UNIVERSIDADE ESTADUAL PAULISTA
"JULIO DE MESQUITA FILHO"
INSTITUTO DE GEOCIÊNCIAS E CIÊNCIAS EXATAS

Relatório PIBIC

Introdução à Simulação de Eventos em Colisões pp a Altas Energias

Gabriel Oliveira Dos Santos
(Gabriel-oliveira.santos@unesp.br)

Prof. Dr. Thiago Rafael Fernandez Perez Tomei
(Thiago.Tomei@unesp.br)

Rio Claro-SP
2025

Resumo

Este trabalho aplica o simulador Geant4 na modelagem das interações entre partículas e matéria, com foco na simulação de uma câmara de nuvens. A simulação foi projetada para detectar e analisar partículas alfa e múons, que geram rastros característicos ao atravessar um meio ionizável. Utilizando o Geant4, foi possível reproduzir a trajetória dessas partículas, validando a precisão dos modelos implementados. As partículas alfa, compostas por dois prótons e dois nêutrons, produziram rastros curtos e espessos, enquanto os múons, devido à sua alta energia e baixa interação com o meio, apresentaram rastros longos e retilíneos. A simulação também destacou a importância da configuração dos materiais e das propriedades físicas do ambiente para a representação realista dos fenômenos físicos. O desenvolvimento do detector seguiu uma abordagem modular, com definição detalhada dos materiais, geometria e processos físicos envolvidos. A implementação incluiu a criação de classes para o gerador primário, fontes de partículas e organização lógica da simulação, garantindo flexibilidade para a execução e análise dos eventos simulados.

Palavras-chave: Simulação, Geant4, Câmara de Nuvens, Física de Partículas

1 Introdução

A simulação computacional desempenha um papel central no avanço científico e tecnológico, possibilitando a investigação de fenômenos físicos complexos que frequentemente desafiam a observação direta em condições laboratoriais. Por meio de modelos matemáticos e computacionais, torna-se possível prever o comportamento de partículas subatômicas em diferentes cenários, analisar padrões de interação com a matéria e otimizar o planejamento de experimentos. Essa abordagem revela-se indispensável em campos como física de partículas, engenharia nuclear, ciência dos materiais e radiologia médica, onde experimentações diretas muitas vezes se mostram inviáveis técnica e economicamente.

Nesse contexto, o *Geometry And Tracking 4* (Geant4) consolida-se como uma ferramenta de simulações para estudos de interação partícula-matéria. Desenvolvido e mantido pela Organização Europeia para a Pesquisa Nuclear (CERN), este *framework* oferece uma plataforma modular e altamente personalizável para simulações estocásticas de transporte de partículas. Sua arquitetura baseada em objetos permite a modelagem precisa de geometrias experimentais, a seleção de modelos físicos específicos e a análise estatística de processos de decaimento e colisão.

As aplicações do Geant4 abrangem um espectro interdisciplinar: desde o projeto de detectores para aceleradores de partículas e experimentos astrofísicos até aplicações médicas como planejamento radioterápico, dosimetria de precisão e desenvolvimento de equipamentos de imagem. Sua flexibilidade permite a integração com técnicas de machine learning e otimização estocástica.

Neste relatório, serão abordados os princípios fundamentais do Geant4, sua arquitetura orientada a objetos e o ecossistema de ferramentas associadas na seção 2. Além disso, será realizado um estudo de caso envolvendo a simulação de uma câmera de nuvens na seção 3. Em seguida, será apresentada uma discussão dos resultados na seção 4, e, por fim, a conclusão na seção 5.

2 Geant4

Nesta pesquisa, foi utilizado como principal referência o minicurso *First steps with Geant4* [1], ministrado por membros do CERN. A modelagem computacional no Geant4 [2] permite a transição entre o mundo físico e o mundo virtual, onde experimentos podem ser replicados, testados e refinados antes de serem realizados em condições laboratoriais reais. Esse processo baseia-se em uma descrição precisa da geometria do ambiente experimental, na definição detalhada das propriedades dos materiais e na especificação dos processos físicos que governam a interação partícula-matéria.

O Geant4 utiliza métodos de Monte Carlo para simular essas interações, permitindo que eventos individuais sejam gerados estocasticamente. Esses eventos seguem distribuições estatísticas que podem ser observadas no mundo real, como a deposição de energia (dE/dx) e as seções de choque (σ). Assim, o Geant4 proporciona previsões quantitativas sobre grandezas físicas, enquanto também oferece uma compreensão qualitativa de como as partículas se propagam e interagem dentro de um meio. Através da visualização dos trajetos das partículas e da análise detalhada dos resultados, os cientistas podem otimizar dispositivos experimentais, refinar hipóteses teóricas e desenvolver novas tecnologias.

O Geant4 é um *framework* flexível, não oferecendo um programa pronto, mas sim um conjunto de ferramentas que o usuário pode configurar de acordo com as necessidades específicas de cada simulação. Isso inclui a definição da geometria do detector, a escolha das partículas envolvidas, os processos físicos a serem considerados e a geração de partículas primárias. O Geant4 também permite a personalização dos parâmetros de simulação através de comandos de interface de usuário (UI), permitindo a modificação de elementos como material e espessura do alvo sem a necessidade de recompilar o código.

A estrutura do Geant4 pode ser representada matematicamente, com os processos físicos descritos pelas equações de interação das partículas com o meio. A equação geral de deposição de energia por uma partícula em um meio pode ser expressa como:

$$\frac{dE}{dx} = \frac{A}{\beta^2}\left(\frac{1}{\ln(2m_e c^2/\varepsilon_{min})} - \frac{1}{\beta^2}\right) \tag{2.1}$$

onde dE/dx é a taxa de deposição de energia, A é uma constante que depende do material, β é a velocidade da partícula dividida pela velocidade da luz, m_e é a massa do elétron, c é a velocidade da luz e ε_{min} é a energia mínima para a produção de pares. Além disso, o cálculo da seção de choque, que descreve a probabilidade de interação entre partículas, pode ser modelado por diferentes fórmulas dependendo do tipo de interação (como a interação eletromagnética ou forte).

O minicurso destacou ainda a criação de uma aplicação de simulação do zero, com um cenário simples: um alvo em forma de caixa feito de silício, com espessura configurável, e uma fonte de partículas que gera, por exemplo, elétrons de 4 MeV, conforme ilustrado na figura 2.1.

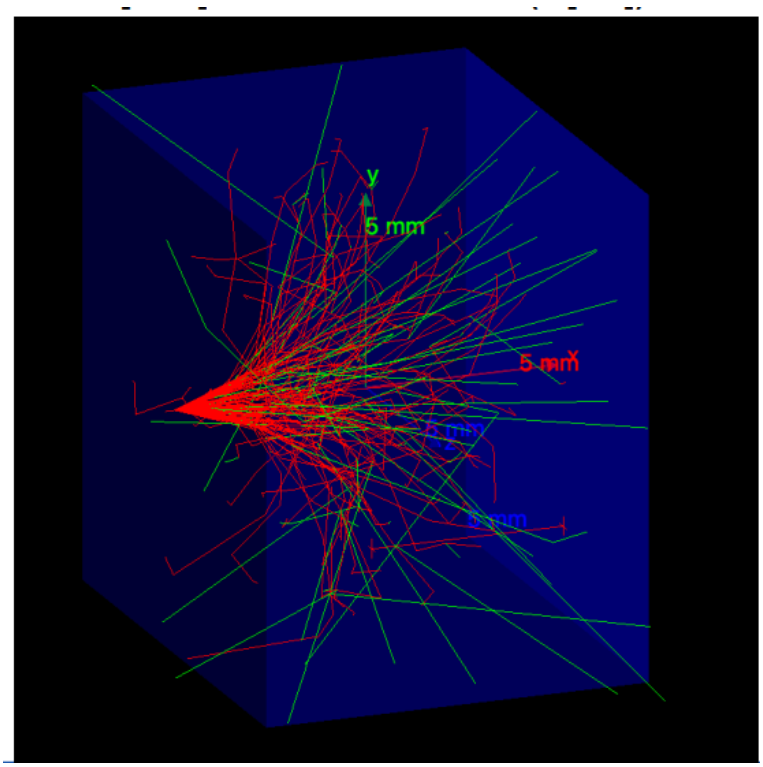

Figura 2.1: Caixa de silício sendo bombardeada por elétrons com energia de 4 MeV.

O objetivo principal dessa simulação foi coletar informações sobre a energia depositada no alvo, evidenciando como o Geant4 pode ser usado para estudar fenômenos físicos de forma detalhada. Para isso, foram abordados os componentes essenciais de uma aplicação Geant4, como o **G4RunManager**, que gerencia o fluxo da simulação, e as classes **G4VUserDetectorConstruction**, **G4VUserPhysicsList** e **G4VUserPrimaryGeneratorAction**, responsáveis, respectivamente, por definir a geometria do detector, os processos físicos e a geração de partículas primárias.

Para utilizar o Geant4, é essencial ter conhecimento em programação na linguagem $C++$ [3], sendo altamente recomendada a utilização de programação orientada a objetos (POO) [4]. A POO facilita a reprodutibilidade e reutilização do código, permitindo, por exemplo, a criação de sensores ou detectores modulares, que podem ser facilmente adaptados ou reutilizados em diferentes simulações.

3 Criação da Câmara de Nuvens

A câmara de nuvens é um dispositivo de detecção de partículas carregadas, desenvolvido no início do século XX pelo físico escocês Charles Wilson [5]. Originalmente concebida para estudar fenômenos atmosféricos, essa tecnologia revolucionou a física de partículas ao permitir a visualização direta da trajetória de partículas subatômicas, como elétrons, pósitrons e múons.

O princípio de funcionamento da câmara de nuvens baseia-se na condensação de um vapor supersaturado quando uma partícula ionizante atravessa o meio. A ionização induzida pela partícula serve como núcleo para a formação de pequenas gotículas, resultando em trilhas visíveis que revelam sua trajetória. Existem dois principais tipos de câmaras de nuvens: a de expansão e a de difusão. A primeira utiliza uma rápida descompressão do vapor para criar a supersaturação necessária, enquanto a segunda mantém um gradiente de temperatura para gerar um ambiente contínuo de detecção.

Desde sua invenção, a câmara de nuvens tem sido amplamente utilizada para o estudo de raios cósmicos e interações fundamentais na física de partículas. A técnica permitiu descobertas históricas, como a detecção do pósitron por Carl Anderson, contribuindo significativamente para a compreensão do Modelo Padrão da física de partículas. Hoje, apesar do desenvolvimento de detectores mais sofisticados, a câmara de nuvens continua sendo uma ferramenta importante para visualização direta de partículas e experimentos didáticos.

Com o uso do Geant4, será possível modelar a propagação de partículas ionizantes dentro da câmara, considerando fenômenos físicos como ionização, difusão e formação de trilhas de condensação. Essa abordagem permite analisar o comportamento dessas partículas, possibilitando uma compreensão detalhada dos mecanismos de formação das trilhas e das interações subatômicas envolvidas no processo.

Como base estrutural da câmara, será adotado o modelo apresentado no vídeo do canal do YouTube Manual do Mundo, intitulado "Fizemos um DETECTOR de RADIAÇÃO CASEIRO"[6]. Esse vídeo oferece uma abordagem prática e acessível para a construção da câmara, servindo como referência para o design.

Por se tratar de uma simulação, os materiais necessários são significativamente simplificados, dispensando o uso de elementos como gelo seco e esponjas embebidas em álcool para criar a saturação, como é comum em experimentos reais. Na simulação, será utilizado um aquário, ou seja, um bloco de vidro com dimensões de $30 \times 30 \times 45$ cm e espessura de 5 mm, que funcionará como a estrutura principal do experimento. Essa abordagem tem como objetivo reduzir a complexidade física envolvida, permitindo que o foco esteja na simulação dos fenômenos de propagação e interação das partículas ionizantes.

A criação do detector para a simulação envolve a implementação de várias classes essenciais dentro do Geant4, incluindo a classe Detector, Gerador Primário e Fonte de Partículas. A primeira etapa do processo consiste na construção da classe do detector, que

exige uma definição clara da geometria e a escolha dos materiais utilizados na simulação.

3.1 Criando o Detector

A criação do Detector pode ser dividida nas seguintes etapas: Geometria do Detector, Materiais e Elementos, Construção da Geometria e Organização Lógica e Física.

3.1.1 Geometria do Detector

Para criar o detector, é necessário definir sua geometria e o espaço onde ele estará inserido. O Geant4 utiliza o conceito de volumes para representar o espaço físico da simulação. Neste caso, o detector é modelado como um volume cúbico (retangular), representado pela classe **G4Box()**. Essa classe cria uma caixa retangular simétrica em todas as direções. O volume do mundo (onde o detector está inserido) é definido por três valores que representam suas dimensões em x,y e z. No programa, essas dimensões são ajustadas para um volume de 2 metros, formando um cubo. Isso representa a sala em que o detector está posicionado. O detector em si é centralizado dentro dessa sala, de forma que ele fique no centro de um espaço cúbico de 2 metros, e o ambiente ao redor do detector é preenchido com ar. Como mostrado na figura 4.4.

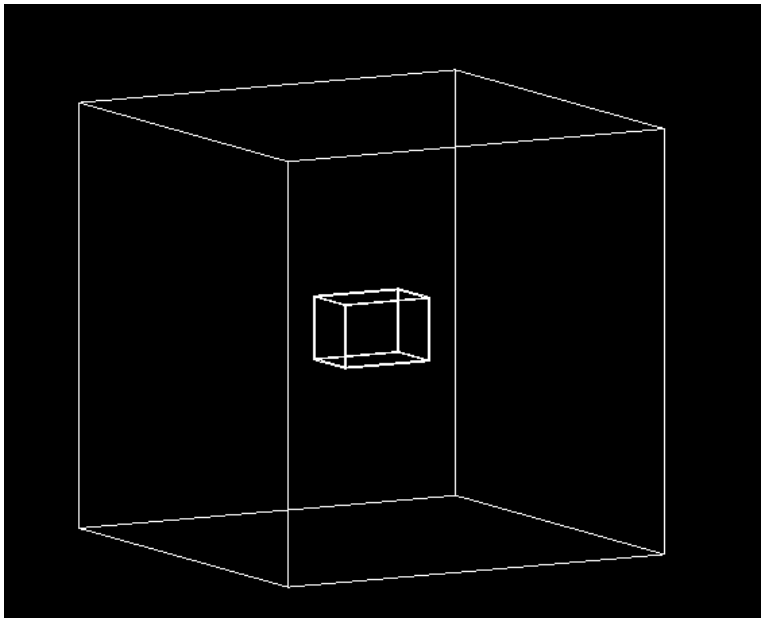

Figura 3.1: Exemplo da sala com o detector.

3.1.2 Materiais e Elementos

Em seguida, os materiais necessários são definidos. O Geant4 fornece uma lista padrão de materiais através da classe **G4NistManager**, que permite acessar e construir materiais como ar e vidro. O vidro é utilizado para representar as paredes do aquário no detector, e o ar é utilizado para preencher a sala ao redor do aquário.

Para o etanol saturado, o Geant4 não oferece um material específico, então é necessário criar o material a partir dos seus componentes químicos. O etanol é composto por carbono (C), hidrogênio (H) e oxigênio (O), que são representados como elementos na simulação, através da classe **G4Element**. O etanol é então criado com a classe **G4Material**, onde são especificadas as proporções dos elementos que compõem o etanol.

Além disso, é importante ajustar a densidade do etanol. Em condições de temperatura e pressão alteradas, o etanol tende a aumentar sua densidade. O valor de densidade do etanol a 20°C é 0.79 g/cm^3, mas como o ambiente simula uma diferença de temperatura, a densidade do etanol é ajustada para 1.05 g/cm^3, refletindo uma maior compactação do líquido em temperaturas mais baixas. Esse ajuste é realizado levando em consideração a mudança na energia cinética das moléculas de etanol.

3.1.3 Construção da Geometria do Aquário

O próximo passo é definir a geometria do aquário, que será o recipiente onde o etanol será colocado. O aquário é modelado com dimensões específicas de $30 \times 30 \times 45$ cm, e sua espessura de vidro é de 5 mm. Para isso, duas caixas são criadas: uma representando o volume externo do aquário e outra representando o volume interno (onde o etanol será colocado).

A classe **G4SubtractionSolid** é utilizada para subtrair a geometria interna da geometria externa, criando assim as paredes do aquário. Isso permite que o espaço interno seja vazio (onde o etanol será inserido) e o volume externo represente o material de vidro.

3.1.4 Organização Lógica e Física

Uma vez definidas as geometrias dos volumes (o quarto e o aquário), é necessário associar a geometria com os materiais. Isso é feito criando volumes lógicos com a classe **G4LogicalVolume**. Cada volume lógico representa uma parte do detector ou do ambiente e associa a geometria à propriedade material correspondente. A classe **G4VPhysicalVolume** define o posicionamento físico desses volumes no espaço. Após a criação dos volumes lógicos, os volumes físicos são instanciados e posicionados dentro do "mundo", ou seja, dentro do ambiente da simulação. No caso do aquário, o volume físico é posicionado dentro da sala, e o etanol é colocado dentro do aquário.

No final a função retorna o volume físico do quarto, que inclui o aquário com o etanol dentro dele, todo posicionado corretamente dentro da sala simulada. Esse processo de construção assegura que todos os elementos estão corretamente posicionados e que os materiais são representados de forma precisa, permitindo que a simulação de partículas ionizantes seja realizada dentro do contexto físico correto.

3.2 Gerador Primário

A classe **GeradorPrimario** é responsável por criar as partículas que serão utilizadas na simulação. No Geant4, essa função é desempenhada por meio da interface **G4VUserPrimaryGeneratorAction**, que define a ação para a criação de partículas primárias (partículas que iniciam a simulação de eventos). A principal tarefa dessa classe é gerar partículas primárias com parâmetros específicos, como tipo, energia e posição de emissão.

3.2.1 Construtor e Destruidor

No construtor da classe **GeradorPrimario**, é instanciada a classe **G4ParticleGun**, que é usada para disparar partículas na simulação. A **G4ParticleGun** precisa ser associada a uma partícula, e, neste caso, é escolhida a partícula **G4Geantino**. O Geantino é uma partícula fictícia que não interage com o meio e é usada principalmente para testar a simulação sem influenciar os resultados. Ele serve como um marcador ou uma partícula "nula", o que pode ser útil para validar o funcionamento do gerador sem a complexidade das interações reais. O objeto **m_particleGun** é, então, inicializado com **G4Geantino::Definition()**, que obtém a definição da partícula Geantino. Isso significa que, por padrão, o gerador de partículas criará geantinos.

No destruidor da classe **GeradorPrimario** , o ponteiro para **m_particleGun** é deletado. O Geant4 usa gerenciamento automático de memória, mas é uma boa prática garantir que objetos dinâmicos sejam explicitamente desalocados quando não forem mais necessários para evitar vazamentos de memória.

3.2.2 Função GeneratePrimaries

A função **GeneratePrimaries** é chamada a cada evento gerado na simulação. Ela é responsável por gerar as partículas primárias que serão inseridas no evento. O método **m_particleGun->GeneratePrimaryVertex(anEvent)** utiliza o **G4ParticleGun** para criar o vértice primário, ou seja, o ponto inicial da trajetória da partícula gerada.

Esse vértice é um ponto no espaço onde a partícula começa a sua trajetória e é definido pelo **m_particleGun**, que é configurado com as propriedades da partícula (neste caso, um geantino). A função **GeneratePrimaryVertex** configura a partícula primária no evento em andamento, fazendo com que ela seja gerada e esteja pronta para interagir com o resto da simulação.

3.3 Fonte de Partículas

A classe **FonteDeParticulas** é responsável por integrar a ação do gerador de partículas dentro da simulação. Ela cria e configura o gerador primário e garante que ele seja utilizado no contexto da simulação.

3.3.1 Função Build

A função **Build** define o gerador primário, criando um objeto da classe **GeradorPrimario** e o associando à simulação como um usuário de ação, por meio do método **SetUserAction()**. Este método insere o gerador primário na cadeia de ações do Geant4, permitindo que ele seja chamado automaticamente durante a execução da simulação.

Através da chamada **SetUserAction(new GeradorPrimario())**, a classe **FonteDeParticulas** está dizendo ao Geant4 que sempre que uma simulação de evento for iniciada, o **GeradorPrimario** será responsável por criar as partículas primárias, neste caso, um geantino.

A classe **FonteDeParticulas** funciona, portanto, como um intermediário entre a configuração do gerador de partículas e o restante da simulação. Ela garante que a ação de geração de partículas seja configurada corretamente para a execução da simulação.

3.4 Main

A função **main** do programa tem como objetivo configurar e inicializar todos os componentes necessários para a execução da simulação de partículas utilizando o Geant4. O processo de inicialização e execução da simulação é feito de maneira sequencial, com a configuração de diferentes módulos do sistema, como o gerenciador de execução, o detector, a fonte de partículas, a lista de física e a visualização. O primeiro passo na função **main** é a criação do **G4RunManager**, que é o núcleo do Geant4. O **runManager** gerencia todo o ciclo de vida da simulação, organizando as etapas de inicialização, execução e finalização. O Geant4 utiliza listas de física, que determinam os processos físicos a serem aplicados aos eventos da simulação. No código em questão, a lista de física chamada "Shielding"é carregada utilizando a **G4PhysListFactory**. Essa fábrica permite acessar listas de física prontas e testadas, como as fornecidas pelo CERN, garantindo que a simulação utilize uma lista de processos físicos adequados para experimentos envolvendo partículas. Com o Gerenciador de Execução criado, o próximo passo é configurar os componentes principais do ambiente de simulação. O código utiliza os seguintes componentes:

- **Detector**: O detector é configurado através da classe **MeuDetector**, que define a geometria do detector, como a câmara de nuvens.
- **Fonte de Partículas**: A classe **FonteDeParticulas** é responsável pela configuração da fonte de partículas, ou seja, ela inicializa o gerador de partículas primárias.
- **Lista de Física**: A lista de física, que contém os processos físicos que serão aplicados, foi configurada como a lista "Shielding"

Após a configuração dos componentes, o código chama o método **runManager->Initialize()**, que inicializa a simulação com todas as configurações previamente definidas.

A função main também configura o motor de visualização para a simulação. O Geant4 oferece suporte a vários motores de visualização, sendo que neste código foi escolhido o uso do **OpenGL**[7] com **Qt**[8]. Essa escolha é vantajosa pois o **OpenGL** com **Qt** permite uma visualização dinâmica e interativa, além de ser multiplataforma, funcionando em sistemas operacionais como Windows, Linux e macOS.

A classe **G4VisExecutive** é responsável por inicializar o sistema de visualização com o suporte ao **OpenGL** e **Qt**, permitindo a exibição da simulação em tempo real e interação dinâmica durante a execução.

A função **main** também permite o controle da simulação através de UI. Para isso, o programa utiliza a classe **G4UIExecutive**, que gerencia a interface de controle da simulação. Dependendo dos parâmetros fornecidos ao programa, a interface pode ser executada no modo interativo com o **Qt** ou, caso o programa receba um arquivo de macro como argumento, ele irá executar o arquivo de macro, configurando automaticamente a simulação conforme o conteúdo do arquivo.

No final da execução da simulação, é importante garantir que não haja vazamentos de memória. Isso é feito deletando todos os objetos criados dinamicamente durante a execução do programa, como o **runManager**,**factory** , **visManager** e **executarUI**.

O **Makefile** [9] facilita a automação do processo de compilação e execução do código, garantindo o gerenciamento adequado de dependências. Os alvos definidos no **Makefile** permitem compilar partes específicas do código sem a necessidade de recompilar tudo a cada alteração, economizando tempo de execução. Abaixo, é apresentada a explicação de cada um dos comandos e alvos definidos:

- **all**: Responsável por gerar o arquivo executável final do programa.
- **compilar-todos** : É útil quando há a necessidade de recompilar todos os arquivos fonte do projeto. recompilará apenas os arquivos afetados.
- **compilar-FonteDeParticulas**: Específico para a classe **FonteDeParticulas**. Ele permite compilar apenas a parte relacionada à geração das partículas sem a necessidade de recompilar o código inteiro.
- **compilar-GeradorPrimario**: Utilizado para compilar especificamente a classe **GeradorPrimario**. Caso apenas essa classe tenha sido alterada
- **compilar-MeuDetector**: Similar aos outros alvos específicos, esse alvo é utilizado para compilar apenas a classe **MeuDetector**.
- **compilar-main**: Compila o arquivo principal do programa, main.cpp.
- **run**: Facilita a execução do programa com configurações específicas de simulação, já que as macros contêm parâmetros predefinidos

Por exemplo, ao alterar apenas a classe **GeradorPrimario**, o comando make **compilar-GeradorPrimario** recompilará apenas essa classe e suas dependências, sem necessidade de recompilar as outras partes do código. Para ter acesso ao código, basta acessar a referência [10].

4 Resultados e Discussão

Agora que o detector foi implementado, podemos iniciar a simulação. Este estudo tem como foco a identificação e análise de partículas alfa e múons originados de raios cósmicos, detectados dentro de uma câmara de nuvens.

As partículas alfa são compostas por um núcleo de hélio, formado por dois prótons e dois nêutrons. Devido à sua alta energia, mas velocidade relativamente baixa em comparação com outras partículas carregadas, as partículas alfa deixam rastros característicos na câmara de nuvens: curtos, retilíneos e espessos, como ilustrado na Figura 4.1.

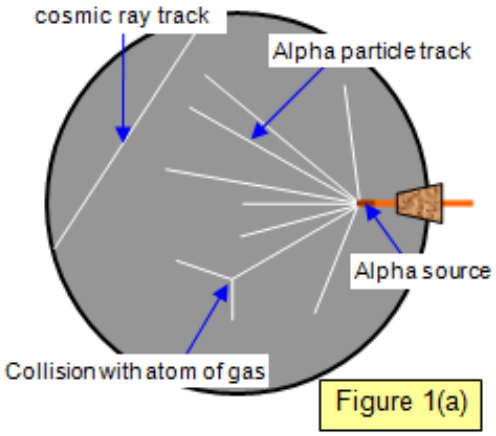

Figura 4.1: Exemplo de partículas alfa.

Na simulação (Figura 4.2), observamos rastros azuis e verdes. Os rastros azuis correspondem a partículas carregadas positivamente, enquanto os verdes representam partículas neutras. O comportamento dos rastros azuis está de acordo com o esperado para partículas alfa no meio simulado. No entanto, na realidade, os rastros verdes não seriam visíveis, uma vez que partículas neutras não ionizam as moléculas do meio e, portanto, não formariam trilhas na câmara de nuvens.

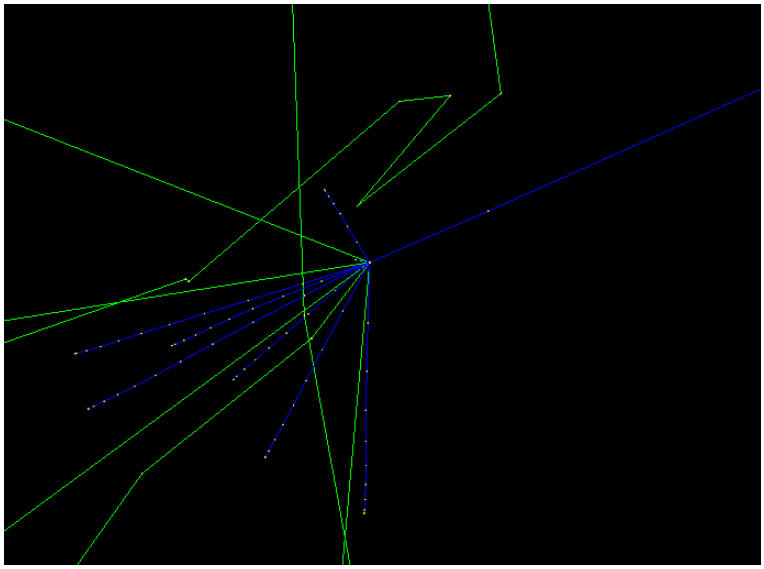

Figura 4.2: Partículas alfa simuladas e detectadas no Geant4.

A ionização e a consequente formação dos rastros ocorrem quando as partículas carregadas interagem com as moléculas do vapor presente na câmara, como etanol, água ou isopropanol. No caso das partículas alfa, essas interações ionizam as moléculas do meio, liberando elétrons e permitindo a condensação do vapor ao longo da trajetória da partícula, formando os rastros visíveis.

Os múons, por sua vez, são partículas elementares da família dos léptons, semelhantes aos elétrons, mas com uma massa aproximadamente 200 vezes maior. Eles são gerados na alta atmosfera a partir do decaimento de mésons (píons e káons), os quais são produzidos na interação de raios cósmicos primários com o ar. Devido à sua alta energia e grande tempo de vida médio, os múons conseguem atravessar grandes distâncias, alcançando o nível do solo e sendo detectáveis em câmaras de nuvens.

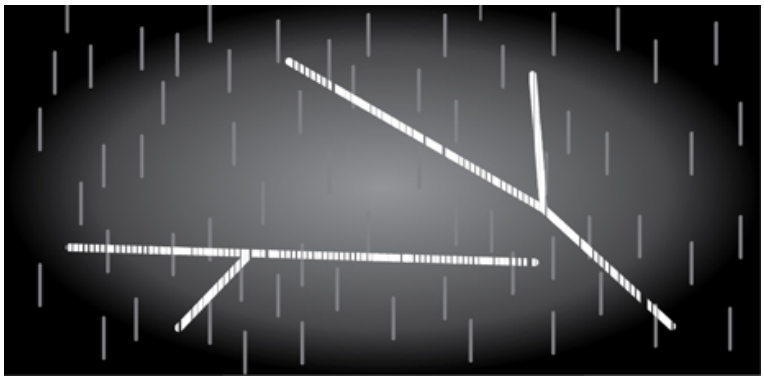

Figura 4.3: Exemplo de múon.

Na simulação (Figura 4.4), observamos diferentes rastros, como os traços vermelhos e verdes. Os rastros avermelhados representam partículas carregadas negativamente, como os múons, enquanto os verdes indicam partículas neutras. Além disso, as partículas que parecem "sair"da linha vermelha principal representam o decaimento do múon em elétrons.

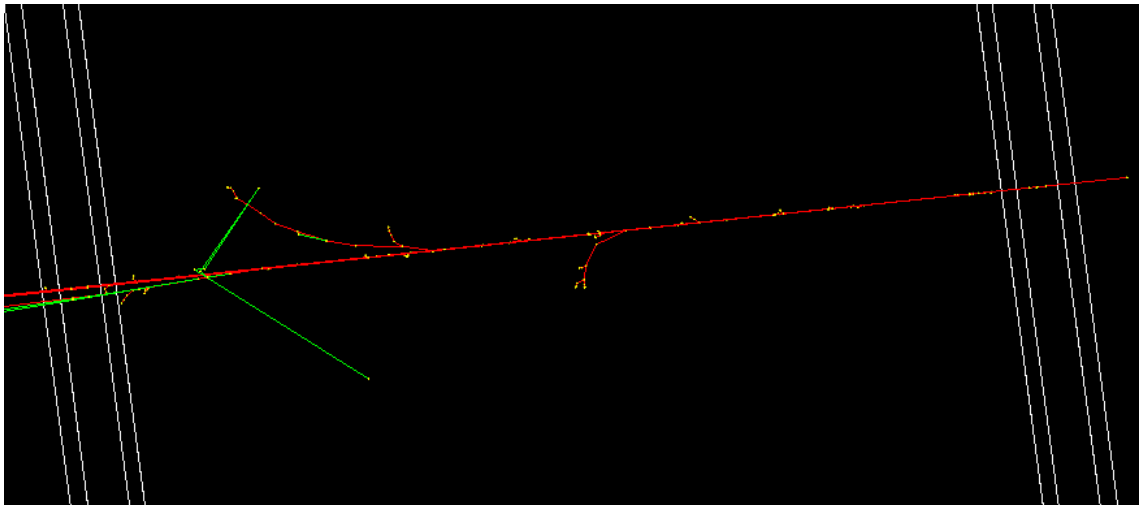

Figura 4.4: Múons simulados no detector no Geant4.

Os múons deixam rastros longos e retilíneos devido à sua alta energia e à baixa interação com o meio. Ao atravessarem a câmara de nuvens, ionizam as moléculas do vapor, permitindo a condensação ao longo de sua trajetória e formando os rastros visíveis. Também é possível observar algumas interações com o meio, como pequenas deflexões e colisões, resultantes da interação dos múons com núcleos atômicos ou elétrons presentes no vapor.

Esse comportamento é consistente com o esperado para partículas de alta energia, que apresentam baixa perda de momento ao longo do percurso, confirmando a identificação dos múons na simulação.

Para um possível aprimoramento da simulação, seria interessante a introdução de um campo magnético. A presença de um campo magnético permitiria a deflexão das partículas carregadas, facilitando a identificação de sua carga. Partículas com cargas opostas se moveriam em direções diferentes devido à força de Lorentz, o que possibilitaria uma discriminação mais clara entre partículas de diferentes naturezas, como partículas alfa e múons, por exemplo. Esse recurso não apenas melhoraria a precisão da identificação das partículas na câmara de nuvens, mas também poderia fornecer informações adicionais sobre sua energia e trajetória, contribuindo para uma análise mais detalhada e confiável dos eventos simulados.

5 Conclusão

A aplicabilidade do Geant4 na simulação das interações entre partículas e matéria possibilita a modelagem detalhada de detectores de grande porte, como os utilizados no LHC, bem como de dispositivos mais simples, como os baseados em câmaras de nuvens. Por meio do desenvolvimento e implementação de um ambiente virtual, foi possível identificar e analisar o comportamento de partículas alfa e múons, evidenciando as características dos rastros deixados na câmara.

Os resultados obtidos confirmaram a capacidade do Geant4 em reproduzir fenômenos físicos observáveis experimentalmente, fornecendo informações precisas sobre a ionização e a formação das trilhas de condensação. Além disso, a flexibilidade do framework permitiu a personalização da simulação, oferecendo maior controle sobre os parâmetros físicos envolvidos.

Como perspectivas futuras, a implementação de um campo magnético poderia aprimorar a identificação das partículas ao induzir deflexões em suas trajetórias, permitindo uma discriminação mais precisa de sua carga e energia. Esse aprimoramento tornaria a simulação ainda mais realista e valiosa para os estudos no campo da física de partículas.

6 Agradecimentos

Sou imensamente grato ao meu orientador, Thiago Rafael Fernandez Perez Tomei, por me proporcionar a oportunidade de pesquisar na área que mais me fascina na ciência. Agradeço não apenas pelo apoio acadêmico, mas também por me ajudar a crescer como cientista e, acima de tudo, como ser humano.

O presente projeto foi desenvolvido com apoio do CNPq através da concessão de bolsa de estudo de Iniciação Cientifica - Código de Financiamento 147150/2024-8

Referências

[1] Mihaly Novak, John Apostolakis, Gabriele Cosmo, Severin Diederichs, Mihaly Novak, and John Apostolakis. G4fs24 day1 session2 "compiling a first geant4 source file"and "geant4 materials", 2024.

[2] J. Allison and et al Amako. Recent developments in geant4. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 835:186–225, November 2016.

[3] ISO/IEC. Iso/iec 14882:2017 - programming languages – c++, 2017. Acesso em: 04 fev. 2025.

[4] Fernanda Farinelli. Conceitos básicos de programação orientada a objetos. *Instituto Federal Sudeste de Minas Gerais*, 2007.

[5] Isabella Marchi Coelho. Detecção e análise física de raios cósmicos através de câmaras de nuvens. 2024.

[6] Manual do Mundo. Fizemos um detector de radiação caseiro, 2025. Acessado em: 04 fev. 2025.

[7] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[8] The Qt Company. QGradient Class, 2021.

[9] Free Software Foundation. Make, 2025. A build automation tool used for automating the compilation of software.

[10] Gabriel Oliveira. Simulação de uma câmera de nuvens utilizando o geant4. `https://github.com/gabriels3t/Simulacao-Camara-De-Nuvens-Geant4`, 2025. Accessed: 2025-02-07.