергетические спектры

Будем записывать данные в NTuple с учетом неидеальности детектора. Ниже приведена таблица с расчетом энергетического разрешения детектора. Энергетическое разрешение считается по формуле (1). Для детектора с рабочим газом Xe энергия образования $\epsilon = 22$ эВ. $F = 0.36$ – фактор Фано; шум электроники $N_{noise} = 100$ событий.

$$\delta E = \sqrt{\frac{F + \delta^2 G}{N_0(E)} + \delta^2 n + \left(\frac{N_{noise}}{N_0(E) G}\right)^2} \tag{1}$$

$$N_0(E) = \frac{E_\gamma}{\epsilon}$$

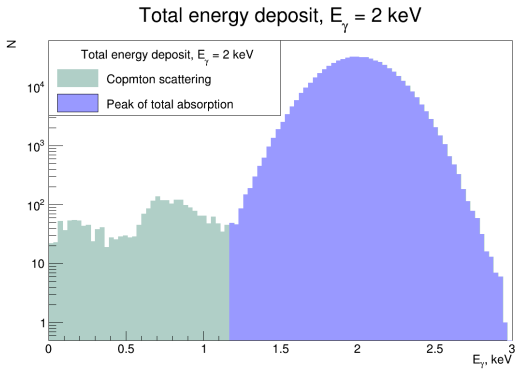
Диапазон измерения составил $2 - 20$ кэВ. Размытие энергии проводилось по следующему алгоритму:

1. Получалось полное энерговыделение в одном событии.
2. С помощью функции RandGauss() разыгрывалась флуктуация энергии для данного энерговыделения:
3. В NTuple записывалось значение с учетом флуктуации.
4. В отдельном коде обработки строятся спектры полного энерговыделения.

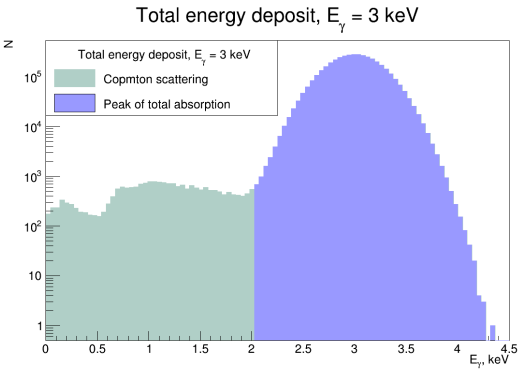
На рисунке 4(a - s) представлены спектры полного энерговыделения в детекторе. Четко видно пики полного поглощения на фоне компотоновских электронов.

Таблица 1: Расчет энергетического разрешения детектора

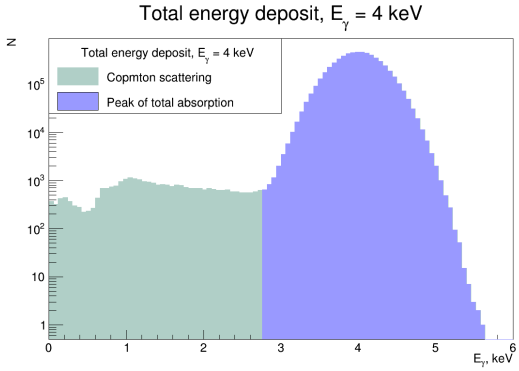
Энергия, кэВ	N_0	Энергетическое разрешение, δE
2	90	0.107475578621378
3	136	0.087790153839204
4	181	0.076126539393302
5	227	0.068200879759722
6	272	0.062370755255257
7	318	0.057853439034886
8	363	0.054222343180648
9	409	0.051222583753846
10	454	0.048691272318558
11	500	0.046518813398452
12	545	0.044628404009604
13	590	0.042964413511677
14	636	0.041485428963993
15	681	0.040159902611657
16	727	0.038963324101006
17	772	0.037876323963065
18	818	0.036883366387159
19	863	0.035971826464179
20	909	0.035131325053291



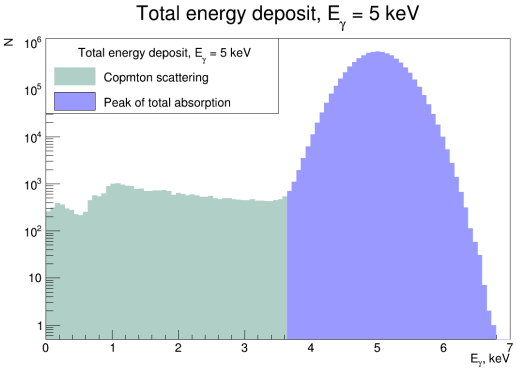
(a) Полное энерговыделение в детекторе, $E_\gamma = 2$ кэВ



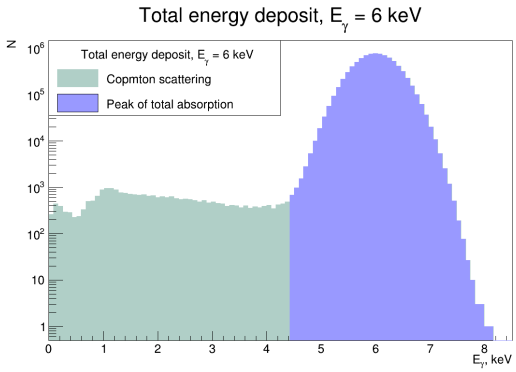
(b) Полное энерговыделение в детекторе, $E_\gamma = 3$ кэВ



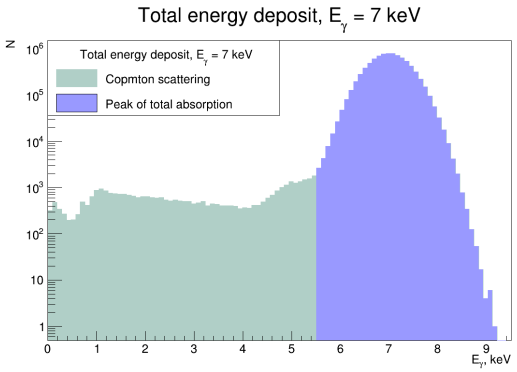
(c) Полное энерговыделение в детекторе, $E_\gamma = 4$ кэВ



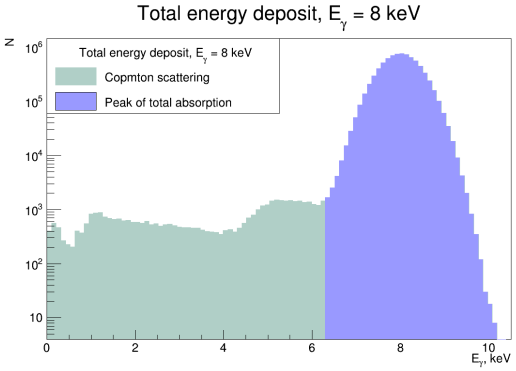
(d) Полное энерговыделение в детекторе, $E_\gamma = 5$ кэВ



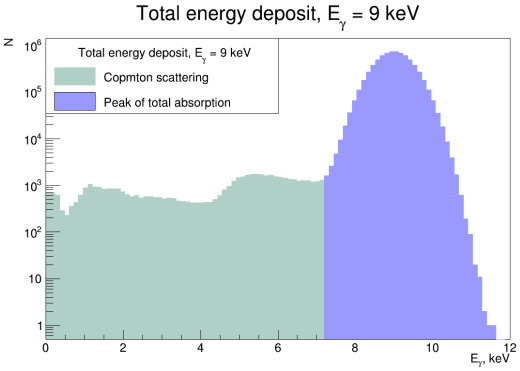
(e) Полное энерговыделение в детекторе, $E_\gamma = 6$ кэВ



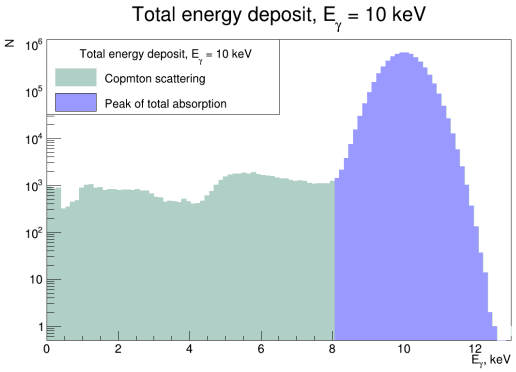
(f) Полное энерговыделение в детекторе, $E_\gamma = 7$ кэВ



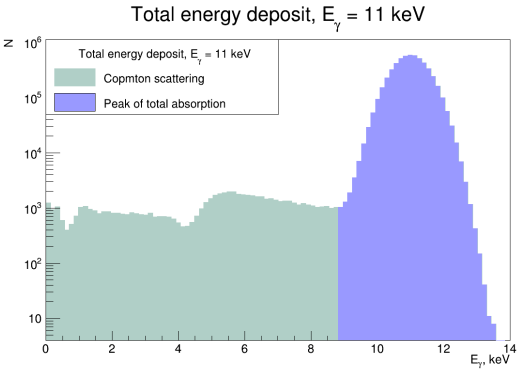
(g) Полное энерговыделение в детекторе, $E_\gamma = 8$ кэВ



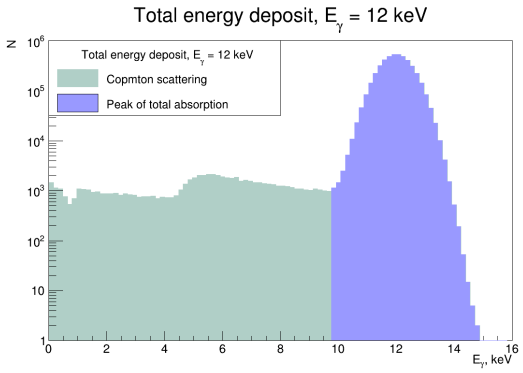
(h) Полное энерговыделение в детекторе, $E_\gamma = 9$ кэВ



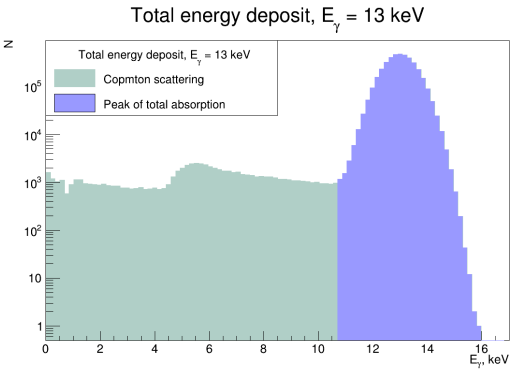
(i) Полное энерговыделение в детекторе, $E_\gamma = 10$ кэВ



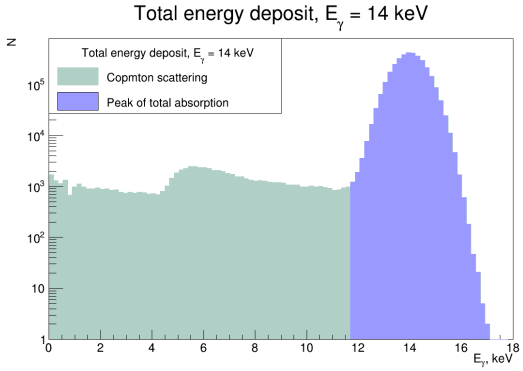
(j) Полное энерговыделение в детекторе, $E_\gamma = 11$ кэВ



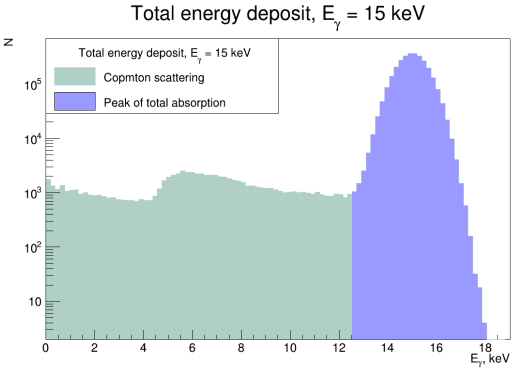
(k) Полное энерговыделение в детекторе, $E_\gamma = 12$ кэВ



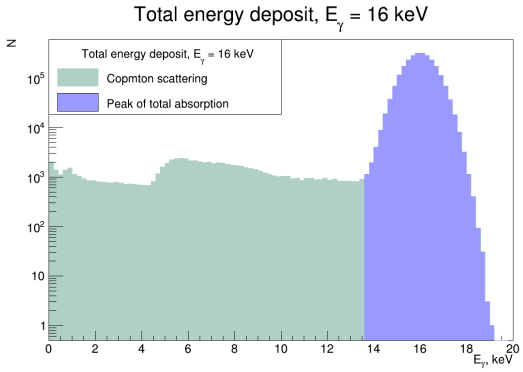
(l) Полное энерговыделение в детекторе, $E_\gamma = 13$ кэВ



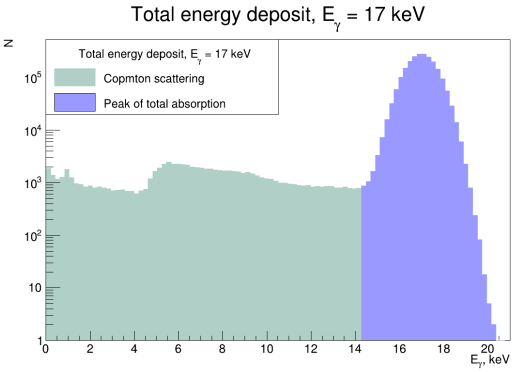
(m) Полное энерговыделение в детекторе, $E_\gamma = 14$ кэВ



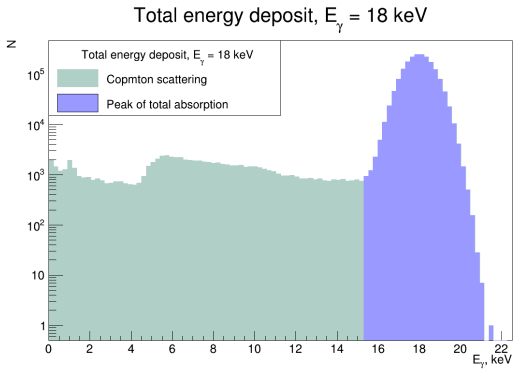
(n) Полное энерговыделение в детекторе, $E_\gamma = 15$ кэВ



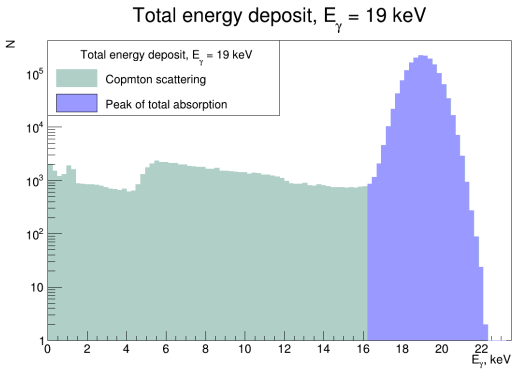
(o) Полное энерговыделение в детекторе, $E_\gamma = 16$ кэВ



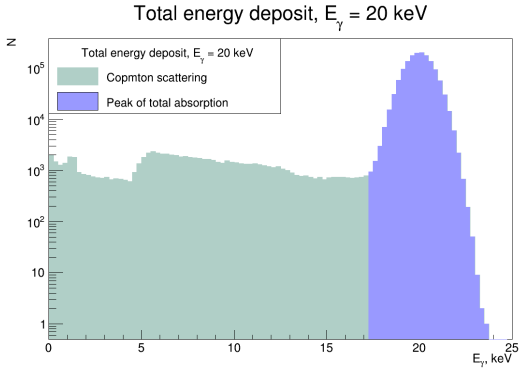
(p) Полное энерговыделение в детекторе, $E_\gamma = 17$ кэВ



(q) Полное энерговыделение в детекторе, $E_\gamma = 18$ кэВ



(r) Полное энерговыделение в детекторе, $E_\gamma = 19$ кэВ



(s) Полное энерговыделение в детекторе, $E_\gamma = 20$ кэВ

Рис. 4: Спектры полного энерговыделения в пропорциональном счетчике $Xe(95\%) + CO_2(5\%)$

3 Расчет эффективности

Эффективность детектора будем рассчитывать по формуле (2).

$$\epsilon_{reg} = \frac{\Delta\Omega}{4\pi} \cdot \frac{N_{reg}}{N_0} \tag{2}$$

$$\Delta\Omega = 2 \cdot \pi \cdot (1 - cos(\theta_{max}))$$

Здесь $\Delta\Omega$ – телесный угол, N_{reg} – число квантов под пиком полного поглощения, N_0 – число начальных частиц. В нашем случае $N_0 = 10^7$. θ_{max} – угол разлета частиц. $cos(\theta_{max}) = 0.98994949$. На рис. 5 приводится зависимость эффективности детектора от энергии.

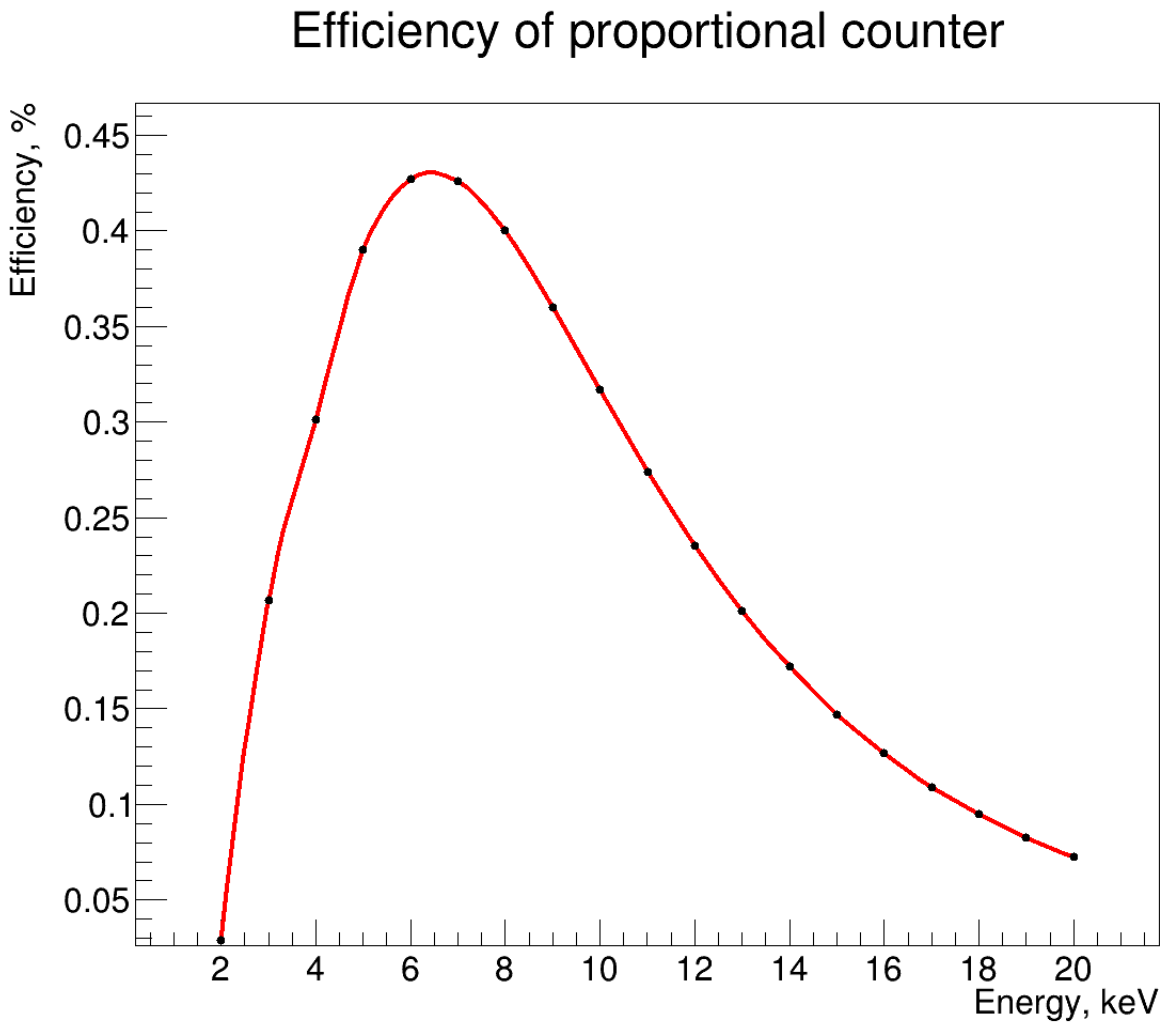


Рис. 5: Зависимость эффективности пропорционального счетчика (Xe(95%) + CO_2(5%)) от энергии

До энергий 6 кэВ наблюдается рост эффективности, затем – экспоненциальный спад. Рост объясняется увеличением числа γ-квантов, выделивших энергию в детекторе. Т.к. эффективность зависит в том числе и от вероятности частиц достичь детектора через воздушную прослойку, то частицы или выделяют часть энергии в воздухе, или задерживаются бериллиевым окном. Более высокоэнергичные частицы уже не так активно задерживаются тонким слоем бериллия и пролетают газ насквозь, поглощаясь уже в задней стенке детектора. Так объясняется падение эффективности газового детектора с ростом энергии.

4 Выводы

В рамках курсового проекта был смоделирован пропорциональный счетчик с рабочим газом Xe(95%) + CO_2(5%). Моделирование проводилось с помощью пакетов GEANT4. Были написаны код для моделирования прохождения γ-квантов через вещество детектора и макрос визуализации. Получены спектры полного энерговыделения для моноэнергетичных линий от 2 до 20 кэВ. Также для каждой энергии γ-квантов вычислена эффективность счетчика и построен соответствующий график. Из него понятно, что эффективность снижается с ростом энергии из-за того, что высокоэнергичные частицы пролетают детектор насквозь, и с малой вероятностью полностью теряют энергию.

Список литературы

[1] Agostinelli S. (Geant4 Collaboration) et al. Geant4 – a simulation toolkit // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2003.

[2] ROOT data analysis framework — open-source code // URL: https://root.cern/

[3] Б.С. Ишханов, И.М. Капитонов, Э.И. Кэбин. Частицы и ядра. Эксперимент // М.: Издательство МАКС Пресс. — 2013.

[4]

5 Приложение

В приложении приводится полный код программы с заголовочными файлами и CMakeLists.txt.

5.1 DetectorConstruction

```
#include <G4VUserDetectorConstruction.hh>
#include "G4LogicalVolume.hh"
#include "GeometrySize.hh"
#include "G4SystemOfUnits.hh"
#include <G4NistManager.hh>
#include <G4Box.hh>
#include <G4PVPlacement.hh>
#include <G4SDManager.hh>
#include "G4SystemOfUnits.hh"
#include <G4Tubs.hh>
#include "G4VSensitiveDetector.hh"
#include "G4RotationMatrix.hh"
#include "G4Material.hh"

using namespace CLHEP;

class DetectorConstruction : public G4VUserDetectorConstruction{
public:
  G4VPhysicalVolume* Construct() override;
  explicit DetectorConstruction(){
    InitializeMaterials();
  }
  //virtual ~DetectorConstruction();
private:
  G4Material * steel;
  G4Material * berillium;
  G4Material * XeCo2;
  G4Material * CO2;
  G4Material * xenon;
  G4Material * air;


```

```
  G4LogicalVolume * detector_logic;
  G4LogicalVolume * detector_logical;
  G4LogicalVolume * tube_with_hole_logic;
  G4LogicalVolume * cylinder_upper_logic;
  G4LogicalVolume * logicWorld;
  G4LogicalVolume * cylinder_cap_logic;
  G4LogicalVolume * cylinder_down_logic;
  G4LogicalVolume * cylinder_seccap_logic;
  G4LogicalVolume * window_logic;
  G4LogicalVolume * gas_logic;

  void InitializeMaterials();
  G4LogicalVolume * CreateDetector();
};
```

```
#include "G4SystemOfUnits.hh"
using namespace CLHEP;

const double hole_lenght = 40 * mm;
const double hole_diameter = 30 * mm;
const double cylinder_lenght = 42 * mm;
const double cylinder_diameter = 32 * mm;
//

const double empty_length = 20 * mm;

const double wind_width = 10 * mm;
const double wind_thick = 40.e-3 * mm;
//           Be    -

const double full_lengh = 100 * mm;
const double full_diameter = 30 * mm;
```

```
#include "DetectorConstruction.hh"

G4VPhysicalVolume* DetectorConstruction::Construct() {
    G4bool checkOverlaps = true;

    G4double world_sizeXYZ = 30 * cm;

```

```
8     auto solidWorld =
9           new G4Box("World",
10                  0.5 * world_sizeXYZ,
11                  0.5 * world_sizeXYZ,
12                  0.5 * world_sizeXYZ);
13
14    logicWorld =
15          new G4LogicalVolume(solidWorld,
16                              air,
17                              "World");
18
19    auto physWorld =
20          new G4PVPlacement(0,
21                            G4ThreeVector(0, 0, 0),
22                            logicWorld,
23                            "World",
24                            0,
25                            false,
26                            0,
27                            checkOverlaps);
28
29    detector_logic = CreateDetector();
30    return physWorld;
31 }
32
33 G4LogicalVolume* DetectorConstruction::CreateDetector() {
34   //rotation matrix (for cylinder and window)
35   G4RotationMatrix * rotm = new G4RotationMatrix();
36   rotm -> rotateZ(-18.20835*deg);
37   G4RotationMatrix * rotwin = new G4RotationMatrix();
38   rotwin -> rotateZ(18.2083*deg);
39
40 auto cylinder_upper_solid =
41       new G4Tubs("cylinder",
42                  0.5 * hole_diameter,
43                  0.5 * cylinder_diameter,
44                  0.5 * hole_lenght,
45                  0,
46                  2 * pi);
47
```

```
48   cylinder_upper_logic =
49          new G4LogicalVolume(cylinder_upper_solid,
50                              steel,
51                              "cylinder_upper");
52
53   auto cylinder_upper_phys =
54          new G4PVPlacement(0,
55                            G4ThreeVector(0, 0, -30 * mm),
56                            cylinder_upper_logic,
57                            "cylinder_upper",
58                            logicWorld,
59                            false,
60                            0);
61
62  //cylinder hat:
63  auto cylinder_cap_solid =
64         new G4Tubs("cylinder",
65                    0,
66                    0.5 * cylinder_diameter,
67                    0.5 * (cylinder_lenght - hole_lenght),
68                    0,
69                    2 * pi);
70
71  cylinder_cap_logic =
72         new G4LogicalVolume(cylinder_cap_solid,
73                             steel,
74                             "cylinder_cap");
75
76  auto cylinder_cap_phys =
77         new G4PVPlacement(0,
78                           G4ThreeVector(0, 0, -50 * mm),
79                           cylinder_cap_logic,
80                           "cylinder_cap",
81                           logicWorld,
82                           false,
83                           0);
84
85 //part of cylinder with Be-window
86     auto tube_with_hole_solid =
87            new G4Tubs("tube_with_hole",
```

```
88                      0.5 * hole_diameter,
89                      0.5 * cylinder_diameter,
90                      0.5 * empty_length,
91                      0,
92                      323.5833 * degree);
93
94    tube_with_hole_logic =
95           new G4LogicalVolume(tube_with_hole_solid,
96                               steel,
97                               "tube_with_hole");
98
99    auto tube_with_hole_phys =
100          new G4PVPlacement(rotm,
101                            G4ThreeVector(0, 0, 0),
102                            tube_with_hole_logic,
103                            "tube_with_hole",
104                            logicWorld,
105                            false,
106                            0);
107
108   auto cylinder_down_solid =
109          new G4Tubs("cylinder",
110                     0.5 * hole_diameter,
111                     0.5 * cylinder_diameter,
112                     0.5 * hole_lenght,
113                     0,
114                     2 * pi);
115
116   cylinder_down_logic =
117          new G4LogicalVolume(cylinder_down_solid,
118                              steel,
119                              "cylinder_down");
120
121   auto cylinder_down_phys =
122          new G4PVPlacement(0,
123                            G4ThreeVector(0, 0, 30 * mm),
124                            cylinder_down_logic,
125                            "cylinder_down",
126                            logicWorld,
127                            false,
```

```
128                         0);
129
130   //one more hat
131   auto cylinder_seccap_solid =
132         new G4Tubs("cylinder",
133                    0,
134                    0.5 * cylinder_diameter,
135                    0.5 * (cylinder_lenght - hole_lenght),
136                    0,
137                    2 * pi);
138
139   cylinder_seccap_logic =
140         new G4LogicalVolume(cylinder_seccap_solid,
141                             steel,
142                             "cylinder_seccap");
143
144   auto cylinder_seccap_phys =
145         new G4PVPlacement(0,
146                           G4ThreeVector(0, 0, 50 * mm),
147                           cylinder_seccap_logic,
148                           "cylinder_seccap",
149                           logicWorld,
150                           false,
151                           0);
152   //Be-window
153   auto window_solid =
154         new G4Tubs("window",
155                    0.5 * hole_diameter - wind_thick,
156                    0.5 * hole_diameter,
157                    0.5 * empty_length,
158                    0,
159                    (360 - 323.5833) * degree);
160
161   window_logic =
162         new G4LogicalVolume(window_solid,
163                             berillium,
164                             "window");
165
166   auto window_phys =
167         new G4PVPlacement(rotwin,
```

```
168                         G4ThreeVector(1 * mm, 0, 0),
169                         window_logic,
170                         "window",
171                         logicWorld,
172                         false,
173                         0);
174   //Gas geometry:
175   auto gas_solid =
176         new G4Tubs("gas",
177                    0,
178                    0.5 * full_diameter,
179                    0.5 * full_lengh,
180                    0,
181                    2*pi);
182
183   gas_logic =
184         new G4LogicalVolume(gas_solid,
185                             XeCo2,
186                             "gas");
187
188   auto gas_phys =
189         new G4PVPlacement(0,
190                         G4ThreeVector(0, 0, 0),
191                         gas_logic,
192                         "gas",
193                         logicWorld,
194                         false,
195                         0);
196
197   return gas_logic;
198 }
199
200 void DetectorConstruction::InitializeMaterials() {
201     auto nist = G4NistManager::Instance();
202     steel = nist -> FindOrBuildMaterial("G4_STAINLESS-STEEL");
203     berillium = nist -> FindOrBuildMaterial("G4_Be");
204     xenon = nist -> FindOrBuildMaterial("G4_Xe");
205     air = nist -> FindOrBuildMaterial("G4_AIR");
206
207     G4Element * oxygen = new G4Element("Oxygen", "O", 8., 16.00*g/mole);
```

```
208     G4Element * carbon = new G4Element("Carbon", "C", 6., 12.00*g/mole);
209
210     double co2_density = 0.60232 * kg/m3;
211     double density = 2.3996 * kg/m3;
212
213     CO2 = new G4Material("CO2", co2_density, 2);
214     CO2 -> AddElement(oxygen, 2);
215     CO2 -> AddElement(carbon, 1);
216
217     XeCo2 = new G4Material("XeCo2", density, 2);
218     XeCo2 -> AddMaterial(CO2, 5*perCent);
219     XeCo2 -> AddMaterial(xenon, 95*perCent);
220 }
```

5.2 PrimaryGeneratorAction

```
1  #include <G4VUserPrimaryGeneratorAction.hh>
2  #include <G4ParticleGun.hh>
3  #include <G4GeneralParticleSource.hh>
4  #include <random>
5
6  class G4ParticleGun;
7  class G4Event;
8
9  class PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction {
10 public:
11   PrimaryGeneratorAction();
12   void GeneratePrimaries(G4Event *anEvent) override;
13   ~PrimaryGeneratorAction() {}
14 private:
15   G4ParticleGun* fParticleGun;
16   G4double cos_theta_gamma;
17   G4double sin_theta_gamma;
18   G4double phi_gamma;
19 };
```

```
1  #include <G4Alpha.hh>
2  #include "G4Gamma.hh"
3  #include <G4Electron.hh>
4  #include <G4DynamicParticle.hh>
```

```
#include "PrimaryGeneratorAction.hh"
#include "G4SystemOfUnits.hh"
#include "G4ParticleTable.hh"
#include "Randomize.hh"

using namespace CLHEP;

void PrimaryGeneratorAction::GeneratePrimaries(G4Event *anEvent) {

   phi_gamma = G4UniformRand() * twopi;
   while (true){
      cos_theta_gamma = -1. + 2.0*G4UniformRand();
      if (cos_theta_gamma <= -0.98994949){
         break;
      }
   }
   sin_theta_gamma = sqrt(1 - cos_theta_gamma*cos_theta_gamma);

   fParticleGun->SetParticlePosition(G4ThreeVector(5 * cm, 0, 0));
   fParticleGun->SetParticleMomentumDirection(G4ThreeVector(cos_theta_gamma, sin_th
   fParticleGun->SetParticleEnergy(20 * keV);
   fParticleGun->SetParticleDefinition(G4Gamma::Definition());
   fParticleGun->GeneratePrimaryVertex(anEvent);
}

PrimaryGeneratorAction::PrimaryGeneratorAction() {
    G4int n_particle = 1;
    fParticleGun = new G4ParticleGun(n_particle);
}
```

5.3 RunAction

```
#ifndef TUPLEID_HH
#define TUPLEID_HH
#include "g4analysis.hh"

struct TupleId{
  G4AnalysisManager * analysisManager;
   G4AnalysisManager * analysis;
};
```

```
#endif
```

```
#include <G4UserRunAction.hh>
//#include "TupleId.hh"
class G4Run;

class RunAction : public G4UserRunAction{
public:
    RunAction();
    void BeginOfRunAction(const G4Run* aRun) override;
    void EndOfRunAction(const G4Run* aRun) override;
};
```

```
#include "RunAction.hh"
#include "g4analysis.hh"
#include <G4VSensitiveDetector.hh>
#include "TupleId.hh"

using namespace CLHEP;

RunAction::RunAction()
 : G4UserRunAction()
{}

void RunAction::BeginOfRunAction(const G4Run *aRun) {
  G4AnalysisManager *analysisManager = G4Analysis::ManagerInstance("root");
  analysisManager->OpenFile("Run_20keV");
   analysisManager -> CreateNtuple("electrons", "electrons");
   analysisManager -> CreateNtupleDColumn("total_energy_deposit");
   analysisManager -> FinishNtuple();
}

void RunAction::EndOfRunAction(const G4Run *aRun) {
  G4UserRunAction::EndOfRunAction(aRun);
  G4AnalysisManager* analysisManager = G4Analysis::ManagerInstance("root");
  analysisManager -> Write();
  analysisManager -> CloseFile(true);
}
```

5.4 SteppingAction

```
1  //SteppingAction.hh
2  #include "G4UserSteppingAction.hh"
3  #include "DetectorConstruction.hh"
4  #include "globals.hh"
5
6  class EventAction;
7
8  class G4LogicalVolume;
9
10 class SteppingAction : public G4UserSteppingAction
11 {
12 public:
13   SteppingAction(DetectorConstruction* detectorConstruction, EventAction* eventActi
14   virtual ~SteppingAction();
15   virtual void UserSteppingAction(const G4Step*);
16 private:
17   EventAction* fEventAction;
18    DetectorConstruction* fDetConstruction;
19 };
```

```
1  #include "SteppingAction.hh"
2  #include "EventAction.hh"
3  #include "G4Step.hh"
4  #include "G4RunManager.hh"
5  #include "G4LogicalVolume.hh"
6  #include "G4Event.hh"
7  #include "G4VPhysicalVolume.hh"
8
9  SteppingAction::SteppingAction(DetectorConstruction* detectorConstruction, EventAct
10        : G4UserSteppingAction(),
11          fDetConstruction(detectorConstruction),
12          fEventAction(eventAction){}
13
14 SteppingAction::~SteppingAction()
15 {}
16
17 void SteppingAction::UserSteppingAction(const G4Step* aStep){
18     if (aStep->GetTrack()->GetVolume()->GetName() == "gas") {
```

```
      G4double energy_negative = (aStep->GetTotalEnergyDeposit() / keV);
      fEventAction -> AddEdep(energy_negative);
  }
}
```

5.5 EventAction

```
//EventAction.hh
#include <G4VSensitiveDetector.hh>
#include "g4analysis.hh"
#include "TupleId.hh"
#include "G4UserEventAction.hh"
#include "globals.hh"

class EventAction : public G4UserEventAction{
public:
  EventAction();
  virtual ~EventAction();

  void BeginOfEventAction(const G4Event *event);
  void EndOfEventAction(const G4Event *event);
  G4double energy_negative;
  void AddEdep(G4double energy) { energy_negative += energy; }
};
```

```
#include "EventAction.hh"
#include "RunAction.hh"
#include "G4RunManager.hh"
#include "G4Event.hh"
#include "G4UnitsTable.hh"
#include "Randomize.hh"
#include "G4Gamma.hh"
#include "G4VProcess.hh"
#include "G4TrackingManager.hh"

EventAction::EventAction()
   : G4UserEventAction()
{}
EventAction::~EventAction() {}

```

```
void EventAction::BeginOfEventAction(const G4Event *event) {
      energy_negative = 0;
}

void EventAction::EndOfEventAction(const G4Event *event) {
    auto analysisManager = G4AnalysisManager::Instance();
    auto id = event -> GetEventID();
      if(energy_negative != 0){
         analysisManager->FillNtupleDColumn(0, 0, energy_negative +
            CLHEP::RandGauss::shoot(0., 0.035131325053291 * energy_negative));
         analysisManager->AddNtupleRow(0);
    }
}
```

5.6 ActionInitialization

```
//ActionInitialization.hh

#include "G4VUserActionInitialization.hh"
//#include "DetectorConstruction.hh"

class DetectorConstruction;

class ActionInitialization : public G4VUserActionInitialization
{
  public:
    explicit ActionInitialization(DetectorConstruction*);
    virtual ~ActionInitialization();

    virtual void BuildForMaster() const;
    virtual void Build() const;
  private:
   DetectorConstruction *DetConstruction;
};
```

```
//ActionInitialization.cc
#include "ActionInitialization.hh"
#include "PrimaryGeneratorAction.hh"
#include "RunAction.hh"
#include "EventAction.hh"
```

```
#include "SteppingAction.hh"
//#include "DetectorConstruction.hh"

ActionInitialization::ActionInitialization(DetectorConstruction *fDetConstruction)
 : G4VUserActionInitialization(),
  DetConstruction(fDetConstruction)
{}

ActionInitialization::~ActionInitialization()
{}

void ActionInitialization::BuildForMaster() const
{}

void ActionInitialization::Build() const
{
   SetUserAction(new PrimaryGeneratorAction());
  SetUserAction(new RunAction());
  auto eventAction = new EventAction();
   SetUserAction(eventAction);
  SetUserAction(new SteppingAction(DetConstruction, eventAction));
}
```

5.7 main и CMakeLists

```
#include <DetectorConstruction.hh>
#include "ActionInitialization.hh"
#include <QGSP_BERT.hh>
#include <G4VisManager.hh>
#include <G4VisExecutive.hh>
#include <G4UIExecutive.hh>
#include <G4UImanager.hh>
#include <RunAction.hh>
#include <random>
#include "G4RunManager.hh"
#include "G4ScoringManager.hh"

using namespace CLHEP;

int main(int argc, char **argv) {
```

```
    std::random_device rd;
    std::uniform_int_distribution<long> uid(0, LONG_MAX);
    long seed = uid(rd);
    HepRandom::setTheSeed(seed);
    //std::cout<<"Seed "<<seed<<std::endl;
    G4UIExecutive *ui = nullptr;
    if(argc == 1) {
        ui = new G4UIExecutive(argc, argv);
    }
    //TupleId *tuple = new TupleId();
   DetectorConstruction *geom = new DetectorConstruction();
  auto runManager = new G4RunManager;
   G4ScoringManager* scoringManager = G4ScoringManager::GetScoringManager();
  runManager->SetUserInitialization(new QGSP_BERT());
  runManager->SetUserInitialization(geom);
   auto actionInitialization = new ActionInitialization(geom);
   runManager->SetUserInitialization(actionInitialization);
  runManager->Initialize();

    G4VisManager* visManager = new G4VisExecutive;
    visManager->Initialise();

    auto UImanager = G4UImanager::GetUIpointer();

    if(!ui) {
        G4String command = "/control/execute ";
        G4String fileName = argv[1];
        UImanager->ApplyCommand(command + fileName);
   } else {
        UImanager->ApplyCommand("/control/execute start0.mac");
        ui->SessionStart();
       delete ui;
    }
    delete visManager;
    delete runManager;
}
```

```
cmake_minimum_required(VERSION 3.8 FATAL_ERROR)
project(Geant4Detector)

```

```
option(WITH_GEANT4_UIVIS "Build example with Geant4 UI and Vis drivers" ON)
if(WITH_GEANT4_UIVIS)
    find_package(Geant4 REQUIRED ui_all vis_all)
else()
    find_package(Geant4 REQUIRED)
endif()

include(${Geant4_USE_FILE})
#include_directories(${PROJECT_SOURCE_DIR}/include)
include_directories(${PROJECT_SOURCE_DIR}/include)
file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh)

add_executable(Geant4Detector main.cc ${sources})
target_link_libraries(Geant4Detector ${Geant4_LIBRARIES})
```

5.8 Макрос визуализации

```
/vis/open OGL
/control/saveHistory

#/vis/verbose errors
#/vis/viewer/set/background white
/vis/geometry/set/colour all -1 0 0 0
/vis/drawVolume World
/vis/scene/add/trajectories
/vis/scene/endOfEventAction accumulate
/vis/scene/add/axes

/vis/geometry/set/colour tube_with_hole 0 0.5 0.5 0.5
/vis/geometry/set/colour cylinder_upper 0 0.5 0.5 0.5
/vis/geometry/set/colour cylinder_down 0 0.5 0.5 0.5
/vis/geometry/set/colour cylinder_cap 0 0.5 0.5 0.5
/vis/geometry/set/colour cylinder_seccap 0 0.5 0.5 0.5

/vis/geometry/set/colour window 1.0 1.0 1.0 0.1
/vis/geometry/set/colour gas 0 0.1 0.1 1.0

/vis/geometry/set/forceSolid tube_with_hole
/vis/geometry/set/forceSolid cylinder_upper
```

```
23 /vis/geometry/set/forceSolid cylinder_down
24 /vis/geometry/set/forceSolid cylinder_cap
25 /vis/geometry/set/forceSolid cylinder_seccap
26
27 /vis/geometry/set/forceSolid window
28 /vis/geometry/set/forceSolid gas
29
30 /vis/scene/add/date
31 /vis/scene/add/hits
32
33 /gun/particle gamma
34 #/gun/energy 8 keV
35 /run/beamOn 10000000
```