UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE FÍSICA

CURSO DE GRADUAÇÃO EM FÍSICA

VÍCTOR CITÓ RODRIGUES

O USO DA PROGRAMAÇÃO COMO FERRAMENTA PARA O ENSINO DE FÍSICA: UMA PROPOSTA METODOLÓGICA

FORTALEZA

2024

VÍCTOR CITÓ RODRIGUES

O USO DA PROGRAMAÇÃO COMO FERRAMENTA PARA O ENSINO DE FÍSICA: UMA PROPOSTA METODOLÓGICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Física do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de licenciado em Física.

Orientador: Prof. Dr. José Alves de Lima Junior.

FORTALEZA

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R617u Rodrigues, Victor Citó.
O uso da programação como ferramenta para o ensino de Física: uma proposta metodológica / Victor Citó Rodrigues. – 2024.
72 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Ciências, Curso de Física, Fortaleza, 2024.
Orientação: Prof. Dr. José Alves de Lima Júnior.

1. cinemática. 2. lançamentos. 3. ensino de Computação. 4. simulador. 5. Python. I. Título.

CDD 530

VÍCTOR CITÓ RODRIGUES

O USO DA PROGRAMAÇÃO COMO FERRAMENTA PARA O ENSINO DE FÍSICA: UMA PROPOSTA METODOLÓGICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Física do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de licenciado em Física.

Aprovada em: 26/09/2024.

BANCA EXAMINADORA

Prof. Dr. José Alves de Lima Junior (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. José Gilvan Rodrigues Maia
Universidade Federal do Ceará (UFC)

Prof. Dr. Fernando Wellysson de Alencar Sobreira
Instituto Federal do Ceará (IFCE)

Aos estudantes e professores que possam fazer bom uso desse trabalho que se faz possível com ajuda de outros estudantes e professores.

AGRADECIMENTOS

Ao Prof. Dr. José Alves de Lima Júnior, pela excelente orientação durante os estágios e o trabalho, personifico-o no agradecimento à Universidade Federal do Ceará pelas horas de estudo e pela estrutura fornecida.

Ao camarada Auro Gabriel, por me apresentar à Computação durante a escola, e ao Prof. Dr. Cesar Lincoln Cavalcante Mattos, pela reapresentação da Computação no ambiente universitário.

Aos meus amigos do grupo Trivial, pelo apoio em todas as dúvidas, questões que me explicaram, trabalhos em grupo, e por toda a experiência compartilhada sobre a sala de aula.

À minha então namorada Flora Dimitria, por tantos momentos alegres, por me incentivar na conclusão dessa etapa de minha vida e por me inspirar a seguir meus anseios atuais, sem me prender a títulos previamente adquiridos.

Aos meus irmãos, Arthur Citó, o Tuca, e Caio Citó, o recém-nomeado tio Caco, por serem fontes inesgotáveis de inspiração acadêmica, profissional e pessoal. Obrigado por me fazerem o caçula dessa família.

Aos meus pais, Henrique Rodrigues, por sempre me incentivar nos estudos e me enviar as principais novidades e informações sobre temas importantes e interessantes, e Hebe Citó, por também sempre me incentivar nos estudos, até mesmo com cobranças tão importantes para a conclusão do presente trabalho, e por todo o cuidado e investimento de ambos.

Aos avós, Cícero Citó e Ângela Citó, por todas as bênçãos recebidas e por me trazerem ao mundo como parte da Citolândia, uma família maravilhosa com tios e primos incríveis que também merecem agradecimentos.

Aos incontáveis professores que me deram a oportunidade do conhecimento, em principal ao professor Idelfrânio Moreira, que, durante a conclusão do meu Ensino Médio, me fez escolher pela docência.

Para ser franco, acho que o problema é que vocês jamais souberam qual é a pergunta. [...] Assim que vocês souberem qual é exatamente a pergunta, vocês saberão o que significa a resposta. (Adams, 2021, p. 264-265)

RESUMO

A Computação tem ganhado cada vez mais espaço nas mais diferentes áreas do conhecimento, dentro da ciência, um novo ramo surge: a Ciência Computacional. Para formar estudantes capazes de compreender e elaborar simulações úteis para o avanço científico, é necessário trabalhar o letramento digital desde a educação básica. O presente trabalho propõe, portanto, uma sequência didática que visa ensinar programação a alunos do Ensino Médio enquanto eles aprendem uma nova área da física: o lançamento de projéteis. Com o uso de fundamentos da Computação Científica, será trabalhado com os estudantes o desenvolvimento de um simulador para aproximar o conteúdo aprendido da realidade ao introduzir a força de resistência do ar. Dessa forma, o ganho de conhecimento na área da Computação se dá de forma aplicada, podendo trazer melhores resultados tanto para a aprendizagem de Física quanto de Computação. A fundamentação teórica desta monografia abrange o desenvolvimento dos conteúdos necessários à aplicação dessa proposta em simbologia do Ensino Superior, bem como sua simplificação para o Ensino Médio, quando possível.

Palavras-chave: cinemática; lançamentos; ensino de Computação; simulador; Python.

ABSTRACT

Computer Science has increasingly gained prominence across various knowledge sites, in Science a new branch rose: the Computational Science. To train students to become able to comprehend and produce useful simulations to the scientific growth, it is necessary to develop the digital knowledge since basic education. This work, thus, proposes a didactic sequence aimed at teaching programming skills to High School students while simultaneously they learn a new field of Physics: the projectile motion. Using concepts of Computational Science, will be developed with the students a simulator to approximate the learnt subject with the reality by introducing the air drag force. This way, learning occurs in an applied manner, which can provide better results in both Computer Science and Physics. The theoretical foundation of this monograph contains the development of all necessary content for implementing this proposal based on a higher education symbolism, as well as its simplification to High School level, where possible.

Keywords: kinematics; projectile motion; Computer Science teaching; simulator; Python.

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

BNCC	Base Nacional Comum Curricular
EDO	Equações Diferenciais Ordinárias
EM	Ensino Médio
MU	Movimento Uniforme
MUV	Movimento Uniformemente Variado
PC	Pensamento Computacional
PDF	*Portable Document Format*
POO	Programação Orientada a Objetos
PSSC	*Physical Science Study Committee*
SI	Sistema Internacional de Unidades

LISTA DE SÍMBOLOS

t	Instante de tempo
V_0	Velocidade inicial
$x(t)$	Posição ao longo de x em função do tempo
$\dot{x}(t)$	Velocidade ao longo de x em função do tempo
$\ddot{x}(t)$	Aceleração ao longo de x em função do tempo
g	Aceleração gravitacional da Terra
F_{Ra}	Força de resistência do ar
S_0	Posição inicial
$S(t)$	Posição em função do tempo
V	Velocidade
$S[n]$	Posição no enésimo intervalo de tempo
ΔT	Intervalo de tempo
V_m	Velocidade média
a	Aceleração
$V[n]$	Velocidade no enésimo intervalo de tempo

SUMÁRIO

1 INTRODUÇÃO

Aprender a elaborar algoritmos é um processo valioso para o desenvolvimento do raciocínio lógico. Aprender uma linguagem de programação pode trazer vantagens como: melhor estruturação do pensamento, melhor desempenho em áreas associadas à Matemática ou Língua Inglesa e melhor capacidade para solucionar problemas (Silveira; Knirsch, 2016).

O desenvolvimento de uma habilidade em programação não somente é importante para o raciocínio lógico matemático, mas é essencial para o letramento digital dos jovens. Afinal, o uso constante de uma tecnologia não implica no seu total entendimento (Scaico *et al.*, 2013).

O uso da computação no meio acadêmico tem se tornado cada vez mais frequente. A Ciência Computacional (ou Computação Científica), como é chamado esse ramo da computação, é aplicada em diversas áreas das ciências e tecnologias. A exemplo, temos o uso da computação na decodificação do genoma humano e na modelagem de sistemas complexos na Física (Tavares, 2014).

O ensino de cinemática no Ensino Médio (EM) é essencial, segundo a Base Nacional Comum Curricular (BNCC). O estudo do movimento dos corpos pode ser identificado em pelo menos 3 habilidades específicas das Ciências da Natureza (Brasil, 2018).

> (EM13CNT101) Analisar e representar as transformações e conservações em sistemas que envolvam quantidade de matéria, de energia e de movimento para realizar previsões em situações cotidianas e processos produtivos que priorizem o uso racional dos recursos naturais.
>
> (EM13CNT204) Elaborar explicações e previsões a respeito dos movimentos de objetos na Terra, no Sistema Solar e no Universo com base na análise das interações gravitacionais.
>
> (EM13CNT306) Avaliar os riscos envolvidos em atividades cotidianas, aplicando conhecimentos das Ciências da Natureza, para justificar o uso de equipamentos e comportamentos de segurança, visando à integridade física, individual e coletiva, e socioambiental.

A cinemática do ensino básico é simplificada para utilizar processos matemáticos conhecidos pelos alunos, como relações de proporcionalidade e resolução de equações. No ensino superior, por sua vez, utiliza conceitos de Cálculo Diferencial e Integral e de Equações Diferenciais Ordinárias (EDO), sobretudo quando é considerada a resistência do ar nos movimentos (Freire *et al.*, 2016).

Mostrada a necessidade do ensino de cinemática no EM, associando-a às vantagens do ensino tecnológico desses jovens, o atual trabalho foi pensado. A interdisciplinaridade da

Computação com Física já demonstrou sucesso em experimentos prévios (Machado *et al.*, 2021). O objetivo desse projeto é usar os conceitos de *algoritmização* para desenvolver um simulador de lançamento oblíquo com resistência do ar.

Tal projeto demonstra aos alunos que com o uso da Computação Científica podemos modelar problemas físicos complexos que necessitam de cálculos avançados de forma simples para obter resultados. Mostrar o processo de fazer ciência de forma mais próxima da realidade pode aproximar alunos da academia.

Após o desenvolvimento do simulador utilizando a linguagem de programação *Python* e o algoritmo desenvolvido em conjunto com os alunos, será realizado experimento prático com lançamento de bolas leves em quadra para comparar os resultados simulados com os resultados experimentais.

No Capítulo 2 será abordada Fundamentação Teórica do trabalho, onde os conteúdos de Física e Computação serão desenvolvidos, bem como o resultado esperado para o aluno e os softwares utilizados serão mostrados. Já no Capítulo 3 é apresentada uma sequência didática e é feito um melhor desenvolvimento dos softwares utilizados com suas respectivas explicações. Ao final do trabalho, no Capítulo 4, as conclusões do trabalho, bem como resultados do simulador produzido serão mostrados.

1.1 Objetivos

O presente trabalho tem, portanto, o objetivo de elaborar uma sequência didática que propõe de ensinar Computação para alunos dos anos iniciais do EM para a construção, em conjunto com o professor, de um simulador de lançamentos oblíquos. Aliada a produção do simulador, as experimentações com análise de vídeo serão ferramentas úteis na validação do simulador.

Essa intertextualidade visa ganhos em ambas as áreas do conhecimento com compreensão das aplicabilidades das equações e conceitos físicos e redução da necessidade de abstração a partir dos alunos, necessidade essa que afasta parte dos alunos das áreas de matérias exatas.

1.1.1 Objetivos específicos

Como objetivos específicos estão listados:

1. Melhorar a compreensão do Movimento Uniforme (MU);

2. Melhorar a compreensão do Movimento Uniformemente Variado (MUV);
3. Desenvolver um pensamento computacional;
4. Compreender as possibilidades e limitações de uma máquina de calcular;
5. Perceber a aplicação da física em modelos de jogos e simuladores.

2 FUNDAMENTAÇÃO TEÓRICA

Os conceitos de cinemática e dinâmica serão necessários para o completo entendimento do trabalho. Portanto, no Capítulo 2, faremos uma fundamentação teórica a partir da literatura sobre Física e Computação descrevendo conceitos necessários para alunos e futuros professores que possam aplicar a mesma sequência didática. Além disso, exemplos de simulações e softwares livres serão apresentados com o fito de validar o algoritmo pensado no projeto.

2.1 Princípios físicos envolvidos na simulação

A cinemática envolvida no movimento simulado permeia o MU e o MUV e pode ser obtida a partir do uso do cálculo diferencial e integral na segunda lei de Newton. Dessa maneira, obtêm-se a aceleração de uma partícula de massa qualquer, bem como sua velocidade e posição em quaisquer instantes de tempo. Ao considerar a resistência do ar é necessária a resolução de EDO para a obtenção de informação de aceleração, velocidade e posição das partículas, que dessa vez depende da massa e de uma constante positiva que modela a resistência do ar (Pereira; Bonfim, 2008; Freire *et al.*, 2016).

2.1.1 Lançamentos em campos gravitacionais uniformes

Considere um projétil sendo lançado a partir da origem dos espaços com velocidade inicial $\vec{V}_0$ que forma um ângulo θ com a horizontal, sua posição no instante de tempo t será dada pelo vetor $\mathbf{r}(t) = (x(t),y(t))$ de forma tal que

$$\begin{cases} \dot{x}(0) = \left|\vec{V}_0\right| cos\theta, & x(0) = 0 \\ \dot{y}(0) = \left|\vec{V}_0\right| sen\theta, & y(0) = 0, \end{cases} \tag{2.1}$$

a Figura 1 mostra graficamente a situação inicial de todos os lançamentos que serão analisados neste trabalho.

Para deduzir a trajetória de um lançamento oblíquo em um campo gravitacional uniforme e livre de resistência do ar precisamos primeiro identificar as forças que atuam sobre o corpo lançado. Nesse caso, apenas a força peso atua sobre o corpo:

$$\vec{F} = -mg\hat{j}. \tag{2.2}$$

Por aplicação direta da segunda lei de Newton na Equação 2.2, conseguimos obter

Figura 1 – Situação inicial do lançamento.

Fonte: elaborado pelo autor.

as componentes de aceleração do corpo em um instante de tempo t:

$$\begin{cases} \ddot{x}(t) = 0 \\ \ddot{y}(t) = -g. \end{cases} \tag{2.3}$$

Observamos, portanto, que no eixo x, o movimento é descrito como um MU, enquanto no eixo y, o movimento é descrito como um MUV com velocidade inicial positiva e aceleração negativa. A análise do eixo vertical é feita a partir da integração de sua componente de aceleração para a modulação de sua componente da velocidade:

$$\dot{y}(t) = \int \ddot{y}(t)\,dt = -gt + \left|\vec{V}_0\right| sen\theta, \tag{2.4}$$

que por sua vez pode ser novamente integrada para a modulação de sua posição vertical

$$y(t) = \int \dot{y}(t)\,dt = -\frac{g}{2}t^2 + \left(\left|\vec{V}_0\right| sen\theta\right)t, \tag{2.5}$$

e da mesma maneira a análise do eixo horizontal pode ser feita obtendo o resultado

$$x(t) = \left(|\vec{V}_0| cos\theta\right)t. \tag{2.6}$$

A posição, $\vec{r}(t)$, do projétil em um instante t pode agora ser escrita da forma

$$\vec{r}(t) = \left(\left|\vec{V}_0\right| cos\theta\, t, -\frac{g}{2}t^2 + \left|\vec{V}_0\right| sen\theta\right)t \tag{2.7}$$

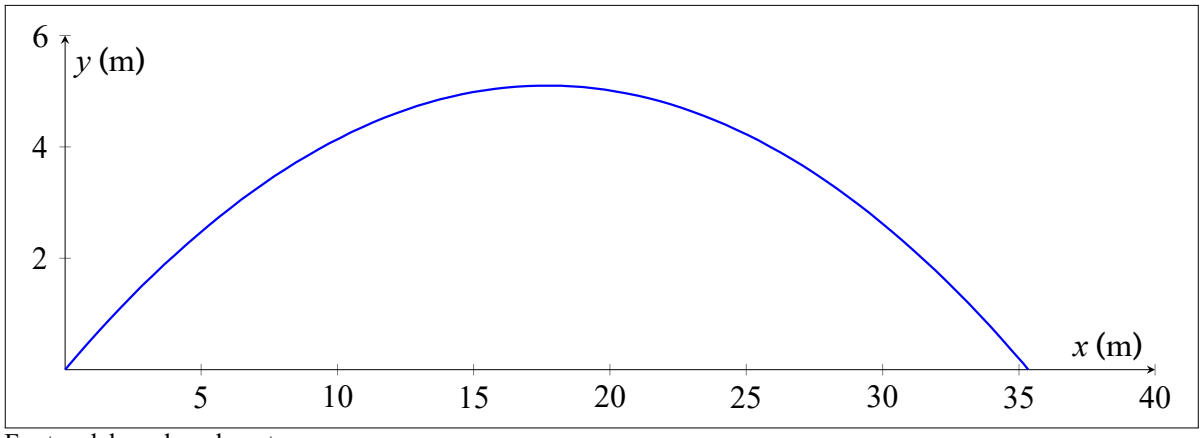
para valores de t contidos no intervalo $0 \leq t \leq t_{queda}$, onde t_{queda} é o instante de tempo tal que $\dot{y}(t_{queda}) = -\dot{y}(0)$, visto que $\mathbf{r}(t_{queda})$ possui componente vertical nula caracterizando fim do lançamento.

Por uma questão de continuidade e parametrização é fácil notar que a curva formada pelos possíveis valores de $\vec{r}(t)$ é uma parábola no intervalo considerado com equação

$$y(x) = x\tan(\theta) - \frac{g}{2v_0^2\cos^2(\theta)}x^2. \tag{2.8}$$

A trajetória descrita pela Equação 2.8 pode ser visualizada graficamente na Figura 2 para valores de V_0, θ e g iguais a, respectivamente, 20 m/s, 30° e 9,8 m/s²

Figura 2 – Gráfico da trajetória de um lançamento oblíquo sem resistência do ar.



Fonte: elaborado pelo autor.

Como uma simplificação para alunos do EM, as equações cinemáticas podem ser deduzidas a partir de equações mais triviais sem a necessidade do uso de cálculo diferencial e integral, chegando a resultado similiar. As ditas funções horárias da posição e velocidade já são amplamente trabalhadas na educação básica em livros-texto e apostilas, bem como seu ensino é fundamentado na BNCC (Brasil, 2018).

2.1.2 Resistência do ar

A resistência do ar é uma força que depende de fatores como a geometria do projétil e sua velocidade. Para fins de aproximação, a força de resistência do ar atua no corpo de forma proporcional à velocidade

$$F_{Ra} = -b\left|\dot{\vec{r}}\right|\hat{\dot{r}}, \tag{2.9}$$

onde b é uma constante positiva de dimensões $[kg/s]$ no Sistema Internacional de Unidades (SI) e $\dot{\vec{r}}$ é a velocidade vetorial do móvel.

Em um movimento unidirecional com início na origem, ou seja, $\vec{r}(0) = 0$, e com velocidade inicial V_0 com ação exclusiva da força de resistência do ar conseguimos moldar a

posição do corpo em função do tempo utilizando conceitos de EDO. Ao aplicar a segunda lei de Newton à Equação 2.9 obtem-se a seguinte EDO:

$$\ddot{\vec{r}} = -\frac{b}{m}\left|\dot{\vec{r}}\right|\hat{r}. \tag{2.10}$$

Como o movimento é unidirecional, o tratamento da equação pode ser escalar. Suasolução inicia-se na redução da ordem da EDO pelo método da substituição, faça $k = \left|\dot{\vec{r}}\right|$

$$\frac{dk}{dt} = -\frac{b}{m}k \tag{2.11}$$

cuja solução trivial nos leva para

$$k = C_1 e^{-\frac{b}{m}t} \tag{2.12}$$

em que C_1 é uma constante de integração. Desfazendo-se a substituição aplicada na Equação 2.11,obteremos

$$\left|\dot{\vec{r}}\right| = C_1 e^{-\frac{b}{m}t} \tag{2.13}$$

uma EDO de primeira ordem que pode facilmente ser resolvida para obtermos **r**(t).

$$|\vec{r}(t)| = -\frac{mC_1}{b}e^{-\frac{b}{m}t} + C_2 \tag{2.14}$$

onde C_2 é uma constante de integração.

Ambas as constantes de integração podem ser obtidas pelos valores iniciais domovimento.

$$|\mathbf{r}(0)| = 0 = -\frac{mC_1}{b}e^{-\frac{b}{m}0} + C_2$$

$$0 = -\frac{mC_1}{b} + C_2$$

$$C_2 = \frac{mC_1}{b} \tag{2.15}$$

$$\left|\dot{\vec{r}}(0)\right| = \left|\vec{V}_0\right| = C_1 e^{-\frac{b}{m}t}$$

$$C_1 = \left|\vec{V}_0\right| \tag{2.16}$$

Com o resultado da Equação 2.16, aplicando-o na Equação 2.15 obtemos seus valores

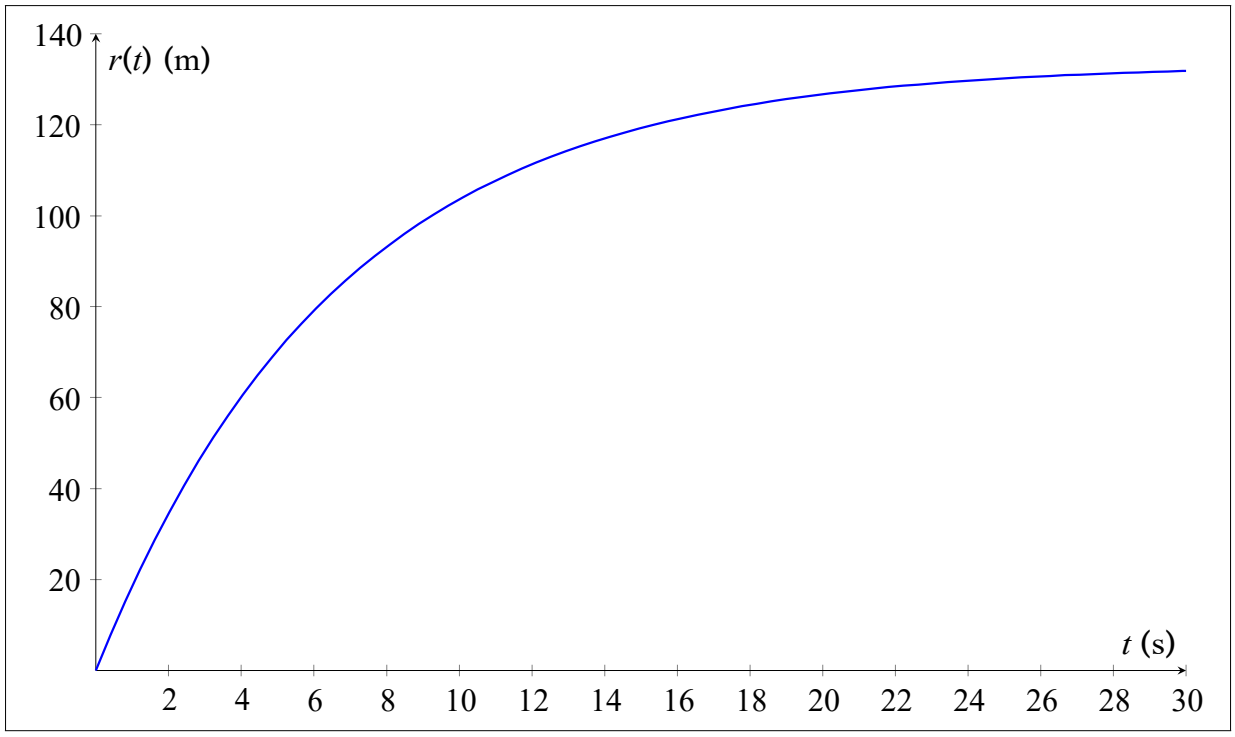
$$\begin{cases} C_1 = V_0 \\ C_2 = \frac{mV_0}{b} \end{cases} \tag{2.17}$$

Por fim, substituindo os achados na Equação 2.17, na Equação 2.14 finalizamos com

$$|\vec{r}(t)| = \frac{m\left|\vec{V}_0\right|}{b}\left(1 - e^{-\frac{b}{m}t}\right), \tag{2.18}$$

equação essa que representa a posição de um móvel em função do tempo. O que pode ser representado graficamente produzindo a Figura 3, que mostra um movimento que inicia-se com grande velocidade e seu módulo reduz com o passar do tempo devido à força de resistência que produz uma desaceleração no móvel.

Figura 3 – Gráfico da posição em função do tempo para um movimento unidirecional sob ação exclusiva da força de resistência do ar.



Fonte: elaborado pelo autor.

Tal solução é impraticável em sala de aula do EM, mas necessária para a modulação do lançamento com resistência do ar. Além dessa resolução, o caso com uma força constante e

proporcional à massa e contrária à velocidade inicial do movimento também se faz necessário de ser resolvido. Esse caso é traduzido como uma EDO não homogênea, tornando sua apresentação no ensino básico ainda mais complicada.

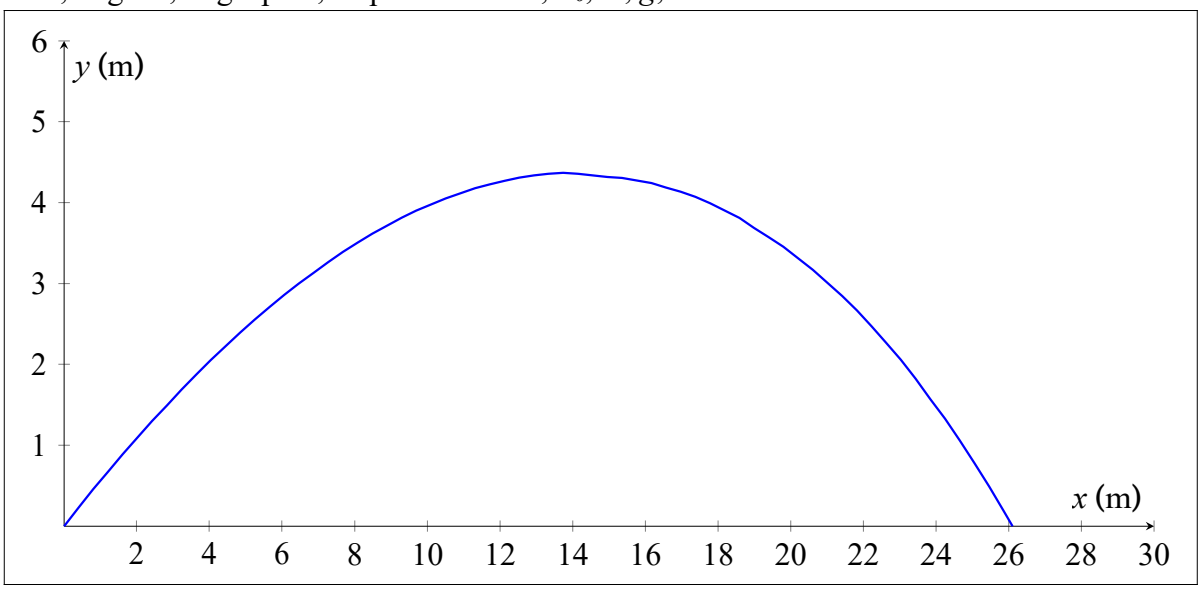
2.1.3 Lançamentos com resistência do ar

Em um lançamento com resistência do ar a trajetória do lançamento deixa de ser parabólica e o alcance dos projéteis passa a depender de suas massas, o que não ocorre nos estudos do ensino básico. A equação da trajetória torna-se não obtível sem conhecimento matemático restrito ao ensino superior.

$$y(x) = \left(tg\theta + \frac{m}{b}\frac{g}{v_0 cos\theta}\right)x + \frac{m^2}{b^2}gln\left(1 - \frac{b}{m}\frac{x}{V_0 cos\theta}\right) \tag{2.19}$$

A Equação 2.19 pode ser obtida pela parametrização das componentes, onde a componente horizontal é obtida na Equação 2.18 e a componente vertical, como já explicada, necessita da resolução de uma EDO não homogênea (Freire *et al.*, 2016; Pereira; Bonfim, 2008). Tal equação pode ser visualizada na Figura 4 para valores de V_0, θ , g, m e b iguais a, respectivamente, 20 m/s, 30°, 9,8 m/s², 2 kg e 0,5 kg/s.

Figura 4 – Gráfico da trajetória considerando resistência do ar cujos parâmetros são 20 m/s, 9,8 m/s², 2kg e 0,5 kg/s para, respectivamente, V_0, θ, g, m e b.

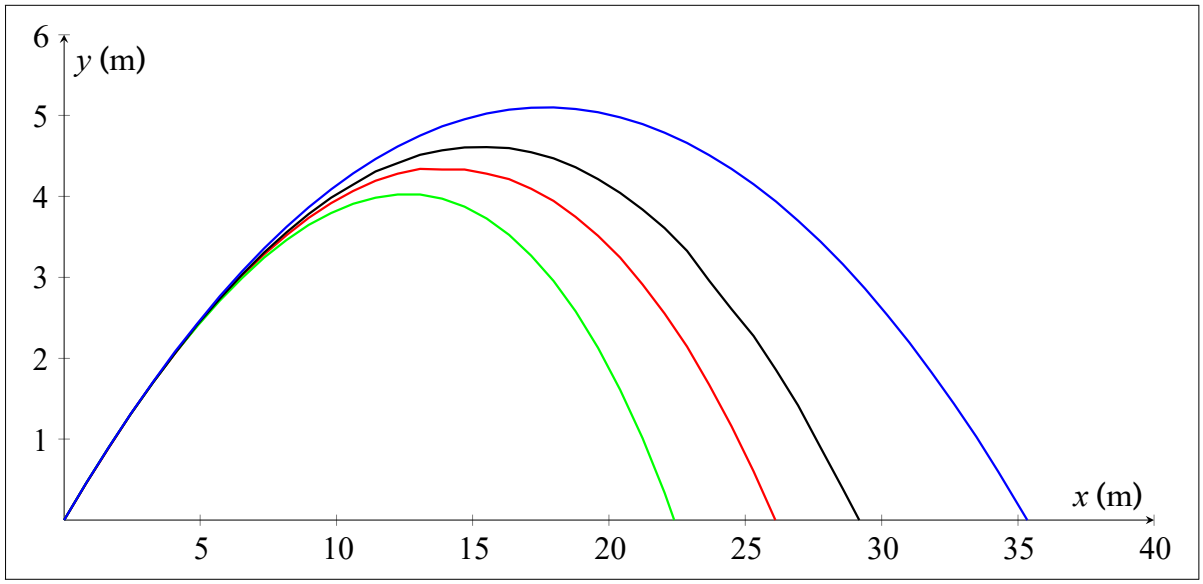


Fonte: elaborado pelo autor.

É possível comparar lançamentos com iguais valores de V_0, θ, g e m e diferentes valores para a constantes b em um único gráfico para melhorar a visualização de como esse fator

afeta a trajetória e o alcance. Esse gráfico está disposto na Figura 5, onde a curva azul mostra a trajetória sem resistência do ar, enquanto as demais curvas mostram, em ordem decrescente de alcance, valores de 0,3, 0,5 e 0,8 kg/s para a constante *b*.

Figura 5 – Gráfico das trajetórias de diferentes lançamentos considerando, em ordem decrescente de alance: Lançamento sem resistência do ar; Lançamento com resistência do ar e constante b = 0,3 kg/s; Lançamento com resistência do ar e constante b = 0,5 kg/s e Lançamento com resistência do ar e constante b = 0,8 kg/s.



Fonte: elaborado pelo autor.

2.2 Introdução ao movimento em simulações computacionais

Imagens estroboscópicas nos dão uma ideia valiosa para simulações computacionais. As telas dos computadores e as calculadoras dos simuladores não conseguem simular eventos no contínuo espaço-tempo, ou seja, todas as simulações, vídeos, jogos e captações de medidas físicas são discretos como uma fotografia estroboscópica e as simulações devem ser submetidas a tempos discretos.

> Nos anos 1950, o uso de computadores digitais estava começando a aumentar em ambientes de trabalho e em laboratórios científicos. Embora fossem caros e com capacidades relativamente limitadas, esses computadores eram freqüentemente utilizados para simular sistemas de processamento de sinais antes da produção em hardware, o que apresentava vantagens em termos de flexibilidade. Entretanto, o processamento não podia ser feito em tempo real, o que limitava as aplicações (Senda *et al.*, 2005).

> Cabe observar que essa evolução só foi possível graças à invenção e subseqüente proliferação dos microprocessadores que permitiram implementações de baixo custo de sistemas de processamento de sinais de tempo discreto. Na década

> de 1980, a tecnologia de circuitos integrados avançou a um nível que permitiu a implementação de microcontroladores em ponto fixo ou ponto flutuante com arquiteturas especialmente projetadas para implementar algoritmos de processamento de sinais de tempo discreto (Senda *et al.*, 2005).

Dessa maneira, jogos e simuladores funcionam calculando a próxima posição de um móvel a partir da posição atual e de um deslocamento. Deslocamento esse calculado a partir da velocidade atual, considerada constante até o próximo cálculo, multiplicada por um intervalo de tempo suficientemente curto. O cálculo aproxima-se da realidade tanto quanto for necessário a partir da variação desse intervalo de tempo considerado.

Em um sistema com tempo contínuo, a posição, $S(t)$, de um móvel em MU pode ser descrita como

$$S(t) = S_0 + Vt, \tag{2.20}$$

enquanto num sistema com tempo discreto será dada por

$$S[n] = S[n-1] + V\Delta T, \tag{2.21}$$

onde ΔT representa o intervalo de tempo entre uma posição e a próxima, com isso o termo $V\,\Delta T$ pode ser simplificado para um único termo que representa o deslocamento do móvel com aquela velocidade, V, naquele intervalo de tempo ΔT.

2.2.1 Exemplos na computação

Na literatura da Computação exemplos da operação citada são comuns, como bem explorado por Nystrom (2014):

Código-fonte 1 – Movimento de patrulha simples.

```
1 void Skeleton::update(double elapsed)
2 {
3   if (patrollingLeft_)
4   {
5     x -= elapsed;
6     if (x <= 0)
7     {
8       patrollingLeft_ = false;
```

```
9           x = -x;
10        }
11     }
12     else
13     {
14        x += elapsed;
15        if (x >= 100)
16        {
17           patrollingLeft_ = true;
18           x = 100 - (x - 100);
19        }
20     }
21  }
```

Essa função *update()* recebe um parâmetro de nome *elapsed* que representa o tempo passado entre a posição anterior e a nova posição. Nesse exemplo específico o móvel pode assumir uma das 101 posições possíveis ao longo de uma direção e move-se exatamente uma posição a cada uma unidade de tempo.

As linhas 5 e 6 do código-fonte 1 representam o cálculo anteriormente apresentado, nesse caso a nova posição, x, recebe seu valor anterior acrescido de $1 \cdot elapsed$ ou decrescido desse valor a depender do sentido do movimento. O cálculo presente nas linhas 9 e 18 apresentam uma inversão do móvel ao chegar nos limites de suas posições.

2.3 *Software* de análise de vídeo

O uso de imagens estroboscópicas aparece em diferentes contextos da educação, desde o *Physical Science Study Committee* (PSSC) até livros brasileiros. Uma evolução das imagens estroboscópicas é a videoanálise, que consiste na análise do movimento a partir de uma sequência de quadros que formam um vídeo (Dias *et al.*, 2018).

O uso, portanto, de imagens estroboscópicas e videoanáise, será apreciado nesse trabalho. Além da simulação feita por meio da Computação que já nos fornecerá uma imagem estroboscópica digitalmente construída, é possível, por meio de *softwares* livres, construir imagens estroboscópicas a partir de eventos reais de forma simples e econômica (Dias *et al.*,

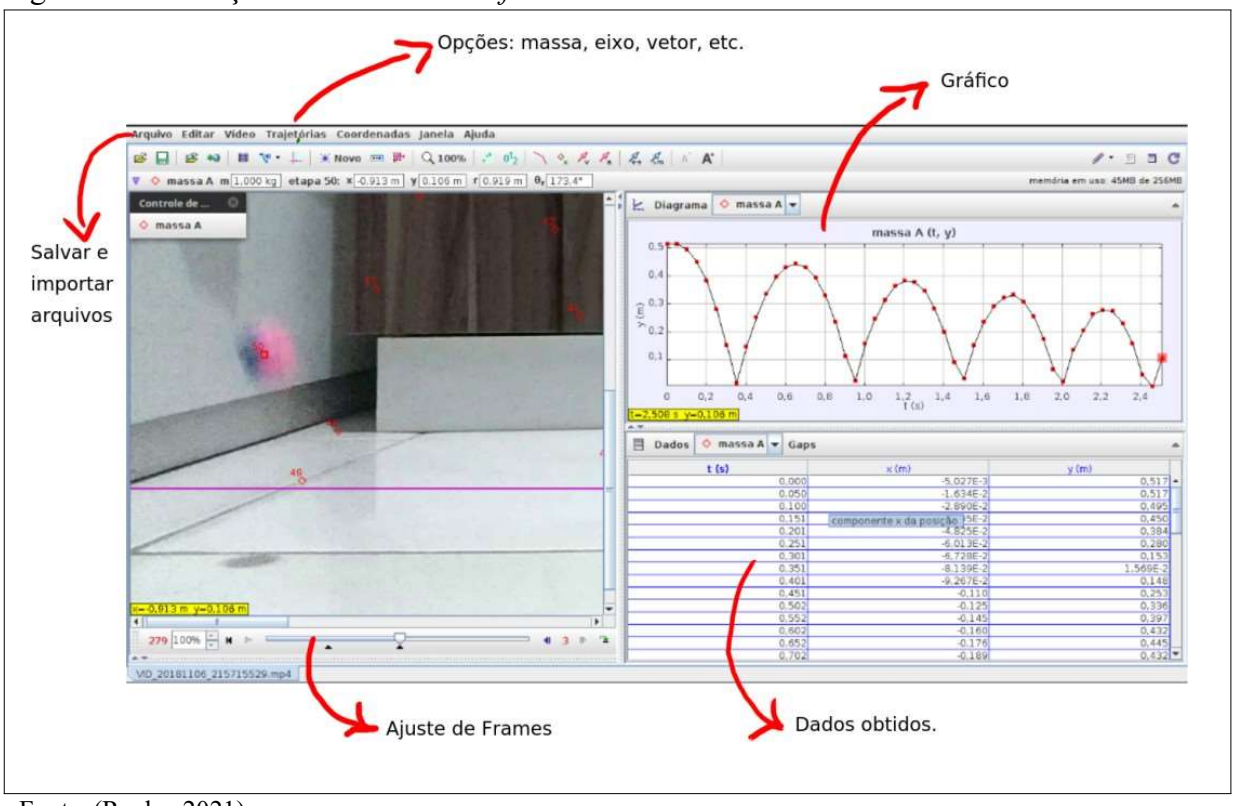
2017). Dessa maneira a comprovação da precisão física do simulador poderá ser realizada.

O *Software Tracker*, consegue produzir análise do movimento a partir de um vídeo digital gravado. Todo vídeo é composto de uma sequência de quadros, *frames*, capturados com intervalos de tempo fixos. O *software* utiliza esses quadros para analisar a movimentação do móvel ofertando ao usuário dados como aceleração, velocidade e posição em função do tempo (Dias *et al.*, 2017).

O uso do *software* escolhido é intuitivo e pode ser aprendido por educadores de forma rápida. O que aumenta a aplicabilidade da proposta metodológica (Dias *et al.*, 2017). Tal aplicativo foi desenvolvido por Brown e possui uma descrição de uso elaborada pelo próprio autor (Brown; Cox, 2009). A Figura 6 contém, de forma simplificada, instruções de uso.

Além de código aberto, o que permite distribuição e modificação em algum nível, o aplicativo dispõe de versão online com alguns vídeos já produzidos. Ambos os formatos podem ser encontrados até a data de publicação desse trabalho via sítio: https://physlets.org/tracker/.

Figura 6 – Descrição da interface do *software* Tracker.



Fonte: (Rocha, 2021)

2.4 Código Alvo

O objetivo do trabalho é que o aluno seja capaz de desenvolver um código para simular o movimento ao longo da direção horizontal utilizando o cálculo disponível em Equação 2.21, bem como ao longo da direção vertical. A parte gráfica do simulador fica a cargo do professor, que pode desenvolvê-la por conta própria ou optar por utilizar o código disponível no Apêndice B.

O código disponibilizado utiliza conceitos de Programação Orientada a Objetos (POO), onde, em *Python*, uma classe *Projectile* foi desenvolvida, mas o cálculo de movimento da classe não foi implementado. A implementação do cálculo de movimento deve ser feita pelos próprios alunos e posteriormente importada pelo professor. A definição da classe citada pode ser encontrada no Apêndice C.

Nesse código disponibilizado, a classe *Projectile* retorna, em uma de suas funções, uma *string*, ou seja, uma sequência de caracteres, contendo sua trajetória. Essa *string* é utilizada para representar graficamente as sucessivas posições do projétil.

É esperado que o aluno seja capaz de desenvolver dois métodos que serão posteriormente importados pela classe *Projectile*, um exemplo de código esperado para os alunos está mostrado a seguir:

Código-fonte 2 – Código esperado para o aluno.

```
1 from Projectile import Projectile # Importa a classe
     desenvolvida pelo professor para poder usar os atributos
      compatíveis
2 def move_horizontally(self: Projectile):
3     self.x_position = self.x_position + self.x_velocity *
         self . delta_time # Cá lculo da posi ção x após um
         intervalo de tempo
4     self . x_velocity = self . x_velocity - self .
         air_drag_constant * self . delta_time # Cá lculo da
         influência do ar na velocidade x após um intervalo
         de tempo
5
6 def move_vertically(self: Projectile):
```

```
7       self.y_position = self.y_position + self.y_velocity *
            self.delta_time # Cálculo da posição y após um
            intervalo de tempo
8       if self.y_velocity >= 0: # Cálculo da influência do ar
            na velocidade y após um intervalo de tempo
9           self.y_velocity = self.y_velocity - self.
                air_drag_constant * self.delta_time
10      else:
11          self.y_velocity = self.y_velocity + self.
                air_drag_constant * self.delta_time
12      self.y_velocity = self.y_velocity + self.
            graivity_acceleration * self.delta_time # Cálculo da
            influência gravitacional na velocidade y após um
            intervalo de tempo
```

O Código-fonte 2 traduz para a linguagem *Python* a Equação 2.21, e também implementa outras equações semelhantes para alterar a velocidade do projétil a depender de uma aceleração constante ou de uma aceleração proporcional à velocidade, como mostrado na Equação 2.9.

É valioso perceber que o simulador implementado e divulgado nos apêndices poderá simular o movimento do projétil mesmo com o código dos alunos desenvolvidos de forma errônea ou incompleta, podendo servir de teste para que o aluno tenha uma melhor compreensão do que está desenvolvendo e de como o computador interpreta seus comandos.

2.5 Ganhos para o aluno

Como citado no Capítulo 1, estudar Computação durante os anos do EM traz ganhos para o aluno nas áreas da matemática e da linguagem. A melhora do aluno na linguagem pode refletir em uma melhor interpretação textual, habilidade valiosa para o bom entendimento da Física e de demais ciências (Lima; Ricardo, 2015).

O Pensamento Computacional (PC) desenvolvido durante o aprendizado de computação pode ser valioso para solucionar problemas, sejam esses problemas cotidianos ou mais complexos (Bordini *et al.*, 2016). Essa melhora na habilidade de solução de problemas

também pode refletir numa melhor capacidade de resolver problemas de ciências, sobretudo na Física.

Como concluído por (Machado *et al.*, 2021), o ensino de Computação proporciona aos alunos condições de serem produtores de tecnologia e não apenas consumidores, tornando-os mais letrados digitalmente e sendo capazes de entender o mundo a sua volta. Além disso, a experimentação é facilitadora para o aprendizado de ciências naturais. Portanto, essa interdisciplinaridade traz oportunidades valiosas para o aprendizado.

3 METODOLOGIA

Nesse capítulo a proposta de sequência didática será apresentada, bem como o uso dos *softwares* escolhidos para o curso. Os planos de aula de cada um dos encontros estão documentados no Apêndice A. É importante para qualquer professor que almeje aplicar tal curso que valorize o trabalho em grupo dos alunos, afinal, nos cursos de computação de ensino técnico e superior é inevitável a colaboração entre os diferentes desenvolvedores de um código (Brito; Bernardo, 2021).

A sequência didática é uma ferramenta útil para a elaboração de atividades de ensino, para Araújo (2013)

> De modo simples e numa resposta direta, sequência didática (doravante SD) é um modo de o professor organizar as atividades de ensino em função de núcleos temáticos e procedimentais.

3.1 Sequência didática

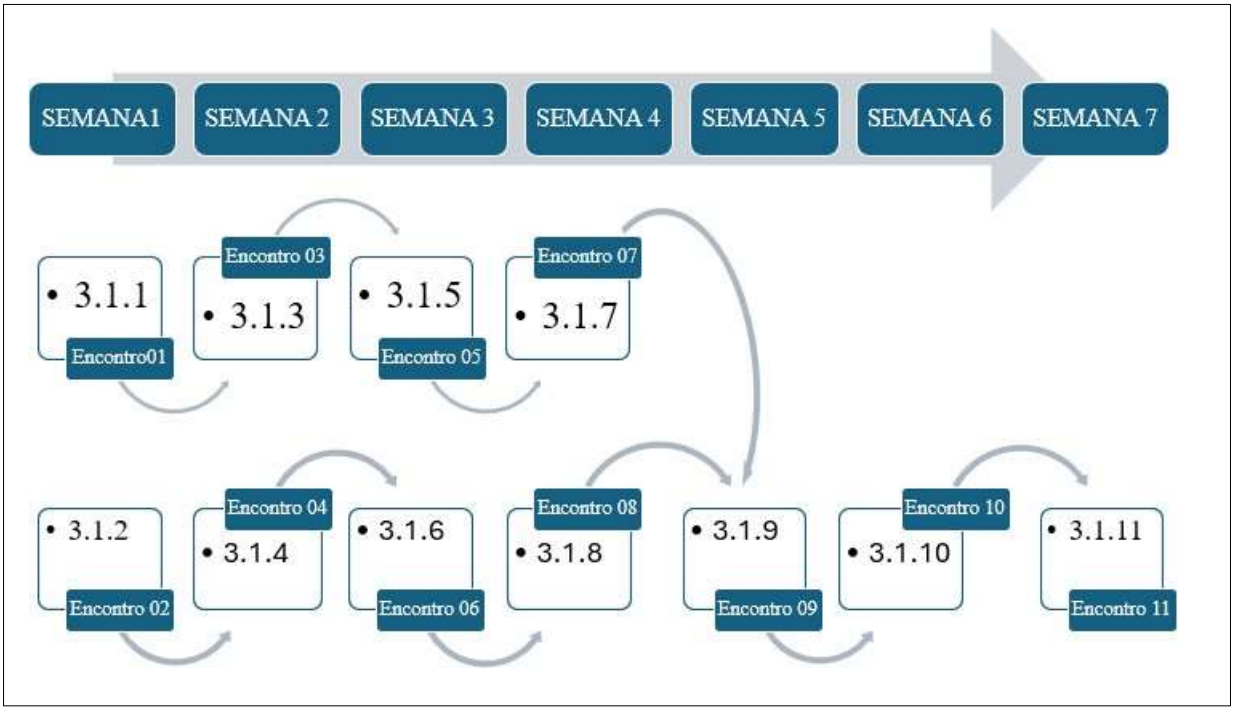
O curso é previsto para alunos que estão tendo ou já tiveram contato recente com os conceitos de lançamentos oblíquos e desejam aprofundar-se em temas de Computação e Mecânica, em principal alunos do 1° ano do EM. Dessa maneira o curso é dividido em onze encontros com duas aulas de 50 minutos cada. Os encontros acontecerão alternando entre ensino de Física e ensino de Computação, sendo quatro focados na Física (subseção 3.1.1, subseção 3.1.3, subseção 3.1.5, subseção 3.1.7), quatro focados em Computação (subseção 3.1.2, subseção 3.1.4, subseção 3.1.6, subseção 3.1.8). Os três encontros restantes serão destinados à construção da simulação, experimentação prática e posterior comprovação da simulação como fiel à realidade.

Os encontros que tratam de Física ocorrem no período curricular do aluno, enquanto os encontros que tratam de Computação e simulação ocorrem no fora do período curricular de forma a não afetar o andamento da disciplina em Física negativamente. O diagrama dessa sequência didática se faz visível na Figura 7.

3.1.1 Encontro 01 - Apresentação do conteúdo de lançamentos

Após a exposição do conteúdo de lançamento horizontal, onde o princípio da independência dos movimentos já foi trabalhado com os alunos, o conteúdo de lançamento oblíquo pode ser iniciado nesse encontro. É valioso utilizar conceitos de trigonometria e decomposição

Figura 7 – Diagrama contendo um cronograma por semana com os encontros a serem realizados em cada uma delas.



Fonte: elaborado pelo autor.

de movimento, como bem desenvolvido na subseção 2.1.1 para resolver problemas iniciais do conteúdo de lançamento oblíquo.

3.1.2 Encontro 02 - Apresentação da Computação como área de estudo

A computação é apresentada aos alunos mostrando inicialmente as possibilidades da programação e tentando fazer clara a melhor das capacidades do computador: computar. Mostrar que o computador consegue fazer cálculos repetitivos de forma rápida e precisa é essencial para os próximos passos. Além disso, é importante, também, comentar sobre como praticar programação fora do ambiente de sala de aula em dispositivos móveis e computadores de mesa.

3.1.3 Encontro 03 - Decomposição e cálculos no lançamento oblíquo

As questões iniciais são resolvidas e as dúvidas são sanadas, consolidando conceitos como tempo de voo e altura máxima. É útil determinar a equação do alcance para que algumas questões sejam solucionadas mais rapidamente.

3.1.4 Encontro 04 - Primeiros scripts

Com o entendimento das possibilidades da programação, o aluno é incentivado, nesse ponto, a criar seus próprios *scripts* para interação com o usuário. Os cálculos devem ser apresentados para os alunos, bem como as ordens de operações aritméticas devem ser relembradas. O extensivo uso de parênteses na descrição de operações aritiméticas é fortemente recomendado nesse ponto.

3.1.5 Encontro 05 - Resolução de questões e aprofundamento em lançamentos

Com a resolução de mais problemas físicos, é possível mostrar aos alunos as simetrias desse movimento. Vale ressaltar aos alunos que o módulo do vetor velocidade é igual para quaisquer dois pontos à mesma altura em relação ao ponto inicial do movimento.

3.1.6 Encontro 06 - Estrutura condicional em scripts

Após a resolução de problemas de programação básicos com ordens simples e diretas, a estrutura condicional deve ser apresentada ao aluno de forma que ele consiga criar *scripts* que executam diferentes trechos de código a partir de diferentes condições.

3.1.7 Encontro 07 - Avaliação da aprendizagem em Física

Nesse encontro, é importante avaliar o aprendizado dos alunos acerca do tema de duas maneiras. Incentivar que os alunos resolvam questões debatendo entre si e com orientação do professor na primeira metade do encontro, e aplicar um teste para que os alunos tentem, individualmente, resolver problemas.

Vale ressaltar que, nesse encontro, assim como nos demais encontros avaliativos (subseção 3.1.9 e subseção 3.1.11), a avaliação deve ser elaborada diante de critérios claros e entendidos pelos alunos. Isso possibilita que os alunos façam uma autoavaliação e tomem consciência de seu aprendizado. Os resultados aqui obtidos devem ser interpretados e devem promover reflexão e posterior evolução do processo de ensino-aprendizagem. Dessa maneira é possível tomar a avaliação como uma etapa do processo de ensino-aprendizagem, não somente de constatação dele (Darsie, 1996).

3.1.8 Encontro 08 - Estruturas de repetição em scripts

Estruturas de repetição passam a ser trabalhadas para que um mesmo trecho do código não precise ser escrito várias vezes para que possa ser executado várias vezes. A melhoria nesse momento vai além da redução do número de linhas de código, pois a repetição pode depender de variáveis que podem mudar ao longo da execução do *script*.

3.1.9 Encontro 09 - Avaliação da aprendizagem em Computação

Nesse encontro é importante avaliar o aprendizado dos alunos acerca do tema de duas maneiras. Incentivar que os alunos resolvam questões debatendo entre si e com orientação do professor na primeira metade do encontro, e aplicar um teste para que os alunos tentem, individualmente, resolver problemas.

3.1.10 Encontro 10 - Apresentação do simulador e produção de scripts

Com a base física e computacional consolidada e validada nos alunos, é possível apresentar aos alunos a estrutura do simulador, explicando brevemente o funcionamento da classe *Projectile* e do *script* encarregado pela renderização do simulador. Nesse momento é possível mostrar ao aluno como o movimento em simulações é calculado e como isso é útil para simular movimentos com resistência do ar. Os alunos devem elaborar códigos para serem implementados no simulador, o professor pode apresentar um código próprio para explicar conceitos adicionais que deseje abordar e que os alunos possam ter neglicenciado.

3.1.11 Encontro 11 - Testes práticos e computacionais

Experimentos reais devem ser feitos e filmados para que, com o *software* Tracker, a videoanálise possa ser feita e comparada com os resultados mostrados pelo simulador. Os códigos dos alunos devem ser testados e validados. Esse momento é um excelente momento de aprendizagem, onde os alunos podem aprender com as correções e comentários do professor, bem como o professor pode comentar sobre conceitos mais aprofundados da programação como vetores, funções e bibliotecas.

3.2 Usando código-fonte

O professor deve se encarregar de elaborar um simulador que aceite implementação de movimento dos alunos, afinal o código que deve ser desenvolvido pelos alunos deve ser responsável apenas pelo cálculo de movimento do projétil. Consta no Apêndice B um código de simulador desenvolvido pelo autor do presente trabalho para uso de demais professores.

Ao mostrar o simulador aos alunos, é valioso mostrar seu funcionamento escrevendo algumas linhas de código e testando-as. A fim de exemplificar, estarão expostos dois possíveis testes a serem realizados com os alunos e uma simulação final considerando a composição dos movimentos.

3.2.1 Movimento uniforme

Para um MU em uma dimensão, traduziremos a Equação 2.21 para Python da seguinte maneira:

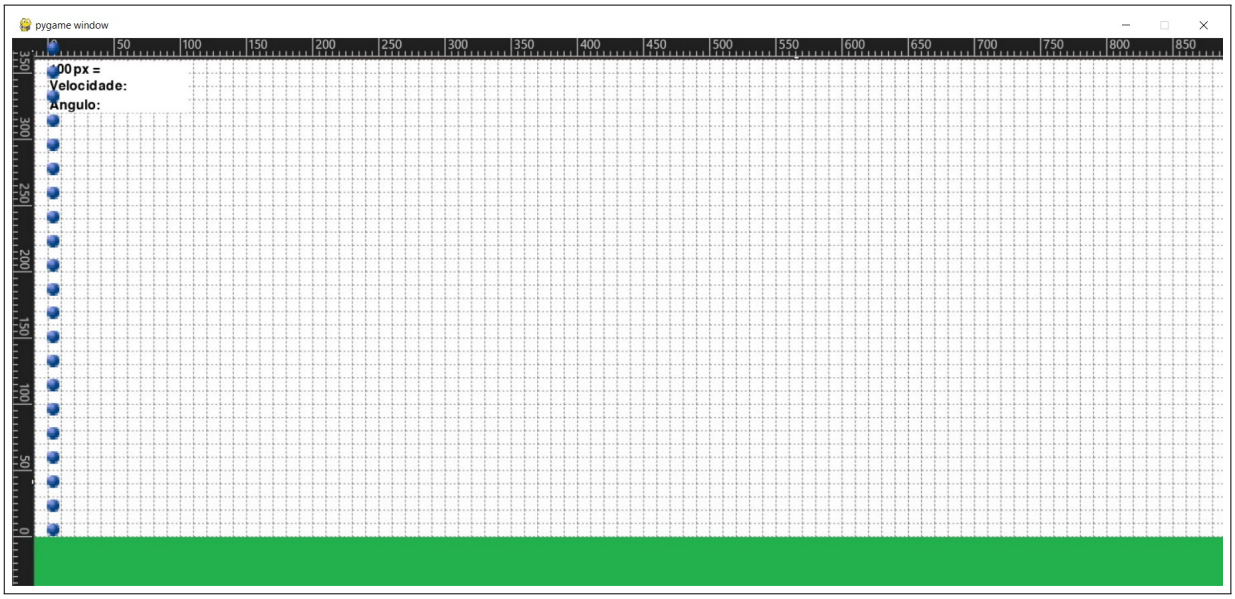
Código-fonte 3 – Movimento Uniforme em uma dimensão.

```
1 def move_horizontally(self):
2     pass
3
4 def move_vertically(self):
5     self.y_position = self.y_position + self.y_velocity *
          self.delta_time
```

Note que a função *move_horizontally(self)* segue definida, mesmo que sem comandos, para que não surjam erros na execução do código. Para que esse teste funcione, alguns ajustes precisam ser feitos no código do simulador visto que, para fins didáticos e de simplificação, muitos cálculos são realizados pensando no lançamento oblíquo e não em um movimento unidimensional.

Finalmente o simulador retorna uma imagem estroboscópica digitalmente gerada que mostra um movimento uniforme na vertical, como mostrado na Figura 8.

Figura 8 – Resultado obtido pelo simulador com Código-fonte 3.



Fonte: elaborado pelo autor.

3.2.2 Movimento Uniformemente Variado

Já para o MUV, também em uma dimensão, continuaremos implementando a Equação 2.21, visto que para curtos intervalos de tempo, como $V_0 \approx V \approx V_m$, o movimento pode ser tratado como um MU para fins de aproximação. No entanto, também será necessário implementar uma outra equação referente à variação da velocidade.

De forma análoga à apresentada na Equação 2.21, é possível obter:

$$V[n] = V[n-1] + a\Delta T. \tag{3.1}$$

Implementando o cálculo da Equação 2.21 e da Equação 3.1 em *Python*, temos:

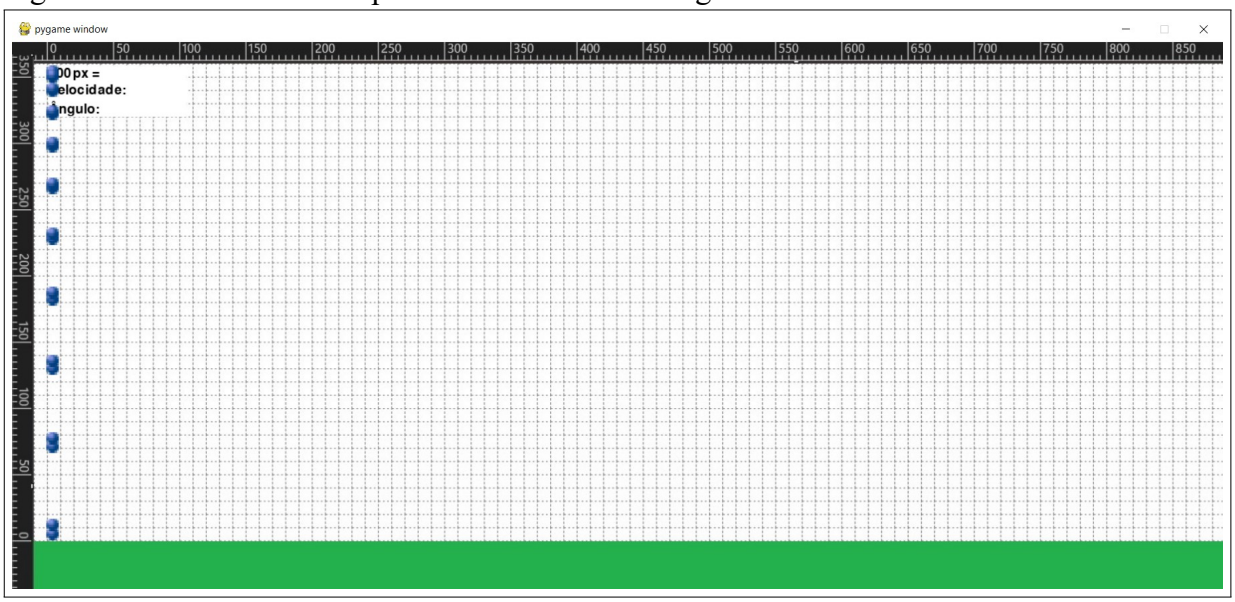
Código-fonte 4 – Movimento Uniformemente Variado em uma dimensão.

```
1 def move_horizontally(self):
2     pass
3
4 def move_vertically(self):
5     self.y_position = self.y_position + self.y_velocity *
          self.delta_time
6     self.y_velocity = self.y_velocity + self.
          graivity_acceleration * self.delta_time
```

Em que, assim como no Código-fonte 3, a função *move_horizontally(self)* segue definida, mas sem comandos para evitar erros de execução no código do simulador.

Como resultado simulado obtemos a Figura 9, uma imagem estroboscópica digitalmente construída de um lançamento vertical em que, devido a sucessivas aproximações, a simetria não é perfeitamente mantida, fazendo com que algumas posições apareçam levemente diferentes do resultado esperado, devido à propagação de erro.

Figura 9 – Resultado obtido pelo simulador com Código-fonte 4.



Fonte: elaborado pelo autor.

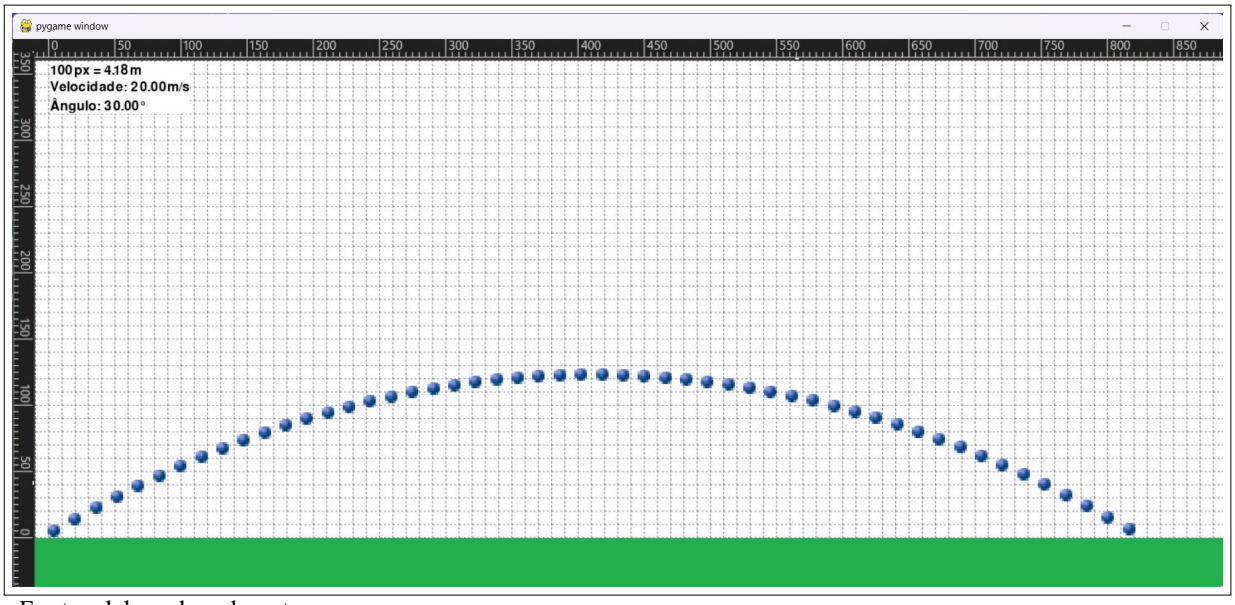
3.2.3 Lançamento oblíquo

Em um lançamento oblíquo com resistência do ar o Código-fonte 2 é implementado no simulador produzindo resultados que podem ser visualizados na Figura 10 e na Figura 11.

É possível observar que o grid quadriculado se mantém, mas a proporção do que esse grid significa modifica-se conforme o lançamento de modo a ocupar da melhor maneira a tela do simulador independentemente do alcance do lançamento, isso pode ser visto no canto superior esquerdo da janela.

A Figura 10 mostra um alcance simulado próximo de 35 m, o que condiz com os cálculos físicos, como mostrado pela Figura 2. A Figura 11 considera um valor de resistência do ar que faz com que seu alcance seja reduzindo em relação ao lançamento anterior. Os artefatos observados abaixo do solo surgem devido ao formato com que a simulação é feita, representando posições durante todo o tempo de voo calculado sem resistência do ar.

Figura 10 – Resultado obtido pelo simulador com Código-fonte 2 e sem resistência do ar.



Fonte: elaborado pelo autor.

Figura 11 – Resultado obtido pelo simulador com Código-fonte 2 e com resistência do ar.

Fonte: elaborado pelo autor.

3.3 Usando o Tracker

A Figura 6 contém de forma simplificada as principais funcionalidades do software Tracker, mas para seu uso em sala de aula no encontro 11 (subseção 3.1.11) será necessário um maior desenvolvimento de suas funcionalidades.

Ao importar um arquivo de vídeo para o *software*, a primeira coisa que se deve fazer é selecionar quais quadros serão utilizados para a análise. Isso é feito na opção "Vídeo > Ajustes de Corte de Vídeo...", onde será escolhido o quadro em que o movimento a ser analisado se

inicia e o quadro em que se encerra a análise.

Após isso, para realizar a análise, será necessário criar um novo ponto de massa, é assim que o programa conseguirá fazer os cálculos necessários. Isso é feito através da opção "Trajetórias > Novo > Ponto de Massa".

Com essas duas etapas concluídas, é possível começar a definir a trajetória do objeto. Isso é feito através da opção "Trajetórias > Massa A (ou a massa cuja análise é necessária) > Trajetória Automática...". Nessa opção, utilizando atalhos descritos na própria janela é possível ensinar ao programa como o objeto aparece na imagem, assim o programa conseguirá localizar o objeto nos próximos quadros. Caso o programa não consiga identificar o objeto em algum quadro ou caso o programa identifique erroneamente, é possível ajustar a marcação manualmente.

Após a realização de todo esse processo, os dados são fornecidos ao usuário conforme a Figura 6, podendo ser alterada a grandeza exposta em cada gráfico ou tabela.

O uso do *software* se faz presente apenas no último encontro para que, a partir da gravação de lançamentos, o resultado obtido via simulador possa ser validado, bem como possa ser determinado o valor da constante *b* para diferentes objetos.

4 CONSIDERAÇÕES FINAIS

O desinteresse de boa parte dos alunos pelo conteúdo de Física muitas vezes ocorre devido à forma como essa disciplina é apresentada aos alunos ao longo dos anos do EM. A necessidade de abstração de conceitos é um dos motivos de desinteresse, visto que a maioria dos conceitos é apresentada de forma idealizada e distante da realidade. Uma forma de contornar essa necessidade é com o uso de simuladores e experimentos práticos.

Demonstrar, portanto, aos alunos como a ciência é feita atualmente utilizando contextos reais em que o aluno tem uma participação ativa na construção do conhecimento se apresenta como uma possibilidade para aumentar o interesse dos alunos.

Propostas metodológicas como a de Rocha (2021) já trazem diferentes formas de abordar temas da Física de forma prática, o presente trabalho se baseia em diversas fontes para propor, também, uma outra maneira de atrair alunos e tornar mais significativa a aquisição de conceitos importantes para a conclusão do EM.

Além de ser uma proposta para aumentar o interesse de alunos pela Física, este curso também oferece uma breve formação em programação, habilidade importante para diferentes áreas do conhecimento.

Espera-se que o estudo desenvolvido para essa Monografia possa auxiliar professores e alunos do EM a melhorarem o processo de ensino-aprendizagem acerca desse tema. O ganho de conhecimento na área da Computação complementa os conhecimentos de Física, assim como a habilidade matemática desenvolvida pela Física enriquece o ensino de programação, criando uma relação mútua de reforço entre as disciplinas. O incentivo ao trabalho em equipe é fundamental para a boa execução do curso proposto.

REFERÊNCIAS

ARAÚJO, D. L. de. O que é (e como faz) sequência didática? **Entrepalavras**, Fortaleza, v. 3, n. 1, p. 322–334, 2013. Disponível em: http://www.entrepalavras.ufc.br/revista/index.php/Revista/article/view/148. Acesso em: 26 set. 2014.

BORDINI, A.; AVILA, C. M. O.; WEISSHAHN, Y.; CUNHA, M. M. da; CAVALHEIRO, S. A. da C.; FOSS, L.; AGUIAR, M. S.; REISER, R. H. S. Computaçao na educação básica no brasil: o estado da arte. **Revista de Informática Teórica e Aplicada**, Porto Alegre, v. 23, n. 2, p. 210–238, 2016. Disponível em: https://seer.ufrgs.br/index.php/rita/article/view/RITA-VOL23-NR2-210. Acesso em: 02 set. 2024

BRASIL. Ministério da Educação. **Base nacional comum curricular**. Brasília, 2018.

BRITO, L.; BERNARDO, J. Classwork: uma ferramenta de acompanhamento em tempo real da contribuição individual de alunos de cursos de computação no desenvolvimento de projetos de software acadêmicos hospedados no github. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 32. **Anais** [...] Porto Alegre, RSl: SBC, 2021. p. 1128–1139. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/18136. Acesso em: 03 mai. 2024

BROWN, D.; COX, A. J. Innovative uses of video analysis. **The Physics Teacher**, *[s.l.]*, v. 47, n. 3, p. 145–150, 2009.

DARSIE, M. M. P. Avaliação e aprendizagem. **Cadernos de pesquisa**, *[s.l.]* n. 99, p. 47–59, 1996.

DIAS, M. A.; VIANNA, D. M.; CARVALHO, P. S. Contribuições de imagens estroboscópicas e da vídeo-análise para intervenções didáticas de investigação. In: SIMPÓSIO NACIONAL DE ENSINO DE FÍSICA, 22. **Anais** [...] São Paulo, SP, 2017.

DIAS, M. A.; VIANNA, D. M.; CARVALHO, P. S. A queda dos corpos para além do que se vê: contribuições das imagens estroboscópicas e da videoanálise para a alfabetização científica. **Ensaio Pesquisa em Educação em Ciências**, Belo Horizonte, MG, v. 20, p. e2947, 2018.

FREIRE, W. H. C.; MEDEIROS, M. L.; LEITE, D.; SILVA, R. M. Lançamento oblíquo com resistência do ar: Uma análise qualitativa. **Revista Brasileira de Ensino de Física**, São Paulo, v. 38, n. 1,p. 1306, 2016.

LIMA, L. G. d.; RICARDO, E. C. Física e literatura: uma revisão bibliográfica. **Caderno Brasileiro de Ensino de Física**, Florianópolis, v. 32, n. 3, p. 577–6, 2015.

MACHADO, V.; AMORIM, T.; BARROS, P. Interdisciplinaridade no ensino de física e computação na educação básica: relato de experiência de um curso de formação inicial e continuada sob a perspectiva na construção de experimentos. In: SBC. Simpósio Brasileiro de Educação em Computação, Jataí. **Anais** […] Jataí, Goiás, 2021. p. 246–254. Disponível em: https://sol.sbc.org.br/index.php/educomp/article/view/14491/14337. Acesso em: 19 abr. 2024.

NYSTROM, R. Sequencing patterns. *In:* NYSTRON, R. **Game Programming Patterns**. *[S. l.]*: Genever Benning, 2014. 139–152 p.

PEREIRA, L. R.; BONFIM, V. Regiões de segurança em lançamento de projéteis. **Revista Brasileira de Ensino de Física**, São Paulo, v. 30, n. 3, p. 3313–1, 2008.

ROCHA, P. S. **Uma proposta de atividade experimental para alunos do fundamental II e médio:** "Egg Drop Competition**"**. 2021 68f. Monografia (Licenciatura em Física) - Centro de Ciências, Universidade Federal do Ceará, Fortaleza, 2021.

SCAICO, P.; LIMA, A. de; AZEVEDO, S.; SILVA, J. B. da; RAPOSO, E. H.; PAIVA, L. F.; ALENCAR, Y.; MENDES, J. P.; SCAICO, A. Ensino de programação no ensino médio: Uma abordagem orientada ao design com a linguagem scratch. **Revista Brasileira de Informática na Educação**, Porto Alegre, v. 21, n. 02, p. 92, 2013. ISSN 2317-6121. Disponível em: http://milanesa.ime.usp.br/rbie/index.php/rbie/article/view/2364. Acesso em: 18 abr. 2024

SENDA, A.; MELO, M. S.; SILVA, M. T.; EISENCRAFT, M.; MELO, M. A. Aplicações de processamento digital de sinais em engenharia elétrica. In: CONGRESSO BRASILEIRO DE ENSINO DE ENGENHARIA, 33. **Anais** [...] Campina Grande, PB: *[s. n.]*, 2005.

SILVEIRA, L. P. N.; KNIRSCH, J. F. Como a programação pode auxiliar no desenvolvimento do raciocínio lógico em crianças, adolescentes e jovens. **Uma Nova Pedagogia para a Sociedade Futura**, Restinga Seca, RS, n. 2, p. 734–737, 2016.

TAVARES, J. N. Computação científica no ensino. **Revista de Ciência Elementar**, Porto, v. 2, n. 3, p. 26–28, 2014.

APÊNDICE A – PLANOS DE AULA

Os seguintes arquivos em *Portable Document Format* (PDF) contêm o plano de aula de cada um dos encontros planejados para o curso, totalizando 11 planos de aula.

Universidade Federal do Ceará

Plano de Aula 01

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Lançamentos oblíquos
Pré-Requisitos:	MU, MUV, decomposição de vetores e lançamentos horizontais
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Conseguir decompor o lançamento oblíquo em dois movimentos independentes entre si: um MU na horizontal e um MUV na vertical. Com isso, conseguir compreender que o tempo que o projétil permanecerá em movimento em qualquer das componentes do movimento é igual ao tempo de voo.

1.2 Específicos

- Decompor a velocidade inicial em componentes ortogonais;
- Calcular o tempo de voo de um projétil lançado;
- Calcular, por meio de mais de uma etapa, o alcance de um lançamento.

2 Conteúdos

- Decomposição do vetor velocidade;
- Movimento Uniforme;
- Movimento Uniformemente Variado;
- Composição de movimento.

3 Procedimentos metodológicos

Apresentação expositiva do conteúdo, aliada a exemplificação e apresentação de simuladores prontos.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;
- Computador equipado com *Python* e/ou simuladores.

5 Avaliação

Os alunos deverão demonstrar compreensão suficiente para, por exemplo, resolver questões iniciais do conteúdo de lançamento oblíquo. Compreensão esta avaliada por meio da capacidade de resolver as questões propostas pelo professor, do tempo decorrido para a resolução e das dúvidas levantadas pelos alunos.

Referências

[1]DOCA, R. H.; BISCUOLA, G. J.; BOAS, N. V. **Tópicos de Física**. São Paulo: Saraiva, 2012.

[2] NUSSENZVEIG, H. Moysés. **Curso de Física Básica 1**. São Paulo: Bulcher, 2011.

Universidade Federal do Ceará

Plano de Aula 02

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	O que é a computação?
Pré-Requisitos:	–
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Compreender as facilidades de uma máquina computadora, assim como suas limitações. Perceber a variedade de áreas do conhecimento que são, atualmente, beneficiadas com o uso da computação. Entender o que é, e quais as aplicabilidades de um algoritimo, e seu uso no cotidiano.

1.2 Específicos

- Diferenciar *hardware* e *software*;
- Diferenciar o pensamento humano do pensamento computacional;
- Entender que o computador executa cálculos de modo rápido e consistente;
- Conseguir escrever um algoritimo.

2 Conteúdos

- *Hardware* e *software*;
- História da Computação;
- Conceitos de algoritimização.

3 Procedimentos metodológicos

Apresentação expositiva do conteúdo, aliada a exemplificação e apresentação de códigos prontos e produzidos durante a aula.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;
- Computador equipado com *Python*.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para, por exemplo, elaborar os seguintes algoritimos:

- Como escolher laranjas?
- Como efetuar a multiplicação de dois inteiros?
- Como adicionar um contato na agenda?
- Como fotografar e publicar a foto de uma paisagem?

Compreensão esta avaliada por meio da capacidade de solucionar as questões propostas pelo professor no tempo decorrido para a resolução e a objetividade do algoritimo dos alunos.

Referências

[1] BANIN, S. L. **Python 3 Conceitos e Aplicações: uma abordagem didática**. São Paulo: Érica, 2018.

Universidade Federal do Ceará

Plano de Aula 03

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Lançamentos oblíquos
Pré-Requisitos:	Conceitos básicos de lançamentos oblíquos
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Entender a dedução e conseguir utilizar equações diretas para grandezas como tempo de voo, alcance e altura máxima do lançamento, grandezas estas consideradas para um lançamento em que o chão é plano e perpendicular ao vetor aceleração gravitacional. Visualizar e compreender o ângulo de 45° como ângulo de maior alcance.

1.2 Específicos

- Equacionar o tempo de voo;
- Equacionar o alcance de um lançamento;
- Equacionar a altura máxima de um lançamento;
- Conseguir utilizar as equações anteriores para resolução de questões.

2 Conteúdos

- Tempo de voo;
- Altura do lançamento;
- Alcance do lançamento.

3 Procedimentos metodológicos

Apresentação expositiva do conteúdo, aliada a exemplificação e apresentação de simuladores prontos produzidos pelo próprio professor ou em acesso à laboratórios virtuais como o Phet Interactive Simulations.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;
- Computador equipado com *Python* e/ou simuladores.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para, por exemplo, resolver questões avançadas do conteúdo de lançamento oblíquo e revisitar questões já resolvidas para resolvê-las de forma mais rápida. Compreensão esta avaliada por meio da capacidade de resolver as questões propostas pelo professor, do tempo decorrido para a resolução e das dúvidas levantadas pelos alunos.

Referências

[1]DOCA, R. H.; BISCUOLA, G. J.; BOAS, N. V. **Tópicos de Física**. São Paulo: Saraiva, 2012.

[2] NUSSENZVEIG, H. Moysés. **Curso de Física Básica 1**. São Paulo: Bulcher, 2011.

Universidade Federal do Ceará

Plano de Aula 04

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Conceitos de algoritmização
Pré-Requisitos:	O que é a computação
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Elaborar algoritmos para solucionar problemas simples. Compreender a utilização das funções de interação do código com o usuário, isto é, as funções *input()* e *print()*, assim como funções de conversão de tipo de variável, tais como *int()*, *float()* e *str()*. Conseguir manejar *strings*.

1.2 Específicos

- Conseguir construir algoritmos simples;
- Conseguir traduzir esses algoritmos para a linguagem *Python*;
- Entender como usar *input()* e *print()*;
- Entender como e quando usar *int()*, *float()* e *str()*;
- Trabalhar com *strings*.

2 Conteúdos

- Algoritmização;
- Funções de entrada;
- Funções de saída;
- Funções de tipos de variáveis;
- Funções de *strings*.

3 Procedimentos metodológicos

Apresentação expositiva do conteúdo, aliada a exemplificação e apresentação de códigos prontos e produzidos durante a aula.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;
- Computador equipado com *Python*.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para, por exemplo, elaborar *scripts* que solucionem os seguintes problemas:

- Receba os dados do usuário e o cumprimente;
- Receba dois valores numéricos e faça a soma deles;
- Converta uma temperatura em °C para °F;
- Receba a velocidade média de um móvel e calcule o tempo de uma viagem de $n$km.

Compreensão esta avaliada por meio da capacidade de solucionar as questões propostas pelo professor, do tempo decorrido para a resolução e da objetividade do algoritmo dos alunos.

Referências

[1] BANIN, S. L. **Python 3 Conceitos e Aplicações: uma abordagem didática**. São Paulo: Érica, 2018.

Universidade Federal do Ceará

Plano de Aula 05

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Lançamentos oblíquos
Pré-Requisitos:	Conceitos básicos de lançamentos oblíquos
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Compreender o motivo de um lançamento vertical, visualizar sua simetria e as demais relações úteis para a resolução de questões do tema de modo mais ágil.

1.2 Específicos

- Entender a relação entre altura máxima e tempo de voo;
- Visualizar o alcance de ângulos complementares;
- Visualizar a simetria do lançamento oblíquo;
- Conseguir usar as relações anteriores para a resolução de questões mais complexas.

2 Conteúdos

- Simetrias no lançamento oblíquo;
- Tempo de voo;
- Altura do lançamento;
- Alcance do lançamento.

3 Procedimentos metodológicos

Apresentação expositiva do conteúdo, aliada a exemplificação e apresentação de simuladores prontos.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;
- Computador equipado com *Python* e/ou simuladores.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para, por exemplo, resolver questões avançadas do conteúdo de lançamento oblíquo e revisitar questões já resolvidas para resolvê-las de forma mais rápida. Compreensão esta avaliada por meio da capacidade de resolver as questões propostas pelo professor, do tempo decorrido para a resolução e das dúvidas levantadas pelos alunos.

Referências

[1]DOCA, R. H.; BISCUOLA, G. J.; BOAS, N. V. **Tópicos de Física**. São Paulo: Saraiva, 2012.

[2] NUSSENZVEIG, H. Moysés. **Curso de Física Básica 1**. São Paulo: Bulcher, 2011.

Universidade Federal do Ceará

Plano de Aula 06

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Estrutura condicional
Pré-Requisitos:	Paradigma estrutural em linguagem *Python*
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Elaborar algoritmos para solucionar problemas que necessitem de decisões. Compreender a sintaxe das estruturas *if*, *else* e *elif*. Conseguir entender aplicações das estruturas condicionais no cotidiano. Entender os operadores lógicos.

1.2 Específicos

- Conseguir construir algoritmos com estruturas condicionais;
- Conseguir traduzir esses algoritmos para a linguagem *Python*;
- Dominar o uso de operadores lógicos.

2 Conteúdos

- Algoritmização;
- Operadores lógicos;
- *if: ... else: ....*

3 Procedimentos metodológicos

Apresentação expositiva do conteúdo, aliada a exemplificação e apresentação de códigos prontos e produzidos durante a aula.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;
- Computador equipado com *Python*.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para, por exemplo, elaborar *scripts* que solucionem os seguintes problemas:

- Receba dois reais e realize uma divisão sem erros;
- Receba as notas de um aluno e diga sua situação em relação a aprovação ou reprovação;
- Converta uma temperatura de °C para °F ou de °F para °C num único *script*.

Compreensão esta avaliada por meio da capacidade de solucionar as questões propostas pelo professor, do tempo decorrido para a resolução e da objetividade do algoritmo dos alunos.

Referências

[1] BANIN, S. L. **Python 3 Conceitos e Aplicações: uma abordagem didática**. São Paulo: Érica, 2018.

Universidade Federal do Ceará

Plano de Aula 07

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Lançamentos oblíquos
Pré-Requisitos:	Lançamentos oblíquos
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Validar o entendimento sobre o tema. Resolver questões variadas sobre o assunto a partir de diferentes métodos.

1.2 Específicos

- Confirmar a consolidação do conteúdo;
- Resolver uma quantidade adequada de questões;
- Sanar dúvidas remanescentes.

2 Conteúdos

- Decomposição do vetor velocidade;
- Composição de movimento;
- Lançamento oblíquo;
- Simetrias no lançamento oblíquo;
- Tempo de voo;
- Altura do lançamento;
- Alcance do lançamento.

3 Procedimentos metodológicos

Apresentação de questões, aliada à resolução destas pelo professor como ferramenta de tira-dúvidas. Orientação de métodos para resolver questões.

4 Recursos didáticos

- Pincel e quadro;

- Projetor e anteparo;
- Computador equipado com *Python* e/ou simuladores;
- Material de questões.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para, por exemplo, resolver questões diversas do conteúdo de lançamento oblíquo. Compreensão esta avaliada por meio da capacidade de resolver questões propostas pelo professor, do tempo decorrido para a resolução e das dúvidas levantadas pelos alunos. Além disso, um teste com 4 ou 5 questões deve ser elaborado para realização dos alunos de modo individual e sem consulta, com o objetivo de avaliar de forma quantitativa o rendimento do processo de ensino e aprendizagem.

Referências

[1]DOCA, R. H.; BISCUOLA, G. J.; BOAS, N. V. **Tópicos de Física**. São Paulo: Saraiva, 2012.

[2] NUSSENZVEIG, H. Moysés. **Curso de Física Básica 1**. São Paulo: Bulcher, 2011.

Universidade Federal do Ceará

Plano de Aula 08

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Estruturas de repetição
Pré-Requisitos:	Paradigma estrutural em linguagem *Python* e estruturas condicionais
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Elaborar algoritmos para solucionar problemas que envolvam decisões e repetições. Compreender a diferença entre as estruturas *for: ...* e *while: ...*, assim como o uso de *arrays* nessas estruturas. Conseguir entender aplicações das estruturas de repetição no cotidiano.

1.2 Específicos

- Conseguir construir algoritmos com estruturas de repetição;
- Conseguir traduzir esses algoritmos para a linguagem *Python*;
- Dominar o uso de operadores lógicos e condições durante as repetições.

2 Conteúdos

- Algoritmização;
- Operadores lógicos;
- *Array*;
- *for: ...* e *while: ...* .

3 Procedimentos metodológicos

Apresentação expositiva do conteúdo, aliada a exemplificação e apresentação de códigos prontos e produzidos durante a aula.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;
- Computador equipado com *Python*.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para, por exemplo, elaborar *scripts* que solucionem os seguintes problemas:

- Receba dois inteiros e multiplique-os sem uso do operador $*$;
- Receba dois inteiros e eleve um ao outro sem o uso do operador $**$;
- Calcule e exponha ao usuário os n primeiros elementos de uma PA, dados o termo inicial e a razão;
- Calcule as sucessivas posições de um corpo dado um intervalo de tempo e uma velocidade constante;
- Calcule os sucessivos valores de velocidade de um corpo dado um intervalo de tempo e uma aceleração constante.

Compreensão esta avaliada por meio da capacidade de solucionar as questões propostas pelo professor, do tempo decorrido para a resolução e da objetividade do algoritmo dos alunos.

Referências

[1] BANIN, S. L. **Python 3 Conceitos e Aplicações: uma abordagem didática**. São Paulo: Érica, 2018.

Universidade Federal do Ceará

Plano de Aula 09

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Paradigma estrutural em linguagem *Python* e estruturas condicionais
Pré-Requisitos:	Paradigma estrutural em linguagem *Python* e estruturas condicionais
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Validar o próprio entendimento do tema. Resolver variadas questões do assunto a partir de diferentes métodos.

1.2 Específicos

- Confirmar a consolidação do conteúdo;
- Resolver uma quantidade adequada de questões;
- Sanar dúvidas remanescentes.

2 Conteúdos

- Algoritmização;
- Operadores lógicos;
- Estrutura condicional;
- Estrutura de repetição.

3 Procedimentos metodológicos

Apresentação expositiva de questões, aliada à resolução delas pelo professor como ferramenta de tira-dúvidas. Orientação de métodos para resolver questões.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;

- Computador equipado com *Python*;
- Material de questões.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para, por exemplo, resolver questões diversas do conteúdo de lançamento oblíquo. Compreensão esta avaliada por meio da capacidade de resolver as questões propostas pelo professor, do tempo decorrido para a resolução e das dúvidas levantadas pelos alunos. Além disso, um teste com 4 ou 5 questões deve ser elaborado para realização dos alunos de modo individual e sem consulta, com o objetivo de avaliar de forma quantitativa o rendimento do processo de ensino e aprendizagem.

Referências

[1] BANIN, S. L. **Python 3 Conceitos e Aplicações: uma abordagem didática**. São Paulo: Érica, 2018.

Universidade Federal do Ceará

Plano de Aula 10

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Funcionalidades do simulador
Pré-Requisitos:	Conceitos básicos de programação e lançamentos oblíquos
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Compreender o funcionamento do simulador, assim como a implementação das funções de movimento no simulador. Compreender quais os cálculos envolvidos na simulação a partir da leitura dos códigos do professor.

1.2 Específicos

- Compreender como o simulador faz seus cálculos;
- Conseguir elaborar algoritmos de implementação de movimento para o simulador;
- Conseguir traduzir esses algoritmos para a linguagem *Python*.

2 Conteúdos

- Algoritmização;
- Lançamentos oblíquos.

3 Procedimentos metodológicos

Apresentação expositiva do simulador, aliada a exemplificação nele e apresentação de códigos prontos produzidos durante a aula.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;
- Computador equipado com *Python*.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para elaborar códigos compatíveis com o simulador e que simulem corretamente o processo físico trabalhado. Compreensão esta avaliada por meio do tempo decorrido para a resolução e da objetividade do algorítimo dos alunos.

Referências

[1]DOCA, R. H.; BISCUOLA, G. J.; BOAS, N. V. **Tópicos de Física**. São Paulo: Saraiva, 2012.

[2]BANIN, S. L. **Python 3 Conceitos e Aplicações: uma abordagem didática**. São Paulo: Érica, 2018.

[3] NUSSENZVEIG, H. Moysés. **Curso de Física Básica 1**. São Paulo: Bulcher, 2011.

Universidade Federal do Ceará

Plano de Aula 11

Dados de Identificação	
Professor:	Víctor Citó Rodrigues
Disciplina:	Física/Computação
Tema:	Testes com o simulador
Pré-Requisitos:	Conceitos básicos de programação e lançamentos oblíquos
Data:	A ser definida
Duração da aula:	100 minutos

1 Objetivos

1.1 Geral

Comparar os resultados de lançamentos reais e tratados com o *software* Tracker com os resultados do simulador. Verificar os resultados do simulador com os resultados esperados em equações já tratadas.

1.2 Específicos

- Tratar as filmagens de lançamentos;
- Simular os lançamentos reais no simulador;
- Comparar os resultados reais, esperados e simulados.

2 Conteúdos

- Programação;
- Lançamentos oblíquos.

3 Procedimentos metodológicos

Utilização do simulador e do *software* Tracker. Apresentação expositiva de equações esperadas.

4 Recursos didáticos

- Pincel e quadro;
- Projetor e anteparo;
- Computador equipado com *Python* e Tracker.

5 Avaliação

Os alunos deverão demonstrar uma compreensão suficiente para comparar os diferentes resultados obtidos durante a simulação.

Referências

[1]DOCA, R. H.; BISCUOLA, G. J.; BOAS, N. V. **Tópicos de Física**. São Paulo: Saraiva, 2012.

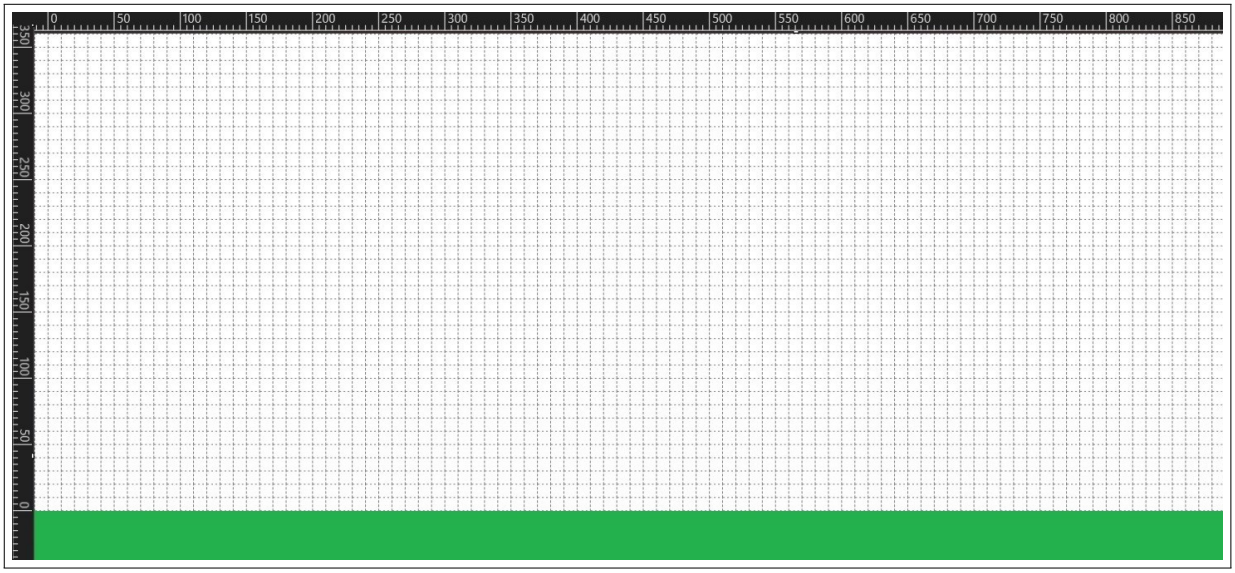
[2]BANIN, S. L. **Python 3 Conceitos e Aplicações: uma abordagem didática**. São Paulo: Érica, 2018.

[3] NUSSENZVEIG, H. Moysés. **Curso de Física Básica 1**. São Paulo: Bulcher, 2011.

APÊNDICE B – JANELA DO SIMULADOR

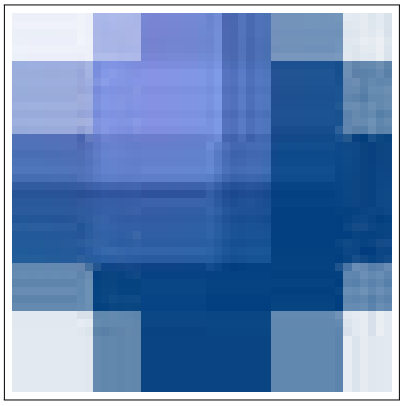
O material abaixo contém os *sprites* utilizados pelo simulador, bem como o código usado para mostrar ao usuário a simulação realizada. A imagem do plano de fundo possui 1458 x 659 *pixels* enquanto a Imagem do projétil possui 15 x 15 *pixels*, o que, quando comparado ao plano de fundo, dá a impressão de um círculo.

Plano de fundo do simulador com grades e réguas.



Fonte: elaborado pelo autor.

Projétil que se moverá ao longo do plano de fundo.



Fonte: elaborado pelo autor.

```
import pygame
from pygame.locals import * # Pygame será a biblioteca utilizada para animar o movimento
from Projectile import Projectile # Importa a classe desenvolvida pelo professor
from math import radians, ceil # Importa cálculos necessários

def show_ratio(x, y, sc): # Função gráfica para expor a razão escolhida
    ran_text = font.render(f'100px = {sc}m', True, (0, 0, 0))
    screen.blit(bgSurface, (x, y))
    screen.blit(ran_text, (x, y))

def show_velocity(x, y, sc): # Função gráfica para expor a velocidade inicial do projétil
    ran_text = font.render(f'Velocidade: {sc}m/s', True, (0, 0, 0))
    screen.blit(bgSurface, (x, y))
    screen.blit(ran_text, (x, y))

def show_angle(x, y, sc): # Função gráfica para expor o ângulo de lançamento do projétil
    ran_text = font.render(f'Ângulo: {sc}°', True, (0, 0, 0))
    screen.blit(bgSurface, (x, y))
    screen.blit(ran_text, (x, y))

projectile_velocity = 50 # Indique o valor da velocidade inicial do lançamento em unidades do SI
launched_angle = 40 # Indique o ângulo utilizado em graus

projectile = Projectile(projectile_velocity, radians(launched_angle), 0.3) # Cria uma instância da classe Projectile que será utilizado no simulador
fligth_time = -2 * projectile.y_velocity / projectile.graivity_acceleration # Obtém o tempo de voo a partir de cálculo conhecido da cinemática
times_shown = 30 # Determina a quantidade de imagens que será vista no resultado final
X_MAX_PIXEL_PG = 1300 # Essas duas constantes estão atreladas ao formado com que o Pygame mostra imagens na tela
Y_MAX_PIXEL_PG = 550 # para nós serve durante o cálculo de proporcionalidade

coordenadasSI = []
last_y = projectile.y_position
for i in range(ceil(Projectile.PARTITIONS_PER_SECOND * fligth_time) + 1): # Obtém as coordenadas, em unidades do SI, que serão mostradas graficamente
    show = Projectile.PARTITIONS_PER_SECOND * fligth_time // times_shown
    if i % show == 0:
        t=i/Projectile.PARTITIONS_PER_SECOND
        coordenadasSI.append([projectile.x_position, projectile.y_position])
        print(coordenadasSI)
        if projectile.y_position > last_y:
```

```
39                  h_max = last_y
40              last_y = projectile.y_position
41          projectile.move()
42
43      if projectile.x_position != 0: # Obtém as possíveis razões entre pixel / m a
        serem utilizadas com o objetivo de ocupar da melhor forma a tela do
        simulador
44          x_ratio = X_MAX_PIXEL_PG / projectile.x_position
45          y_ratio = Y_MAX_PIXEL_PG / h_max
46      y_ratio = Y_MAX_PIXEL_PG / h_max
47
48      selected_ratio = x_ratio
49      if x_ratio < y_ratio: # Escolhe a melhor razão dentre as possíveis
50          selected_ratio = x_ratio
51          display_ratio = 850 / projectile.x_position
52      else:
53          selected_ratio = y_ratio
54          display_ratio = 350 / h_max
55
56      coordenadasPixel = []
57      for i in coordenadasSI:
58          coordenadasPixel.append([i[0] * selected_ratio, i[1] * selected_ratio])
        # Converte as coordenadas antes em unidades do SI para pixel
59
60      pygame.init() # Inicia a janela que suportará o simulador
61      font = pygame.font.Font('freesansbold.ttf', 16)
62      bgSurface = pygame.Surface((165, 20))
63      bgSurface.fill((255, 255, 255))
64      DISPLAY_LENGTH = 1458
65      DISPLAY_WIDTH = 659
66      screen = pygame.display.set_mode((DISPLAY_LENGTH, DISPLAY_WIDTH))
67      backgroundImage = pygame.image.load('background.jpg')
68      ballImage = pygame.image.load('ball.png')
69      clock = pygame.time.Clock()
70      screen.blit(backgroundImage, (0, 0))
71      show_ratio(45, 30, f'{100 / display_ratio:.2f}')
72      show_velocity(45, 50, f'{projectile_velocity:.2f}')
73      show_angle(45, 70, f'{launched_angle:.2f}')
74      pygame.display.update() # Conclui a configuração da janela e atualiza-a
75
76      for coordinate in coordenadasPixel: # Desenha na tela o projétil em cada uma
        das posições em uma taxa de 24 quadros por segundo
77          clock.tick(24)
78          tupleCoordinate = (int(f'{coordinate[0] + 42:.0f}'), int(f'{-
        coordinate[1] + 584:.0f}'))
79          screen.blit(ballImage, tupleCoordinate)
80          pygame.display.update()
81
82      while True: # Mantém a janela aberta até que seja reiniciada pelo usuário
```

```
83          for event in pygame.event.get():
84              if event.type == QUIT:
85                  pygame.quit()
86                  Break
87
```

APÊNDICE C – CLASSE PROJECTILE

O código abaixo contém a definição da classe *Projectile*, utilizada para simplificar a interação do aluno com o simulador

```
from math import sin, cos
import moves # Importa o código de movimentação do aluno

class Projectile:
    graivity_acceleration   =   -10   #   Define   a   aceleração   gravitacional
    PARTITIONS_PER_SECOND = 200 # Define a quantidade de cálculos feito
pelo simulador a cada um segundo
    delta_time = 1/PARTITIONS_PER_SECOND # Define o intervalo de tempo
entre cada posição

    def____init__(self,  velocity:  float,  angle:  float,  air_drag_constant:
float=0, x_position: int=0, y_position: int=0) -> None:
        """velocity in m/s, angle in radians, air_drag_constant in m/s,
x_position and y_position in meters"""
        # run validation
        assert velocity >= 0, f"Invalid velocity value"
        assert angle >= 0, f"Invalid angle value"
        assert air_drag_constant >= 0, f"Invalid air drag constant value"

        # assign do self
        self.x_velocity  =  velocity  *  cos(angle)
        self.y_velocity  =  velocity  *  sin(angle)
        self.air_drag_constant = air_drag_constant
        self.x_position        =        x_position
        self.y_position = y_position
        self.trajectory = [(self.x_position, self.y_position)]

def___str__(self) -> str:
return f"({self.x_position:.2f}, {self.y_position:.2f})"

    def      move_horizontally(self):
        moves.move_horizontally(self)

    def      move_vertically(self):
        moves.move_vertically(self)

    def              move(self):
        self.move_horizontally()
        self.move_vertically()
        self.trajectory.append((self.x_position, self.y_position))
```