UNIVERSIDADE FEDERALDO PARÁ - UFPA

CAMPUS DE ANANINDEUA - CANAN

FACULDADE DE FÍSICA - FACFIS

FABIO JOSÉ DE SOUSA SOBRINHO

INTEGRAÇÃO DE ANIMAÇÕES EM PYTHON NO ENSINO DE FÍSICA: UM ESTUDO SOBRE TECNOLOGIAS MULTIMÍDIA NO ENSINO MÉDIO

ANANINDEUA – PARÁ

12/2024

FABIO JOSÉ DE SOUSA SOBRINHO

INTEGRAÇÃO DE ANIMAÇÕES EM PYTHON NO ENSINO DE FÍSICA: UM ESTUDO SOBRE TECNOLOGIAS MULTIMÍDIA NO ENSINO MÉDIO

Trabalho de Conclusão de Curso, apresentado à Faculdade de Física, do Campus Universitário de Ananindeua da Universidade Federal do Pará, como requisito para a obtenção do grau de Licenciatura Plena em Física.

Orientador: Prof. Msc. Francisco das Chagas de Oliveira Cacela Filho

ANANINDEUA – PARÁ

12/2024

FABIO JOSÉ DE SOUSA SOBRINHO

INTEGRAÇÃO DE ANIMAÇÕES EM PYTHON NO ENSINO DE FÍSICA: UM ESTUDO SOBRE TECNOLOGIAS MULTIMÍDIA NO ENSINO MÉDIO

Trabalho de Conclusão de Curso orientado pelo Prof. Msc. Francisco das Chagas de Oliveira Cacela Filho, apresentado ao Curso de Licenciatura em Física da Universidade Federal do Pará, Campus Ananindeua, como requisito para obtenção de grau em Licenciatura em Física.

Data de apresentação: 04/12/2024.

Conceito: Excelente

BANCA EXAMINADORA

Prof. Msc. Francisco das Chagas de Oliveira Cacela Filho
Orientador – FACFIS/CANAN/UFPA

Profa. Dra. Shirsley Joany dos Santos da Silva
Examinadora Interna – FACFIS/CANAN/UFPA

Profa. Dr. Vicente Ferrer Pureza Aleixo
Examinadora Interna – FACFIS/CANAN/UFPA

FICHA CATALOGRÁFICA

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

S725i Sousa, Fabio José de Sousa Sobrinho.
INTEGRAÇÃO DE ANIMAÇÕES EM PYTHON NO ENSINO DE FÍSICA: UM ESTUDO SOBRE TECNOLOGIAS MULTIMÍDIA NO ENSINO MÉDIO / Fabio José de Sousa Sobrinho Sousa. — 2024.
47 f. : il. color.

Orientador(a): Prof. Me. Francisco das Chagas de Oliveira Cacela Filho Cacela
Trabalho de Conclusão (Graduação) - Universidade Federal do Pará, Campus Universitário de Ananindeua, Curso de Ciência e Tecnologia, Ananindeua, 2024.

1. Python. 2. Animações . 3. Ensino de Física . 4. TICS. 5. Manim. I. Título.

CDD 530.07

Dedico este trabalho à minha família e aos amigos que sempre me apoiaram nos momentos em que mais precisei.

AGRADECIMENTOS

Agradeço a Deus e a todos que contribuíram de alguma forma para a minha jornada: família, amigos e professores. Não tenho palavras para descrever esse sentimento e espero poder retribuí-los algum dia. Assim, deixo uma grande frase para se inspirar na vida: "O trabalho duro vence o talento natural quando o talento natural não trabalha duro."

" Se eu vi mais longe, foi porque estava
sobre os ombros de gigantes."
(Isaac Newton)

RESUMO

Este trabalho apresenta uma proposta didática para o ensino de Física no ensino médio, utilizando animações desenvolvidas por meio da biblioteca *Manim*, da linguagem de programação *Python*. O desafio central do estudo reside na dificuldade dos alunos em visualizar e compreender conceitos abstratos de Física utilizando métodos tradicionais, especialmente em um contexto onde há pouca integração de tecnologias digitais nas salas de aula. A pesquisa foi conduzida em duas turmas do 2º e 3º anos do Ensino Médio, com o total de 31 estudantes. O trabalho foi realizado no Colégio Aspecto, localizado em Ananindeua, Pará. A metodologia envolveu a aplicação de questionários para avaliar a percepção dos alunos sobre o uso da programação nas aulas de Física, comparando a eficácia de métodos de ensino tradicionais com o uso de animações interativas. As principais contribuições da integração das Tecnologias de Informação e Comunicação (TICs) no ensino de física incluem a introdução de animações como uma ferramenta pedagógica para facilitar a visualização dos conceitos físicos, promovendo maior engajamento e desenvolvimento cognitivo dos alunos. Os resultados indicaram que as animações foram essenciais para a melhoria da compreensão dos conteúdos, tornando as aulas mais dinâmicas e acessíveis. Como principal resultado, observou-se que os alunos relataram um aumento no interesse e no engajamento com o uso da programação, destacando o papel das animações na facilitação do aprendizado de conceitos complexos em Física.

Palavras-chaves: Ensino de Física; Programação; Python; Animações; TICs.

ABSTRACT

This work presents an didactic proposal for teaching Physics in high school, utilizing animations developed through the Manim library in the Python programming language. The central challenge of the study lies in the difficulty students face in visualizing and understanding abstract concepts in Physics using traditional methods, especially in a context where there is little integration of digital technologies in classrooms. The research was conducted in two classes of 2nd and 3rd-year high school students, totaling 31 learners, at Colégio Aspecto, located in Ananindeua, Pará. The methodology involved administering questionnaires to assess students' perceptions of using programming in Physics classes, comparing the effectiveness of traditional teaching methods with the use of interactive animations. The main contributions of integrating Information and Communication Technologies (ICT) in teaching Physics include the introduction of animations as a pedagogical tool to facilitate the visualization of physical concepts, promoting greater student engagement and cognitive development. The results indicated that animations were essential for improving content comprehension, making lessons more dynamic and accessible. As a key outcome, students reported an increase in interest and engagement with the use of programming, highlighting the role of animations in facilitating the learning of complex concepts in Physics.

Keywords: Physics Teaching; Programming; Python; Animations; ICTs.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

BNCC	Base Nacional Comum Curricular
PCNs	Parâmetros Curriculares Nacionais
MANIM	Mathematical Animation Engine

SUMÁRIO

1 INTRODUÇÃO

A assimilação de conceitos de Física no ensino médio é frequentemente desafiadora para os alunos devido à complexidade da matéria e à falta de recursos interativos. Atualmente, o ensino de Física nas escolas enfrenta desafios como a redução da carga horária e a ausência de aulas práticas, o que dificulta o aprendizado interdisciplinar e prático (Moreira, 2018). Além disso, métodos tradicionais, somente aulas expositivas, limitam o engajamento e a compreensão dos alunos, uma vez que conceitos abstratos muitas vezes não são visualizados de maneira eficiente.

A linguagem de programação Python surge como uma ferramenta promissora para o ensino de Ciências, proporcionando interatividade e flexibilidade na criação de recursos didáticos. A proposta deste trabalho utiliza Biblioteca *Manim*, desenvolvida na linguagem Python, para criar animações que facilitam a visualização de conceitos físicos complexos, como energia e movimento, ampliando a capacidade dos alunos de aplicar e compreender os conteúdos. De acordo com Campos e Souza Junior (2020), o Manim é uma ferramenta poderosa e acessível que, com sua sintaxe amigável e flexível em Python, permite a criação de animações personalizadas, sendo especialmente útil para representar visualmente conceitos matemáticos complexos. Sua capacidade de gerar gráficos de alta qualidade e animações suaves faz com que seja uma excelente opção para educadores, pesquisadores e outros interessados em diversas áreas das ciências. Coluci (2021) propõe o uso do *Manim* para criar animações voltadas a conteúdos de Matemática do ensino médio, com o objetivo de aumentar o engajamento dos alunos em Ciências Exatas. Além disso, estão sendo desenvolvidos cursos de extensão para capacitar professores no uso dessas animações como ferramentas de apoio à aprendizagem. Segundo Lopes *et al*., (2023), um projeto voltado à formação de professores e futuros professores de Matemática teve como foco principal a manipulação do software GeoGebra e sua aplicação na modelagem e resolução de problemas matemáticos do Ensino Médio

A linguagem de programação Python destaca-se por sua sintaxe simples, que facilita o desenvolvimento de habilidades computacionais e lógicas, permitindo que os alunos se concentrem na estrutura dos problemas ao invés de detalhes técnicos complexos (Borges, 2014; Colpo, 2015). Essa proposta pretende promover o avanço do ensino de Física ao integrar a programação como uma ferramenta didática prática e visual, como tem sido

sugerido por Da Silva Marcolino & Laranjeira (2021), que apontam o uso de Python como eficaz para introduzir conceitos básicos de algoritmos e lógica.

Como principal objetivo deste estudo, pretende-se desenvolver e implementar animações interativas como ferramenta didática para facilitar o aprendizado dos alunos, promovendo uma construção do conhecimento de forma dinâmica, acessível e estimulante.

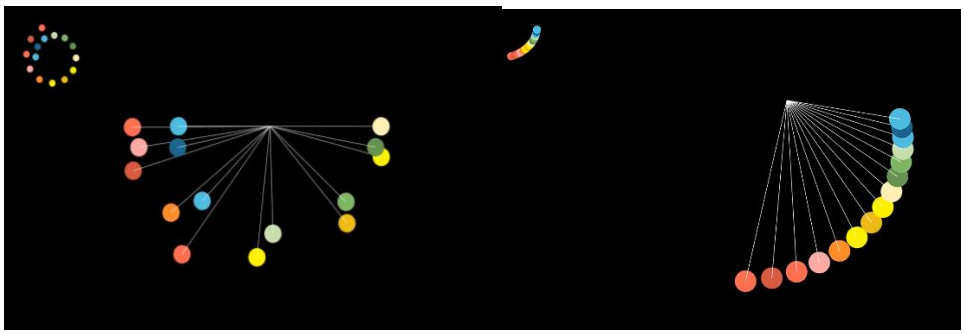
2 MATERIAIS E MÉTODOS

O estudo foi realizado em duas turmas do 2º e 3º anos do Ensino Médio, na escola "Colégio Aspecto", localizada em Ananindeua, Pará, com a participação de 31 estudantes. A principal metodologia consistiu no uso de animações desenvolvidas com a biblioteca *Manim* para complementar as aulas expositivas de Física.

As aulas foram divididas em dois blocos: *i*) o primeiro bloco seguiu a metodologia expositiva tradicional, e *ii*) no segundo bloco foram empregadas animações programadas em Python, que exploraram conceitos como oscilação e movimento. Essa abordagem está alinhada com a Base Nacional Comum Curricular (BNCC) e as competências e habilidades EM13CNT103, que recomendam a caracterização de ondas mecânicas por meio dos conceitos de amplitude, comprimento de onda, frequência, velocidade de propagação e ressonância (Brasil, 2018).

Durante as aulas práticas, foram realizadas simulações que incluíram a criação de animações como "A Dança dos Pêndulos" (ver Figura 1) e a solução de “Problemas de Geometria” (ver Figura 2). O uso dessas animações como ferramenta pedagógica foi inspirado nos estudos de Jesus & Barros (2014), que ressaltam a eficácia das simulações visuais no engajamento dos alunos e na melhoria da compreensão de fenômenos físicos complexos.

Figura 1: Animação a “Dança dos pêndulos”

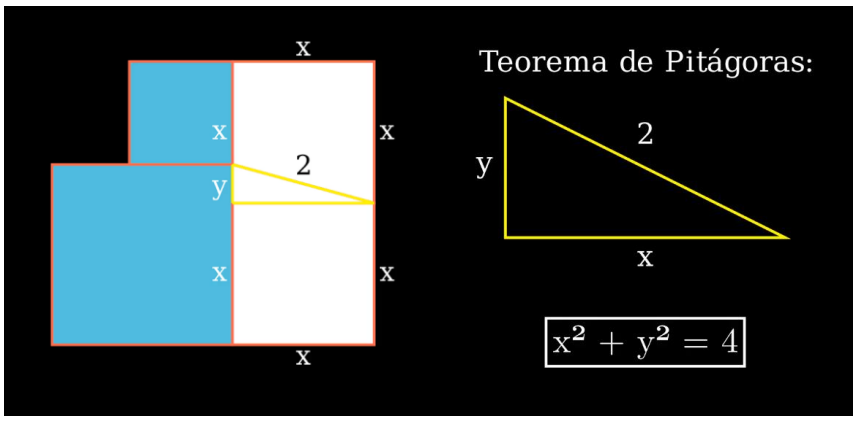


Fonte: Dos próprios autores.

A animação “Dança dos Pêndulos”, mostrada na Figura 1, ilustra a formação de padrões harmônicos de pêndulos simples através do movimento sincronizado de vários pêndulos com comprimentos diferentes. Cada pêndulo oscila em seu próprio ritmo, mas, ao serem alinhados, eles criam uma sequência visual semelhante a uma "dança". O comprimento

de cada pêndulo determina sua frequência, e, conforme o tempo passa, eles entram e saem de fase, gerando padrões que mudam constantemente. Esse fenômeno é um exemplo de como variações sutis nas condições iniciais podem resultar em uma oscilação complexa, porém harmônica.

Figura 2: “Problema de geometria”



Fonte: Dos próprios autores.

A animação “Problema de geometria”, mostrado na figura 2, determina a soma das áreas de dois quadrados azuis em uma figura composta por quatro quadrados, onde a medida de um segmento destacado em vermelho, representando a hipotenusa de um triângulo retângulo, era conhecida. A solução foi desenvolvida a partir das dimensões dos catetos do triângulo e aplicando o Teorema de Pitágoras. A expressão obtida, que relaciona as áreas dos quadrados azuis com as variáveis X e Y dos lados dos quadrados brancos, levou ao resultado final. Essa animação ilustrou como o *Manim* pode ser uma ferramenta poderosa para visualização e resolução de problemas matemáticos, contribuindo para uma compreensão mais clara e intuitiva de conceitos geométricos.

As atividades foram realizadas em sala de aula, permitindo que os alunos participassem ativamente, fizessem perguntas e esclarecessem dúvidas sobre o funcionamento e a criação das animações. Além disso, os estudantes puderam visualizar as animações tanto em seus celulares quanto em tablets, proporcionando uma experiência mais interativa e acessível para todos.7

3 RESULTADOS E DISCUSSÕES

Os resultados da coleta de dados da pesquisa quantitativa do tipo, descritiva-exploratória indicaram que, a maioria dos alunos que participou da pesquisa afirmaram que as aulas com animações foram mais dinâmicas e proporcionaram uma melhor compreensão dos tópicos abordados, como o movimento oscilatório e as transformações de energia. A porcentagem de alunos que demonstraram maior interesse e engajamento com as aulas também aumentou consideravelmente após o uso das animações.

Comparando os resultados obtidos com a literatura, estudos como os de Souza & Pires (2012) corroboram a ideia de que ferramentas visuais aumentam o interesse dos alunos em disciplinas que envolvem conceitos abstratos, como a Física. Em comparação com métodos tradicionais, que não utilizam tecnologias interativas, o uso das Tecnologias de Informação e Comunicação (TICs) com animações proporcionou uma visualização mais concreta dos fenômenos, o que facilitou a retenção dos conceitos e estimulou o desenvolvimento de habilidades investigativas.

Em termos práticos, os alunos relataram que as animações os ajudaram a compreender conceitos que antes eram considerados difíceis de assimilar apenas com aulas teóricas-expositivas. Além disso, o uso de Python como ferramenta de criação de animações incentivou o desenvolvimento de novas competências computacionais, algo que contribui para a formação interdisciplinar dos estudantes. De acordo com Heckler *et al*., (2007):

> "As animações e simulações são consideradas, por muitos, a solução dos vários problemas que os professores de Física enfrentam ao tentar explicar para seus alunos fenômenos demasiado abstratos para serem ‘visualizados’ através de uma descrição em palavras, e demasiado complicados para serem representados através de uma única figura. Elas possibilitam observar em alguns minutos a evolução temporal de um fenômeno que levaria horas, dias ou anos em tempo real, além de permitir ao estudante repetir a observação sempre que o desejar" (Heckler *et al*., 2007, p. 267).

Os resultados obtidos com os questionários aplicados nas turmas demonstram o impacto das animações e das tecnologias educacionais na compreensão dos conceitos de Física pelos alunos. Esses dados foram coletados por meio dos questionários elaborados pelos autores da pesquisa, refletindo as experiências e percepções dos estudantes sobre o uso de animações nas aulas. Como mostrado no quadro 1 abaixo:

Quadro 1: Questionário aplicado em sala de aula.

Perguntas	Alternativas
Questão 01- Qual é o seu nível de conhecimento em Python?	Nenhum conhecimento - Básico - Intermediário - Avançado
Questão 02- Você já viu animações de física em suas aulas?	Sim- Não
Questão 03- Como você avaliaria o impacto das animações na sua compreensão dos conceitos de física?	Muito útil – Útil - Pouco útil - Nada útil
Questão 04- Quais conceitos de física você gostaria de ver mais animados?	Movimento e força - Energia e trabalho - Ondas e som- Eletromagnetismo- Outro: _____
Questão 05- Você se sentiu mais engajado durante as aulas que utilizaram animações?	Sim, muito mais - Sim, um pouco mais - Não, não mudou - - Não, fiquei menos engajado
Questão 06- Quais ferramentas ou plataformas você utilizou para visualizar animações de física? (Selecione todas as que se aplicam)	Simuladores online - Vídeos educativos - Softwares - específicos - Não usei nenhuma
Questão 07- Você estaria interessado em aprender a criar animações de física usando Python?	Sim - Talvez - Não
Questão 08- Qual é o seu método preferido de aprendizado?	Aulas expositivas - Animações e vídeos - Prática e experimentação - Leitura de textos
Questão 09- Você já experimentou usar alguma ferramenta de programação para entender física?	Sim - Não
Questão 10- Que recursos você considera mais importantes para ajudar no aprendizado de física? (Selecione todos os que se aplicam)	Animações - Aulas práticas - Materiais de leitura Discussões em grupo - Outro: _____

Fonte: Dos próprios autores.

A questão 01 demonstra o nível de conhecimento dos alunos sobre a linguagem Python, essencial para a criação de animações didáticas. Os resultados revelaram que a maioria dos alunos do 2º e 3º anos não têm conhecimento sobre Python. 83,33% dos alunos do 3º ano e 76,92% do 2º ano não possuem familiaridade com a linguagem. Uma pequena

parcela dos alunos tem conhecimento básico, o 2º ano apresentou 23,08% de conhecimento da linguagem, e o 3º apenas 11,11%. Somente 7,69% dos alunos do 2º ano possuem conhecimento intermediário, e nenhum aluno se classifica como avançado. Isso sugere a necessidade de fortalecer o uso das TICs e do ensino de Python, especialmente em níveis mais avançados. Ao discutir o uso da linguagem de programação Python no ensino de Física, é crucial considerar os desafios enfrentados, como a falta de recursos nas escolas, a curva de aprendizado e a necessidade de adaptações curriculares. Esses fatores, semelhantes às limitações apontadas por Moreira (2017) sobre as Tecnologias Digitais de Informação e Comunicação (TDIC), demandam atenção e investimento das instituições educacionais e dos formuladores de políticas públicas (Yamaguti, 2024).

Na questão 02 do questionário, abordamos a exposição prévia dos alunos a animações nas aulas de Física. A pesquisa revela que 60% dos alunos do 3º ano já tiveram contato com animações, enquanto 40% não tiveram essa experiência. No 2º ano, a situação é ainda mais marcante, com 93% dos alunos afirmando que não viram animações em aulas de Física. No total, 42% dos alunos assistiram a animações, enquanto 58% não tiveram essa oportunidade. Esses dados indicam que, apesar de um maior acesso a animações por parte dos alunos do 3º ano, a maioria dos estudantes ainda não foi exposta a essa ferramenta didática, sugerindo um uso limitado de animações nas aulas de Física.

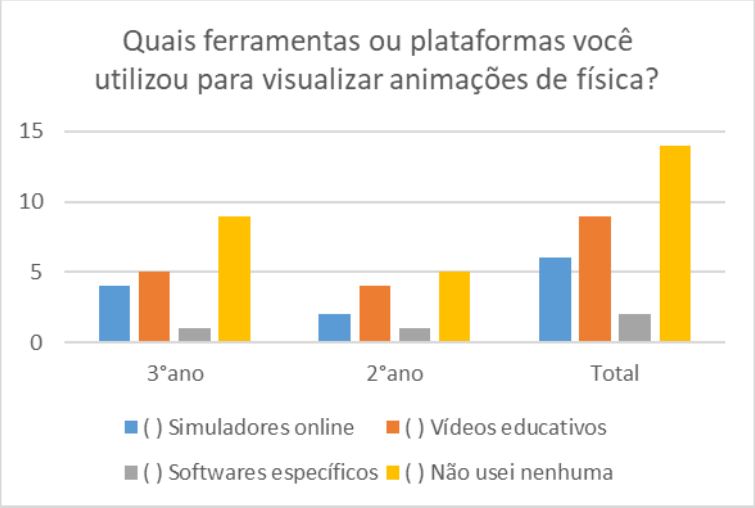
A questão 03 do questionário reflete como os alunos avaliaram o impacto das animações na compreensão dos conceitos de Física. A análise dos dados revela que a maioria dos alunos considera positivamente o impacto das animações na compreensão dos conceitos físicos. No 3º ano, 50% dos alunos consideram as animações "muito úteis", enquanto 30% dos alunos do 2º ano compartilham essa opinião. No total, 80% dos alunos acreditam que as animações são muito úteis. Por outro lado, 27% os consideram "úteis", e não houve nenhuma avaliação de "pouco útil" ou "nada útil".

A questão 04 explora o interesse dos alunos em aprender a criar animações de Física utilizando Python. Indicou-se um interesse significativo dos alunos em aprender a criar animações de física usando Python. No 3º ano, 40% dos alunos mostraram interesse, enquanto 67% no 2º ano expressaram vontade de aprender. No total, 56% dos alunos estão interessados, e 22% estão indecisos. Apenas 19% dos alunos não demonstraram interesse. Esses resultados sugerem uma boa oportunidade de explorar o aprendizado de animações como ferramenta educacional, especialmente entre os alunos do 2º ano.

Na questão 05, os dados revelam quais conceitos de Física os alunos gostariam de ver com animações. "Ondas e Som" foi o conceito mais mencionado, seguido por "Eletromagnetismo" e "Movimento e Força". O conceito mais popular entre os estudantes é "Ondas e Som", com 31,4% das preferências, enquanto "Outro" é o menos escolhido, com 5,7%. As demais preferências são distribuídas entre "Eletromagnetismo" (20%), "Movimento e Força" (14,3%), e "Energia e Trabalho" (11,4%).

A questão 06 ilustra o nível de engajamento dos alunos durante as aulas que utilizaram animações. Nesse contexto, engajamento refere-se ao grau de envolvimento, participação ativa e interesse demonstrado pelos alunos durante as atividades em sala de aula. Os dados revelam que a maioria dos estudantes do 2º e 3º anos se mostrou mais interessada nas aulas com esse recurso. Aproximadamente 45,5% dos alunos relataram sentir-se "muito mais" envolvidos, sendo 24,2% do 3º ano e 21,2% do 2º ano. Além disso, 33,3% dos alunos indicaram um aumento "um pouco mais" no entusiasmo, com a maioria deles pertencendo ao 3º ano (30,3%). Apenas 9,1% afirmaram que o nível de envolvimento "não mudou", e 6,1% relataram sentir-se "menos engajados". Esses dados indicam que as animações contribuíram de forma positiva para o engajamento dos alunos, incentivando uma participação mais ativa, maior interesse e melhor interação com os conteúdos, especialmente no 3º ano.

Figura 3: Questão 07.

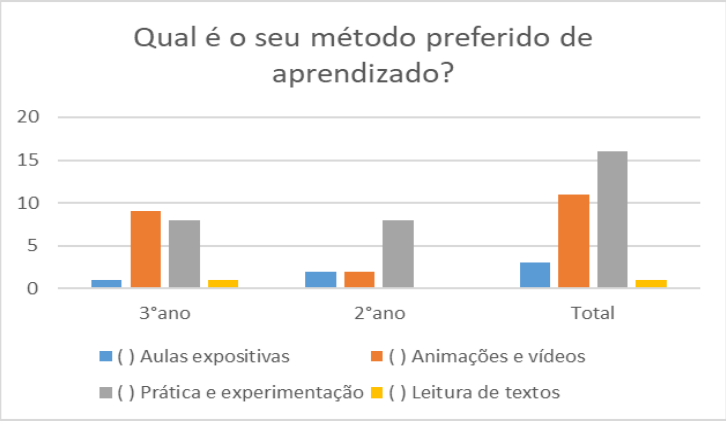


Fonte: Dos próprios autores.

A figura 3 (questão 07) apresenta as ferramentas ou plataformas que os alunos utilizaram para visualizar animações de Física. O gráfico acima revela as ferramentas ou

plataformas utilizadas pelos alunos do 2º e 3º anos para visualizar animações de física. A maioria dos estudantes (46,9%) afirmou não ter utilizado nenhuma ferramenta. Entre aqueles que usaram recursos, 25% optaram por vídeos educativos, sendo a segunda escolha mais comum. Simuladores online foram utilizados por 18,8% dos alunos, enquanto softwares específicos foram os menos usados, com apenas 6,3% de adesão. Esses dados sugerem que, embora existam opções de tecnologia disponíveis, quase metade dos alunos não aproveitou essas ferramentas para visualização de animações de física.

Figura 4: Questão 08.



Fonte: Dos próprios autores.

Na figura 4 (questão 08) são apresentados os métodos de aprendizado preferidos pelos alunos. Os dados demonstram que os métodos de aprendizado mais preferidos pelos alunos do 3º e 2º ano são animações e vídeos e prática e experimentação. No 3º ano, animações e vídeos lideram com 40%, seguidos por prática e experimentação (30%). No 2º ano, a prática é a favorita com 60%, e animações têm 20%. No total, prática e experimentação é o método mais escolhido (45%), seguido por animações e vídeos (35%). Aulas expositivas e leitura de textos são menos populares, com 15% e 5%, respectivamente.

A questão 09 revela quantos alunos utilizaram ferramentas de programação para aprender Física. A maioria dos alunos não utilizou ferramentas de programação para aprender física. No 3º ano, apenas 20% responderam "Sim" e 80% "Não". No 2º ano, a situação é ainda mais limitada, com apenas 5% tendo experimentado essas ferramentas. No total, cerca de 15% dos alunos usaram programação, enquanto 85% não a exploraram. Isso indica que o uso de programação no ensino de física é pouco aproveitado entre os alunos.

A questão 10 ilustra os recursos que os alunos consideram mais importantes para o aprendizado de Física. De acordo com os dados obtidos, no 3º ano, as animações e as aulas práticas se destacam, com cerca de 10 alunos preferindo animações e 7 alunos escolhendo aulas práticas. No 2º ano, cerca de 4 alunos optaram por animações, e 4 alunos escolheram aulas práticas. Em geral, as aulas práticas são o recurso mais mencionado, com aproximadamente 14 alunos, seguidas por animações, que receberam cerca de 12 preferências. Outro recurso bem avaliado foi o uso de materiais de leitura, com cerca de 13 votos no total, embora essa preferência seja mais evidente no 3º ano. Discussões em grupo e a categoria "outro" receberam menos atenção, com apenas alguns votos em cada. Em resumo, os alunos consideram principalmente as aulas práticas e as animações como os recursos mais importantes para o aprendizado de Física, sendo que os materiais de leitura também recebem destaque, especialmente no 3º ano. De acordo com Wambua *et al*., (2020) animações de qualidade impactam positivamente o processo de ensino e aprendizagem, evidenciado pela melhoria nos resultados dos alunos. O estudo comparou as médias de desempenho entre grupos de controle e experimental, e a análise estatística revelou uma diferença significativa entre essas médias.

4 CONSIDERAÇÕES FINAIS

Este estudo evidenciou que a maioria dos estudantes do 2º e 3º anos do ensino médio não possui familiaridade com a linguagem de programação Python. O interesse em aprender a programar animações, no entanto, foi notável, especialmente entre os alunos do 2º ano, o que indica um potencial significativo para a implementação dessa ferramenta como recurso didático nas aulas de Ciências. Embora poucos alunos tenham tido contato prévio com animações em sala de aula, a maioria dos participantes reconheceu a importância dessa tecnologia para facilitar a compreensão dos conceitos de Física. As animações e vídeos foram identificados como métodos de ensino altamente eficazes, seguido da prática e experimentação, que foi se mostrou valorizada pelos alunos do 2º ano. Isso reforça combinar diferentes abordagens didáticas para atender às necessidades de aprendizado de diferentes perfis de estudantes é fundamental.

Esse fato sugere que a incorporação de tecnologias educacionais, como a programação em Python e o uso de animações, ainda enfrenta barreiras, tanto em termos de acesso, quanto de desenvolvimento de habilidades técnicas por parte dos alunos. O uso dessas ferramentas

pode facilitar a compreensão de conceitos abstratos, e promover o desenvolvimento de competências interdisciplinares, como pensamento computacional e resolução de problemas, essenciais para a formação dos alunos em um mundo cada vez mais digital e conectado.

5 REFERÊNCIAS

BRASIL. **Base Nacional Comum Curricular.** Brasília*: MEC*, 2018. Disponível em: http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf. Acesso em: 21 out. 2024.

BORGES, L. E. **Python para desenvolvedores: aborda Python 3.3.** Novatec Editora, 2014.

CAMPOS, Arthur Ferreira; DE SOUZA JUNIOR, Arlindo José. Conhecimento tecnológico com a matemática na produção de vídeos interativos. Revista BOEM, v. 12, n. 22, p. e0110-e0110.

COLPO, R. A. M.; FARIA, AU de; MACHADO, A. F. O ensino de física no ensino médio intermediado por programação em linguagem Python. **Encontro Nacional de Pesquisa em Educação em Ciências**, 2015.

COLUCI, VITOR R. Animações de conceitos da teoria de erros usando *Manim*/Python. **Revista Brasileira de Ensino de Física**, v. 44, p. e20210239, 2021.

DA SILVA MARCOLINO, J.; LARANJEIRA, D. R. O processo de ensino e aprendizagem em programação inicial de computadores com linguagem Python. **Revista Multidisciplinar Pey Këyo**,v. 7, n. 3, p. 19-24, 2021.

HECKLER, Valmir; SARAIVA, Maria de Fátima Oliveira; OLIVEIRA FILHO, Kepler de Souza. Uso de simuladores, imagens e animações como ferramentas auxiliares no ensino/aprendizagem de óptica. **Revista Brasileira de Ensino de Física**, v. 29, p. 267-273, 2007.

JESUS, V. L. B. de; BARROS, M. A. J. As múltiplas faces da dança dos pêndulos. **Revista Brasileira de Ensino de Física**, 2014.

LOPES, Érika Maria Chioca et al. Um passo adiante: Matemática e extensão em uma proposta para oficinas online utilizando o GeoGebra. **Revista Conexão UEPG**, v. 19, n. 1, p. 20, 2023.

MANIM COMMUNITY. ***Manim*** **Community**. Disponível em: https://www.manim.community/. Acesso em: 21 out. 2024.

MOREIRA, M. A. Uma análise crítica do ensino de Física. **Estudos avançados**, v. 32, n. 94, p. 73-80, 2018.

PIRES, M. A.; VEIT, E. A. Tecnologias de Informação e Comunicação para ampliar e motivar o aprendizado de Física no Ensino Médio. **Revista Brasileira de Ensino de Física**, v. 28, n. 2, p. 123-130, jun. 2006.

WAMBUA, Joseph M.; TWOLI, Nicholas W.; MAUNDU, John N. **The effect of animations in learning and performance of physics at secondary school level**. IOSR Journal of Research & Method in Education (IOSR-JRME), v. 10, n. 3, ser. VII, p. 25-32, maio-jun. 2020.23

YAMAGUTI, MATEUS XAVIER. Ensino de física potencializado pela aprendizagem da linguagem de programação python: **uma sequência didática baseada nos três momentos pedagógicos**. 2024.

APÊNDICE A: Carta de autorização para a realização da pesquisa no colégio Aspecto

Autorização para Realização de Pesquisa

À Direção do Colégio Aspecto,

Venho, por meio desta, solicitar autorização para a realização de uma pesquisa nas dependências do Colégio Aspecto, com o objetivo de desenvolver um estudo acadêmico relacionado ao aprimoramento do processo de ensino-aprendizagem. A pesquisa será conduzida conforme os princípios éticos e respeitará todas as normas estabelecidas pela instituição.

Informações sobre os responsáveis:
Discente: Fabio José de Sousa Sobrinho – Matricula: 202074140008
Orientador: Prof. Msc. Francisco de Chagas de Oliveira Cacela Filho

Comprometemo-nos a preservar a integridade dos participantes e a confidencialidade das informações obtidas. Os resultados da pesquisa serão disponibilizados à instituição, caso solicitado.

Agradecemos pela atenção e colocamo-nos à disposição para quaisquer esclarecimentos.

Atenciosamente,

Fabio José de Sousa Sobrinho

Assinatura do Discente

Francisco Cacela Filho.

Assinatura do Orientador

APÊNDICE B: Carta de autorização aceita sobre pesquisa realizada pelo colégio Aspecto.

Eu, DENIZE BENTES DA SILVA, Diretora Pedagógica do Colégio Aspecto, Venho, por meio desta, autorizar a realização de uma pesquisa nas dependências do Colégio Aspecto, com o objetivo de desenvolver um estudo acadêmico relacionado ao aprimoramento do processo de ensino-aprendizagem. O acadêmico se responsabilizará em conduzir sua pesquisa conforme os princípios éticos e respeitará todas as normas estabelecidas pela instituição.

Informações sobre os responsáveis:

Discente: Fabio José de Sousa Sobrinho – Matricula: 202074140008
Orientador: Prof. Msc. Francisco de Chagas de Oliveira Cacela Filho

Agradecemos pela atenção e colocamo-nos à disposição para quaisquer esclarecimentos.

Atenciosamente,

Denize Bentes da silva.
Gestora pedagógica!

Assinatura da Gestora
Denize Bentes da Silva
Gestora Pedagógica
Registro- 10867

APÊNDICE C: Certificado de Trabalho/apresentação no evento III Encontro de Ciências da Natureza no Marajó (Artigo)

Verifique o código de autenticidade **97177808.5510823.3.7.8488303114032668** em https://www.even3.com.br/documentos

ACADEMY

CERTIFICADO

Certificamos que o trabalho INTEGRAÇÃO DE ANIMAÇÕES EM PYTHON NO ENSINO DE FÍSICA: UM ESTUDO SOBRE TECNOLOGIAS MULTIMÍDIA NO ENSINO MÉDIO, dos autores Fabio José de Sousa Sobrinho, Shirsley Joany dos Santos da Silva, Lorena Gomes Corumbá e Francisco das Chagas de Oliveira Cacela Filho foi apresentado de forma oral no III Encontro de Ciências da Natureza no Marajó, e será publicado na modalidade de artigo científico.

29/10/2024 a 09/11/2024

Josiney Farias de Araújo
coordenador

CNPJ: 28.612.931/0001-60
Rod. Aug. Montenegro, N.º 4300 – Parque Verde – Belém/PA
Contatos: (91) 98900-7710 /e-mail: kpiacademy594@gmail.com

ANEXO A: ANIMAÇÕES NO MANIM PYTHON

TUTORIAL: CRIANDO E PLOTANDO ANIMAÇÕES COM MANIM NO REPLIT

O Manim (Mathematical Animation Engine) é uma biblioteca Python poderosa para criar animações matemáticas e visuais. Segue um tutorial para configurar e usar o Manim no Replit.

Passo 1: Configurando o Ambiente no Replit

1. Crie um novo projeto no Replit:
   - Vá para Replit (https://replit.com).
   - Escolha a linguagem Python.
2. Instale o Manim:
   - No terminal do Replit, instale as dependências necessárias:
   pip install manim
   - Certifique-se de que o Replit suporta bibliotecas pesadas como o Manim (em alguns casos, pode ser necessário ativar suporte adicional ou usar outra plataforma, como Google Colab).

Passo 2: Estrutura do Código para Animações

1. Crie um arquivo Python, por exemplo, main.py.
2. Escreva o seguinte código básico para gerar uma animação simples, no exemplo abaixo:

```
from manim import *

class CircleAnimation(Scene):
    def construct(self):
        # Cria um círculo
        circle = Circle(radius=2, color=BLUE)
        self.play(Create(circle)) # Animação de criação do círculo

        # Adiciona um movimento de rotação
        self.play(Rotate(circle, angle=PI)) # Roda o círculo em 180°
        self.play(FadeOut(circle)) # Faz o círculo desaparecer
```

Passo 3: Executando o Manim no Replit

1. No terminal, execute o seguinte comando para renderizar a animação:
   manim -pql main.py CircleAnimation
   - -pql: Renderiza a animação em qualidade baixa (rápido para testes).
   - Substitua CircleAnimation pelo nome da classe que você criou.

2. Após renderizar, o Replit exibirá um link para visualizar o vídeo gerado.

Passo 4: Melhorando sua Animação

- Adicione mais objetos geométricos (retângulos, polígonos, etc.).
- Experimente com textos e fórmulas matemáticas:
  text = Text('Hello, Manim!', color=GREEN)
  self.play(Write(text))
- Crie animações complexas combinando movimentos, transformações e aparições.

Dicas Finais

Manim requer boa configuração gráfica. Se o Replit não funcionar bem para renderizar vídeos, você pode migrar o projeto para outra plataforma, como Google Colab ou sua máquina local.
Consulte a documentação oficial do Manim (https://docs.manim.community/) para exemplos e tutoriais avançados.

ANEXO B: CÓDIGO PARA ANIMAÇÃO DE PÊNDULOS EM MANIM, EXPLICAÇÃO DETALHADA

Este documento explica detalhadamente o código que cria uma animação de pêndulos utilizando a biblioteca Manim. O objetivo do código é ilustrar o movimento de diferentes pêndulos e sua relação com conceitos físicos como período, amplitude e fase.

Código da Animação:

```
from manim import *
import math
import numpy as np

# Constante da gravidade
```

```
Gravidade = 9.81

class Video(MovingCameraScene):
    def construct(self):
        # Cria os pêndulos
        self.pendulos = self.criar_pendulos()
        # Configura a câmera e executa a animação
        self.play(self.camera.frame.animate.set(width=2.8).move_to([0, 1.75, 0]),
run_time=0.01)
        self.animar()

    def criar_pendulos(self):
        # Frequências dos ciclos por minuto, alterando os parametros do range, teremos aumento
na quantidade de pendulos oscilando.
        ciclos_por_minuto = [x for x in range(50, 70)]
        # Calcula os períodos dos pêndulos
        periodos = [60.0 / x for x in ciclos_por_minuto]
        # Calcula os comprimentos baseados nos períodos
        comprimentos = [(periodo / 2 / math.pi) ** 2 * Gravidade for periodo in periodos]
        # Cria uma lista de objetos Pendulo
        pendulos = [Pendulo(comprimento) for comprimento in comprimentos]
        return pendulos

    def animar(self):
        # Listas para armazenar elementos gráficos
        todos_os_pontos, todos_os_pontos_fase, todos_os_pontos_delta_fase = [], [], []

        # Cores para os pêndulos
        cores = [
            RED, RED_E, RED_C, RED_A, ORANGE, YELLOW, YELLOW_E, YELLOW_C,
            YELLOW_A, GREEN_E, GREEN_C, GREEN_A, BLUE, BLUE_E, BLUE_C,
            BLUE_A, PURPLE, PURPLE_E, PURPLE_C, PURPLE_A
        ]

        for i, pendulo in enumerate(self.pendulos):
            # Cria um ponto que representa o pêndulo, radius vai definir o tamanho da esfera do
pendulo.
            ponto = Circle(radius=0.02, color=cores[i % len(cores)]).move_to(pendulo.posicao)
            ponto.pendulo = pendulo
            todos_os_pontos.append(ponto)

            # Cria um círculo para a visualização da fase
            circulo_delta_fase = Circle(
```

```
            radius=pendulo.comprimento / 2.5, color="GREY", arc_center=[-1.0, 2.3, 0],
stroke_width=0.005
        )
        ponto_delta_fase = Circle(
            radius=0.01, color=cores[i % len(cores)], stroke_width=1.0
        ).set_fill(cores[i % len(cores)],
opacity=1.0).move_to(circulo_delta_fase.point_from_proportion(0.0))

        ponto_delta_fase.pendulo = pendulo
        ponto_delta_fase.circulo = circulo_delta_fase
        todos_os_pontos_delta_fase.append(ponto_delta_fase)

    # Define a taxa de atualização
    taxa = 2 / 3.0

    # Função para atualizar a posição do ponto do pêndulo
    def atualizar_posicao(mob, dt):
        mob.pendulo.atualizar_posicao(dt * taxa)
        mob.move_to(mob.pendulo.posicao)

    for ponto in todos_os_pontos:
        ponto.add_updater(atualizar_posicao)

    # Função para atualizar a linha entre o pêndulo e o ponto fixo
    def atualizar_linha(linha, dt):
        linha.put_start_and_end_on([0, 2, 0], todos_os_pontos[linha.indice].pendulo.posicao)

    for i, ponto in enumerate(todos_os_pontos):
        linha = Line(start=[0, 2, 0], end=ponto.pendulo.posicao, stroke_width=0.2)
        linha.indice = i
        linha.add_updater(atualizar_linha)
        self.add(ponto)
        self.add(linha)

    # Função para atualizar o ponto na visualização da fase
    def atualizar_ponto_fase(ponto, dt):
        ponto.move_to(ponto.circulo.point_from_proportion(ponto.pendulo.fase % 1))

    for ponto_delta_fase in todos_os_pontos_delta_fase:
        ponto_delta_fase.add_updater(atualizar_ponto_fase)
        self.add(ponto_delta_fase)

    # Espera enquanto os pêndulos oscilam
```

```
        self.wait(60. / taxa)

class Pendulo:
    def __init__(self, comprimento):
        # Parâmetros do pêndulo
        self.amplitude_maxima = math.radians(90)
        self.comprimento = comprimento
        self.escalonamento = 2.0
        self.deslocamento = 0.0
        self.deslocamento_posicao = np.array([0.0, 2.0, 0.0])
        self.posicao = np.array([0.0, 0.0, 0.0])
        self.tempo = 0
        dt = 0.0
        self.fase = 0

        # Atualiza a posição inicial
        self.atualizar_posicao(dt)

    def atualizar_posicao(self, dt):
        # Incrementa o tempo
        self.tempo += dt
        # Calcula o período do pêndulo
        periodo = 2 * math.pi * math.sqrt(self.comprimento / Gravidade)
        # Calcula o ângulo de oscilação no instante atual
        theta = self.amplitude_maxima * math.cos(2 * math.pi / periodo * self.tempo +
self.deslocamento)
        # Calcula as coordenadas da posição
        x, y = self.comprimento * self.escalonamento * math.sin(theta), 0 - (self.comprimento *
self.escalonamento * math.cos(theta))
        self.posicao = np.array([x, y, 0.0]) + self.deslocamento_posicao
        # Calcula a fase
        self.fase = self.tempo / periodo
```

1. Importação de Bibliotecas

O código importa bibliotecas essenciais para o funcionamento da animação:

- `manim`: Para a criação e manipulação de objetos animados.

- `math`: Para cálculos matemáticos como seno, cosseno e pi.

- `numpy`: Para operações com vetores e arrays.

2. Constantes Físicas

O código define a constante `Gravidade` com o valor de 9.81 m/s², representando a aceleração gravitacional usada nos cálculos do movimento dos pêndulos.

3. Classe Video
A classe `Video` herda de `MovingCameraScene` e é responsável por criar a cena animada. Abaixo estão os principais métodos da classe:

3.1 Método construct()
- Cria os pêndulos utilizando o método `criar_pendulos()`.
- Configura a posição inicial da câmera.
- Chama o método `animar()` para executar a animação.

3.2 Método criar_pendulos()
Este método gera uma lista de objetos `Pendulo` com diferentes comprimentos baseados nas frequências definidas. Passos principais:
- Define as frequências dos ciclos por minuto.
- Calcula os períodos usando a fórmula: `T = 60 / frequência`.
- Determina os comprimentos com base no período usando a fórmula do pêndulo simples.
- Cria objetos `Pendulo` com os comprimentos calculados.

3.3 Método animar()
Este método controla a animação dos pêndulos e elementos relacionados. As principais funcionalidades incluem:
- Criação de pontos representando os pêndulos.
- Criação de linhas conectando os pontos ao ponto fixo.
- Atualização contínua das posições e fases usando funções de updater.

4. Classe Pendulo
A classe `Pendulo` define as propriedades e o comportamento de um pêndulo individual. Os principais elementos são:

4.1 Atributos
- `comprimento`: Comprimento do pêndulo calculado com base no período.
- `amplitude_maxima`: Ângulo máximo de oscilação, definido como 90 graus.
- `escalonamento`: Escala para ajustar as dimensões da animação.
- `fase`: Posição na oscilação em termos de ciclo.

4.2 Método atualizar_posicao()

Este método calcula a nova posição do pêndulo no espaço para cada instante de tempo:
- Calcula o período com `T = 2 * π * √(comprimento / gravidade)`.
- Determina o ângulo de oscilação usando funções trigonométricas.
- Atualiza a posição do pêndulo no espaço com base no ângulo calculado.
- Atualiza a fase do pêndulo com o tempo.

5. Conclusão

O código é um exemplo de como conceitos físicos podem ser visualizados usando ferramentas computacionais. A animação dos pêndulos demonstra a relação entre comprimento, período e frequência, servindo como uma poderosa ferramenta educacional.

ANEXO C: CÓDIGO PARA PROBLEMA DE GEOMETRIA EM MANIM, EXPLICAÇÃO DETALHADA

Esse código em *Manim* é uma animação criada para ilustrar um problema de geometria que envolve o Teorema de Pitágoras. O foco está em representar e explorar áreas de quadrados dentro de um contexto visual, utilizando um triângulo retângulo.

Código da Animação:

```
from manim import *


class Obmep(Scene):

    def construct(self):
        obmep_position = UP * 3.0
        obmep = Text("OBMEP 2023 - Nível 3",
                     font_size=40,
                     font="Times New Roman")
        obmep.move_to(obmep_position)
        self.add(obmep)
```

```
quadradomaior_position = LEFT * 1.0 + DOWN * 0.5
quadradomaior = Square(color=RED, side_length=2.8)
quadradomaior.set_fill(color=BLUE_C, opacity=1.0)
quadradomaior.move_to(quadradomaior_position)

quadradomedio1_position = quadradomaior.get_center(
) + RIGHT * 2.5 + DOWN * 0.3
quadradomedio1 = Square(color=RED, side_length=2.2)
quadradomedio1.set_fill(color=WHITE, opacity=1.0)
quadradomedio1.move_to(quadradomedio1_position)

quadradomedio2_position = quadradomaior.get_center(
) + RIGHT * 2.5 + UP * 1.9
quadradomedio2 = Square(color=RED, side_length=2.2)
quadradomedio2.set_fill(color=WHITE, opacity=1.0)
quadradomedio2.move_to(quadradomedio2_position)

quadradomenor_position = quadradomaior.get_center(
) + RIGHT * 0.6 + UP * 2.2
quadradomenor = Square(color=RED, side_length=1.6)
quadradomenor.set_fill(color=BLUE_C, opacity=1.0)
quadradomenor.move_to(quadradomenor_position)

linha = Line(start=quadradomaior.get_corner(UR),
             end=quadradomedio1.get_corner(UR),
             color=PURE_RED)
linha2 = Line(start=quadradomedio1.get_corner(UL),
              end=quadradomaior.get_corner(UR),
              color=YELLOW)
linha3 = Line(start=quadradomedio1.get_corner(UL),
              end=quadradomedio1.get_corner(UR),
              color=YELLOW)
linha4 = Line(start=quadradomaior.get_corner(UR),
```

```
        end=quadradomedio1.get_corner(UR),
        color=YELLOW)

numero = Text("2", font_size=30, font="Times New Roman", color=BLACK)
numero.move_to(linha.get_center() + UP * 0.3)

X1 = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X1.move_to(quadradomedio1.get_center() + LEFT * 1.3)

X2 = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X2.move_to(quadradomedio1.get_center() + RIGHT * 1.3)

X4 = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X4.move_to(quadradomedio1.get_center() + DOWN * 1.3)

X5 = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X5.move_to(quadradomedio2.get_center() + UP * 1.3)

X6 = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X6.move_to(quadradomedio2.get_center() + LEFT * 1.3)

X7 = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X7.move_to(quadradomedio2.get_center() + RIGHT * 1.3)

Y = Text("y", font_size=30, font="Times New Roman", color=WHITE)
Y.move_to(quadradomedio1.get_center() + LEFT * 1.3 + UP * 1.3)

# Definindo os pontos do triângulo retângulo
ponto1 = np.array([-2, -1, 0])  # Ponto inferior esquerdo
ponto2 = np.array([2, -1, 0])  # Ponto inferior direito
ponto3 = np.array([-2, 1, 0])  # Ponto superior esquerdo

# Criando o triângulo retângulo com os três pontos
```

```
triangulo = Polygon(ponto1, ponto2, ponto3, color=YELLOW)

# Centralizando o triângulo na cena
triangulo.move_to(ORIGIN + UP * 1.0)

self.play(Create(quadradomaior))
self.play(Create(quadradomedio1))
self.play(Create(quadradomedio2))
self.play(Create(quadradomenor))
self.play(Create(linha))
self.play(Create(numero))

self.wait(4)

grupo1 = VGroup(obmep)
grupo2 = VGroup(X1, X2, X4, X5, X6, X7, Y)
grupo3 = VGroup(linha2, linha3, linha4)
grupo4 = VGroup(quadradomaior, quadradomedio1, quadradomedio2,
            quadradomenor, linha, numero, X1, X2, X4, X5, X6, X7,
            Y, linha2, linha3, linha4)

Xt = Text("x", font_size=30, font="Times New Roman", color=WHITE)
Xt.move_to(triangulo.get_center() + DOWN * 1.3)

Yt = Text("y", font_size=30, font="Times New Roman", color=WHITE)
Yt.move_to(triangulo.get_center() + LEFT * 2.3)

numero2 = Text("2", font_size=30, font="Times New Roman", color=WHITE)
numero2.move_to(triangulo.get_center() + UP * 0.5)

pit = Text("Teorema de Pitágoras:",
        font_size=30,
        font="Times New Roman",
```

```
        color=WHITE)
pit.move_to(ORIGIN + UP * 2.5)

formula = Tex(r"x² + y² = 4", color=WHITE)
formula.move_to(ORIGIN + DOWN * 1.5)

retangulo = SurroundingRectangle(formula,
                        color=WHITE,
                        fill_color=WHITE)

grupoT = VGroup(triangulo)

quadradomaiorT_position = ORIGIN + LEFT * 0.5 + DOWN * 0.5
quadradomaiorT = Square(color=RED, side_length=2.8)
quadradomaiorT.set_fill(color=BLUE_C, opacity=1.0)
quadradomaiorT.move_to(quadradomaiorT_position)

quadradomedio1T_position = quadradomaiorT.get_center(
) + RIGHT * 2.5 + DOWN * 0.3
quadradomedio1T = Square(color=RED, side_length=2.2)
quadradomedio1T.set_fill(color=WHITE, opacity=1.0)
quadradomedio1T.move_to(quadradomedio1T_position)

quadradomedio2T_position = quadradomaiorT.get_center(
) + RIGHT * 2.5 + UP * 1.9
quadradomedio2T = Square(color=RED, side_length=2.2)
quadradomedio2T.set_fill(color=WHITE, opacity=1.0)
quadradomedio2T.move_to(quadradomedio2T_position)

quadradomenorT_position = quadradomaiorT.get_center(
) + RIGHT * 0.6 + UP * 2.2
quadradomenorT = Square(color=RED, side_length=1.6)
quadradomenorT.set_fill(color=BLUE_C, opacity=1.0)
```

```
quadradomenorT.move_to(quadradomenorT_position)

linhaT = Line(start=quadradomaior.get_corner(UR),
            end=quadradomedio1.get_corner(UR),
            color=PURE_RED)
linha2T = Line(start=quadradomedio1.get_corner(UL),
             end=quadradomaior.get_corner(UR),
             color=YELLOW)
linha3T = Line(start=quadradomedio1.get_corner(UL),
             end=quadradomedio1.get_corner(UR),
             color=YELLOW)
linha4T = Line(start=quadradomaior.get_corner(UR),
             end=quadradomedio1.get_corner(UR),
             color=YELLOW)

numeroT = Text("2", font_size=30, font="Times New Roman", color=BLACK)
numero.move_to(linha.get_center() + UP * 0.3)

X1T = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X1T.move_to(quadradomedio1T.get_center() + LEFT * 1.3)

X2T = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X2T.move_to(quadradomedio1T.get_center() + RIGHT * 1.3)

X4T = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X4T.move_to(quadradomedio1T.get_center() + DOWN * 1.3)

X5T = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X5T.move_to(quadradomedio2T.get_center() + UP * 1.3)

X6T = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X6T.move_to(quadradomedio2T.get_center() + LEFT * 1.3)
```

```
X7T = Text("x", font_size=30, font="Times New Roman", color=WHITE)
X7T.move_to(quadradomedio2T.get_center() + RIGHT * 1.3)

YT = Text("y", font_size=30, font="Times New Roman", color=WHITE)
YT.move_to(quadradomedio1T.get_center() + LEFT * 1.3 + UP * 1.3)

grupoq = VGroup(quadradomaiorT, quadradomenorT, quadradomedio1T,
            quadradomedio2T, X1T, X2T, X4T, X5T, X6T, X7T, YT)

self.play(ReplacementTransform(grupo1, grupo2))
self.wait(3)

self.play(Create(grupo3))

self.wait(3)

self.play(ReplacementTransform(grupo4, grupoT))

self.play(Write(Xt))

self.play(Write(Yt))

self.play(Write(numero2))

self.play(Write(pit))

self.play(Write(formula))

self.play(Create(retangulo))

grupoT = VGroup(triangulo, Xt, Yt, numero2, pit, formula, retangulo)

self.wait(1)
```

```
self.play(ReplacementTransform(grupoT, grupoq))

X1t = Text("x+y", font_size=30, font="Times New Roman", color=WHITE)
X1t.move_to(quadradomaiorT.get_center() + DOWN * 1.7)

X2t = Text("x+y", font_size=30, font="Times New Roman", color=WHITE)
X2t.move_to(quadradomaiorT.get_center() + LEFT * 2.0)

X3t = Text("x-y", font_size=30, font="Times New Roman", color=WHITE)
X3t.move_to(quadradomenorT.get_center() + LEFT * 1.2)

X4t = Text("x-y", font_size=30, font="Times New Roman", color=WHITE)
X4t.move_to(quadradomenorT.get_center() + UP * 1.0)

self.play(Write(X1t))

self.play(Write(X2t))

self.play(Write(X3t))

self.play(Write(X4t))

grupos = VGroup(quadradomedio1T, quadradomedio2T, X1T, X2T, X4T, X5T,
            X6T, X7T, YT)

self.play(FadeOut(grupos))

area = Text("Área azul",
        font_size=30,
        font="Times New Roman",
        color=WHITE)
area.move_to(ORIGIN + RIGHT * 2.0)
```

```
self.play(Write(area))

area1 = Text("Área azul:",
            font_size=30,
            font="Times New Roman",
            color=WHITE)
area1.move_to(ORIGIN + UP * 2.5)

solucao = Text(r"(x+y)² + (x-y)²",
              font_size=30,
              font="Times New Roman",
              color=WHITE)
solucao.move_to(ORIGIN + UP * 1.8)

solucao1 = Text(r"x² + 2xy + y² + x² - 2xy + y²",
               font_size=30,
               font="Times New Roman",
               color=WHITE)
solucao1.move_to(ORIGIN + UP * 1.1)

solucao2 = Text(r"x² + y² + x² + y²",
               font_size=30,
               font="Times New Roman",
               color=WHITE)
solucao2.move_to(ORIGIN + UP * 0.4)

grupoF = VGroup(quadradomaiorT, quadradomenorT, area, X1t, X2t, X3t,
               X4t)

self.play(ReplacementTransform(grupoF, area1))

self.play(Write(solucao))
```

```
self.wait(1.5)
self.play(Write(solucao1))
self.wait(1.5)
self.play(Write(solucao2))

retangulo1 = SurroundingRectangle(solucao2,
                        color=WHITE,
                        fill_color=WHITE)

grupoe = VGroup(solucao2, area1, solucao1, solucao)

solucaof = Text(r"4+4",
            font_size=30,
            font="Times New Roman",
            color=WHITE)
solucaof.move_to(ORIGIN + UP * 0.4)

self.wait(1)

solucaoF = Text(r"8",
            font_size=30,
            font="Times New Roman",
            color=WHITE)
solucaoF.move_to(ORIGIN)

self.wait(1)

self.play(ReplacementTransform(grupoe, solucaof))

self.play(ReplacementTransform(solucaof, solucaoF))

retanguloF = SurroundingRectangle(solucaoF,
                        color=WHITE,
```

```
                    fill_color=WHITE)

    self.play(Create(retanguloF))

    self.wait(2)
```

1. Título e posição do texto "OBMEP 2023 - Nível 3":

O código começa criando o texto "OBMEP 2023 - Nível 3", que é movido para uma posição acima da cena, representando o contexto do problema dentro de uma competição de matemática.

2. Criação dos quadrados:

O código desenha uma sequência de quadrados de diferentes tamanhos, com cores e posicionamentos específicos:

Um quadrado maior em azul e vermelhos.

Quadrados menores (em branco) dentro do quadrado maior.

Esses quadrados são posicionados de maneira a formar uma figura geométrica que representa uma decomposição visual de áreas.

3. Linhas conectando os quadrados:

Linhas são desenhadas para conectar os quadrados, ajudando a destacar as relações entre os diferentes elementos geométricos. As linhas de cores diferentes (vermelho e amarelo) mostram as interações geométricas entre os quadrados.

4. Texto e variáveis:

O código usa variáveis representadas por letras "x" e "y", posicionadas nas diferentes partes da figura, e também adiciona números (como "2") para ajudar na visualização do problema.

O propósito dessas letras é representar os lados dos quadrados e do triângulo, fundamentais para a explicação do Teorema de Pitágoras.

5. Criação do triângulo retângulo:

O código também cria um triângulo retângulo amarelo, utilizando o Teorema de Pitágoras como base para explicar a relação entre os lados dos quadrados.

O triângulo é centralizado na cena para servir de base para a explicação visual.

6. Fórmula do Teorema de Pitágoras:

Após a introdução dos quadrados e do triângulo, a animação exibe a fórmula $x^2+y^2 = 4$, com o número 4 resultante da soma das áreas dos quadrados. Essa fórmula é destacada para ajudar os alunos a entenderem a relação entre os lados do triângulo.

7. Solução passo a passo:

A animação segue com uma explicação visual da solução do problema, incluindo expandir a expressão $(x+y)^2 + (x-y)^2$, simplificando os termos e mostrando como chegar à resposta final de 8.

Ao longo do processo, retângulos são desenhados ao redor das expressões para ajudar na visualização.

8. Resultado final:

No final, a animação revela a solução final, que é 8, com um retângulo ao redor para destacar o resultado.

Em resumo, o código apresenta um exercício de geometria visual em que quadrados e um triângulo retângulo são usados para ilustrar e explicar o Teorema de Pitágoras de forma dinâmica, tornando o conceito mais acessível para o público-alvo (possivelmente estudantes de um nível de matemática básico ou intermediário, como os participantes da OBMEP).