UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

MAX RAFAEL VIANA RAULINO

DIVERSIFICAÇÃO DAS ESTRATÉGIAS DE ENSINO EM MECÂNICA CLÁSSICA: UTILIZAÇÃO DE UMA FERRAMENTA EM LINGUAGEM PYTHON COMO ALTERNATIVA RELEVANTE PARA SIMULAÇÃO COMPUTACIONAL

MOSSORÓ
2021

MAX RAFAEL VIANA RAULINO

DIVERSIFICAÇÃO DAS ESTRATÉGIAS DE ENSINO EM MECÂNICA CLÁSSICA: UTILIZAÇÃO DE UMA FERRAMENTA EM LINGUAGEM PYTHON COMO ALTERNATIVA RELEVANTE PARA SIMULAÇÃO COMPUTACIONAL

Monografia apresentada à Universidade Federal Rural do Semi-Árido – UFERSA, Centro de Ciências Exatas e Naturais, para obtenção do título de Bacharel em Interdisciplinar em CIência e Tecnologia.

Orientador: Prof. Dr. Fabricio de Figueredo Oliveira.

MOSSORÓ

2021

R245d Raulino, Max Rafael Viana.
Diversificação das estratégias de ensino em Mecânica Clássica: Utilização de uma ferramenta em linguagem Python como alternativa relevante para simulação computacional / Max Rafael Viana Raulino. - 2021.
38 f. : il.

Orientador: Fabricio de Figueredo Oliveira.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência e Tecnologia, 2021.

1. Aprendizagem significativa. 2. Estrategias de ensino. 3. Python. 4. Mecânica Classica. I. Oliveira, Fabricio de Figueredo, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Mossoró / Setor de Informação e Referência
Bibliotecária: Keina Cristina Santos Sousa e Silva
CRB: 15/120

MAX RAFAEL VIANA RAULINO

DIVERSIFICAÇÃO DAS ESTRATÉGIAS DE ENSINO EM MECÂNICA CLÁSSICA: UTILIZAÇÃO DE UMA FERRAMENTA EM LINGUAGEM PYTHON COMO ALTERNATIVA RELEVANTE PARA SIMULAÇÃO COMPUTACIONAL

Monografia apresentada à Universidade Federal Rural do Semi-Árido – UFERSA como requisito para obtenção do título de Bacharel em Interdisciplinar em Ciência e Tecnologia.

Defendida em:17/11/2021.

BANCA EXAMINADORA

FABRICIO DE FIGUEREDO OLIVEIRA:64905896304

Assinado de forma digital por FABRICIO DE FIGUEREDO OLIVEIRA:64905896304
Dados: 2021.11.17 09:05:13 -03'00'

Prof. Dr. Fabricio Figueredo de Oliveira (UFERSA)
Presidente

SUBENIA KARINE DE MEDEIROS:02450251464

Assinado de forma digital por SUBENIA KARINE DE MEDEIROS:02450251464
Dados: 2021.11.18 16:51:52 -03'00'

Profa. Dra. Subênia Karine de Medeiros (UFERSA)
Membro Examinador

LAZARO LUIS DE LIMA SOUSA:66761476372

Assinado de forma digital por LAZARO LUIS DE LIMA SOUSA:66761476372
Dados: 2021.11.20 09:21:46 -03'00'

Prof. Dr. Lázaro Luis de Lima Sousa (UFERSA)
Membro Examinador

RESUMO

No contexto de adaptação das estratégias de ensino de Mecânica Clássica, este trabalho tem a finalidade de discutir conceitos relacionados à aprendizagem significativa e estratégias metacognitivas na potencialização da aprendizagem com o objetivo de avaliar e justificar a inserção de ferramentas computacionais. Baseado nas dinâmicas envolvidas na aprendizagem, é abordada uma perspectiva de ensino-aprendizagem com a utilização de mecanismos e proposições didáticas nas abordagens de tópicos específicos relacionados à Mecânica Clássica. Diante disso, é apresentada uma ferramenta pedagógica como alternativa para a diversificação das estratégias de ensino de Mecânica Clássica integrando a linguagem de programação Python para realizar simulações de fenômenos físicos direcionados sobretudo aos tópicos de Lançamento de Projéteis e Colisões de Partículas.

Palavras-chave: Aprendizagem significativa. Estratégias de ensino. Python. Mecânica Clássica.

ABSTRACT

In the context of adapting Classical Mechanics teaching strategies, this work aims to discuss concepts related to meaningful learning and metacognitive strategies in enhancing learning in order to evaluate and justify the insertion of computational tools. Based on the dynamics involved in learning, a teaching-learning perspective is approached with the use of mechanisms and didactic propositions in the approach of specific topics related to Classical Mechanics. Therefore, a pedagogical tool is presented as an alternative for the diversification of Classic Mechanics teaching strategies, integrating the Python programming language to perform simulations of physical phenomena directed mainly to the topics of Projectile Launch and Particle Collisions.

Key words: Meaningful learning. Teaching Strategies. Python. Classic Mechanics.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Desafios no ensino de Física

É um grande desafio para os docentes contextualizar, elucidar e problematizar conceitos físicos muitas vezes abstratos e contraintuitivos. Tendo em vista os diferentes níveis de proficiência e diferentes ritmos de aprendizagem apresentados pelos discentes, saber selecionar proposições didáticas capazes de gerar curiosidade, motivar e estimular o interesse, independentemente do nível de conhecimento dos estudantes, é vital para a garantia de resultados satisfatórios.

De maneira equivalente, torna-se também um processo desafiador para os estudantes, enquanto protagonistas e autorreguladores de seus processos mentais, ter consciência dos próprios processos cognitivos. É indispensável, portanto, que os alunos reconheçam seus processos mentais e tenham consciência do seu próprio grau de compreensão para alcançar um bom desempenho. (HOLT, 1982 apud BORUCHOVITCH, 1999).

Diante deste contexto, contar com ferramentas e recursos auxiliares às metodologias de ensino é fundamentalmente necessário para ampliar e diversificar as estratégias no ensino de Física, de modo a tornar estas abordagens mais significativas e possibilitar também que sejam contempladas diferentes formas de aprendizado.

Cabe ainda destacar que os aspectos relacionados ao ensino de Física que foram mencionados, direcionados tanto aos professores quanto aos estudantes, apresentam uma estreita relação e permitem compreender algumas dinâmicas existentes no processo ensino-aprendizagem. O docente, ao utilizar recursos pedagógicos de maneira adequada e não-arbitrária, possibilita que o estudante obtenha maior consciência de seus processos cognitivos e, consequentemente, uma aprendizagem mais significativa com desempenhos satisfatórios. Portanto, as ferramentas dão suporte para o docente desenvolver práticas pedagógicas mais significativas, refletindo na compreensão e na consciência dos processos cognitivos dos estudantes.

Esta abordagem evidencia a importância de inserir novos contextos de ensino e

a imprescindibilidade de contemplar novas práticas utilizando ferramentas pedagógicas que auxiliem o processo de ensino e aprendizagem de conceitos de Mecânica Clássica. Dessa forma, esta prática oportuniza aos estudantes uma autoavaliação e investigação como forma de melhorar sua percepção sobre os conhecimentos prévios consolidados e suas limitações conceituais sobre os princípios discutidos, à medida que, quando aplicadas ferramentas didáticas adequadas, possibilita aos discentes desenvolverem competências de caráter científico e melhorem outros aspectos cognitivos.

Diante disso, o presente trabalho de pesquisa busca propor uma das possíveis soluções às questões levantadas, apresentando uma proposta de utilização de um produto educacional desenvolvido em linguagem de programação Python voltado especialmente para os tópicos de Lançamento de Projéteis e Colisões de Partículas.

2 OBJETIVOS

2.1 Objetivo geral

Este trabalho tem como objetivo evidenciar as principais justificativas para o uso de ferramentas adaptadas às tendências atuais de ensino que auxiliem as práticas pedagógicas no contexto de tópicos de Mecânica Clássica. Por conseguinte, desenvolver uma ferramenta que permitá explorar os resultados e facilitar a investigação de fenômenos físicos.

2.2 Objetivos específicos

- Apresentar um produto educacional adaptado às tendências atuais em ensino e aprendizagem desenvolvido em linguagem de programação Python.
- Ressaltar os recursos disponíveis e as vantagens do uso da ferramenta como alternativa para a diversificação das estratégias de ensino dos tópico de Lançamento de projéteis e Colisão Unidimensional, em contraste com as práticas pedagógicas tradicionais.

3 DIVERSIFICAÇÃO DAS ESTRATÉGIAS DE ENSINO

De acordo com Moreira (2021), pode-se dizer que a complementaridade que há entre a Física Teórica e a Física Experimental é essencial para o aprendizado significativo dos conceitos físicos. Neste contexto, fica claro que simulações e análises baseadas em evidências, quando abordadas de maneira adequada e não-arbitrária, permitem aos discentes desenvolverem competências importantes como investigação e pensamento de caráter científico para formular e validar suas hipóteses sobre os conceitos físicos discutidos. O mais preocupante, contudo, é constatar que abordagens dissociadas da preocupação de problematizar os conceitos físicos e estimular o teor crítico e pensamento científico dos alunos, ou seja, abordagens orientadas a uma aprendizagem mecânica predominantemente teórica e desvinculada de aplicações, limitam o potencial do aluno para relacionar os novos conteúdos às experiências já existentes em sua estrutura cognitiva.

3.1 Ferramentas adaptadas às tendências atuais de ensino

Ampliar possibilidades pedagógicas por meio do uso das tecnologias digitais de informação e comunicação (TDICs) é uma alternativa de diversificar as estratégias de ensino de Física. Portanto, a utilização de uma linguagem de programação é um recurso que pode auxiliar o docente a promover abordagens criativas e mais significativas em sala de aula.

A linguagem de programação Python é uma das possíveis alternativas capazes de adaptar mais facilmente situações reais em determinadas abordagens no ensino de Física utilizando a simulação, justificando o seu uso no desenvolvimento da ferramenta educacional apresentada neste trabalho de pesquisa. Nesse sentido, o produto educacional apresentado é encarado como uma alternativa relevante para a diversificação das estratégias em ensino de Mecânica Clássica e pode ser inserida no contexto de problematização das temáticas abordadas em sala de aula pelos docentes a partir de proposições didáticas que favoreçam a aprendizagem dos tópicos de lançamento de projéteis e colisões unidimensionais.

4 LINGUAGEM DE PROGRAMAÇÃO PYTHON

Criada por Guido van Rossum no início dos anos 1990, a linguagem de programação Python é uma linguagem interpretada e orientada a objetos. Considerada uma linguagem de alto nível, é um ambiente excelente para construir diversos tipos de aplicação de análise e para propósito geral (GONZÁLEZ DUQUE, 2014; MCKINNEY, 2019).

Os programas desenvolvidos em Python apresentam uma sintaxe simples que se assemelham a pseudocódigos; isto é, uma maneira genérica de desenvolver um código, sem a necessidade de dominar uma sintaxe específica. E esta é uma das vantagens para iniciantes, visto que começar nesta linguagem torna-se bastante simples.

4.1 Bibliotecas Python utilizadas no programa

Bibliotecas em Python são coleções de códigos e comandos, previamente escritos por outros programadores, que cumprem objetivos específicos e estão disponíveis para uso em outros *scripts*. As bibliotecas são bastante úteis pois facilitam o processo de programação suprimindo a necessidade de reescrever comandos usados com frequência.

As escolhas das bibliotecas Python usadas para o desenvolvimento do ambiente virtual foram baseadas em critérios como: praticidade do uso, quantidade de informações disponíveis para consulta (documentação) e quanto à presença nas distribuições Python para *Windows*. As bibliotecas Python utilizadas no programa são apresentadas a seguir.

4.1.1 Matplotlib

A matplotlib (`https://matplotlib.org/`) é a biblioteca Python mais amplamente utilizada para gerar visualizações gráficas e será utilizada para fornecer ao usuário um recurso de visualização gráfica dentro do ambiente virtual de simulações. Esta biblioteca permite personalizar os estilos de gráfico para visualização estática e dinâmica, e foi empregada para fornecer ao usuário visualização dos resultados da simulação.

4.1.2 NumPy

A biblioteca NumPy (`https://numpy.org/`), abreviatura para Numerical Python, é um pacote para computação numérica que oferece diversas funções matemáticas e bastante eficiente para manipular uma grande quantidade de informações. Esta biblioteca foi utilizada também pois oferece algumas rotinas básicas para manipulação de matrizes e é bastante eficiente para computação numérica. Na construção de gráficos, por exemplo, foi necessária para integrar com a biblioteca matplotlib

4.1.3 Math

Math é um módulo matemático e numérico presente na distribuição Python que oferece funções matemáticas, da quais foram utilizadas funções trigonométricas e métodos para conversão de ângulos, sobretudo para a implementação das equações cinemáticas.

4.1.4 Tkinter

É uma biblioteca que já acompanha a distribuição Python (versão *windows*) e permite criar interfaces gráficas de usuário com componentes gráficos de alto nível. A biblioteca Tkinter é versátil, fácil de usar e apresenta vasta documentação.

4.2 Etapas para o desenvolvimento do produto educacional

A primeira atividade desenvolvida para a realização deste trabalho de pesquisa foi uma revisão bibliográfica relacionada à diversificação das estratégias de ensino de Física e sobre aspectos relacionados à aprendizagem. Foram consultados artigos de revistas especializadas em ensino de Fisica e trabalhos do campo da psicologia educacional para avaliar as produções acadêmicas na mesma temática de diversificação das estratégias de ensino de Física.

A etapa seguinte consistiu em selecionar uma linguagem de programação adequada no que diz respeito ao propósito do trabalho e à facilidade de seu uso. Utilizou-se a linguagem de programação Python para possibilitar a realização de simulações

de fenômenos físicos direcionados a lançamento oblíquo e colisão unidimensional de partículas. Estas escolhas foram baseadas no contexto de diversificação das estratégias de ensino de Física de modo que propõe-se uma ferramenta para relacionar conceitos específicos de Mecânica Clássica, oferecendo à comunidade acadêmica um ambiente virtual para simulação.

Em seguida, foram selecionados os tópicos de Mecânica Clássica e os critérios de escolha apoiaram-se nos livros didáticos em consonância com a ementa do componente curricular de modo que, através das duas aplicações propostas, fosse possível abranger conteúdos relevantes e de unidades distintas.

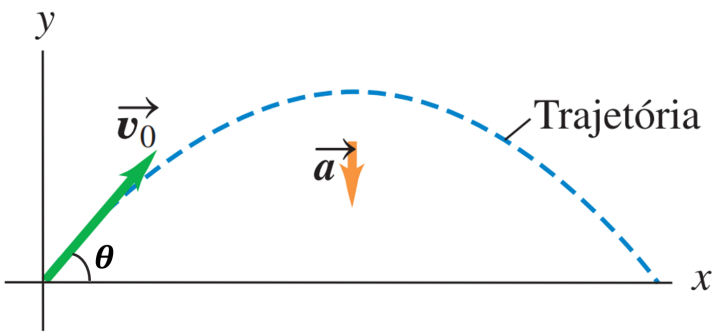
5 DISCUSSÃO DOS PRINCÍPIOS FÍSICOS

O objetivo desta seção é discutir alguns princípios físicos importantes relacionados às aplicações de Mecânica Clássica propostas para contextualizar e facilitar a compreensão da construção do programa em Python. Inicialmente, serão discutidos os princípios físicos e conceitos relacionados ao movimento de projéteis. Em seguida, serão apresentados alguns conceitos relacionados momento linear, impulso a colisões.

5.1 Movimento em duas dimensões

Considere uma partícula de massa m lançada com velocidade inicial de módulo $v_0 > 0$ de uma região próxima à superfície terrestre, ou seja, uma altura muito pequena se comparada ao raio da Terra ($6370\,\mathrm{km}$), como mostra a Figura 5.1. Dessa maneira, o campo gravitacional pode ser considerado uniforme e com valor de $\mathrm{g} = 9,81\,\mathrm{m\,s^{-2}}$.

Figura 5.1: Lançamento da partícula.



Fonte: FREEDMAN, Adaptado (2016, p 82.)

Desprezando os efeitos da resistência do ar e assumindo que, após o lançamento a partir de um ângulo $\theta \in (0; \frac{\pi}{2})$, a trajetória da partícula ocorre exclusivamente pela ação da força gravitacional, as equações de movimento do projétil são consequência imediata da aplicação da segunda lei de Newton, como segue

$$\vec{F} = m\vec{a} \tag{1}$$

Em coordenadas retangulares, temos:

$$F_x = ma_x \text{ e } F_y = ma_y$$

Como a partícula segue uma trajetória determinada exclusivamente pela ação da gravidade, então:

$$\frac{d^2x}{dt^2} = 0 \tag{2}$$

$$\frac{d^2y}{dt^2} = -\,\mathrm{g} \tag{3}$$

Considerando as condições iniciais dadas por:

$$(x_0, y_0) = (0, 0)\,, \qquad \vec{v}(0) = v_0 \cos(\theta) t\,\vec{i} + v_0 \sin(\theta)\,\vec{j} \tag{4}$$

As expressões (2) e (3) representam equações diferenciais ordinárias de segunda ordem e as soluções aplicando as condições iniciais definidas em (4), podem ser obtidas por integração e são dadas por:

$$x(t) = v_0 \cos(\theta) t \tag{5}$$

$$y(t) = v_0 \sin(\theta) - \frac{1}{2}\,\mathrm{g}t^2 \tag{6}$$

As equações (5) e (6) descrevem a posição do projétil em qualquer instante t. A partir dessas equações, podemos determinar a *equação da trajetória* em termos de x e y eliminando t e supondo que $(x_0, y_0) = (0, 0)$

$$y(x) = \tan(\theta) x - \frac{\mathrm{g}}{2v_0^2 \cos^2(\theta)} x^2 \tag{7}$$

Além disso, escrevemos

$$v_y = v_{0y} + a_y t \tag{8}$$

Ao atingir o ponto mais elevado da trajetória, a partícula assume velocidade nula na direção vertical. Utilizando a equação (8), determinamos o valor de t para a condição

em que $v_y = 0$. Portanto, escrevemos

$$0 = v_0 \sin(\theta) - \mathrm{g}t$$

$$t = \frac{v_0 \sin(\theta)}{\mathrm{g}}$$

Como a partícula percorre a mesma distância vertical em um mesmo intervalo de tempo, o tempo total necessário para que seja atingida a altura máxima (em $v_y = 0$) e a partícula retorne à posição inicial em relação ao eixo Oy pode ser expresso como segue

$$t = 2\frac{v_0 \sin(\theta)}{\mathrm{g}} \tag{9}$$

Substituindo (9) em (5), obtemos

$$A = x(t) = (v_0 \cos\theta)\frac{2v_0 \sin(\theta)}{\mathrm{g}}$$

$$A = \frac{v_0^2}{\mathrm{g}} 2 \sin(\theta)\cos(\theta)$$

Usando a identidade trigonométrica $2\sin(\theta)\cos(\theta) = \sin(2\theta)$, decorre que

$$A = \frac{v_0^2}{\mathrm{g}} \sin(2\theta) \tag{10}$$

A expressão (10) relaciona o *alcance horizontal* A com o ângulo inicial de lançamento θ e será utilizada no programa para fornecer ao usuário o alcance horizontal do lançamento.

5.2 Colisão unidimensional de partículas

Considere uma partícula de massa m constante. Substituindo a aceleração $\vec{a}$ pela derivada $\frac{dv}{dt}$, a segunda lei de Newton pode ser escrita na forma

$$\sum \vec{F} = m\frac{d\vec{v}}{dt} = \frac{d}{dt}(m\vec{v}) \tag{11}$$

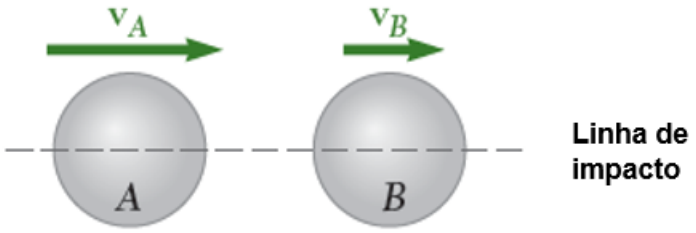
o vetor mv é chamado de quantidade de movimento linear. O *momento linear* (ou quantidade de movimento linear) de uma partícula é uma grandeza vetorial $\vec{P}$ definida pela equação

$$\vec{P} = m\vec{v} \tag{12}$$

em que m é a massa e $\vec{v}$ é a velocidade da partícula. Como m é uma grandeza escalar positiva, decorre da equação (12) que $\vec{P}$ e $\vec{v}$ possuem a mesma orientação.

Agora, considere duas partículas de massas m_A e m_B que se movem ao longo de um mesmo eixo horizontal com velocidades conhecidas $\vec{v}_A$ e $\vec{v}_B$, como mostrado na Figura 5.2.

Figura 5.2: Impacto central direto



Fonte: BEER, Adaptado (2012, p. 825).

Em uma colisão unidimensional, os movimentos das partículas antes e após a colisão ocorrem ao longo de uma única direção. Neste contexto, serão feitas as seguintes considerações:

- Os centros de massas das partículas estão localizados ao longo da linha de impacto
- As velocidades das partículas são orientadas ao longo da linha de impacto.

Considerando as duas partículas como um único sistema isolado e , portanto, na ausência de forças externas, o momento linear do sistema permanece constante. Logo, escrevemos

$$m_A\vec{v}_A + m_B\vec{v}_B = m_A\vec{v}'_A + m_B\vec{v}'_B$$

O símbolo (’) empregado nas equações representa uma condição pós-colisão, ou seja, a velocidade da partícula após a colisão. Dessa forma, não deve ser confundido com o operador diferencial.

Pela consideração que as velocidades das partículas estão orientadas ao longo da linha de impacto, a expressão pode ser reescrita pelas coordenadas escalares

$$m_A v_A + m_B v_B = m_A v'_A + m_B v'_B \tag{13}$$

O intuito é determinar as velocidades v'_A e v'_B após a colisão. De maneira mais rigorosa que as abordagens dos livros didáticos de Mecânica Clássica, ambas as partículas se deformarão dutante o impacto. É necessário estabelecer uma relação adicional para determinar as velocidades v'_A e v'_B.

Dessa forma, podemos associar um coeficiente de restituição expressando-o como sendo

$$e = \frac{|v'_B - v'_A|}{|v_A - v_B|} \tag{14}$$

onde:

- $|v'_B - v'_A|$ representa o módulo da velocidade relativa das duas partículas após a colisão;
- $|v_A - v_B|$ representa o módulo da velocidade relativa de aproximação das partícuas antes da colisão.

As equações (13) e (14) são utilizadas para determinar as velocidades finais das partículas.

6 PRODUTO EDUCACIONAL

O produto educacional que será apresentado na sequência é uma proposta de ferramenta didática complementar às estratégias no ensino de tópicos relativos à disciplina de Mecânica Clássica. Esta ferramenta é direcionada ao estudo dos assuntos de Lançamento de Projéteis e Colisões, destinando-se tanto aos docentes como aos estudantes deste componente curricular.

6.1 Interface gráfica desenvolvida com Tkinter

Desenvolvido usando a linguagem de programação Python, este ambiente emprega especialmente a biblioteca Tkinter para implementação de uma interface gráfica de usuário (GUI), fornecendo representações gráficas que melhoram a interação entre o usuário e o programa. A utilização da biblioteca Tkinter permite criar janelas customizáveis de fácil modificação e dispõe de métodos capazes de gerar desenhos com diversas geometrias, inserir imagens, adicionar textos instrucionais para o usuário e inúmeros recursos adicionais.

O principal objetivo da construção da interface gráfica é facilitar a realização de simulações baseadas nos princípios físicos apresentados na disciplina de Mecânica Clássica permitindo ao usuário definir e modificar os parâmetros iniciais de simulação. A interface gráfica inicial desenvolvida com a biblioteca Tkinter é apresentada na Figura 6.1.

A ferramenta pode ser utilizada em sala de aula como recurso auxiliar pelos docentes em conjunto com os alunos, adaptando as metodologias e possibilitando que sejam contempladas diferentes formas de aprendizado através de propostas didáticas alternativas sobre os temas. Através das simulações, os alunos podem também melhorar o rendimento e entendimento durante a resolução de exercícios propostos nos livros, validando suas hipóteses e respostas com os resultados obtidos nas simulações. Diferentemente de um gabarito no final da seção de um livro didático, no qual se limita a exprimir apenas o valor final correto, a ferramenta permite ao usuário uma visualização mais aprofundada do fenômeno físico e dos resultados.

Figura 6.1: Janela inicial do ambiente virtual.

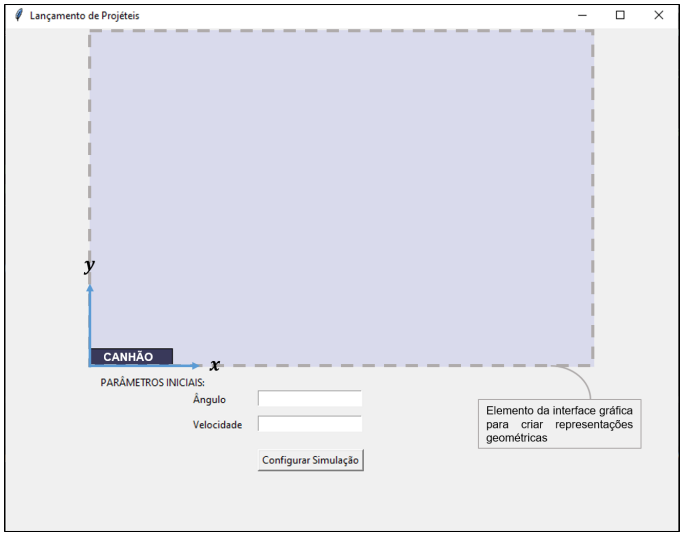
Fonte: Autoria própria (2021).

A estrutura da interface foi organizada para permitir que o usuário selecione as opções 'Colisão' ou 'Movimento de Projéteis', que correspondem a botões com atribuições específicas definidas no programa. Baseado nesta escolha, o usuário é direcionado ao ambiente de simulação correspondente ao botão selecionado.

A Figura 6.2 mostra a janela de simulação do canhão virtual que surge para o usuário ao ser selecionada a opção 'Movimento de Projéteis'. Dentro deste ambiente, ocorrem as simulações de lançamentos bidimensionais de partículas.

O canhão, representado por um retângulo, está posicionado no canto inferior esquerdo do elemento responsável por criar representações geométricas dentro da interface gráfica. A partícula será lançada a partir da origem do sistema de coordenadas adotado para o lançamento, o qual coincide com o centro de rotação do canhão. É importante mencionar que o canhão é meramente ilustrativo e não interfere diretamente no movimento da partícula.

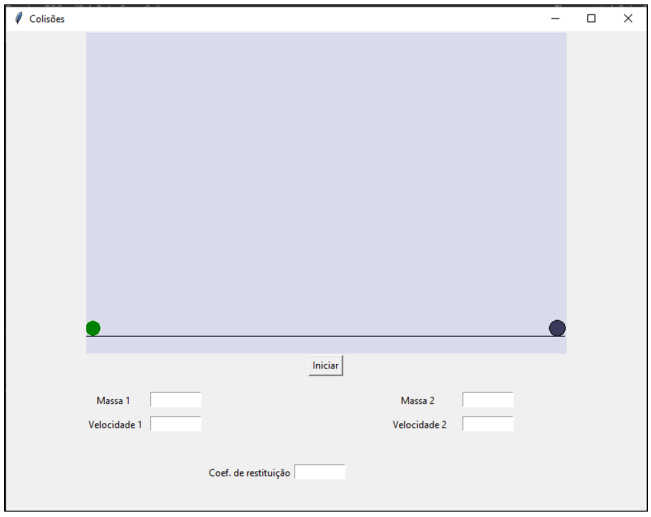
Figura 6.2: Janela de simulação do canhão virtual.



Fonte: Autoria própria (2021).

Na Figura 6.3 é mostrada a janela de simulação de colisões unidimensionais de partículas, quando a opção ’Colisão’ é selecionada.

Figura 6.3: Janela de simulação do ambiente de colisões.



Fonte: Autoria própria (2021).

Semelhante ao ambiente para simulação de lançamento de projéteis, este ambiente possui um elemento responsável por criar as figuras geométricas e os efeitos visuais de simulação, dentro do qual são representadas duas partículas e uma linha de referência que indicará a orientação do movimento.

6.2 Elementos da interface gráfica

A interface gráfica desenvolvida utilizando a biblioteca Tkinter é composta por diversos elementos gráficos individuais, denominados *widgets*. Os *widgets* se referem a qualquer elemento que pode interagir com o usuário em uma interface gráfica, como: janelas, botões, campos de entrada de dados, caixa de texto e imagens. São atribuídos a estes elementos funcionalidades específicas que permitirão melhorar a interação entre o usuário e o programa.

Em Python, todos os elementos representam objetos e são instanciados a partir da classe *Tk( )*. Esta classe presente na biblioteca Tkinter fornece diversos recursos configuráveis para montar uma interface gráfica e os principais elementos que compõem a interface do ambiente virtual desenvolvido neste trabalho são apresentados a seguir.

6.2.1 Label

Label é um *widget* usado para inserir imagens e textos dentro da interface gráfica importantes para a navegação do usuário. Este elemento foi utilizado para inserir textos instrucionais e informações dentro do ambiente virtual para facilitar o entendimento do usuário e transmitir alguns resultados da simulação.

6.2.2 Entry

Todos os parâmetros iniciais de simulação informados pelo usuário são recolhidos por meio de um elemento da interface gráfica denominado Entry, a partir do qual são trabalhados dentro do programa. Este *widget* é um campo de entrada de dados e é sempre utilizado em conjunto com elementos do tipo Label.

6.2.3 Canvas

O widget Canvas é um elemento do Tkinter que permite criar elementos geométricos e modificá-los de acordo com a necessidade da aplicação, gerando os efeitos visuais desejados. Esta ferramenta é muito vantajosa no desenvolvimento de simulações e pequenas animações. Dentro deste elemento, serão desenhadas representações geométricas

para os elementos da simulação (retângulos, círculos e linhas), apagados e movidos para promover uma animação em cada aplicação abordada.

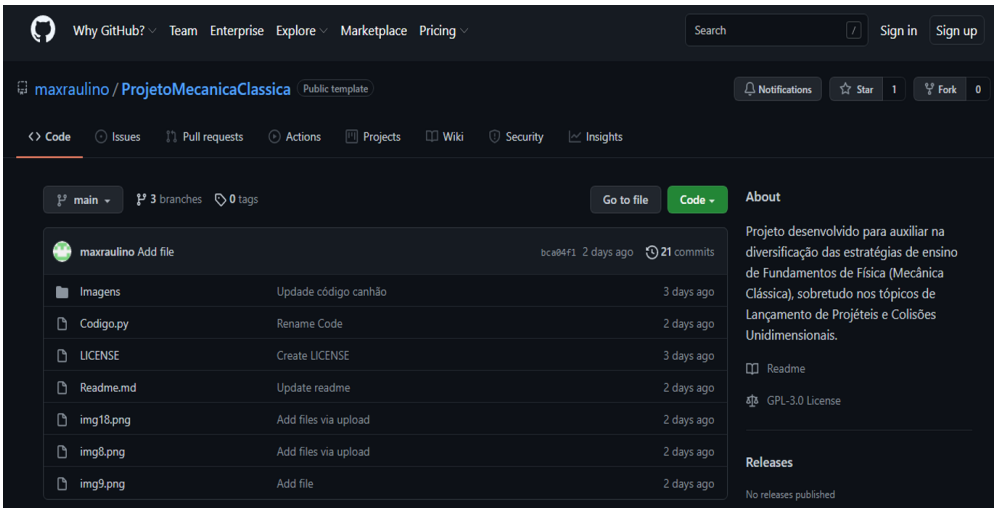
6.2.4 Button

O *widget* Button é um elemento que estará ligado uma instrução particulares a partir de funções e métodos dentro do programa para executar rotinas específicas. Este elemento foi utilizado dentro da interface para para criar botões de seleção para permitir ao usuário escolher a opção de interesse e controlar as simulações.

6.3 Armazenamento do projeto na plataforma Github

Através de uma conta na plataforma Github, foi criado um repositório do projeto para facilitar manutenção, implementação de melhorias, gerenciamento de mudanças no código e também para permitir que outros desenvolvedores contribuam otimizando e criando novos recursos em colaboração neste ambiente.

Figura 6.4: Repositório no Github.



Fonte: Autoria própria (2021).

Através do site, é possível consultar documentação e extrair informações sobre o projeto. Além da descrição do projeto, estão presentes também todos os arquivos desenvolvidos que podem ser acessados através do seguinte link: `https://github.com/maxraulino/ProjetoMecanicaClassica`.

A hospedagem de um repositório (privado ou público) nesta plataforma e a contribuição em outros repositórios podem ser realizadas por qualquer usuário que possua cadastro na plataforma, que concede criação ilimitada de repositórios públicos gratuitamente. Ao acessar o link do repositório no Github, o usuário é direcionado para a página com todos os arquivos do projeto exibida na Figura 6.4.

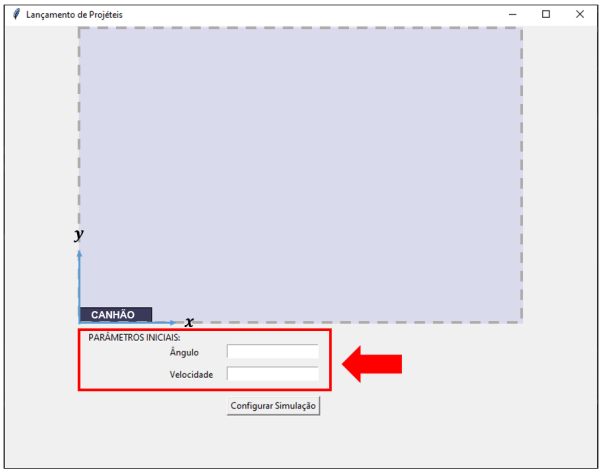
6.4 Procedimento para tornar o produto acessível

Com o objetivo de facilitar a difusão, divulgação e utilização da ferramenta, adotou-se um procedimento para transformar o código python (formato .py) em um arquivo executável (formato .exe) para possibilitar os alunos e docentes utilizarem esta ferramenta sem necessariamente ter o programa Python instalado em sua máquina e sem a exigência de entender a linguagem. O propósito disso é fornecer uma ferramenta pronta para uso e de fácil utilização para otimizar este processo. Caso contrário, seria exigido do usuário um conhecimento básico prévio sobre a linguagem de programação Python e a instalação do programa. Dessa forma, o programa se torna mais acessível para alunos e professores. O arquivo no formato (.exe) também foi armazenado juntos com os outros arquivos do projeto no repositório do Github.

7 CANHÃO VIRTUAL

Para configurar a simulação, o usuário deverá inserir os valores iniciais do lançamento nos campos de entrada de dados disponíveis na tela, como mostra a Figura 7.1.

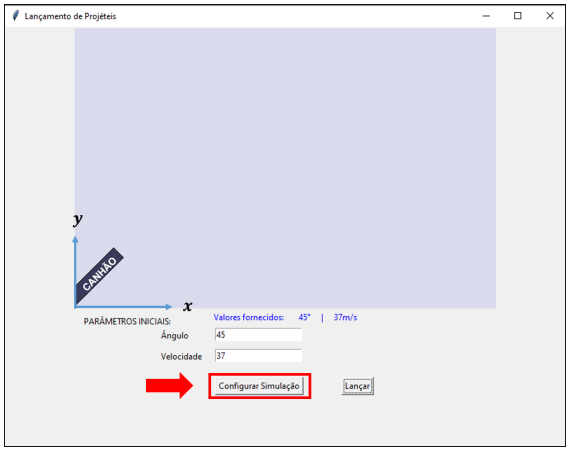
Figura 7.1: Campos de entrada dos parâmetros iniciais de lançamento.



Fonte: Autoria própria (2021).

Cada parâmetro de lançamento inserido pelo usuário é inspecionado dentro do programa através de um controle de fluxo para verificar e validar os valores fornecidos. Ao selecionar o botão 'Configurar Simulação', o canhão é articulado com base no valor de ângulo fornecido, conforme mostra a Figura 7.2.

Figura 7.2: Opção Configurar Simulação selecionada no ambiente de simulação.

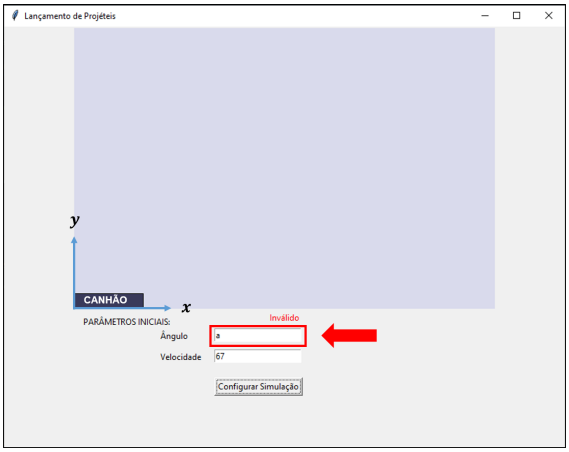


Fonte: Autoria própria (2021).

Caso os parâmetros sejam digitados incorretamente ou apresentem incoerências,

é exibida uma mensagem de erro ao usuário. Ao digitar uma letra no campo destinado ao ângulo de lançamento, por exemplo, o programa reconhece a inconsistência no valor fornecido e exibe uma mensagem de texto para o usuário, permitindo a correção do parâmetro. Na Figura 7.3 é apresentada uma situação em que um dos valores digitados não é válido.

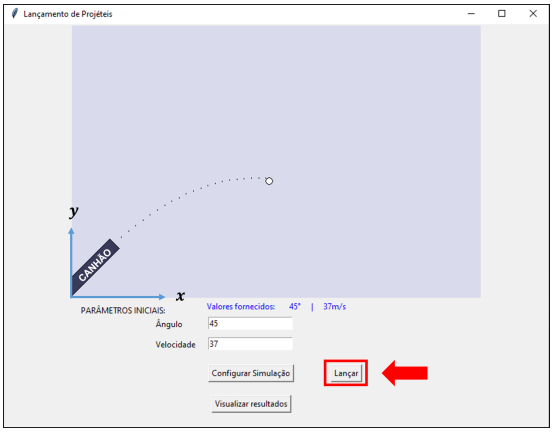
Figura 7.3: Parâmetro inicial de simulação inválido.



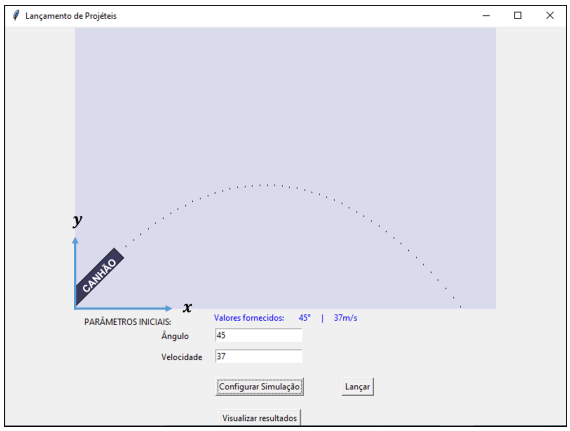
Fonte: Autoria própria (2021).

Após inserir parâmetros válidos, aparece na tela o botão 'Lançar' que ao ser clicado, efetua o lançamento da partícula na configuração fornecida pelo usuário, como mostra a Figura 7.4.

Figura 7.4: Opção Lançar selecionada no ambiente de simulação.



(a) Janela de simulação após o lançamento.

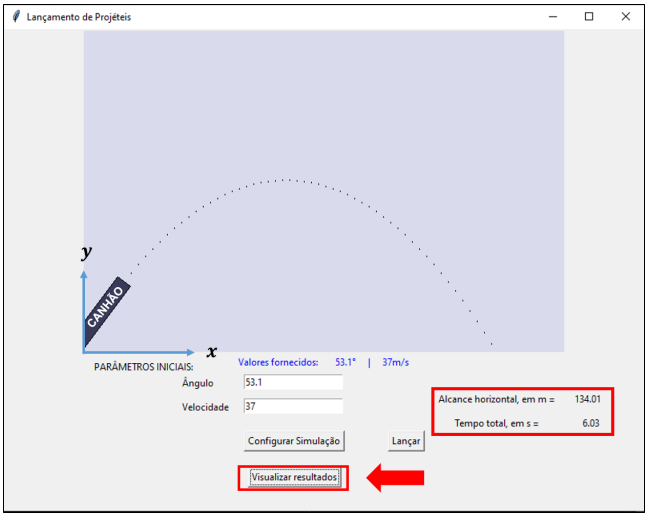


(b) Janela de simulação ao final do lançamento.

Fonte: Autoria própria (2021).

Ao final da simulação, é possível obter os valores do alcance horizontal e do tempo total da trajetória da partícula selecionando o botão 'Visualizar resultados'. A Figura 7.5 mostra um exemplo dos resultados obtidos para um ângulo de lançamento de $53.1°$ e velocidade de 37 m/s.

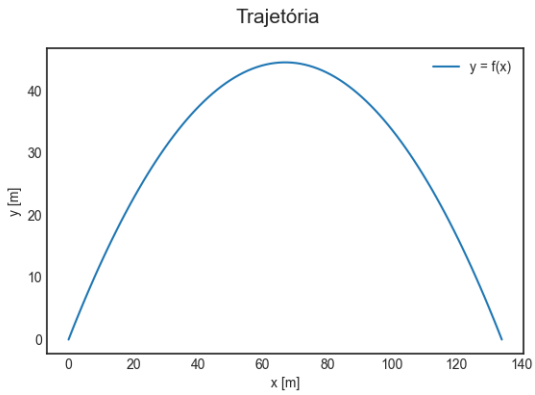
Figura 7.5: Visualização dos resultados da simulação.



Fonte: Autoria própria (2021).

Para os parâmetros em questão, são retornados os valores de $134,01$m para o alcance horizontal e $6,03$ s para o tempo total da trajetória da partícula. Além dos valores de alcance horizontal e tempo total da trajetória, surge uma janela ao lado do ambiente de simulação com um gráfico 2D no qual o usuário pode visualizar a equação da trajetória e visualizar a altura máxima que a partícula atingiu, como apresenta a Figura 7.6.

Figura 7.6: Recurso de visualização gráfica após lançamento.

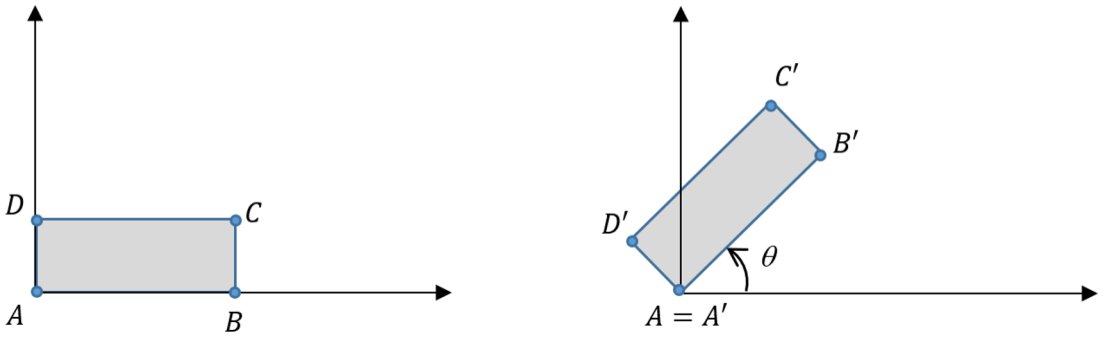


Fonte: Autoria própria (2021).

7.1 Articulação do canhão

Inicialmente, considere uma figura geométrica plana de formato retangular para representar o canhão virtual no plano (Figura 7.7a) desenvolvido em Python com o objetivo de simular movimentos bidimensionais de partículas. Fixado um sistema de coordenadas, o elemento representativo para o canhão pode ser definido pelas coordenadas de seus vértices A, B, C e D.

Figura 7.7: Elemento representativo para o canhão.



(a) Posição inicial do elemento.

(b) Rotação do elemento em torno do ponto A.

Fonte: Autoria própria (2021).

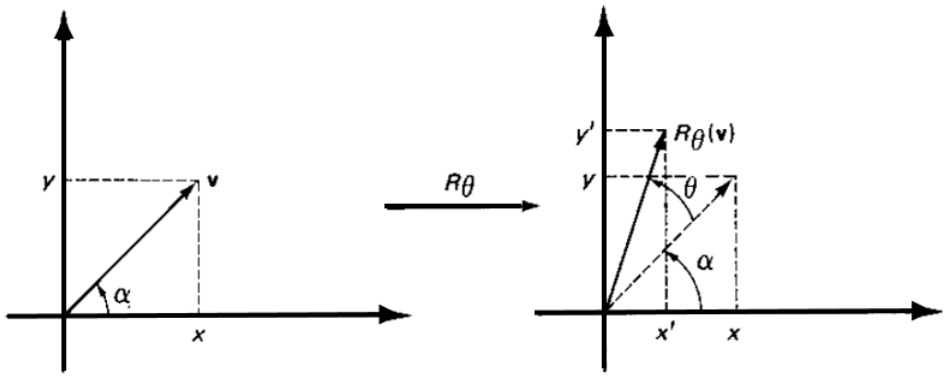
Suponha que deseja-se girar de um ângulo θ o elemento representativo, no sentido anti-horário (Figura 7.7b). Por convenção, considere que a rotação ocorra em torno do ponto A coincidente com a origem do sistema de coordenadas. O problema consiste em determinar as novas coordenadas B', C' e D' dos vértices do retângulo após a rotação em torno do ponto especificado.

A descrição matemática da articulação do canhão, representada pelo movimento de rotação do retângulo, suscita uma aplicação de Álgebra Linear, mais precisamente uma transformação linear no plano. É conveniente, portanto, revisar alguns tópicos importantes envolvendo as ferramentas matemáticas exigidas para o desenvolvimento do trecho de código associado ao canhão virtual.

7.1.1 Rotação no Plano e Operador Matricial de Rotação de $\mathbb{R}^2$

Seja $v = (x, y)$ um vetor em $\mathbb{R}^2$ de norma $\|\vec{v}\| = r$, como mostra a Figura 7.8. Queremos encontrar a imagem de v pela rotação de um ângulo θ.

Figura 7.8: Rotação de um vetor no plano.



Fonte: BOLDRINI (1984, p. 149).

Para o vetor **v**, podemos escrever as seguintes relações

$$x = r\cos(\alpha) \qquad \text{e} \qquad y = r\sin(\alpha)$$

E para **v'**, temos:

$$x' = r\cos(\alpha + \theta) \qquad \text{e} \qquad y' = r\sin(\alpha + \theta)$$

Relembrando as seguintes propriedades trigonométricas, escrevemos:

$$\cos(\alpha + \theta) = \cos(\alpha)\cos(\theta) - \sin(\alpha)\sin(\theta)$$

$$\sin(\alpha + \theta) = \sin(\alpha)\cos(\theta) + \sin(\theta)\cos(\alpha)$$

Portanto,

$$x' = r\cos(\alpha + \theta) = r(\cos\alpha\cos\theta - \sin\alpha\sin\theta)$$

$$= (r\cos\alpha)\cos\theta - (r\sin\alpha)\sin\theta$$

$$x' = x\cos\theta - y\sin\theta \tag{15}$$

De maneira análoga,

$$y' = r\sin(\alpha+\theta) = r(\sin\alpha\cos\theta + \sin\theta\cos\alpha)$$

$$= (r\sin\alpha)\cos\theta + (r\cos\alpha)\sin\theta$$

$$y' = x\sin\theta + y\cos\theta \tag{16}$$

As relações (15) e (16) são denominadas *equações de rotação* em $\mathbb{R}^2$, de forma que o *operador* $R_\theta : \mathbb{R}^2 \to \mathbb{R}^2$ é definido por:

$$R_\theta(x,y) = (x',y') = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$$

Assim, temos uma transformação $R_\theta : \mathbb{R}^2 \to \mathbb{R}^2$ que leva um vetor $v = (x,y)$ do plano em outro vetor de coordenadas $v' = (x',y')$ de mesmo módulo, pela rotação de um ângulo θ.

A articulação do canhão dentro do código Python consiste em redesenhar um retângulo dentro do Canvas utilizando as novas coordenadas de seus vértices obtidas pela rotação do ângulo θ fornecido pelo usuário como parâmetro inicial de lançamento. A configuração inicial do canhão representa a posição em que $\theta = 0$.

Para realizar a rotação do canhão, foi utlizado o método *matriz_rotacao*. Este método é responsável por retornar as novas coordenadas *self.xp* e *self.yp* dos vértices do canhão e é aplicado a cada vértice separadamente. A sintaxe do método é apresentada a seguir:

```
def matriz_rotacao(self, x, y):
    self.xp = (x * math.cos(self.var)) - (y * math.sin(self.var))
    self.yp = (y * math.cos(self.var)) + (x * math.sin(self.var))
    return (self.xp, self.yp)
```

O método *matriz_ rotacao* relaciona-se com os aspectos matemáticos discutidos e é uma aplicação em linguagem Python das equações (15) e (16). Os termos *math.sin* e *math.cos* presentes no trecho de código são as funções trigonométricas seno e cosseno e estas funções são oferecidas e acessadas pelo módulo *math*.

Define-se um objeto do tipo Canvas com dimensões 600 x 400, em pixels e dentro dele é desenhado um retângulo no seu canto inferior esquerdo. As posições dos vértices nesta configuração são tomadas como padrão e sempre que o código é executado, o retângulo estará nessa posição. Após o usuário definir o ângulo, o retângulo representativo é redesenhado de acordo com o valor fornecido.

O ângulo de lançamento é determinado previamente pelo usuário e a partir desta escolha, o canhão é ajustado para realizar o lançamento do projétil. São criados dentro do Canvas (elemento da biblioteca Tkinter para representação de geometrias) um retângulo e um círculo para representar, respectivamente, o canhão e a partícula.

O referido método exige dois argumentos iniciais (atributos), os quais correspondem às coordenadas x e y do retângulo em sua posição original ($\theta = 0$) e são definidas inicialmente dentro do programa. Após as operações matemáticas, o método retorna os pares ordenados dos vértices do canhão após a rotação de um ângulo.

O ângulo de rotação, denotado por *self.var*, é definido inicialmente pelo usuário, junto com a velocidade inicial do lançamento através de um *Entry* presente na interface que recolherá o valor fornecido e passará para o método *matriz_ rotacao*.

7.2 Apresentação dos resultados na tela

Os resultados de tempo total de trajetória e de alcance horizontal apresentados ao final de lançamento são estruturados no Python e representam as expressões (9) e (10). O trecho de código que relaciona estes parâmetros é mostrado na sequência.

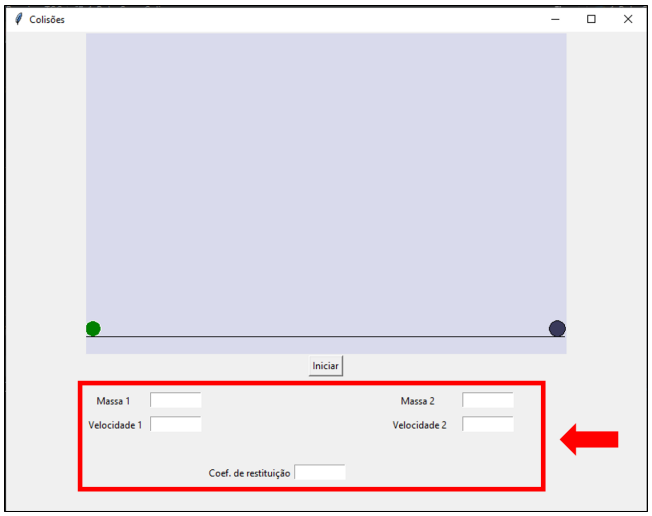
```
#Alcance horizontal
self.R = (math.sin(2*self.angle)*(self.speed)**2)/g
self.R = round(self.R, 2)
#Tempo total
self.t = 2 * self.speed * math.sin(self.angle)/g
self.t = round(self.t, 2)
```

As variáveis *self.R* e *self.t* são exibidas dentro de um elemento *Label* após o botão 'Visualizar resultados' ser selecionado.

8 SIMULADOR DE COLISÕES UNIDIMENSIONAIS

Para realizar a simulação no ambiente de colisões, o usuário deverá inserir os parâmetros iniciais de acordo com o Sistema Internacional de Unidades (SI) nos campos indicados na Figura 8.1. São exigidos ao usuário valores para as massas das partículas, velocidades iniciais e coeficiente de restituição.

Figura 8.1: Campos de entrada de parâmetros no ambiente de colisões.



Fonte: Autoria própria (2021).

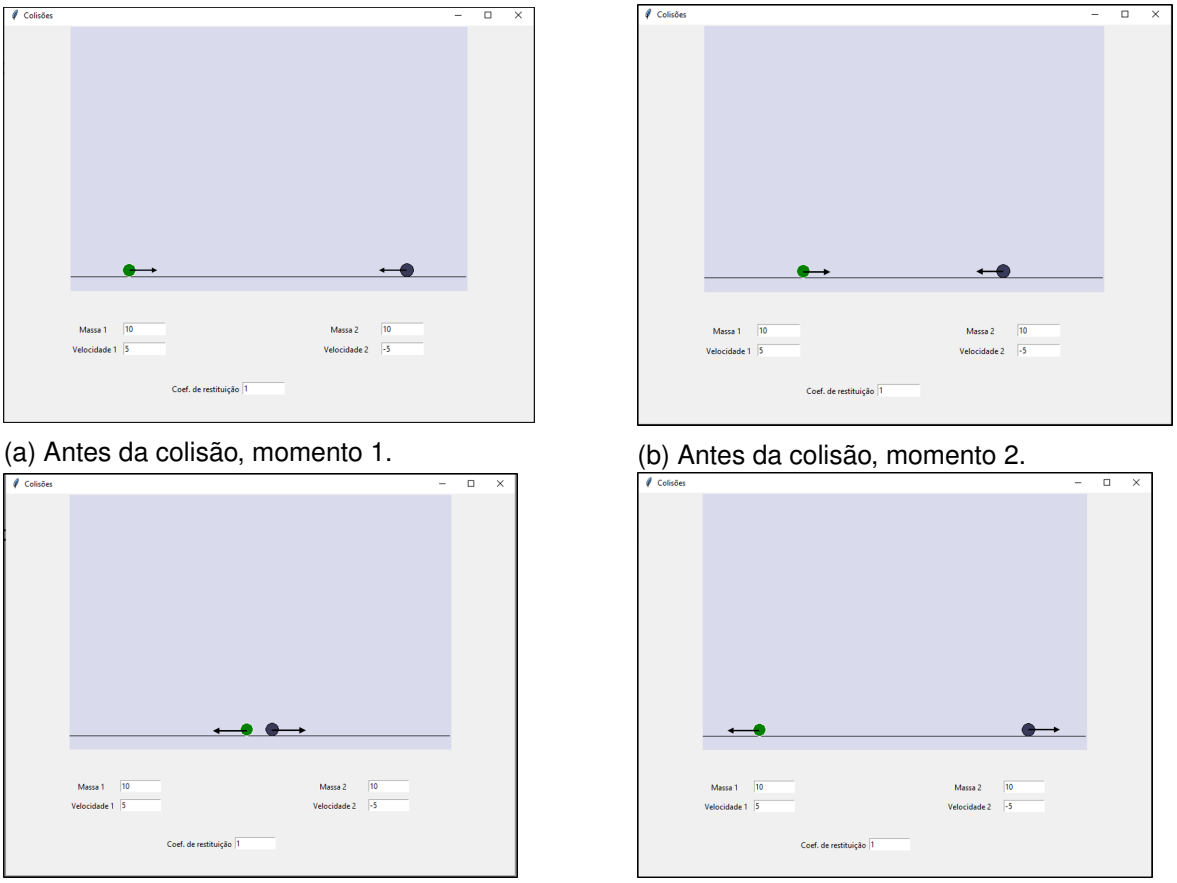
Após inserir todos os valores válidos e selecionar a opção ‘Iniciar’ (Figura 8.2), é dado início à simulação da colisão unidimensional.

Figura 8.2: Botão Lançar no ambiente de colisões.

Fonte: Autoria própria (2021).

A Figura 8.3 apresenta uma demonstração utilizando valores para exemplificar a navegação no simulador. São atribuídos valores semelhantes para as massas das duas partículas ($m_1 = m_2 = 10\,\mathrm{kg}$) e também para as velocidades iniciais ($|\vec{v}_1| = |\vec{v}_2| = 5\,\mathrm{m/s}$), com coeficiente de restituição $e = 1$ (Colisão perfeitamente elástica).

Figura 8.3: Simulação de uma colisão elástica entre duas partículas.



(a) Antes da colisão, momento 1.

(b) Antes da colisão, momento 2.

(c) Depois da colisão, momento 1.

(d) Depois da colisão, momento 2.

Fonte: Autoria própria (2021).

Apesar das velocidades das duas partículas possuírem mesmo módulo, o sinal negativo que acompanha a velocidade da partícula 2 indica que ambas as partículas respeitam a orientação de um eixo comum e que, neste caso, é positivo para a direita. Logo, para uma velocidade $v > 0$ fornecido a partícula se deslocará para a direita. Caso contrário, em que $v < 0$, o deslocamento ocorrerá para a esquerda.

Inicialmente, são importadas as bibliotecas que oferecem os recursos necessárias

para o devido funcionamento do programa. A maior parte das bibliotecas utilizadas possui documentação vasta e de fácil acesso na internet a partir dos sites oficiais. O trecho de código usado para importar os módulos e bibliotecas necessário é apresentado a seguir:

```
# Importando as Bibliotecas necessárias
from tkinter import *
import matplotlib.pyplot as plt
import math
import numpy as np
```

A importação dos pacotes acontece por meio do comando *import* seguido do nome da biblioteca. Os trechos precedidos pelo símbolo (#) representam comentários e não fazem parte diretamente do código. Ao utilizar o símbolo (#), todo o restante da linha a partir de sua utilização será entendido pelo interpretador Python como um comentário e durante a execução, não será interpretado.

Na sequência, são definidos parâmetros importantes dentro do programa para definir dimensões e posições dos elementos da interface gráfica, como mostra a sintaxe a seguir:

```
# Definindo variáveis para tamanho do canvas e janela principal:
larg_canvas = 600
alt_canvas = 400
# Definindo variáveis para os círculos:
# Bola I
raio = 20
p1 = (0, alt_canvas - (2 * raio))
pos_x1, pos_y1 = p1
velocidade1 = (5, 0)
# Bola II
p2 = ((larg_canvas - raio), alt_canvas - (2 * raio))
pos_x2, pos_y2 = p2
velocidade2 = (5, 0)
```

São definidos neste trecho de código variáveis para configurar as dimensões do canvas e as posições das partículas dentro deste elemento. O elemento canvas é criado com 600 pixesl de altura e 400 pixels de largura e dentro deste ambiente são desenhadas as duas partículas em forma de círculos na configuração inicial de simulação, ou seja, cada partícula localizada em uma extremidade do canvas.

Os círculos foram criados dentro do canvas utilizando o método *create_ oval* e as posições iniciais das partículas são definidas no início do código, junto com os outros parâmetros importantes. Na sequência, é mostrado um exemplo da sintaxe do método *create_ oval* utilizada no programa para a partícula 1 (à esquerda no canvas).

```
self.bola1 = self.canvas.create_oval(self.pos_x1,
                                      self.pos_y1,
                                      self.pos_x1 + self.raio,
                                      self.pos_y1 + self.raio,
                                      fill='green',
                                      outline='white',
                                      tag='bola1'
                                      )
```

Para fazer a animação das partículas, os círculos são sequencialmente excluídos e redesenhados em novas posições de acordo com os parâmetros fornecidos inicialmente pelo usuário. Utilizando o mesmo método *create oval*, mas com valores diferentes para as posições, os valores posicionais são reiteradamente atualizados a cada 10 ms (milissegundos), gerando o efeito visual de colisão.

9 CONCLUSÃO

A partir deste trabalho, buscou-se evidenciar as justificativas pelas quais a utilização de ferramentas didáticas vinculadas às tecnologias são importantes alternativas que devem ser implementadas para melhoria das práticas pedagógicas na disciplina de Mecânica Clássica. Diante da discussão proposta, fica evidente a importância da utilização de ferramentas alternativas como recurso complementar nas práticas didáticas.

De face ao que foi discutido, foi apresentado um ambiente virtual que conciliam a linguagem de programação Python com duas propostas para o auxílio no estudo: Um Canhão virtual para lançamento de projéteis para o estudos de movimentos bidimensionais e Um Simulador de colisões unidimensionais para estudo de conceitos relacionados a Princípio de impulso e quantidade de movimento.

Os objetivos propostos neste trabalho foram alcançados tendo em vista que desenvolveu-se uma implementação em Python para aplicações na disciplina de Mecânica Clássica. Evidenciar a importância da utilização de novas ferramentas e consequentemente buscar a diversificação das estratégias pedagógicas e implementar exemplos práticos que possam ser motivadores para novas ferramentas.

Mostrou-se que a linguagem de programação Python é uma poderosa ferramenta e a justificativa para o seu uso neste trabalho se baseia na implementação de recursos de visualização detalhada dos resultados com auxílio de gráficos. A linguagem dispõe de diversas bibliotecas e especialmente uma biblioteca que foi utilizada para visualizações gráficas (matplotlib), permitindo explorar os resultados e facilitar a investigação de fenômenos físicos.

Diante disso, espera-se que o referido trabalho contribua para a diversificação das estratégias em ensino de Mecânica Clássica e funcione como um meio motivador para induzir futuras pesquisas e a elaboração de novas ideias para que outros trabalhos no mesmo sentido sejam desenvolvidos e que a diversificação das estratégias de ensino de Física tornem-se uma realidade.

REFERÊNCIAS

BEER, Johnston. Cornwell, **Mecânica Vetorial para Engenheiros: Dinâmica**, 9ª Edição, 2012. Editora Bookman.

BOLDRINI, José Luiz et al. **Álgebra Linear** 3º Edição, 1984. São Paulo: Editora Harbra – Harper & Row do Brasil.

BORUCHOVITCH, Evely. Estratégias de aprendizagem e desempenho escolar: considerações para a prática educacional. **Psicologia: reflexão e crítica**, v. 12, n. 2, p. 361-376, 1999.

FREEDMAN, Roger A.; YOUNG, Hugh D. **Física I: mecânica**. 14. ed. São Paulo: Pearson Education do Brasil, 2008. v. 1.

FREIRE, Wilson Hugo C. et al. Lançamento oblíquo com resistência do ar: Uma análise qualitativa. **Revista Brasileira de Ensino de Física**, v. 38, 2016.

GONZÁLEZ DUQUE, Raúl. Python para todos. 2014.

MCKINNEY, Wes. Python para análise de dados: **Tratamento de dados com Pandas, NumPy e IPython**. Novatec Editora, 2019.

MEIRA, Damiao Pedro; KAMASSURY, Jorge Kysnney Santos; MEIRA, Rose Caldas de Souza. Uma discussão sobre o coeficiente de restituição. **Revista Brasileira de Ensino de Física**, v. 39, 2017.

MOREIRA, Marco Antonio. Desafios no ensino da física. **Revista Brasileira de Ensino de Física**, v. 43, 2021.

RESNICK, Robert; WALKER, Jearl; HALLIDAY, David. **Fundamentos de física: mecânica**. 9. ed. Rio de Janeiro: LTC, 2012. v. 1.