**JUCIANE GONÇALVES MAIA**

# ENSINO DE GRAVITAÇÃO POR MEIO DE SIMULAÇÃO 3D: DESENVOLVIMENTO DE UM SIMULADOR PARA A PLATAFORMA SIMUFÍSICA®

**JI-PARANÁ, RO**
**ABRIL DE 2026**

**JUCIANE GONÇALVES MAIA**

# ENSINO DE GRAVITAÇÃO POR MEIO DE SIMULAÇÃO 3D: DESENVOLVIMENTO DE UM SIMULADOR PARA A PLATAFORMA SIMUFÍSICA®

Trabalho de Conclusão de Curso apresentado ao Departamento de Física de Ji-Paraná, Universidade Federal de Rondônia, Campus de Ji-Paraná, como parte dos quesitos para a obtenção do Título de Licenciada em Física, sob orientação do Prof. Dr. Marco Polo Moreno de Souza.

**JI-PARANÁ, RO**
**ABRIL DE 2026**

Catalogação da Publicação na Fonte
Fundação Universidade Federal de Rondônia - UNIR

M217e Maia, Juciane Goncalves.
Ensino de gravitação por meio de simulação 3D: Desenvolvimento de um simulador para a plataforma SimuFísica / Juciane Goncalves Maia. - Ji-Paraná, 2026.

83f.: il.

Orientação: Prof. Dr. Marco Polo Moreno de Souza.

Trabalho de Conclusão de Curso (Graduação em Física) - Campus Ji-Paraná, Fundação Universidade Federal de Rondônia, Ji-Paraná, 2026.

1. Ensino de Física. 2. Gravitação Universal. 3. Simulador 3D. 4. Tecnologia Educacional. 5. SimuFísica®. I. Souza, Marco Polo Moreno de. II. Título.

Biblioteca Setorial de Ji-Paraná CDU 37:53

Bibliotecário(a) Bruno Crespo Soares CRB-11/1151

MINISTÉRIO DA EDUCAÇÃO
FUNDAÇÃO UNIVERSIDADE FEDERAL DE RONDÔNIA
DEPARTAMENTO ACADÊMICO DE FÍSICA - JI-PARANÁ

# ATA DE DEFESA DE CONCLUSÃO DE CURSO

As 14:14 horas do dia 02 do mês de abril de 2026 foi realizada, no Laboratório de Pesquisa em Ensino de Física (LAPEF), a Sessão de Apresentação e Defesa do Trabalho de Conclusão de Curso intitulado **Ensino de Gravitação por meio de simulação 3D: Desenvolvimento de um simulador para a plataforma SimuFísica®**, apresentado pela acadêmica **JUCIANE GONÇALVES MAIA**. O trabalho foi julgado **aprovado** pelos docentes Marco Polo Moreno de Souza, Vanessa Delfino Kegler e Walter Trennephol Junior, todos do Departamento de Física, Campus Ji-Paraná, da Universidade Federal de Rondônia - UNIR, com nota **10,0** como requisito parcial para obtenção do título de LICENCIADA EM FÍSICA, com ressalvas para correções a serem feitas pela aluna antes de submeter a versão definitiva para o fechamento da nota na disciplina: Trabalho de Conclusão de Curso.

Ji-Paraná/RO, 02 de abril de 2026.

Aprovado pela Banca Examinadora constituída pelos seguintes professores:

ORIENTADOR: Prof. Dr. Marco Polo Moreno de Souza
Aprovado ( x ) Reprovado ( )

AVALIADORA 1: Profa. Dra. Vanessa Delfino Kegler
Aprovado ( x ) Reprovado ( )

AVALIADOR 2: Prof. Dr. Walter Trennepohl Junior
Aprovado ( x ) Reprovado ( )

Reaberta a sessão pública, o orientador proclamou os resultados e encerrou a sessão, da qual foi lavrada a presente ata que vai assinada pelos membros da banca.

Documento assinado eletronicamente por **WALTER TRENNEPOHL JUNIOR**, **Docente**, em 02/04/2026, às 17:49, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **VANESSA DELFINO KEGLER**, **Docente**, em 03/04/2026, às 10:35, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **MARCO POLO MORENO DE SOUZA**, **Docente**, em 07/04/2026, às 11:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.unir.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2580369** e o código CRC **56442B38**.

**DEDICATÓRIA**

Dedico este trabalho às minhas pequenas, Helissa e Hellen. Se hoje busco os segredos das leis que regem o cosmos, é para mostrar a vocês que o conhecimento é a única força capaz de nos levar além de qualquer horizonte. Vocês são a motivação por trás de cada linha deste código e a luz que ilumina o meu caminho acadêmico. Amo vocês além das estrelas.

## AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Marco Polo Moreno de Souza, por sua mentoria perspicaz, pela paciência e pela confiança depositada neste projeto desde a sua concepção. Sua expertise foi o alicerce necessário para que eu pudesse convergir o rigor da física teórica às exigências da tecnologia. Para além das equações, espero ter absorvido a essência de ser um docente que transborda amor pelo que faz.

Aos docentes do Departamento de Física do Campus de Ji-Paraná, com quem tive a honra de conviver e aprender. Cada aula ministrada e cada desafio proposto representaram alicerces fundamentais na construção do meu saber e na consolidação da minha identidade acadêmica. Ninguém transita por nossa trajetória sem deixar marcas, e os ensinamentos aqui colhidos serão bússolas por toda a minha vida.

À Universidade Federal de Rondônia (UNIR), ambiente de pluralidade e excelência que possibilitou minha formação. Reitero meu orgulho e gratidão: viva as universidades federais, baluartes da ciência e patrimônios inalienáveis do povo brasileiro, que resistem e iluminam o interior do país com pesquisa de qualidade e compromisso social. Aos amigos que fiz nesta caminhada na UNIR, meu muito obrigada pelo companheirismo e pelas trocas que tornaram a jornada mais leve.

À minha família e aos meus amigos, que compuseram uma rede de apoio inestimável. Meu agradecimento mais profundo e sincero àqueles que, com generosidade, cuidaram das minhas filhas para que eu pudesse dedicar as horas necessárias a este estudo. Sem esse amparo logístico e emocional, este trabalho jamais teria transcendido o campo das ideias para se tornar realidade.

Em especial às minhas filhas, que são o norte das minhas escolhas, o motivo da minha resiliência e minha maior fonte de inspiração. Todo esse esforço é, e sempre será, por vocês.

**EPÍGRAFE**

"Se eu vi mais longe, foi por estar sobre ombros de gigantes." — Isaac Newton

**RESUMO**

Este trabalho apresenta o desenvolvimento e a análise de um simulador tridimensional de gravitação destinado ao ensino de Física, concebido para integração à plataforma educacional *SimuFísica*®. O simulador proposto permite explorar a dinâmica gravitacional entre dois e três corpos em um ambiente tridimensional interativo, possibilitando a visualização de órbitas e das interações resultantes do princípio da superposição das forças gravitacionais. O motor físico da simulação foi implementado com base na solução numérica das equações diferenciais do movimento utilizando o método de Runge-Kutta de quarta ordem (RK4), garantindo estabilidade e precisão nos cálculos. A visualização gráfica foi desenvolvida com a biblioteca *Three.js*, permitindo a execução da ferramenta diretamente em navegadores web. A validação do simulador foi realizada por meio da comparação entre resultados numéricos obtidos pela simulação e parâmetros astronômicos reais, demonstrando elevada fidelidade na reprodução de órbitas planetárias. Do ponto de vista educacional, o simulador constitui um objeto de aprendizagem que favorece metodologias ativas, permitindo que estudantes investiguem fenômenos gravitacionais por meio da manipulação de parâmetros físicos e da observação direta dos resultados. Dessa forma, o trabalho contribui tanto para o fortalecimento do ecossistema *SimuFísica*® quanto para a ampliação do uso de simuladores desenvolvidos no Brasil para o ensino de Física.

**Palavras-chave**: Ensino de Física. Gravitação Universal. Simulador 3D. Tecnologia Educacional. *SimuFísica*®.

**ABSTRACT**

This work presents the development and analysis of a three-dimensional gravitational simulator designed for physics education, conceived for integration into the educational platform *SimuFísica*®. The proposed simulator allows the exploration of gravitational dynamics in two- and three-body systems within an interactive three-dimensional environment, enabling the visualization of orbital motion and the interactions resulting from the superposition principle of gravitational forces. The simulation's physics engine was implemented based on the numerical solution of the equations of motion using the fourth-order Runge–Kutta (RK4) method, ensuring stability and accuracy in the calculations. Graphical visualization was developed using the *Three.js* library, allowing the tool to run directly in web browsers. The simulator was validated by comparing numerical results obtained from the simulations with real astronomical parameters, demonstrating high fidelity in reproducing planetary orbits. From an educational perspective, the simulator constitutes an interactive educational resource that supports active learning methodologies, allowing students to investigate gravitational phenomena through the manipulation of physical parameters and the direct observation of the results. In this way, the work contributes both to strengthening the *SimuFísica*® ecosystem and to expanding the use of educational simulators developed in Brazil for physics education.

**Keywords**: Physics Education. Universal Gravitation. 3D Simulation. Educational Technology. *SimuFísica*®.

## LISTA DE TABELAS

## LISTA DE FIGURAS

**SUMÁRIO**

# 1 INTRODUÇÃO

O ensino de Física é frequentemente desafiador, sobretudo ao abordar conceitos abstratos como gravitação. A jornada para compreender a atração entre os corpos e a estrutura do cosmos mobilizou a ciência por séculos, culminando em teorias de imenso poder explicativo.

Desde a formulação da lei da Gravitação Universal por Isaac Newton [1], que unificou a "física terrestre" e a "física celeste", até a revolucionária Teoria da Relatividade Geral de Albert Einstein [2], o entendimento da gravitação evoluiu significativamente. Contudo, a sofisticação matemática e a natureza não-visualizável desses modelos representam uma barreira significativa para a aprendizagem, especialmente para estudantes do Ensino Médio e dos primeiros anos do Ensino Superior.

Para transpor essa barreira, a integração de tecnologias educacionais tem se mostrado uma estratégia eficaz, convertendo abstrações teóricas em experiências interativas e visuais. Dentre essas tecnologias, simuladores e laboratórios virtuais se destacam, pois permitem que os alunos explorem fenômenos físicos simulados em um ambiente controlado, seguro e acessível, promovendo uma aprendizagem mais significativa [3]. Nesse cenário, a plataforma *SimuFísica*®[1] [4] surge como um ecossistema digital dedicado a centralizar e disponibilizar objetos de aprendizagem interativos e de alta qualidade para o ensino de Física [5]. A simulação tridimensional (3D), em particular, emerge como um recurso representativo, uma vez que a gravitação é um fenômeno inerentemente tridimensional, e uma representação em três dimensões permite explorar a dinâmica orbital de forma mais autêntica e imersiva.

O objetivo central deste trabalho é, portanto, apresentar o desenvolvimento de um simulador 3D de gravitação para integrar e expandir o acervo da plataforma *SimuFísica*®. Implementado com a biblioteca *Three.js*[2] [6], o simulador busca facilitar a compreensão de conceitos fundamentais, como o princípio da superposição no problema de $N$-corpos [7] e a natureza caótica do problema dos três corpos [8], promovendo um aprendizado ativo [9] e investigativo [10]. Sob essa ótica, a ferramenta deixa de ser um mero recurso visual para tornar-se um laboratório virtual, onde o aluno é encorajado a formular hipóteses e testar variáveis, processo central para a alfabetização científica e a construção ativa do saber [11].

Diferente de abordagens puramente teóricas, este estudo propõe a validação da ferramenta sob o prisma da fidedignidade científica, requisito indispensável para que o simulador atue como um objeto de aprendizagem confiável [12]. A aplicabilidade em sala de aula é discutida a partir da capacidade da ferramenta em sustentar metodologias ativas, onde o professor deixa de ser o único detentor do saber para se tornar um mediador de investigações guiadas pela visualização interativa [13]. Com isso, espera-se contribuir para a modernização das práticas de ensino de Física e para o fortalecimento do ecossistema *SimuFísica*® como um polo de recursos didáticos que despertem o protagonismo e a curiosidade científica dos discentes [14].

Para conduzir o leitor por essa jornada, este trabalho está estruturado da seguinte forma: primeiramente, serão apresentados os fundamentos teóricos da gravitação [15] e da pedagogia baseada em simulações [16, 17]; em seguida, detalhamos a metodologia de desenvolvimento e as tecnologias empregadas; por fim, são expostos os resultados da validação numérica do motor de física e discutidas as perspectivas de integração e uso pedagógico da ferramenta no contexto educacional.

[1] *SimuFísica*® é uma plataforma brasileira, acessível via simufisica.com, que funciona como um ambiente virtual de aprendizagem, gratuito e de livre acesso. É dedicada ao ensino de Física para o Ensino Médio e Superior, e serve como ecossistema para a integração da ferramenta desenvolvida neste trabalho.

[2] Biblioteca JavaScript de código aberto que simplifica a criação e exibição de gráficos 3D em navegadores web. Ela atua como uma camada de abstração sobre a *webgl*, facilitando a manipulação de cenas, câmeras, luzes e objetos tridimensionais.

## 1.1 JUSTIFICATIVA

A relevância deste trabalho de conclusão de curso (TCC) se consolida sobre um tripé de contribuições que se interceptam: a pedagógica, a tecnológica e a científica [18]. Cada uma dessas dimensões justifica a necessidade e a importância do desenvolvimento da ferramenta de simulação proposta [19].

Na dimensão pedagógica, o trabalho ataca uma dificuldade de aprendizagem historicamente consolidada no ensino de Física [20]: a abstração da gravitação [21] e da mecânica [22]. Conceitos como a natureza vetorial das forças, o princípio da superposição em sistemas de múltiplos corpos [23] e a visualização de órbitas tridimensionais representam um abismo cognitivo para muitos estudantes [24]. A ferramenta proposta visa mitigar essa dificuldade ao adotar uma abordagem construtivista, na qual o aluno deixa de ser um receptor passivo de informações para se tornar um agente ativo na construção do seu conhecimento [25]. Ao permitir a experimentação livre — alterando massas, velocidades e posições iniciais, o simulador fomenta o levantamento de hipóteses e a observação direta de suas consequências [26]. Essa metodologia está em plena conformidade com as competências da Base Nacional Comum Curricular (BNCC), que incentivam o uso de tecnologias digitais (TDIC[3]) e o desenvolvimento do pensamento científico, crítico e criativo [27].

Sob a perspectiva tecnológica e de inovação, o TCC agrega valor direto ao ecossistema da plataforma *SimuFísica*®. Enquanto muitas simulações educacionais se limitam a representações bidimensionais, a presente proposta inova ao oferecer uma experiência tridimensional imersiva para um tópico fundamental da Física. O desenvolvimento com tecnologias web modernas e de código aberto[4], notavelmente a biblioteca *Three.js*, representa uma escolha estratégica: garante-se a acessibilidade e a ampla distribuição da ferramenta, que pode ser executada em qualquer navegador moderno sem a necessidade de instalação de software específico [28]. Dessa forma, o projeto não apenas amplia o portfólio da plataforma *SimuFísica*®, mas também contribui para democratizar o acesso a recursos educacionais de qualidade.

Finalmente, na dimensão científica e acadêmica, o trabalho oferece um "laboratório virtual" para a exploração de um dos problemas mais clássicos e fundamentais da física: o problema de N-corpos [29, 30]. A simulação permite uma aproximação intuitiva e visual a um tema de elevada complexidade matemática, tornando conceitos avançados, como a estabilidade de sistemas planetários e a natureza caótica do problema dos três corpos [31], acessíveis a um público mais amplo. Além disso, a pesquisa em si, ao desenvolver e validar a eficácia de uma nova ferramenta didática, pode contribuir para a área de Ensino de Física, fornecendo dados e um estudo de caso sobre o potencial das simulações 3D baseadas na web para a educação científica.

Embora existam simuladores 3D robustos que exploram a gravitação, como o *software* comercial *Universe Sandbox* [32] ou ferramentas disponibilizadas pela NASA [33], estes frequentemente possuem um caráter de *sandbox*[5] ou de demonstração em larga escala. A proposta

[3] TDIC é a sigla para Tecnologias Digitais da Informação e Comunicação. O termo, amplamente utilizado em contextos educacionais e na BNCC, abrange o conjunto de hardware, software, aplicativos e mídias digitais que podem ser utilizados para criar, processar, armazenar e comunicar informações, servindo como ferramentas para potencializar o processo de ensino-aprendizagem.

[4] Software de "código aberto" (*open source*) é aquele cujo código-fonte é disponibilizado publicamente, permitindo que qualquer pessoa o veja, modifique e distribua. Essa filosofia promove a colaboração, a transparência e a inovação, além de geralmente não ter custos de licença, o que é importante para projetos educacionais e de pesquisa.

[5] No contexto de software e jogos, um ambiente "*sandbox*" (caixa de areia) é aquele que oferece grande liber-

deste trabalho, em contraste, foca no desenvolvimento de uma ferramenta com um objetivo pedagógico específico, permitindo a manipulação controlada de parâmetros para investigar as órbitas resultantes da interação de três corpos [34], um tema classicamente desafiador [35]. A futura integração ao *SimuFísica*® garante que o recurso esteja alinhado a um contexto educacional e curricular brasileiro e que tem se destacado pela facilidade de utilização e resultados no ensino de física.

## 1.2 OBJETIVOS

### 1.2.1 Objetivo geral

Desenvolver e validar uma simulação computacional 3D interativa sobre a dinâmica da gravitação universal, concebida como uma ferramenta pedagógica para facilitar a visualização de fenômenos complexos e promover a aprendizagem ativa, integrando-a ao ecossistema da plataforma *SimuFísica*®.

### 1.2.2 Objetivos específicos

- Fundamentar o trabalho em uma revisão bibliográfica sobre a história e a física da gravitação, o problema de N-corpos e o uso de simulações no ensino de ciências;
- Projetar a arquitetura do software utilizando a linguagem *UML (Unified Modeling Language)*[6], visando um design modular, escalável e de fácil manutenção;
- Estruturar o ciclo de desenvolvimento do software com base na metodologia ágil *Scrum*[7], para garantir entregas iterativas e adaptáveis;
- Implementar o método numérico de *Runge-Kutta de 4ª Ordem (RK4)*[8] para solucionar, computacionalmente, as equações diferenciais que governam o movimento dos corpos;
- Construir a interface gráfica e a visualização 3D com a biblioteca *Three.js*, priorizando a usabilidade e a interatividade do usuário;

dade ao usuário para explorar, criar e modificar o sistema sem um objetivo predefinido. Embora excelente para a exploração livre, pode ser menos eficaz para investigações pedagógicas guiadas e focadas em conceitos específicos.

[6] Linguagem de Modelagem Unificada. É uma linguagem gráfica padronizada para visualizar, especificar, construir e documentar os artefatos de um sistema de software, utilizada na fase de planejamento da arquitetura da simulação.

[7] Um *framework* ágil para desenvolver e manter produtos complexos. Baseia-se em ciclos de desenvolvimento curtos chamados *Sprints*, e seus pilares são a transparência, a inspeção e a adaptação.

[8] Um método numérico iterativo amplamente utilizado para a solução de equações diferenciais ordinárias. Foi o algoritmo escolhido no projeto por seu equilíbrio entre precisão computacional e estabilidade para simular as trajetórias.

## 1.3 ESTRUTURA DO TRABALHO

Este Trabalho de Conclusão de Curso está organizado em seis capítulos, estruturados de forma a conduzir o leitor desde a fundamentação conceitual até a validação técnica da ferramenta desenvolvida.

O **Capítulo 2**, fundamentação teórica, estabelece o alicerce científico da pesquisa, revisitando a evolução histórica do pensamento gravitacional — de Platão a Einstein — e detalhando a física do problema de $N$-corpos, sob a ótica do princípio da superposição e da dinâmica caótica.

O **Capítulo 3** descreve o ecossistema *SimuFísica*® e sua relevância no contexto educacional brasileiro e como o desenvolvimento do simulador gravitação 3D contribuirá.

O **Capítulo 4** descreve a metodologia de desenvolvimento sob dois prismas: a modelagem física, através da implementação do método numérico de *Runge-Kutta* de 4ª Ordem (RK4), e a engenharia de software. São descritas as tecnologias empregadas, como a biblioteca *Three.js*, além da arquitetura do sistema planejada via UML.

O **Capítulo 5** detalha os resultados e validação, constituindo o núcleo técnico e experimental do trabalho. Apresenta-se o confronto entre os dados gerados pelo motor de física do simulador e os parâmetros astronômicos reais, atestando a fidedignidade científica da ferramenta para uso em sala de aula.

Finalmente, o **Capítulo 6** apresenta as considerações finais, onde são sintetizadas as contribuições deste estudo, as limitações identificadas e o *roadmap*[9] para a integração definitiva da ferramenta à plataforma de destino, apontando direções para expansões futuras em física moderna.

[9]Roadmap: termo utilizado para representar um planejamento estratégico visual contendo etapas, metas e direcionamentos previstos para o desenvolvimento futuro de um projeto ou sistema.

# 2 FUNDAMENTAÇÃO TEÓRICA

## 2.1 UMA BREVE HISTÓRIA DO ESTUDO DA GRAVITAÇÃO

A compreensão da gravidade é um pilar da Física e da nossa visão de mundo. Longe de ser um conceito estático, nossa noção de gravidade evoluiu através de milênios de observação, debate filosófico e revoluções científicas. Esta seção apresenta uma cronologia, considerando as principais teorias e modelos cosmológicos desenvolvidos ao longo da história, desde as primeiras concepções cosmológicas que separavam os céus da Terra, passando pela unificação de Newton [36], até as fronteiras da física contemporânea, onde a gravidade é entendida como a própria geometria do espaço-tempo [37].

### 2.1.1 Do problema de Platão às leis de Kepler

A astronomia grega buscava compreender e descrever racionalmente os movimentos observados no céu, conciliando observação empírica e princípios filosóficos. Para os filósofos gregos, especialmente os vinculados à tradição platônica, o cosmos era entendido como uma estrutura perfeita e ordenada, de modo que os movimentos celestes deveriam necessariamente ocorrer de maneira circular e uniforme, considerados os movimentos mais perfeitos da natureza. Entretanto, as observações astronômicas revelavam comportamentos complexos dos planetas (do grego *planētēs*, "errantes"), como variações de velocidade e mudanças aparentes de direção, incompatíveis com a simplicidade esperada pelos modelos filosóficos da época [38].

Nesse contexto surgiu o chamado "Problema de Platão", cujo objetivo era explicar matematicamente os movimentos irregulares observados no céu utilizando apenas combinações de movimentos circulares uniformes. O grande desafio intelectual da astronomia grega era reconciliar a perfeição filosófica dos movimentos circulares e uniformes com as observações dos planetas, que exibiam trajetórias complexas e irregulares no céu, incluindo o notório movimento retrógrado[1] [39]. A primeira grande solução matemática foi proposta por Eudoxo de Cnido (c. 408-355 a.C.), que concebeu um sistema de esferas concêntricas[2], ver Fig. 2.1. Neste modelo, cada planeta estava preso a uma esfera que girava, e o eixo dessa esfera, por sua vez, estava conectado a uma esfera maior com uma rotação diferente, e assim por diante. Com uma combinação de várias esferas para cada planeta, era possível reproduzir seus movimentos complexos [40].

Aristóteles (384-322 a.C.) adotou e expandiu essa ideia, transformando-a em um modelo físico composto por 49 esferas concêntricas que procuravam explicar os movimentos de todos os corpos celestes, ver Fig. 2.2. Nesse sistema, a esfera mais externa era a das estrelas fixas, a qual transmitia o movimento a todas as esferas inferiores, enquanto seu próprio giro era provocado por uma entidade de natureza sobrenatural [41].

[1] Fenômeno aparente observado quando um planeta parece inverter temporariamente sua direção de deslocamento no céu em relação às estrelas de fundo, devido às diferenças entre as velocidades orbitais da Terra e dos demais planetas.

[2] Conjunto de esferas transparentes e centradas na Terra utilizadas na cosmologia antiga para representar o movimento dos astros.

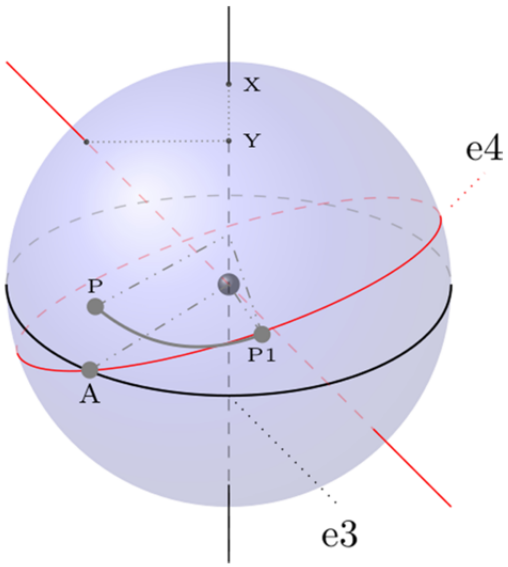


**Figura 2.1:** Representação do modelo matemático das esferas concêntricas proposto por Eudoxo de Cnido para explicar os movimentos aparentes dos corpos celestes. Nesse sistema, os planetas eram associados a esferas que realizavam movimentos circulares uniformes em diferentes eixos de rotação. A combinação desses movimentos permitia reproduzir trajetórias complexas observadas no céu, incluindo variações aparentes de posição. Na figura, o planeta inicia seu movimento no ponto A sobre o equador representado em vermelho e posteriormente é transferido para o equador em preto até alcançar o ponto P. Esse modelo representa uma das primeiras tentativas matemáticas de descrever o movimento planetário na astronomia grega. Fonte: [42].

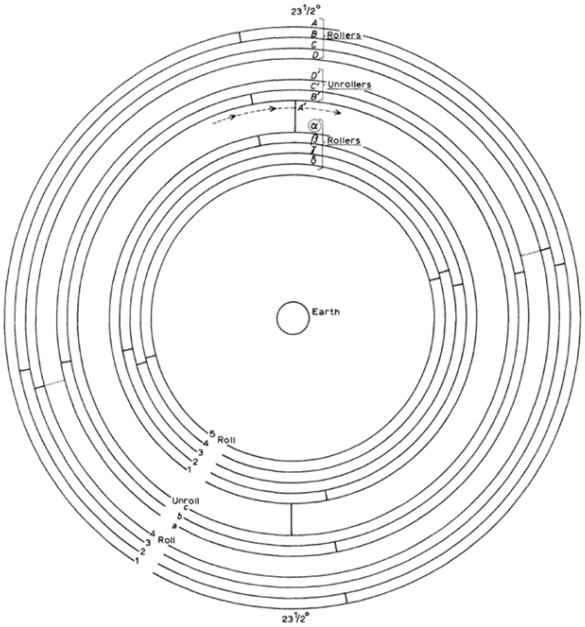


**Figura 2.2:** Representação do modelo cosmológico aristotélico baseado em esferas concêntricas. Nesse sistema, a Terra ocupa o centro do universo, enquanto os corpos celestes são transportados por esferas cristalinas que giram ao seu redor. Cada planeta possui um conjunto de esferas responsáveis por reproduzir seus movimentos observados, incluindo variações aparentes como retrogradações. A esfera mais externa corresponde à das estrelas fixas, cujo movimento uniforme é transmitido às esferas inferiores, sendo, por sua vez, associado a uma causa de natureza metafísica. Esse modelo representa uma das primeiras tentativas de descrever o cosmos por meio de uma estrutura geométrica e física integrada. Fonte: [43].

O ápice e a síntese definitiva do modelo modelo geocêntrico[3], no entanto, vieram com Cláudio Ptolomeu (c. 100-170 d.C.), conforme ilustrado na Fig. 2.3. Em sua obra monumental, o *Almagesto* [44], ele aperfeiçoou as ideias de seus predecessores, como Apolônio e Hiparco, para construir um modelo geométrico de imenso poder preditivo [45]. Em vez de apenas esferas, Ptolomeu utilizou um conjunto de artifícios geométricos para ajustar a teoria às observações cada vez mais precisas [46]:

**O Excêntrico**: Para explicar por que o Sol parecia mover-se mais rápido em algumas épocas do ano, Ptolomeu propôs que o centro da órbita solar (o deferente[4]) não estava na Terra, mas em um ponto ligeiramente deslocado, o excêntrico[5].

**O Epiciclo**: Para explicar o movimento retrógrado, o planeta se movia em um círculo menor, o epiciclo, cujo centro, por sua vez, percorria o círculo maior (deferente) ao redor da Terra. A combinação desses dois movimentos criava a "laçada" no céu observada pelos astrônomos. Tanto o aparente movimento para trás dos planetas em alguns períodos do ano como os epiciclos podem ser vistos através do simulador Movimento Retrógrado dos Planetas (ver Fig. 2.4).

**O Equante**: Para dar conta das variações de velocidade restantes, Ptolomeu introduziu um terceiro ponto, o equante. A velocidade angular do centro do epiciclo era uniforme não em relação ao centro do deferente, mas em relação a este novo ponto. Essa consideração foi necessária para se adequar aos movimentos observados dos planetas.

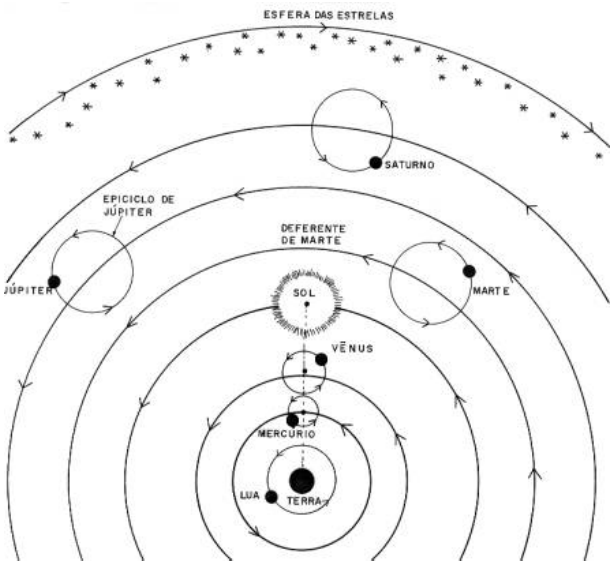


**Figura 2.3:** Representação do modelo geocêntrico de Cláudio Ptolomeu, detalhado em sua obra magna, o *Almagesto* (século II d.C.). O sistema coloca a Terra imóvel no centro do universo, com a Lua, Mercúrio, Vênus, Sol, Marte, Júpiter e Saturno orbitando ao seu redor. Para explicar o movimento retrógrado dos planetas, Ptolomeu utilizou um sistema complexo de epiciclos (círculos menores) girando sobre deferentes (círculos maiores). Fonte: Adaptado de Oliveira Filho e Saraiva (2014) [47].

[3]Modelo cosmológico no qual a Terra é considerada o centro do universo, enquanto os demais corpos celestes realizam movimentos ao seu redor. Esse modelo predominou na astronomia antiga e medieval, sendo amplamente sistematizado por Cláudio Ptolomeu por meio do uso de deferentes e epiciclos para explicar os movimentos aparentes dos planetas.

[4]Elemento fundamental do modelo geocêntrico ptolomaico, o deferente consistia em uma circunferência maior centrada próxima à Terra sobre a qual o centro do epiciclo de um planeta realizava seu movimento orbital, permitindo descrever matematicamente irregularidades observadas nos movimentos planetários aparentes.

[5]No contexto da astronomia geocêntrica, o excêntrico correspondia a uma circunferência cujo centro não coincidia exatamente com a Terra. Esse artifício geométrico foi introduzido para explicar as variações aparentes de velocidade dos planetas observadas no céu.

O sistema ptolomaico, com sua combinação de deferentes, epiciclos e equantes, tornou-se o modelo padrão da astronomia por mais de 1400 anos. Sua longevidade se deve ao seu sucesso: ele não apenas descrevia os movimentos celestes, mas permitia prever com razoável precisão as posições futuras dos planetas, eclipses e outros eventos astronômicos [48].

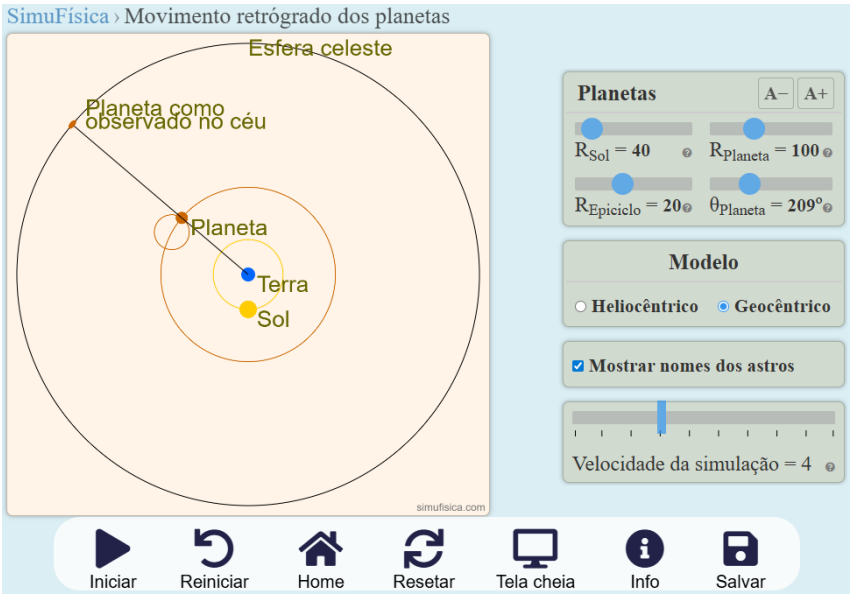


**Figura 2.4:** Simulação do movimento retrógrado dos planetas no modelo geocêntrico, desenvolvida na plataforma SimuFísica. A representação utiliza o sistema de deferentes e epiciclos empregado na astronomia ptolomaica para descrever os movimentos aparentes dos planetas observados a partir da Terra. Nesse modelo, o planeta realiza um movimento circular em um epiciclo, enquanto o centro desse epiciclo se desloca ao longo de um deferente, permitindo reproduzir fenômenos como a retrogradação planetária. Link para a simulação configurada: `https://simufisica.com/3lfszS`.

A primeira grande ruptura conceitual veio com Nicolau Copérnico (1473-1543). Em sua obra *De Revolutionibus Orbium Coelestium* [49], publicada postumamente, Copérnico propôs um sistema sistema heliocêntrico[6], motivado pela busca por uma maior harmonia e simplicidade matemática em relação ao complexo modelo de Ptolomeu, conforme ilustrado na Fig. 2.5.

Primeiramente, o modelo copernicano ofereceu uma explicação natural e elegante para o movimento retrógrado dos planetas. Esse fenômeno complexo, que exigia os epiciclos de Ptolomeu, foi explicado como um mero efeito de perspectiva, resultante do movimento relativo da Terra ao ultrapassar planetas mais distantes (como Marte) ou ao ser ultrapassada por planetas mais internos (como Vênus). Em segundo lugar, e talvez sua contribuição mais poderosa, o sistema heliocêntrico permitiu, pela primeira vez na história, determinar a arquitetura do Sistema Solar. Utilizando a distância Terra-Sol como uma nova unidade de medida (a Unidade Astronômica), Copérnico foi capaz de calcular tanto os períodos orbitais verdadeiros de cada planeta quanto as dimensões relativas de suas órbitas, alcançando uma precisão notável para a época, como demonstram os dados compilados em sua obra [50].

Contudo, apesar de seu avanço conceitual, Copérnico permaneceu atado ao dogma platônico das órbitas perfeitamente circulares. Como as órbitas planetárias são, na verdade, elípticas, ele foi forçado a reintroduzir um sistema de epiciclos e excêntricos para ajustar seu modelo às observações. Isso resultou em um sistema que, em termos de complexidade de cálculo e preci-

[6]Modelo cosmológico no qual o Sol ocupa a posição central do sistema, enquanto os planetas, incluindo a Terra, realizam movimentos orbitais ao seu redor. Esse modelo foi sistematizado por Nicolau Copérnico no século XVI e representou uma ruptura com a tradição geocêntrica predominante na astronomia antiga e medieval.

são preditiva, não era significativamente superior ao de Ptolomeu, o que dificultou sua aceitação inicial pela comunidade astronômica [51].

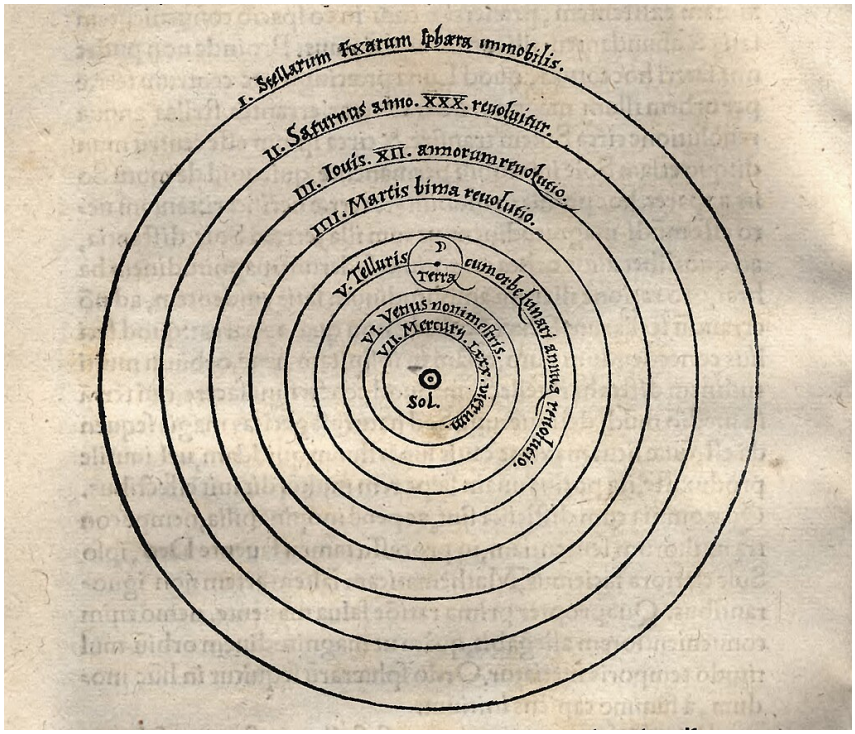


**Figura 2.5:** Diagrama do sistema Heliocêntrico conforme apresentado na obra de Nicolau Copérnico, *De revolutionibus orbium coelestium*, publicada em 1543. O modelo posiciona o Sol no centro, com os planetas conhecidos na época (Mercúrio, Vênus, Terra com a Lua, Marte, Júpiter e Saturno) orbitando ao seu redor. Fonte: Nicolau Copérnico (1543) [49].

Além das dificuldades técnicas, o modelo enfrentou forte oposição por desafiar o senso comum e dogmas religiosos profundamente enraizados. Questões como "se a Terra se move, por que não sentimos seu movimento?" e o conflito com interpretações literais das Escrituras Sagradas levaram a acusações de heresia. Ciente da controvérsia, Copérnico retardou a publicação de sua obra por décadas, vindo a recebê-la impressa apenas em seu leito de morte [52]. Ainda assim, ao retirar a Terra de sua posição privilegiada, Copérnico deu o passo conceitual decisivo que iniciou a Revolução Científica, abrindo caminho para as futuras contribuições de Tycho Brahe, Kepler e Galileu.

A ponte entre o debate teórico e a solução empírica foi construída pelo astrônomo dinamarquês Tycho Brahe (1546-1601). Confrontado com o dilema entre a elegância matemática do sistema copernicano e as fortes objeções físicas e filosóficas a uma Terra móvel, Tycho dedicou sua vida a resolver a questão através de dados observacionais de precisão sem precedentes [53].

O principal argumento científico de Tycho contra o heliocentrismo era a ausência da paralaxe estelar[7], ou seja, o desvio ou mudança aparente na posição de uma estrela próxima quando observada de diferentes pontos da órbita da Terra ao redor do Sol. Ele raciocinou, corretamente, que se a Terra orbitasse o Sol, as estrelas mais próximas deveriam exibir um desvio aparente em sua posição ao longo do ano. Com seus instrumentos meticulosamente

[7]Fenômeno astronômico caracterizado pela mudança aparente na posição de uma estrela em relação ao fundo de estrelas mais distantes, causada pela observação a partir de diferentes posições da órbita terrestre ao redor do Sol. A paralaxe estelar constitui uma das principais evidências observacionais do movimento da Terra no sistema heliocêntrico.

construídos, capazes de medir ângulos com precisão sem precedentes de até um sexto de minuto de arco em instrumentos fixos [54], ele não encontrou tal efeito. Isso o levou a duas possíveis conclusões: ou a Terra estava, de fato, imóvel, ou as estrelas estavam a uma distância tão vasta que a paralaxe era pequena demais para ser medida. Embora a segunda opção seja a correta, na época ela parecia implausível, e Tycho favoreceu a primeira, mantendo uma Terra estacionária [55].

Como solução, Tycho propôs seu próprio sistema cosmológico, um modelo híbrido geo-heliocêntrico[8], conforme ilustrado na Fig. 2.6. Neste sistema, a Terra permanecia fixa no centro do universo, com o Sol e a Lua orbitando-a. No entanto, os cinco planetas então conhecidos (Mercúrio, Vênus, Marte, Júpiter e Saturno) orbitavam o Sol. Este arranjo engenhoso preservava as vantagens matemáticas do modelo de Copérnico para explicar o movimento retrógrado, ao mesmo tempo que mantinha a Terra imóvel, em conformidade com a física aristotélica e as Escrituras. Durante suas observações, um evento chave fortaleceu sua crítica ao cosmos antigo: a passagem de um grande cometa em 1577. Ao medir a paralaxe do cometa, Tycho demonstrou que ele estava localizado muito além da órbita da Lua, estilhaçando a noção aristotélica de esferas celestes perfeitas e imutáveis [56, 57].

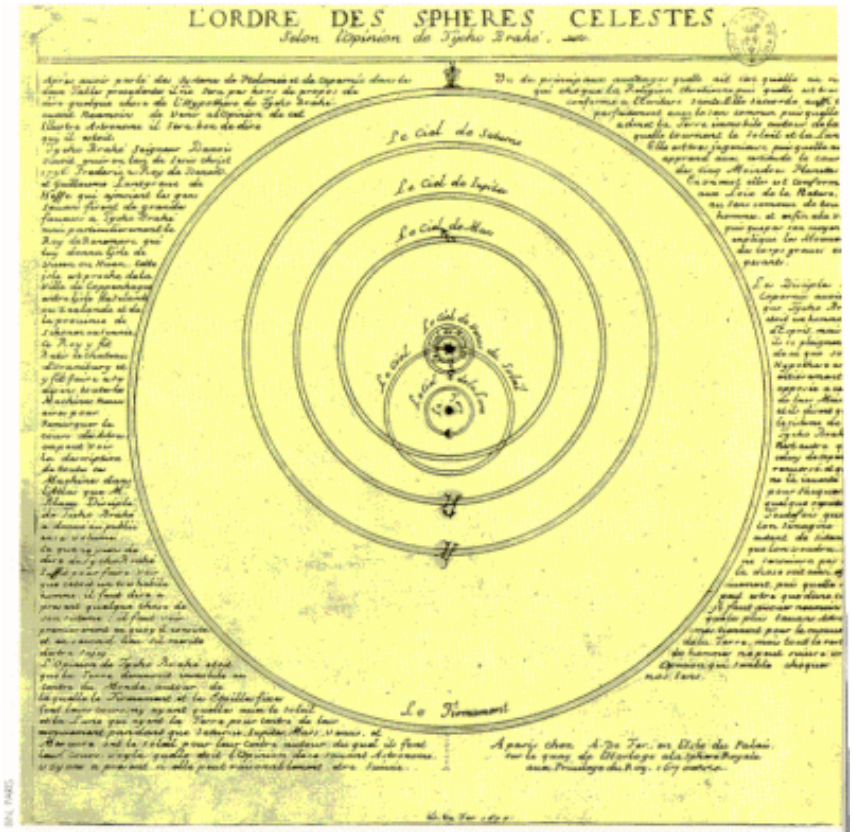


**Figura 2.6:** O sistema híbrido de Tycho Brahe buscava conciliar duas visões: por um lado, mantinha a premissa do geocentrismo aristotélico, explicando por que os objetos caem em direção à Terra; por outro, incorporava a eficiência geométrica do modelo de Copérnico para justificar o movimento retrógrado dos planetas. Fonte: Fer (1705) [58].

Embora o modelo de Tycho tenha sido um beco sem saída na história da cosmologia, seu

[8]Modelo cosmológico proposto por Tycho Brahe no final do século XVI, no qual a Terra permanecia imóvel no centro do universo, enquanto o Sol e a Lua orbitavam a Terra, e os demais planetas orbitavam o Sol. Esse sistema buscava conciliar as observações astronômicas da época com a ausência observável da paralaxe estelar e com a tradição cosmológica geocêntrica.

legado duradouro não foi sua teoria, mas a qualidade dos seus dados astronômicos. Financiado pelo rei Frederico II da Dinamarca, ele construiu o magnífico observatório de Uraniborg[9], onde, por mais de 20 anos, compilou o mais completo e preciso catálogo de estrelas e posições planetárias já visto. Esses dados, especialmente os da órbita de Marte, se tornariam a base empírica indispensável para que seu assistente, Johannes Kepler (1571-1630), finalmente decifrasse as verdadeiras leis do movimento planetário [59].

Foi Johannes Kepler quem herdou esse tesouro de dados. Diferente de seu mentor, Kepler não era primariamente um observador, mas um matemático genial e um teórico profundamente convicto de que o cosmos era governado por uma harmonia geométrica divina. Em sua obra de juventude, Mysterium Cosmographicum (1597) [60], ele já havia tentado, sem sucesso, relacionar as órbitas dos seis planetas conhecidos com os cinco sólidos platônicos[10] [61]. Ao receber os dados de Tycho, ele encontrou a oportunidade de testar suas ideias com a mais pura e precisa evidência empírica disponível.

A tarefa designada por Tycho a Kepler foi monumental: decifrar a órbita de Marte. Por anos, Kepler labutou sob o paradigma copernicano, tentando ajustar os dados a uma órbita circular. Ele chegou a um modelo que previa as posições de Marte com um erro de apenas oito minutos de arco. Para qualquer astrônomo anterior, tal precisão seria um triunfo, e o erro seria descartado como imprecisão observacional. Mas Kepler, conhecendo o rigor de Tycho, tomou a decisão corajosa que mudaria a história da ciência: ele confiou nos dados. Ele postulou que a discrepância não era um erro, mas uma verdade sobre a natureza. Esse humilde respeito pela evidência o forçou a abandonar o dogma de 2.000 anos da perfeição circular e a buscar a verdadeira forma da órbita [62].

Essa busca o levou à descoberta de que a órbita de Marte (e, por extensão, a de todos os planetas) não era um círculo, mas uma elipse. Mais do que isso, Kepler foi o primeiro a introduzir a física na astronomia, procurando uma causa dinâmica para esses movimentos. Ele teorizou que uma força emanava do Sol, varrendo os planetas e fazendo com que suas velocidades variassem: mais rápidas quando próximos ao Sol (periélio) e mais lentas quando distantes (afélio). Essas percepções foram a base de suas duas primeiras leis, publicadas em sua obra-prima de 1609, Astronomia Nova [63]:

1. **A Lei das Órbitas**: Os planetas se movem em órbitas elípticas, com o Sol ocupando um dos focos.

2. **A Lei das Áreas**: A linha que conecta um planeta ao Sol varre áreas iguais em intervalos de tempo iguais. Esta lei captura matematicamente a variação da velocidade orbital do planeta, uma consequência direta da "força" solar que Kepler imaginou.

Dez anos depois, em sua obra *Harmonices Mundi* [64], Kepler finalmente encontrou a relação matemática que unificava todo o sistema solar em uma única e harmoniosa estrutura. Sua terceira lei estabeleceu uma poderosa conexão entre a distância de um planeta ao Sol e a velocidade com que ele completa sua órbita [65]:

[9] Observatório astronômico construído por Tycho Brahe em 1576 na ilha de Hven, financiado pelo rei Frederico II da Dinamarca. Uraniborg tornou-se um dos mais importantes centros de observação astronômica da Europa antes da invenção do telescópio, sendo responsável pela obtenção de medições extremamente precisas dos movimentos planetários e estelares.

[10] Conjunto de cinco poliedros regulares convexos — tetraedro, cubo, octaedro, dodecaedro e icosaedro — caracterizados por possuírem faces congruentes formadas por polígonos regulares e o mesmo número de faces encontrando-se em cada vértice. Na tradição filosófica grega, especialmente em Platão, esses sólidos eram associados à harmonia e à estrutura fundamental do cosmos.

3. **A Lei dos Períodos** (ou Lei Harmônica): O quadrado do período orbital de um planeta $(T)$ é diretamente proporcional ao cubo do semieixo maior de sua órbita $(a)$, ou seja, $T^2 \propto a^3$.

As leis de Kepler forneceram uma descrição cinemática completa e espantosamente precisa do sistema solar [66]. Elas representaram o triunfo definitivo da evidência empírica sobre o dogma filosófico e da matemática rigorosa sobre a especulação [67]. No entanto, a causa dinâmica por trás desses movimentos, a natureza da "força" que os regia, permanecia um profundo mistério, preparando o terreno para a síntese de Isaac Newton.

### 2.1.2 A síntese Newtoniana: a força universal

Se as leis de Kepler descreveram a cinemática dos céus, coube a Isaac Newton (1643-1727) fornecer a dinâmica subjacente. A unificação da "física terrestre" e da "física celeste" foi a sua genialidade, um processo de maturação intelectual que se estendeu por mais de duas décadas e culminou em sua obra monumental, *Philosophiæ Naturalis Principia Mathematica* (1687) [68].

A gênese dessa unificação é frequentemente associada à famosa anedota da maçã, que teria ocorrido por volta de 1666, durante o período em que Newton se retirou para sua casa em Woolsthorpe devido à peste. Conforme os relatos que o próprio Newton compartilhou em sua velhice, o *insight* crucial não foi "descobrir" a gravidade, um termo já conhecido desde a antiguidade para descrever a tendência dos corpos de cair, mas sim conjeturar sobre sua universalidade. Ao observar a queda da maçã, ele se perguntou se a mesma força que a puxava para o chão não poderia se estender muito além, até a órbita da Lua, sendo a causa que a impedia de escapar em uma trajetória retilínea [69], conforme o princípio da inércia que ele estudara em Descartes [70].

Essa ideia o levou a realizar um primeiro cálculo quantitativo, o "teste da Lua"[11]. Partindo da hipótese de que a força da gravidade diminuiria com o inverso do quadrado da distância, ele comparou a aceleração da Lua em sua órbita (interpretando seu movimento curvo como uma "queda" contínua em direção à Terra) com a aceleração de um corpo na superfície terrestre. Embora o resultado inicial não tenha sido perfeitamente concordante, possivelmente pelo uso de um valor impreciso para o raio da Terra, o cálculo demonstrou a viabilidade da ideia. Na mesma época, utilizando a terceira lei de Kepler, ele também deduziu que a força que mantém os planetas em suas órbitas deveria seguir essa mesma relação do inverso do quadrado da distância [71].

Contudo, a formulação final da Lei da Gravitação Universal exigiu um amadurecimento conceitual significativo. Durante seus primeiros estudos, Newton ainda concebia a dinâmica orbital sob a influência cartesiana, como um equilíbrio entre uma tendência centrífuga e uma gravidade solar. Uma mudança fundamental, segundo historiadores como I. Bernard Cohen [72], foi estimulada pela sua correspondência com Robert Hooke (1679-1680), que o levou a abandonar a ideia de equilíbrio e adotar o conceito de uma única força atrativa e central

[11] O "teste da Lua" consistiu em comparar a aceleração necessária para manter a Lua em sua órbita (calculada a partir de seu período e distância) com a aceleração da gravidade na superfície da Terra, ajustada pela lei do inverso do quadrado da distância. A concordância dos valores validou a hipótese de que a mesma força atuava em ambos os casos.

(centrípeta) que continuamente desvia o planeta de sua trajetória inercial [73]. Foi esse novo paradigma dinâmico que permitiu a Newton dar um significado físico às leis de Kepler [74].

No *Principia*, Newton rompeu definitivamente com a separação aristotélica, postulando que quaisquer dois corpos no universo se atraem mutuamente, conforme representado na Fig. 2.7. Matematicamente, a intensidade desta força, $F_g$, é expressa como:

$$F_g = G\frac{m_1 m_2}{r^2}, \tag{2.1}$$

onde $m_1$ e $m_2$ são as massas dos corpos, $r$ é a distância entre seus centros de massa, e $G$ é a constante de gravitação universal. O valor de $G$, determinado experimentalmente, é extremamente pequeno, o que explica por que a força gravitacional só é significativa para corpos de grande massa, como planetas e estrelas. O valor atualmente recomendado pelo *Committee on Data for Science and Technology* (CODATA) é [75]:

$$G = 6.67430(15) \times 10^{-11}\text{m}^3\text{kg}^{-1}\text{s}^{-2}. \tag{2.2}$$

Sendo uma grandeza vetorial, a força de atração que o corpo 2 exerce sobre o corpo 1, denotada por $\vec{F}_{12}$, pode ser escrita de forma mais completa:

$$\vec{F}_{12} = -G\frac{m_1 m_2}{r_{12}^2}\,\hat{r}_{12}, \tag{2.3}$$

onde $r$ é a magnitude da distância ($|\vec{r_1} \to \vec{r_2}|$) e $\hat{r}_{12}$ é o vetor unitário que aponta do corpo 1 para o corpo 2 conforme ilustrado no Fig. 2.7. O sinal negativo indica a natureza atrativa da força, que age sempre na direção oposta ao vetor de posição relativa. De acordo com a Terceira Lei de Newton (ação e reação) [76], a força que o corpo 1 exerce sobre o corpo 2, $\vec{F}_{21}$, é igual em módulo e oposta em sentido, ou seja, $\vec{F}_{21} = -\vec{F}_{12}$.

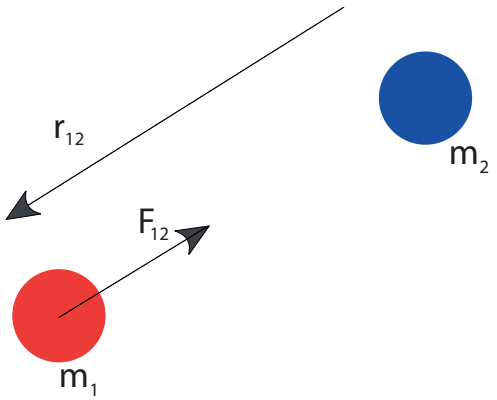


**Figura 2.7:** Ilustração da força de atração gravitacional exercida no corpo de massa $m_1$ devido à presença de um corpo com massa $m_2$. O vetor $\vec{r}_{12}$ está orientado na direção que une os dois corpos, no sentido do corpo 2 para o corpo 1.

O maior triunfo da lei da gravitação de Newton foi sua capacidade de, quando combinada com suas três leis do movimento, derivar matematicamente as leis empíricas de Kepler a partir de primeiros princípios. Isso demonstrou que as órbitas elípticas, a variação de velocidade dos planetas e a relação entre período e distância não eram meras coincidências, mas consequências diretas de uma única lei universal. Isso não apenas explicou as leis de Kepler, mas lhes conferiu um status de certeza que não possuíam anteriormente na comunidade científica [77].

O paradigma newtoniano representou uma profunda transformação na ciência ao fornecer uma estrutura matemática capaz de descrever e prever com grande precisão o movimento dos corpos celestes. Um dos exemplos mais conhecidos desse poder preditivo foi a descoberta do planeta Netuno, em 1846, motivada pela análise das perturbações observadas na órbita de Urano [78]. Esse episódio tornou-se um marco na história da Astronomia por demonstrar a capacidade da mecânica newtoniana de indicar a existência de corpos ainda não observados diretamente.

Entretanto, a interpretação histórica desse evento não é consensual. Alguns autores argumentam que a concordância entre as previsões teóricas e a posição observada de Netuno foi, em parte, favorecida por circunstâncias específicas e não exclusivamente pelo sucesso dos cálculos realizados. Entre esses autores, Peirce [79] sustentou que a descoberta envolveu um componente de coincidência favorável, ao analisar as hipóteses empregadas nos cálculos orbitais da época.

Apesar desse debate histórico, a teoria da gravitação de Newton permaneceu, por mais de dois séculos, como a principal descrição dos fenômenos gravitacionais. Ainda hoje, constitui uma ferramenta fundamental para inúmeras aplicações práticas, incluindo a determinação de órbitas de satélites artificiais, missões espaciais e problemas de engenharia aeroespacial [80].

### 2.1.3 A revolução Einsteiniana e fronteiras modernas

No início do século XX, apesar do merecido sucesso da teoria de Newton, algumas fissuras em seu alicerce começavam a aparecer. Duas questões principais se destacavam: uma observacional, a precessão anômala do periélio de Mercúrio[12] (um pequeno desvio em sua órbita que a mecânica newtoniana não conseguia justificar) [81], e outra teórica, a incompatibilidade da ação instantânea à distância com a recém-formulada Teoria da Relatividade Especial (1905) [82]. Para Newton, a gravidade se propagava instantaneamente; se o Sol desaparecesse, a Terra sentiria o efeito no mesmo instante. Para Einstein, nada poderia viajar mais rápido que a luz [83]. Essa contradição fundamental levou Albert Einstein (1879-1955) a uma reconceitualização completa da gravidade. Em 1915, ele apresentou ao mundo sua Teoria da Relatividade Geral [84].

Nesta nova visão, a gravidade não é uma força no sentido tradicional, mas uma manifestação da curvatura do tecido do espaço-tempo, causada pela presença de massa e energia. A analogia clássica é a de uma bola de boliche (representando um corpo massivo) colocada sobre um lençol esticado (representando o espaço-tempo), que cria uma depressão na qual objetos menores, como bolas de gude, têm suas trajetórias desviadas — elas "orbitam" a depressão. A matéria dita a geometria do espaço, e essa geometria dita o movimento da matéria. A teoria é encapsulada matematicamente nas Equações de Campo de Einstein, que relacionam a geometria do espaço-tempo (através do tensor de curvatura de Einstein, $G_{\mu\nu}$) à distribuição de matéria e energia (através do tensor de energia-momento, $T_{\mu\nu}$). A famosa máxima de John Archibald Wheeler resume a essência da teoria: "O espaço-tempo diz à matéria como se mover; a matéria diz ao espaço-tempo como se curvar" [85].

[12] O periélio é o ponto da órbita de um planeta mais próximo do Sol. A teoria de Newton previa que a órbita de Mercúrio giraria (precessão) lentamente devido à influência de outros planetas, mas as observações mostravam um pequeno desvio de 43 segundos de arco por século em relação a essa previsão, uma anomalia que permaneceu inexplicada até a Relatividade Geral.

A Relatividade Geral não só explicou com perfeição a anomalia da órbita de Mercúrio, mas também previu fenômenos inéditos. Dois deles foram cruciais para sua aceitação:

1. A deflexão da luz por corpos massivos, confirmada pela expedição de Arthur Eddington e equipes durante o eclipse solar total de 29 de maio de 1919. As observações, realizadas em Sobral, no Ceará (Brasil), e na Ilha do Príncipe [86, 87], foram decisivas para corroborar a teoria de Einstein e torná-lo uma celebridade mundial.

2. A existência de buracos negros, regiões do espaço-tempo com uma curvatura tão extrema que nada, nem mesmo a luz, pode escapar [88].

As previsões da Relatividade Geral continuaram a ser confirmadas com precisão crescente. Um exemplo cotidiano de sua validade é o Sistema de Posicionamento Global (GPS), que não funcionaria sem as correções relativísticas para a passagem do tempo nos satélites em órbita [89]. Em 14 de setembro de 2015, a colaboração LIGO/Virgo detectou diretamente ondas gravitacionais — ondulações no tecido do espaço-tempo — geradas pela fusão de dois buracos negros a mais de um bilhão de anos-luz de distância. Esta descoberta, publicada na *Physical Review Letters* em 2016 [90], inaugurou uma nova era na astronomia, permitindo-nos "ouvir" os eventos mais violentos do cosmos. Mais recentemente, em 2019, a colaboração Event Horizon Telescope (EHT) capturou a primeira imagem da "sombra" de um buraco negro supermassivo no centro da galáxia M87[13], fornecendo uma prova visual direta da existência desses objetos previstos por Einstein [91].

Apesar desses triunfos, a história da gravitação não está terminada. A Relatividade Geral é uma teoria clássica e é incompatível com a outra grande teoria da física moderna, a Mecânica Quântica. A busca por uma teoria da gravidade quântica, que unifique esses dois pilares e descreva a gravidade em escalas de energia extremas (como no centro de um buraco negro ou nos instantes iniciais do Big Bang), é o maior desafio em aberto da física fundamental. Teorias candidatas, como a Teoria das Cordas (que postula que as partículas fundamentais são vibrações de cordas em dimensões extras) e a Gravidade Quântica em *Loop* (que propõe que o próprio espaço-tempo é granulado ou "quantizado"), representam a fronteira atual desta longa e fascinante jornada científica [92].

## 2.2 O PROBLEMA DE N-CORPOS E O PRINCÍPIO DA SUPERPOSIÇÃO

Já sabemos como a força da gravidade se comporta quando temos apenas dois corpos. Como ficaria quando um certo corpo de massa $m_1$ está na presença de outros dois corpos, de massas $m_2$ e $m_3$?

A interação gravitacional é uma força que age entre pares de corpos. Nesse caso, o corpo 1 (de massa $m_1$) sente de forma independente cada uma das forças exercidas pelos corpos 2 e 3. A força resultante é a soma vetorial das duas forças. Esse é o chamado princípio da superposição [93]. Generalizando para um objeto na presença de outros $N$ objetos, escrevemos

$$\vec{F}_R = \vec{F}_1 + \vec{F}_2 + \vec{F}_3 + \cdots + \vec{F}_N, \tag{2.4}$$

ou, de forma sintética,

[13] A Messier 87 (M87) é uma galáxia elíptica gigante situada a uma distância de $16,8 \pm 0,8$ Mpc da Terra. É conhecida por seu jato relativístico e por abrigar um buraco negro supermassivo central com massa estimada em $M = (6,5 \pm 0,7) \times 10^9 M_\odot$ [91].

$$\vec{F}_R = \sum_{k=1}^{N} \vec{F}_k. \tag{2.5}$$

### 2.2.1 O problema dos três corpos e a natureza caótica

Enquanto o problema de dois corpos, sob a gravitação newtoniana, possui uma solução analítica exata que descreve órbitas cônicas (elipses, parábolas ou hipérboles) [94], o mesmo não ocorre quando um terceiro corpo é introduzido. O Problema dos Três Corpos consiste em determinar o movimento de três corpos celestes, considerando apenas sua interação gravitacional mútua, a partir de suas posições e velocidades iniciais.

A complexidade surge do fato de que as equações de movimento para cada corpo dependem, a todo instante, da posição dos outros dois. Isso cria um sistema de equações diferenciais acopladas e não-lineares que não possui uma solução geral em forma fechada [35]. No final do século XIX, o matemático Henri Poincaré, ao estudar este problema, descobriu que o sistema exibe uma dependência sensível às condições iniciais, uma característica fundamental dos sistemas caóticos [95]. Isso significa que uma alteração ínfima na posição ou velocidade inicial de um dos corpos pode levar a trajetórias drasticamente diferentes em um tempo futuro [96].

A ausência de uma solução analítica torna a simulação numérica, como a desenvolvida neste trabalho, a única ferramenta viável para explorar a rica e complexa dinâmica do sistema de três corpos, que pode resultar em órbitas estáveis, trocas de parceiros orbitais ou a ejeção de um dos corpos do sistema [97].

### 2.2.2 O princípio da superposição na gravitação

Matematicamente, em um sistema de N corpos, a força resultante $\vec{F}_{\text{res},i}$ sobre o $i$-ésimo corpo é:

$$\vec{F}_{\text{res},i} = \vec{F}_{i\leftarrow j} + \vec{F}_{i\leftarrow k} + \cdots = \sum_{j\neq i} \vec{F}_{i\leftarrow j}. \tag{2.6}$$

Esta equação é a base para o cálculo da aceleração $\vec{a}_i$ na equação 4.2. Na implementação da simulação, a cada passo de tempo, o "motor de física" itera sobre cada corpo e, para cada um deles, calcula a soma vetorial das forças exercidas por todos os outros corpos. Apenas após a determinação dessa força resultante é que o método de Runge-Kutta é aplicado para calcular a evolução da posição e da velocidade. A correta aplicação do princípio da superposição é, portanto, fundamental para a validade física da simulação.

O objetivo desta seção é derivar as equações diferenciais (equações do movimento) que regem o movimento de cada um dos objetos em um sistema de três corpos. Esse tipo de problema não tem solução analítica exata, mas é possível resolver numericamente (de forma aproximada) usando o processamento computacional.

O sistema de três corpos está apresentado na Fig. 2.8. Basicamente, para chegar nas equações do movimento, precisamos aplicar a segunda lei de Newton para cada um dos três

corpos. Escrevemos então

$$\vec{F}_1^R = m_1\ddot{\vec{r}}_1 \quad (2.7)$$
$$\vec{F}_2^R = m_2\ddot{\vec{r}}_2 \quad (2.8)$$
$$\vec{F}_3^R = m_3\ddot{\vec{r}}_3 \quad (2.9)$$

Nas equações acima, $\vec{F}_k^R$, $m_k$ e $\vec{r}_k$ são a força resultante agindo no $k$-ésimo corpo, e $m_k$ e $\vec{r}_k$ são suas massas e seus vetores posição no espaço tridimensional, respectivamente.

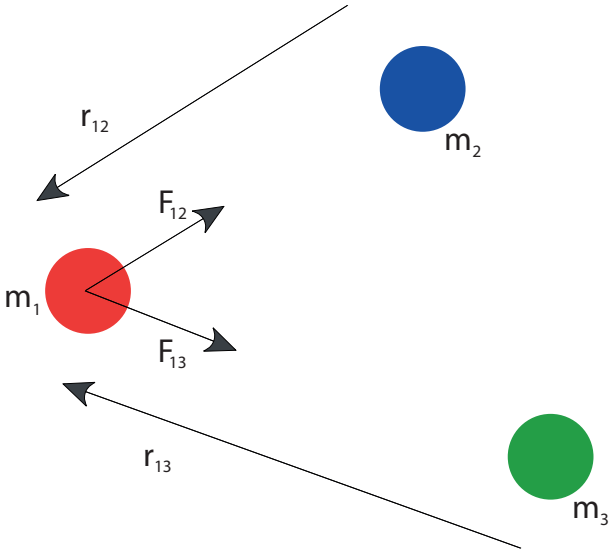


**Figura 2.8:** Ilustração da força de atração gravitacional exercida no corpo de massa $m_1$ devido à presença de um corpo com massa $m_2$ e de um corpo com massa $m_3$. O vetor $\vec{r}_{12}$ está orientado na direção que une o corpo de massa $m_1$ com $m_2$ no sentido do corpo 2 para o corpo 1, valendo o mesmo para o vetor $\vec{r}_{13}$.

Pelo princípio da superposição, cada uma das forças resultantes das equações acima é uma soma vetorial das forças exercidas pelos outros dois corpos no terceiro corpo. Por exemplo, a força resultante no corpo 1 é dada por

$$\vec{F}_1^R = \vec{F}_{12} + \vec{F}_{13} \quad (2.10)$$

Uma forma de chegar nas equações do movimento para cada uma das três componentes ($x$, $y$ e $z$) de cada um dos corpos é decompor as forças nessas três componentes. Ou seja, a equação acima resulta nas seguintes equações:

$$(F_1^R)_x = m_1\ddot{x}_1 \quad (2.11)$$
$$(F_1^R)_y = m_1\ddot{y}_1 \quad (2.12)$$
$$(F_1^R)_z = m_1\ddot{z}_1, \quad (2.13)$$

onde $(F_1^R)_p$ é a $p$-ésima componente da força resultante no corpo 1. O procedimento para os outros corpos é análogo, de forma que chegamos em

$$(F_2^R)_x = m_2\ddot{x}_2 \quad (2.14)$$
$$(F_2^R)_y = m_2\ddot{y}_2 \quad (2.15)$$
$$(F_2^R)_z = m_2\ddot{z}_2 \quad (2.16)$$

$$(F_3^R)_x = m_3\ddot{x}_3 \quad (2.17)$$
$$(F_3^R)_y = m_3\ddot{y}_3 \quad (2.18)$$
$$(F_3^R)_z = m_3\ddot{z}_3 \quad (2.19)$$

## 2.3 FUNDAMENTAÇÃO PEDAGÓGICA: O USO DE SIMULAÇÕES E A BNCC

O desenvolvimento de uma ferramenta tecnológica para a educação, como a proposta neste trabalho, supera o desafio técnico e deve ser firmemente ancorado em princípios pedagógicos que justifiquem seu uso em sala de aula [98]. Esta seção discute o papel dos simuladores como instrumentos para uma aprendizagem ativa e investigativa [99], seu alinhamento com a Base Nacional Comum Curricular (BNCC) e como a plataforma *SimuFísica*® materializa esses conceitos no contexto brasileiro.

### 2.3.1 Simulações como ferramentas para a aprendizagem ativa

A utilização de simulações computacionais no ensino de Física representa uma ruptura com o modelo tradicional de transmissão de conhecimento. Em vez de serem receptores passivos de informação, os estudantes se tornam agentes ativos na construção do seu próprio entendimento. Esta abordagem se alinha com teorias construtivistas da aprendizagem, que postulam que o conhecimento é construído através da interação do indivíduo com o mundo [100].

Simuladores como os da plataforma *SimuFísica*® são particularmente eficazes por permitirem:

- **A visualização do invisível:** Conceitos fundamentais da Física, como forças, campos e trajetórias complexas, são abstratos e de difícil visualização. As simulações tornam esses elementos concretos e manipuláveis, auxiliando na criação de modelos mentais mais acurados [101].
- **A experimentação segura e ilimitada:** Os estudantes podem realizar "experimentos" que seriam impossíveis, perigosos ou caros em um laboratório de verdade. Eles podem explorar cenários extremos, como a colisão de corpos celestes ou o comportamento da matéria em condições ideais (ex: sem atrito), sem qualquer obstáculo [102].
- **A aprendizagem por descoberta e investigação:** O caráter interativo das simulações incentiva a formulação de hipóteses. O estudante pode se perguntar "O que acontece se eu dobrar a massa deste corpo?" ou "Qual velocidade inicial é necessária para que este planeta entre em órbita?". Ao testar essas perguntas e observar os resultados imediatamente, o estudante engaja em um ciclo de investigação científica autêntica [103].

### 2.3.2 O papel das simulações interativas no ensino de física

A integração de tecnologias digitais no processo de ensino-aprendizagem é uma diretriz fundamental para a educação contemporânea. A Base Nacional Comum Curricular (BNCC), nesse sentido, incentiva a incorporação de recursos tecnológicos, como as simulações computacionais, como meio para superar desafios pedagógicos e construir uma aprendizagem mais dinâmica e significativa para os estudantes.

No cenário internacional, um vasto ecossistema de plataformas se consolidou, oferecendo laboratórios virtuais que permitem a exploração de fenômenos físicos. Destacam-se o *PhET Interactive Simulations* [104], conhecido por sua abordagem investigativa, e o *GeoGebra* [105], que integra geometria, álgebra e cálculo de forma interativa. Outras ferramentas como *Algodoo* [106], *The Physics Classroom* [107] e *Open Source Physics* (OSP) [108] também fornecem recursos robustos para a visualização de conceitos. No Brasil, iniciativas como o *Laboratório Virtual de Física da UFC* [109] demonstram o investimento nacional na área. É nesse contexto de inovação que se insere a plataforma *SimuFísica*® [4], um ambiente virtual focado nas necessidades do ensino de Física no Ensino Médio e Superior do país.

Dentre os diversos temas da Física, o ensino da Gravitação Universal apresenta desafios pedagógicos notáveis, dada a sua natureza abstrata e a dificuldade em visualizar as interações multipartículas no espaço tridimensional. Pesquisas corroboram que o uso de simulações é particularmente eficaz na abordagem de temas que envolvem campos e interações à distância, como os campos gravitacionais, elétricos e magnéticos [110].

### 2.3.3 Vantagens das simulações 3D no ensino de gravitação

A utilização de simulações tridimensionais no ensino de Física tem se consolidado como uma estratégia didática particularmente eficaz para a compreensão de fenômenos cuja natureza é essencialmente espacial. No caso específico da gravitação, essa abordagem apresenta vantagens pedagógicas significativas em relação às representações bidimensionais tradicionais, uma vez que os sistemas gravitacionais envolvem movimento orbital, profundidade, orientação espacial, vetores posição e interações simultâneas entre múltiplos corpos distribuídos em diferentes planos.

Estudos comparativos entre mídias 2D e 3D evidenciam que ambientes tridimensionais favorecem o desenvolvimento da visualização espacial, da rotação mental e da construção de modelos cognitivos mais robustos, contribuindo para uma aprendizagem conceitual mais próxima da realidade. Recursos 3D apresentam maior eficácia no aprimoramento das habilidades espaciais dos estudantes, aspecto essencial para a compreensão de fenômenos astronômicos e gravitacionais [111].

No ensino de gravitação, a superioridade das simulações 3D torna-se ainda mais evidente, pois a própria interação gravitacional depende da distância espacial entre os corpos, expressa classicamente em três dimensões. Enquanto representações bidimensionais limitam a percepção de profundidade e tendem a reduzir a complexidade do fenômeno a um plano cartesiano, a modelagem tridimensional permite ao estudante explorar órbitas inclinadas, perturbações gravitacionais, planos orbitais distintos e comportamentos caóticos, especialmente no contexto do problema dos três corpos.

Além da fidelidade física, a simulação 3D favorece uma postura investigativa por parte do estudante, que pode manipular variáveis como massa, posição inicial e velocidade, observar diferentes perspectivas do sistema e formular hipóteses sobre a evolução dinâmica das órbitas. Essa possibilidade de interação fortalece princípios das metodologias ativas, promovendo maior engajamento, autonomia intelectual e aprendizagem significativa.

Outro aspecto central reside na aproximação entre a representação didática e a realidade dos sistemas celestes. Embora as órbitas dos planetas do Sistema Solar apresentem relativa coplanaridade[14], elas não são rigorosamente planas. Dados da NASA demonstram que essas inclinações orbitais variam desde aproximadamente 0,77°, no caso de Urano, até 7,00° em Mercúrio, enquanto Plutão apresenta uma inclinação significativamente maior, da ordem de 17,16° [112]. Esses valores evidenciam que mesmo sistemas considerados quase coplanares possuem natureza tridimensional, reforçando a limitação conceitual de representações exclusivamente bidimensionais.

**Tabela 2.1:** Inclinação orbital dos planetas do Sistema Solar em relação ao plano da eclíptica

| **Planeta** | **Inclinação (°)** |
|---|---|
| Mercúrio | 7,00 |
| Vênus | 3,39 |
| Terra | 0,00 |
| Marte | 1,85 |
| Júpiter | 1,30 |
| Saturno | 2,49 |
| Urano | 0,77 |
| Netuno | 1,77 |
| Plutão | 17,16 |

Fonte: NASA Planet Compare [112].

A Tabela 2.1 reforça que os corpos celestes não se movem em um plano rigorosamente único, mas em um espaço tridimensional no qual pequenas inclinações já são suficientes para alterar significativamente a percepção espacial das órbitas. Sob essa perspectiva, as simulações 3D não representam apenas um aprimoramento visual em relação às simulações 2D, mas uma representação epistemologicamente mais fiel do fenômeno físico estudado.

Nesse contexto, as simulações tridimensionais transcendem a função meramente ilustrativa e consolidam-se como instrumentos pedagógicos capazes de integrar visualização, experimentação e modelagem computacional. Para o ensino da gravitação, essa abordagem representa uma evolução metodológica significativa, pois aproxima o estudante do comportamento real dos sistemas físicos, potencializando a compreensão de conceitos complexos como força gravitacional, conservação do momento angular, órbitas elípticas, perturbações orbitais e dinâmica caótica.

[14] Diz-se que um conjunto de corpos é coplanar quando suas órbitas estão contidas, total ou aproximadamente, em um mesmo plano geométrico. No Sistema Solar, os planetas apresentam pequenas inclinações orbitais em relação ao plano da eclíptica, sendo frequentemente considerados aproximadamente coplanares para fins de modelagem e simulação.

# 3 O ECOSSISTEMA *SIMUFÍSICA*® E SEU VALOR PEDAGÓGICO

## 3.1 O SIMUFÍSICA®: HISTÓRICO E PUBLICAÇÕES ASSOCIADAS

O *SimuFísica*® se propõe a tornar o ensino de física mais interativo e eficaz, disponibilizando um portfólio de simulações que cobrem diversas áreas da física, como mecânica, termologia, óptica, ondulatória e eletromagnetismo. Na Fig. 3.1 podemos visualizar a *interface* da plataforma. Desenvolvida com tecnologias web como HTML5, JavaScript[1] e PHP[2], a plataforma garante ampla acessibilidade, funcionando diretamente no navegador em diversos dispositivos (computadores, *tablets* e *smartphones*) sem a necessidade de instalação de *plugins* ou *softwares* adicionais.

A consolidação e validação acadêmica de suas aplicações ocorreram por meio de uma série de publicações em periódicos especializados e servidores de *preprints*[3]. Esta seção detalha os principais aplicativos da plataforma que foram objeto dessas publicações, abrangendo áreas como eletromagnetismo, mecânica e termodinâmica, e destacando tanto simuladores de fenômenos físicos quanto ferramentas de análise experimental em tempo real.

### 3.1.1 Motor elétrico – *SimuFísica*®: um aplicativo para o ensino de eletromagnetismo

O ensino de eletromagnetismo pode ser otimizado com o uso de simulações interativas que demonstrem a aplicação de conceitos teóricos de forma didática e visual. Nesse contexto, foi desenvolvido o "Motor elétrico", um aplicativo de simulação da plataforma *SimuFísica*® projetado para ser utilizado em salas de aula dos ensinos médio e superior [113].

O aplicativo consiste em um simulador de motor elétrico de corrente contínua, que apresenta uma bobina com espiras retangulares imersa em um campo magnético uniforme gerado por um par de ímãs, como pode ser visto na Fig. 3.2. Uma das principais vantagens do simulador é a interatividade, que o diferencia de vídeos e animações. Ele permite ao usuário configurar uma variedade de parâmetros e condições iniciais, como a tensão da fonte, a intensidade do campo magnético, o número de espiras e a massa da bobina e de forma investigativa analisar como o simulador reage.

[1] HTML5 (*HyperText Markup Language 5*) é a linguagem de marcação usada para estruturar o conteúdo de uma página web (títulos, parágrafos, etc.), enquanto JavaScript é a linguagem de programação que permite adicionar interatividade e lógica a essa página. Juntas, elas formam a base para a criação de aplicações web complexas e interativas, como as simulações da plataforma.

[2] PHP (um acrônimo recursivo para *PHP: Hypertext Preprocessor*) é uma linguagem de script de código aberto amplamente utilizada no desenvolvimento web. Diferente do JavaScript, que é executado no lado do cliente (navegador), o PHP é processado no lado do servidor, sendo responsável por tarefas como a comunicação com bancos de dados, processamento de formulários e geração dinâmica de conteúdo HTML.

[3] Servidores de preprints são repositórios digitais de acesso aberto onde pesquisadores disponibilizam versões preliminares de seus artigos científicos antes de serem submetidos ou publicados em periódicos com revisão por pares (*peer review*). Essas plataformas permitem a disseminação rápida de resultados de pesquisa e o estabelecimento de prioridade intelectual sobre novas descobertas. No campo da Física, o repositório arXiv.org, mantido pela *Cornell University*, é a plataforma mais consolidada e utilizada mundialmente.


**Figura 3.1:** Interface da plataforma *SimuFísica*®, ferramenta educacional interativa que reúne diversas simulações computacionais e laboratórios virtuais para o ensino de Física no ensino médio e superior. A versão apresentada (2.1.2, de 17/05/2025) foi desenvolvida por colaboradores da Universidade Federal de Rondônia (UNIR), com apoio de agências como CNPq, FAPERO e UNIR.

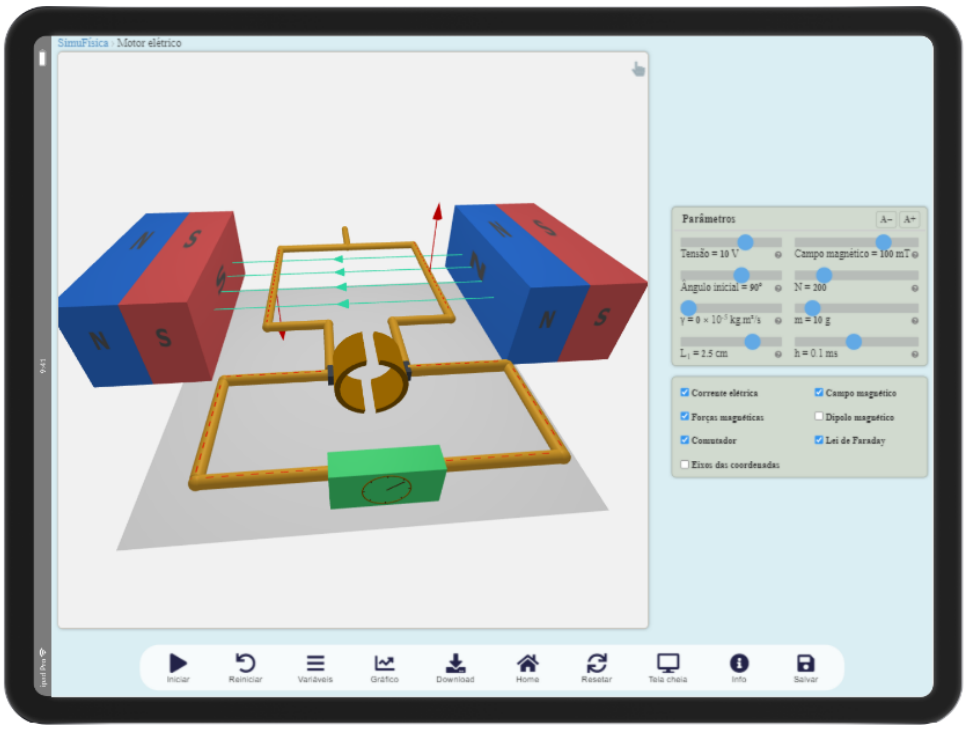


**Figura 3.2:** Interface do aplicativo Motor Elétrico, exibindo os ímãs, a bobina e os vetores de força magnética (setas vermelhas) e campo magnético (setas verdes). Link para acesso: https://simufisica.com/simulacoes/motor-eletrico/.

A simulação é regida por um sistema de equações diferenciais ordinárias que descrevem a dinâmica do motor. O torque que impulsiona a rotação da bobina é gerado pelas forças magné-

ticas que atuam nos lados das espiras. O módulo desse torque é dado por $\tau = iNL_1L_2B\sin\theta$, onde $i$ é a corrente, $N$ o número de espiras, $L_1$ e $L_2$ as dimensões da espira, $B$ o campo magnético e $\theta$ o ângulo entre o vetor normal ao plano da bobina e $B$. O aplicativo também considera a força eletromotriz induzida pela Lei de Faraday. Para simplificar o cálculo e garantir o bom funcionamento em dispositivos com menor capacidade de processamento.

O uso do aplicativo em ambiente educacional é justificado por sua capacidade de facilitar a compreensão de conceitos abstratos de forma interativa. A ferramenta está alinhada às competências da Base Nacional Comum Curricular (BNCC) para o Ensino Médio, que preveem a compreensão do funcionamento de motores elétricos e o uso de aplicativos digitais no processo de aprendizagem. No Ensino Superior, pode ser utilizado para aprofundar conceitos como força magnética, lei de Faraday e dipolo magnético [113].

### 3.1.2 O simulador conservação de energia mecânica – *SimuFísica*®

O princípio da conservação da energia mecânica é um dos tópicos da Física cujo ensino pode ser significativamente enriquecido com o apoio de recursos didáticos interativos. Alinhado a essa necessidade e às diretrizes da Base Nacional Comum Curricular (BNCC), que incentivam o uso de simulações computacionais, a plataforma *SimuFísica*® disponibiliza o aplicativo Conservação de Energia Mecânica [14, 114], mostrado na Fig. 3.3.

Este simulador foi projetado para ilustrar as transformações entre as energias cinética, potencial gravitacional e potencial elástica. A ferramenta simula o deslocamento de um objeto puntiforme ao longo de uma trajetória unidimensional com diferentes níveis de altura, permitindo a análise de como a energia se transforma e se conserva.

O simulador é configurável, permitindo ao usuário ajustar parâmetros como as alturas ao longo da trajetória, a massa do objeto, a velocidade inicial, a constante elástica de uma mola e até mesmo o coeficiente de atrito em um trecho específico. Em tempo real, o aplicativo exibe a velocidade e a altura do objeto, além de um gráfico de barras que representa as diferentes formas de energia e a energia mecânica total, facilitando a compreensão intuitiva dos princípios físicos envolvidos.

Para agilizar o uso em sala de aula, o aplicativo conta com configurações pré-montadas, através do botão “Exemplos”, que ilustram cenários como “Vale com atrito” e “Descida com mola”. A ferramenta se destaca como um *software* com grande potencial para apoiar a resolução de problemas e a visualização de fenômenos abstratos relacionados à conservação de energia mecânica [114]. As equações diferenciais que regem dinâmica do simulador podem ser encontradas no apêndice da Ref. [115].

### 3.1.3 Explorando trocas de calor com o simulador de calorimetria – *SimuFísica*®

O aplicativo Calorimetria é um simulador interativo projetado para o ensino de processos de trocas de calor, sendo aplicável tanto no ensino médio quanto nos anos iniciais da graduação. Esta ferramenta (Fig. 3.4) possibilita a exploração dinâmica e audiovisual de fenômenos como o aquecimento de líquidos e o estabelecimento do equilíbrio térmico. Há dois modos de operação

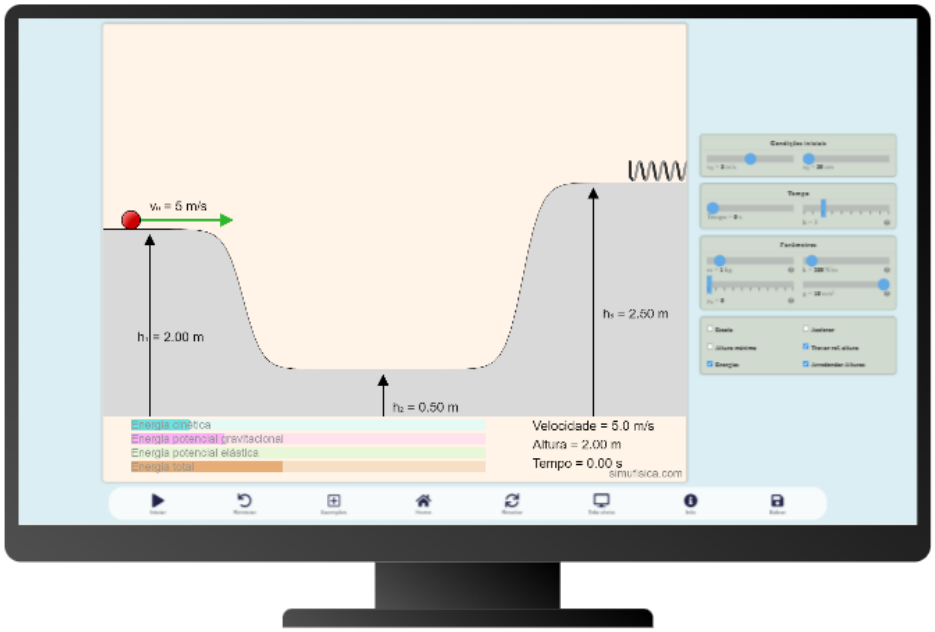


**Figura 3.3:** Interface do aplicativo Conservação de Energia Mecânica da plataforma *SimuFísica*®, desenvolvido para auxiliar o ensino da conservação da energia mecânica. O simulador permite visualizar, em tempo real, as transformações entre energia cinética, potencial gravitacional e potencial elástica, além de possibilitar a configuração de parâmetros físicos e a análise gráfica das energias do sistema. Acesso disponível em: `https://simufisica.com/simulacoes/conservacao-energia-mecanica/`.

principais: Aquecimento e Equilíbrio Térmico. No modo de Aquecimento, o usuário pode observar a transferência de energia para um líquido aquecido por uma chama com potência ajustável, configurando parâmetros como massa, calor específico e calor latente de vaporização. O segundo modo simula a imersão de uma esfera metálica aquecida em um líquido mais frio, permitindo analisar a troca de calor até que o equilíbrio térmico seja alcançado [116].

A física implementada no simulador é baseada nas equações de calor sensível ($Q = mc\Delta T$), calor latente de vaporização ($Q = mL$) e na lei de resfriamento de Newton, que descreve a evolução térmica do sistema. Essa fundamentação teórica garante a fidelidade das representações, permitindo que os estudantes comparem resultados de cálculos analíticos com os dados gerados pela simulação, como demonstrado pelos exemplos de aplicação presentes no artigo.

Recursos pedagógicos como gráficos da evolução temporal das temperaturas, efeitos sonoros e configurações predefinidas (por exemplo, “Aquecimento de Etanol”) enriquecem a experiência de aprendizagem e promovem o engajamento. A ferramenta se mostra, portanto, um recurso valioso para apoiar a instrução conceitual e para o desenvolvimento de atividades investigativas, alinhando-se às modernas abordagens do ensino de ciências [116].

### 3.1.4 Pêndulo filmado - *SimuFísica*®: uma ferramenta para medições em tempo real

O estudo de pêndulos é fundamental no ensino de física, mas muitos estudantes apresentam dificuldade em conectar o movimento físico observado com suas representações gráficas

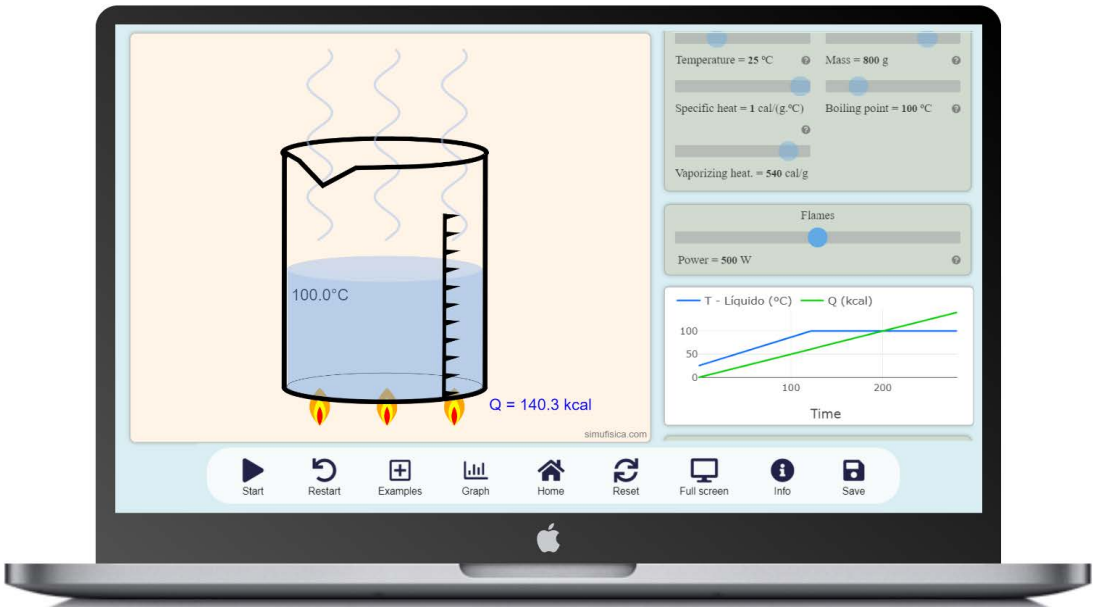


**Figura 3.4:** Interface do simulador Calorimetria da plataforma *SimuFísica*®, configurado no modo Aquecimento. O aplicativo permite visualizar processos de transferência de calor, incluindo aquecimento de líquidos e equilíbrio térmico, por meio da configuração de parâmetros físicos e da análise gráfica da evolução térmica do sistema. Acesso disponível em: `https://simufisica.com/en/calorimetry/`.

abstratas. Para fechar essa lacuna, foi incluído na plataforma *SimuFísica*® o aplicativo Pêndulo Filmado, ferramenta baseada em visão computacional[4] que permite a medição em tempo real do movimento oscilatório de um pêndulo [3].

O aplicativo funciona diretamente no navegador em computadores, *tablets* e *smartphones*, utilizando a câmera do dispositivo para capturar e analisar o movimento. Desenvolvida com a biblioteca *OpenCV.js*[5], a aplicação detecta automaticamente a posição do pêndulo e exibe, em tempo real, o gráfico do ângulo em função do tempo, além de estimativas do período de oscilação (ver Fig. 3.5). Essa abordagem em tempo real representa uma vantagem sobre ferramentas tradicionais de videoanálise, que geralmente exigem uma análise manual quadro a quadro após a gravação.

O aplicativo demonstra a eficácia da aplicação em diversos contextos experimentais, como a medição do período, a determinação da aceleração da gravidade e a análise de oscilações amortecidas, com resultados que mostram excelente concordância com as previsões teóricas [3].

[4] Visão Computacional é um campo da inteligência artificial que treina computadores para interpretar e compreender o mundo visual. Usando imagens digitais de câmeras e vídeos, as máquinas podem identificar e processar objetos com precisão — neste caso, a posição do pêndulo.

[5] *OpenCV.js* é uma biblioteca de visão computacional para *JavaScript*, compilada diretamente do código-fonte original em C++ da *OpenCV* através do *Emscripten* para o formato *WebAssembly* (Wasm). Esta abordagem permite a execução de algoritmos com performance próxima à nativa no navegador, sendo uma ferramenta poderosa para aplicações web que exigem análise de vídeo em tempo real, realidade aumentada ou processamento de imagem interativo no lado do cliente.


**Figura 3.5:** Interface do aplicativo Pêndulo Filmado da plataforma *SimuFísica*® em execução em um *smartphone*. A aplicação utiliza visão computacional para detectar automaticamente o movimento do pêndulo e exibir, em tempo real, o gráfico do ângulo em função do tempo, além de estimativas do período de oscilação. Acesso disponível em: `https://simufisica.com/simulacoes/pendulo-filmado/`.

## 3.2 RELEVÂNCIA NO CONTEXTO EDUCACIONAL BRASILEIRO

A relevância da plataforma *SimuFísica*® se manifesta em múltiplos níveis, especialmente ao considerarmos os desafios da educação no Brasil.

- **Alinhamento curricular e linguístico:** Diferentemente de muitas plataformas internacionais que são apenas traduzidas, o *SimuFísica*® foi concebido por e para o contexto brasileiro. Seu conteúdo, exemplos e linguagem dialogam diretamente com a realidade de estudantes e professores do país, bem como com as competências e habilidades preconizadas pela Base Nacional Comum Curricular (BNCC).

- **Democratização do acesso a laboratórios:** Em um cenário onde muitas escolas públicas carecem de laboratórios de física bem equipados, a plataforma funciona como um laboratório virtual. Ela permite que estudantes realizem experimentos e simulações, manipulem variáveis e visualizem fenômenos físicos que, de outra forma, seriam restritos à abstração teórica do quadro e do livro didático. Este aspecto é fundamental para a democratização de uma educação científica mais prática e investigativa.

- **Recurso para formação de professores:** Além de ser uma ferramenta para o estudante, a plataforma serve como um recurso para a formação inicial e continuada de professores de Física, que podem explorar novas abordagens didáticas e metodologias ativas baseadas em simulação.

No cenário mundial, enquanto plataformas como PhET [104] estabeleceram o padrão para simulações interativas, o *SimuFísica*® se destaca por seu foco regionalizado, demonstrando que o Brasil pode desenvolver suas próprias tecnologias educacionais adaptadas para atender

a necessidades curriculares e culturais específicas, um movimento crescente na comunidade global de *EdTech*[6].

## 3.3 O SIMULADOR GRAVITAÇÃO 3D NO ECOSSISTEMA *SIMUFÍSICA*®

O desenvolvimento da simulação do problema de três corpos, tema deste trabalho, foi planejado especificamente para agregar valor e expandir as fronteiras da plataforma *SimuFísica*® de maneiras significativas:

1. **Inovação em visualização e interação:** Embora a plataforma SimuFísica® já disponibilize um competente simulador bidimensional para a gravitação (Fig. 3.6), que permite a manipulação de variáveis como massa, velocidade e número de corpos, a transição para um ambiente tridimensional (3D) trará novas perspectivas para o aprendizado, visto que fenômenos orbitais, em especial o problema de três ou mais corpos, envolvem dinâmicas que nem sempre se restringem a um único plano, apresentando inclinações orbitais fundamentais para a compreensão do sistema. Além disso, um simulador tridimensional possibilita maior interatividade com a cena via rotação e zoom, permitindo que o estudante desenvolva uma intuição espacial mais aprimorada, visualizando a dinâmica sob múltiplas perspectivas e construindo um modelo mental mais próximo à natureza tridimensional do universo.

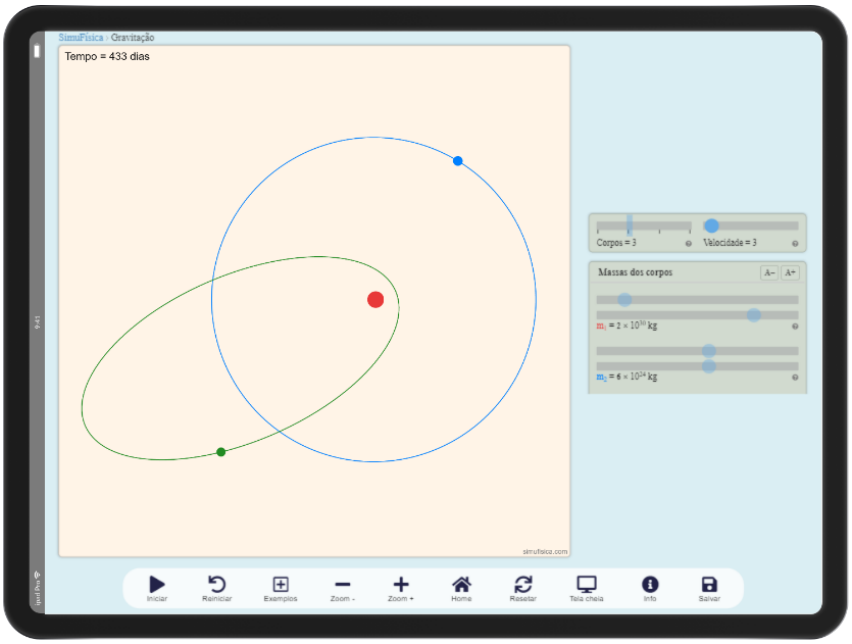


**Figura 3.6:** *Interface* do Simulador Gravitação da plataforma SimuFísica® em execução em um *tablet*. Link para acesso: https://simufisica.com/simulacoes/gravitacao/.

2. **Abordagem de um tópico avançado e desafiador:** A gravitação, em particular o problema de N-corpos, é um tópico de alta complexidade conceitual. Ao incorporar uma fer-

[6]O termo *EdTech* (amálgama das palavras inglesas *education* e *technology*) refere-se ao uso estratégico de recursos tecnológicos — incluindo *softwares*, *hardwares* e metodologias digitais — com o objetivo de facilitar a aprendizagem.

ramenta tridimensional para explorar este tema, o *SimuFísica*® amplia seu alcance para além do currículo padrão do Ensino Médio, tornando-se um recurso ainda mais valioso para disciplinas de Mecânica Clássica no Ensino Superior e para estudantes interessados em aprofundar seus conhecimentos.

3. **Exploração de fenômenos caóticos:** Conforme discutido na subseção 2.2.1, o problema de três corpos é um exemplo clássico de sistema caótico. A simulação proposta permite que os usuários visualizem de forma direta e intuitiva a "dependência sensível às condições iniciais", um conceito essencial em sistemas dinâmicos. Esta ferramenta oferece uma oportunidade na plataforma para discutir a natureza da previsibilidade e do caos, temas que raramente são abordados de forma experimental no ensino básico ou superior.

Em suma, este TCC não se limita a criar um software isolado, mas a enriquecer um ecossistema educacional já consolidado e relevante. A nova simulação pretende fortalecer o *SimuFísica*®, introduzindo novas dimensões de visualização, complexidade temática e profundidade conceitual, e reforçando sua posição como uma plataforma de vanguarda no ensino de Física no Brasil.

### 3.3.1 Alinhamento com a BNCC

O simulador Gravitação 3D contribui diretamente para o desenvolvimento de várias Habilidades da Base Nacional Comum Curricular (BNCC) [27], notadamente:

**EM13CNT201** *"Analisar e discutir modelos, teorias e leis propostos em diferentes épocas e culturas para comparar distintas explicações sobre o surgimento e a evolução da Vida, da Terra e do Universo com as teorias científicas aceitas atualmente."*

O simulador Gravitação 3D permite representar e explorar modelos de descrição do movimento dos corpos celestes, possibilitando relacionar essas representações com diferentes explicações históricas sobre a estrutura do Universo. Ao simular sistemas gravitacionais, como o movimento de planetas ao redor de estrelas ou de satélites ao redor de planetas, os estudantes podem discutir como modelos científicos foram sendo modificados ao longo do tempo, passando de interpretações baseadas em concepções geocêntricas para modelos heliocêntricos fundamentados em leis físicas.

**EM13CNT202** *"Analisar as diversas formas de manifestação da vida em seus diferentes níveis de organização, bem como as condições ambientais favoráveis e os fatores limitantes a elas, com ou sem o uso de dispositivos e aplicativos digitais (como softwares de simulação e de realidade virtual, entre outros)."*

O simulador Gravitação 3D pode contribuir para a discussão das condições físicas que tornam um planeta potencialmente habitável. Por meio da simulação de sistemas planetários, é possível investigar como a distância entre um planeta e sua estrela influencia sua dinâmica orbital e, consequentemente, as condições ambientais associadas à presença de água líquida em sua superfície. Essa abordagem permite introduzir o conceito de zona habitável, região ao redor de uma estrela onde as condições de temperatura podem ser compatíveis com a existência de água no estado líquido. Ao variar parâmetros como massa da estrela, distância orbital e velocidade inicial dos corpos, os estudantes podem

explorar como essas variáveis influenciam no tamanho das órbitas e discutir sua relação com as condições favoráveis ou limitantes ao surgimento e à manutenção da vida.

**EM13CNT204** *"Elaborar explicações, previsões e cálculos a respeito dos movimentos de objetos na Terra, no Sistema Solar e no Universo com base na análise das interações gravitacionais, com ou sem o uso de dispositivos e aplicativos digitais (como softwares de simulação e de realidade virtual, entre outros)."*

Essa habilidade é diretamente contemplada pelo uso do simulador Gravitação 3D. A ferramenta permite investigar o movimento de corpos submetidos à interação gravitacional, analisando como variáveis como massa, posição inicial e velocidade influenciam as trajetórias resultantes. Ao realizar simulações virtuais, os estudantes podem formular hipóteses sobre o comportamento do sistema, elaborar previsões e comparar essas previsões com os resultados obtidos nas simulações.

**EM13CNT209** *"Analisar a evolução estelar associando-a aos modelos de origem e distribuição dos elementos químicos no Universo, compreendendo suas relações com as condições necessárias ao surgimento de sistemas solares e planetários, suas estruturas e composições e as possibilidades de existência de vida, utilizando representações e simulações, com ou sem o uso de dispositivos e aplicativos digitais (como softwares de simulação e de realidade virtual, entre outros)."*

O simulador Gravitação 3D também pode contribuir para a discussão de aspectos relacionados à formação e evolução de sistemas planetários. Ao explorar sistemas gravitacionais com múltiplos corpos, os estudantes podem analisar como as interações gravitacionais influenciam a organização e a estabilidade de sistemas planetários ao longo do tempo. Essas explorações permitem discutir, em nível introdutório, processos associados à formação de sistemas solares e às condições físicas que podem favorecer a existência de planetas em órbitas estáveis ao redor de estrelas. Dessa forma, o simulador pode servir como recurso didático para relacionar conceitos de gravitação com temas mais amplos da Astronomia, como a formação de sistemas planetários e a evolução do Universo.

**Atividade Sugerida: O Desafio da Órbita Estável.** O professor pode utilizar o simulador para propor um desafio em grupo: "Dado um sistema com duas estrelas em órbita binária (configuração inicial predefinida), qual deve ser a posição e a velocidade inicial de um terceiro corpo (um planeta) para que ele permaneça em uma órbita estável ao redor do par?".

Nesta atividade, os estudantes são levados a:

1. **Investigar:** Eles precisam explorar diferentes configurações, testando hipóteses de forma colaborativa.

2. **Analisar e argumentar:** Ao observar os resultados — que podem ser uma órbita estável, uma interação caótica ou a ejeção do planeta —, os grupos devem analisar o que ocorreu e argumentar sobre o porquê de suas escolhas terem levado àquele resultado.

3. **Desenvolver habilidades:** A atividade mobiliza diretamente as habilidades EM13CNT204 e EM13CNT209 da BNCC, além de promover o raciocínio lógico, a resolução de problemas e o trabalho em equipe.

# 4 O SIMULADOR GRAVITAÇÃO 3D: SOLUÇÃO NUMÉRICA, PLANEJAMENTO E DESENVOLVIMENTO

## 4.1 METODOLOGIA NUMÉRICA: A SOLUÇÃO DO PROBLEMA DE N-CORPOS

A simulação do movimento de três ou mais corpos sob a influência mútua da gravidade, conhecido como problema de N-Corpos, é um desafio clássico da física e da matemática. Exceto por casos muito específicos, não existe uma solução analítica exata para este problema. Portanto, a trajetória dos corpos deve ser determinada através de métodos de integração numérica. Esta seção detalha o tratamento matemático e o algoritmo utilizado para resolver as equações de movimento no "motor de física" da simulação desenvolvida.

### 4.1.1 A equação de movimento e a necessidade do método numérico

O fundamento dinâmico da simulação é a segunda lei de Newton, combinada com a lei da Gravitação Universal. Para um corpo $i$ de massa $m_i$ e vetor posição $\vec{r}_i$, a força resultante $\vec{F}_i$ exercida sobre ele por todos os outros corpos $j$ é dada por:

$$\vec{F}_i = \sum_{j \neq i} G \frac{m_i m_j}{|\vec{r}_j - \vec{r}_i|^3} (\vec{r}_j - \vec{r}_i), \tag{4.1}$$

onde $G$ é a constante gravitacional. Pela segunda lei de Newton, $\vec{F}_i = m_i \vec{a}_i$, onde a aceleração $\vec{a}_i$ é a segunda derivada temporal do vetor posição, $\vec{a}_i = \frac{d^2\vec{r}_i}{dt^2}$. Combinando as equações, obtemos a equação diferencial de segunda ordem que governa o movimento de cada corpo:

$$\frac{d^2\vec{r}_i}{dt^2} = \sum_{j \neq i} G \frac{m_j}{|\vec{r}_j - \vec{r}_i|^3} (\vec{r}_j - \vec{r}_i). \tag{4.2}$$

A equação 4.2 é uma equação diferencial ordinária (EDO) vetorial de segunda ordem. Métodos numéricos padrão, como os da família *Runge-Kutta*, são projetados para resolver EDOs de primeira ordem. Portanto, é necessário primeiro converter esta equação em um sistema de EDOs de primeira ordem.

### 4.1.2 Transformação para um sistema de equações de primeira ordem

A conversão é realizada através da definição de um vetor de estado para cada corpo. Na Mecânica Clássica, o estado de um corpo em qualquer instante de tempo $t$ é completamente descrito por sua posição $\vec{r}(t)$ e sua velocidade $\vec{v}(t)$. Definimos a velocidade como a primeira derivada da posição:

$$\frac{d\vec{r}}{dt} = \vec{v}. \tag{4.3}$$

Esta é a nossa primeira EDO de primeira ordem. A segunda EDO é obtida definindo a aceleração como a primeira derivada da velocidade. A aceleração, $\vec{a}(\vec{r})$, é a função do lado direito da equação 4.2, que depende das posições de todos os corpos na cena:

$$\frac{d\vec{v}}{dt} = \vec{a}(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \sum_{j \neq i} G \frac{m_j}{|\vec{r}_j - \vec{r}_i|^3}(\vec{r}_j - \vec{r}_i). \tag{4.4}$$

Com as equações 4.3 e 4.4, transformamos uma EDO vetorial de segunda ordem em um sistema de duas EDOs vetoriais de primeira ordem acopladas. Agora, podemos aplicar o método de Runge-Kutta para resolver este sistema.

### 4.1.3 O método de *Runge-Kutta* de 4ª ordem (RK4)

Embora existam métodos mais simples, como o método de Euler, eles acumulam erros rapidamente em simulações orbitais, levando a resultados instáveis (órbitas que espiralam para fora ou para dentro). O método de Runge-Kutta de 4ª ordem (RK4) foi escolhido por oferecer um excelente equilíbrio entre precisão, estabilidade e custo computacional [117].

Para uma EDO de primeira ordem genérica $y' = f(t, y)$, o método RK4 avança a solução de um tempo $t_n$ para $t_n + h$ (onde $h$ é o passo de tempo) através da fórmula:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{4.5}$$

onde os coeficientes $k$ são aproximações da inclinação em diferentes pontos do intervalo de tempo $h$:

$$k_1 = f(t_n, y_n) \tag{4.6}$$
$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \tag{4.7}$$
$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \tag{4.8}$$
$$k_4 = f(t_n + h, y_n + hk_3) \tag{4.9}$$

Para aplicar o RK4 ao nosso sistema, tratamos a posição $\vec{r}$ e a velocidade $\vec{v}$ como as variáveis a serem atualizadas. O algoritmo é aplicado simultaneamente a ambas, para cada corpo $i$, a cada passo de tempo $h$:

1. **Calcular os coeficientes $k_1$ (no início do intervalo):**

$$\vec{k}_{1r} = \vec{v}_n \tag{4.10}$$
$$\vec{k}_{1v} = \vec{a}(\vec{r}_n) \tag{4.11}$$

2. **Calcular os coeficientes $k_2$ (no ponto médio, usando $k_1$):**

$$\vec{k}_{2r} = \vec{v}_n + \frac{h}{2}\vec{k}_{1v} \tag{4.12}$$
$$\vec{k}_{2v} = \vec{a}(\vec{r}_n + \frac{h}{2}\vec{k}_{1r}) \tag{4.13}$$

3. **Calcular os coeficientes $k_3$ (no ponto médio, usando $k_2$):**

$$\vec{k}_{3r} = \vec{v}_n + \frac{h}{2}\vec{k}_{2v} \tag{4.14}$$

$$\vec{k}_{3v} = \vec{a}(\vec{r}_n + \frac{h}{2}\vec{k}_{2r}) \tag{4.15}$$

4. **Calcular os coeficientes $k_4$ (no final do intervalo, usando $k_3$):**

$$\vec{k}_{4r} = \vec{v}_n + h\vec{k}_{3v} \tag{4.16}$$

$$\vec{k}_{4v} = \vec{a}(\vec{r}_n + h\vec{k}_{3r}) \tag{4.17}$$

5. **Calcular o estado final (posição e velocidade no tempo $t_{n+1}$):**

$$\vec{r}_{n+1} = \vec{r}_n + \frac{h}{6}(\vec{k}_{1r} + 2\vec{k}_{2r} + 2\vec{k}_{3r} + \vec{k}_{4r}) \tag{4.18}$$

$$\vec{v}_{n+1} = \vec{v}_n + \frac{h}{6}(\vec{k}_{1v} + 2\vec{k}_{2v} + 2\vec{k}_{3v} + \vec{k}_{4v}) \tag{4.19}$$

Note que para calcular cada $\vec{a}(\dots)$, é necessário conhecer as posições de *todos* os corpos naquele sub-passo específico, o que evidencia o acoplamento do sistema.

## 4.2 MODELAGEM DA SIMULAÇÃO COM UML

Antes da fase de codificação, foi fundamental estabelecer uma arquitetura de software para a simulação, planejada e modelada com a *Linguagem de Modelagem Unificada* (UML) [118]. A UML é uma linguagem gráfica padronizada na engenharia de software, utilizada para visualizar, especificar, construir e documentar os artefatos[1] de um sistema.

A adoção desta metodologia foi crucial não apenas por boas práticas de desenvolvimento, mas também pela natureza do projeto: uma simulação física envolve interações complexas entre objetos, cálculos contínuos e uma lógica de estado que precisa ser clara e bem definida. O uso da UML permitiu traduzir os conceitos abstratos da física gravitacional em um design de software concreto e modular[2], garantindo a clareza do escopo e facilitando a manutenção futura da ferramenta.

Para representar o sistema de forma abrangente, foram selecionados quatro diagramas essenciais que, em conjunto, fornecem uma visão completa da arquitetura sob duas perspectivas distintas:

- **Visão estrutural:** Descreve a estrutura estática do sistema, ou seja, as classes, seus atributos, métodos e os relacionamentos entre elas.
- **Visão comportamental:** Mostra a dinâmica do sistema, focando em como os objetos interagem entre si e como os fluxos de trabalho são executados para alcançar as funcionalidades desejadas.

[1] Em engenharia de software, artefatos são os produtos gerados durante o processo de desenvolvimento, que podem incluir diagramas (como os de UML), documentos de requisitos, código-fonte e os próprios executáveis do software.

[2] A modularidade é um princípio de design de software que consiste em dividir um sistema em componentes independentes e intercambiáveis, chamados módulos. Cada módulo encapsula uma responsabilidade específica, facilitando a manutenção, o teste e a reutilização do código.

### 4.2.1 Diagrama de casos de uso

O Diagrama de Casos de Uso modela as funcionalidades do sistema sob a perspectiva de um ator[3] externo, definindo o escopo do software ao responder à pergunta: "O que o sistema faz?". Conforme ilustrado na Fig. 4.1, este diagrama delimita as interações essenciais que um usuário, seja ele estudante ou professor, pode realizar com o simulador.

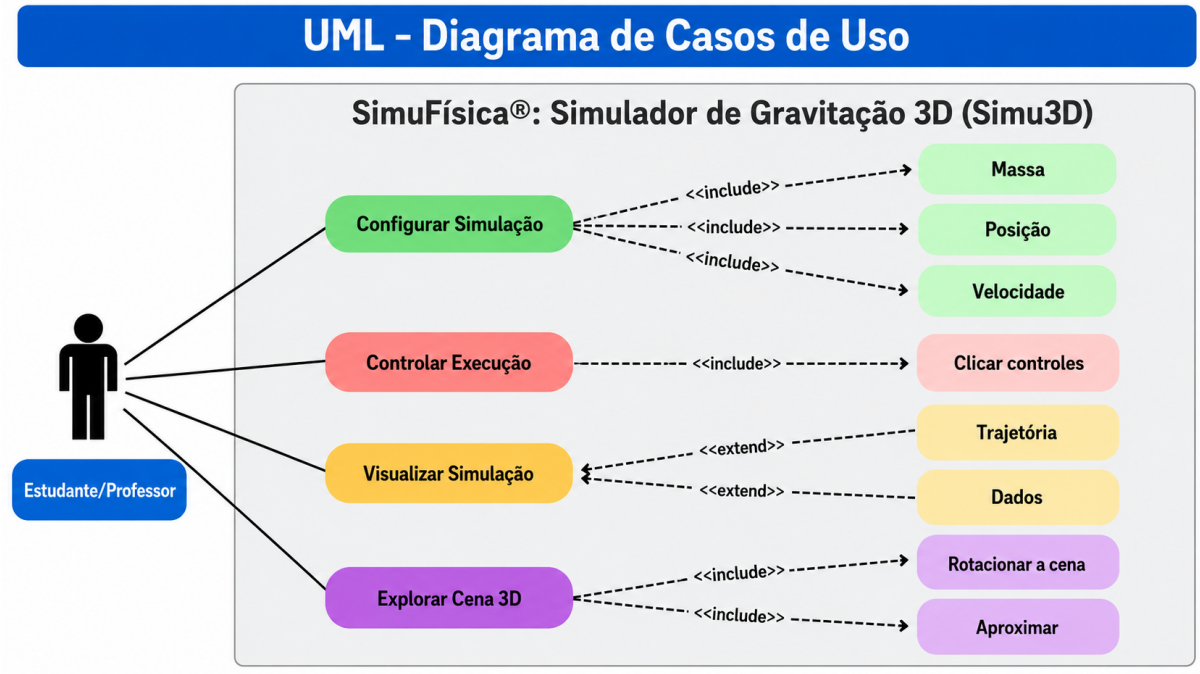


**Figura 4.1:** Diagrama de Casos de Uso do sistema *SimuFísica*®: Simulador de Gravitação 3D (Simu3D), representando as principais interações entre o ator (estudante/professor) e as funcionalidades centrais do simulador, incluindo configuração da simulação, controle da execução, visualização dos resultados e exploração da cena tridimensional.

As funcionalidades representadas no Diagrama de Casos de Uso descrevem as principais interações entre o usuário e o simulador. A funcionalidade **Configurar Simulação** permite ao usuário definir as condições iniciais do sistema, incluindo massa, posição tridimensional $(x, y, z)$ e componentes da velocidade $(v_x, v_y, v_z)$ de cada corpo participante da simulação. Já a funcionalidade **Controlar Execução** possibilita o gerenciamento do fluxo temporal do experimento, permitindo iniciar, pausar, continuar ou reiniciar a simulação conforme necessário.

A funcionalidade **Visualizar Simulação** corresponde à principal interface de saída do sistema, apresentando a renderização tridimensional do movimento orbital obtido a partir da integração numérica das equações físicas implementadas. Complementarmente, a funcionalidade **Explorar Cena 3D** oferece recursos de interação espacial, permitindo ao usuário rotacionar a câmera, aplicar *zoom* e transladar a cena, favorecendo a análise do comportamento dinâmico dos corpos sob diferentes perspectivas.

[3]Em UML, um ator representa um papel desempenhado por uma entidade externa que interage com o sistema. Pode ser um usuário humano, outro sistema de software ou até mesmo um dispositivo de hardware. Neste caso, representa o estudante ou o professor que irá interagir com a simulação.

### 4.2.2 Diagrama de classes

O Diagrama de Classes constitui o pilar da modelagem estrutural em UML, descrevendo a arquitetura estática do sistema ao detalhar as classes[4], seus membros e os relacionamentos de interdependência. No desenvolvimento deste simulador, a modelagem ilustrada na Fig. 4.2 foi fundamental para organizar o código em componentes coesos, aderindo ao Princípio da Responsabilidade Única (SRP)[5].

A arquitetura foi projetada sob uma variação do padrão *Model-View-Controller* (MVC)[6], promovendo a separação entre a lógica, a interface do usuário e o controle de fluxo. As responsabilidades de cada classe estão sintetizadas na Tabela 4.1.

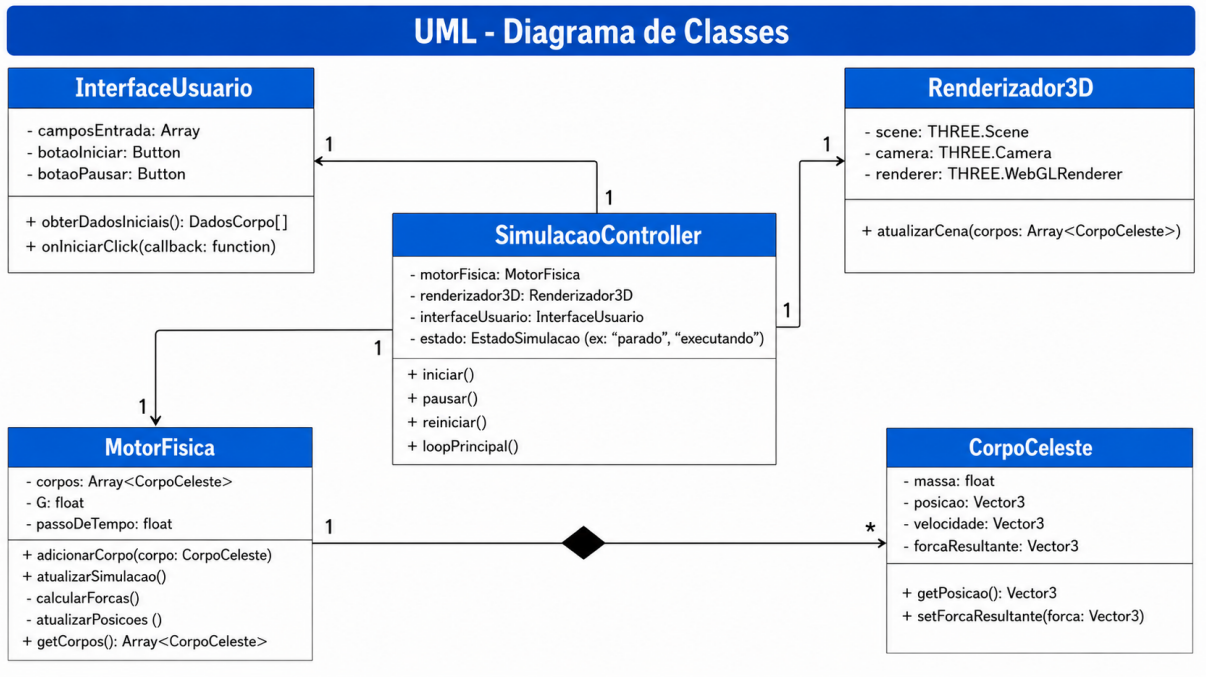


**Figura 4.2:** Diagrama de classes detalhando a arquitetura estática e os relacionamentos fundamentados no padrão MVC.

A conectividade entre as classes reforça o padrão MVC através de simbologias específicas da UML. A conexão de "Associação" (linha contínua) indica que o `SimulacaoController` utiliza os serviços das demais classes como mediador central, mantendo, contudo, ciclos de vida independentes.

Já a "Composição" (losango preenchido) denota uma dependência existencial rigorosa: a relação entre `MotorFisica` e `CorpoCeleste` estabelece que os corpos celestes não possuem razão de existir fora do domínio do motor que gerencia a simulação. O dinamismo resultante dessa estrutura manifesta-se no fluxo de dados ilustrado na Fig. 4.3.

[4]No paradigma de Programação Orientada a Objetos (POO), uma classe funciona como um projeto ou "molde" para a criação de objetos, definindo atributos (estado) e métodos (comportamento).

[5]O SRP (*Single Responsibility Principle*) estabelece que uma classe deve possuir apenas uma razão para mudar, garantindo que sua função seja única e bem definida dentro do ecossistema do software.

[6]O Model-View-Controller (MVC) é um padrão de arquitetura de software originalmente concebido na década de 1970 para o desenvolvimento de interfaces gráficas de usuário. Ele estrutura a aplicação em três partes: o *Model* (os dados e regras), a *View* (a representação visual desses dados) e o *Controller* (que processa as entradas do usuário para manipular o *Model*.

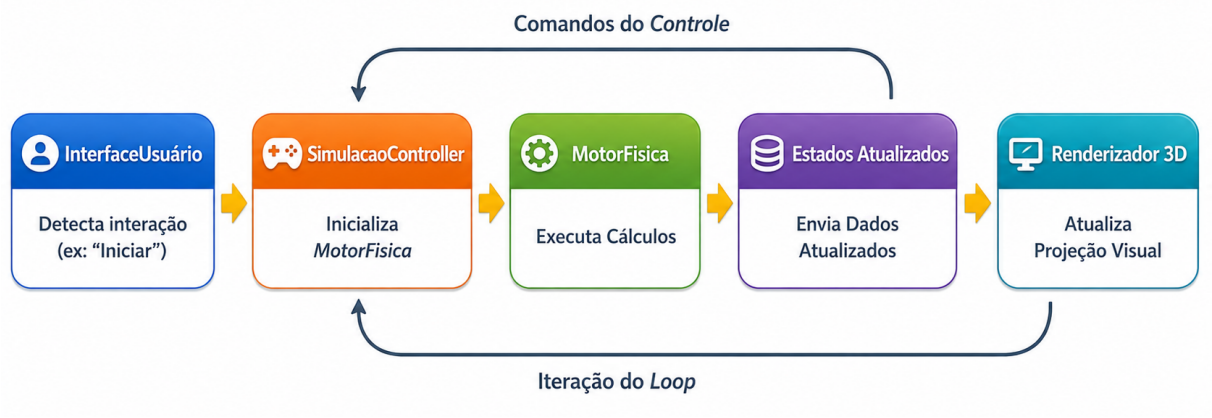


**Figura 4.3:** Representação do fluxo dinâmico de dados entre os componentes da arquitetura MVC.

Esta arquitetura, definida pelos relacionamentos e pelo fluxo de dados, garante o baixo acoplamento entre os componentes. Isso resulta em um sistema coeso, escalável e de fácil manutenção, permitindo, por exemplo, que o motor de renderização seja alterado sem impactar a lógica física da simulação.

**Tabela 4.1:** Arquitetura do Sistema e Responsabilidades das Classes.

| **Classe / Componente** | **Papel (MVC)** | **Descrição e Responsabilidades** |
|---|---|---|
| `SimulacaoController` | *Controller* | Atua como o *Controller* e orquestrador do sistema. É responsável por instanciar e coordenar os demais componentes, gerenciar o estado geral da simulação (parado, executando, pausado) e conter o *loop* principal. Ele faz a ponte entre a interface do usuário e o motor de física, garantindo que permaneçam desacoplados. |
| `InterfaceUsuario (UI)` | *View* | Parte da *View*, esta classe encapsula os elementos da interface gráfica (campos de entrada, botões). Sua função é capturar as ações do usuário e notificar o *Controller*, sem ter conhecimento algum sobre a lógica da simulação. |
| `Renderizador3D` | *View* | A outra parte da *View*, responsável pela representação visual da simulação. Utilizando a biblioteca *Three.js*, ele cria e gerencia a cena 3D. Sua única função é atualizar a tela com base nos dados que recebe do *Controller*, sem alterar o estado da simulação. |
| `MotorFisica` | *Model* | Componente central do *Model*. Esta classe encapsula toda a lógica e os dados da simulação gravitacional. Contém a lista de corpos e os métodos para calcular as forças e atualizar posições e velocidades, operando de forma totalmente independente da interface gráfica. |
| `CorpoCeleste` | *Model* | Classe de dados (*Data Object*) que também compõe o *Model*. Armazena os atributos intrínsecos de cada corpo: massa, posição, velocidade e a força resultante que atua sobre ele em um instante. |

### 4.2.3 Diagrama de sequência

Enquanto o Diagrama de Classes mostra a estrutura estática, o Diagrama de Sequência modela o comportamento dinâmico do sistema. Ele ilustra como os objetos interagem ao longo do tempo através da troca de mensagens, focando em um cenário de uso específico.

O diagrama apresentado na Figura 4.4 detalha a sequência de eventos mais importantes do simulador, que se inicia quando o usuário solicita a execução da simulação.

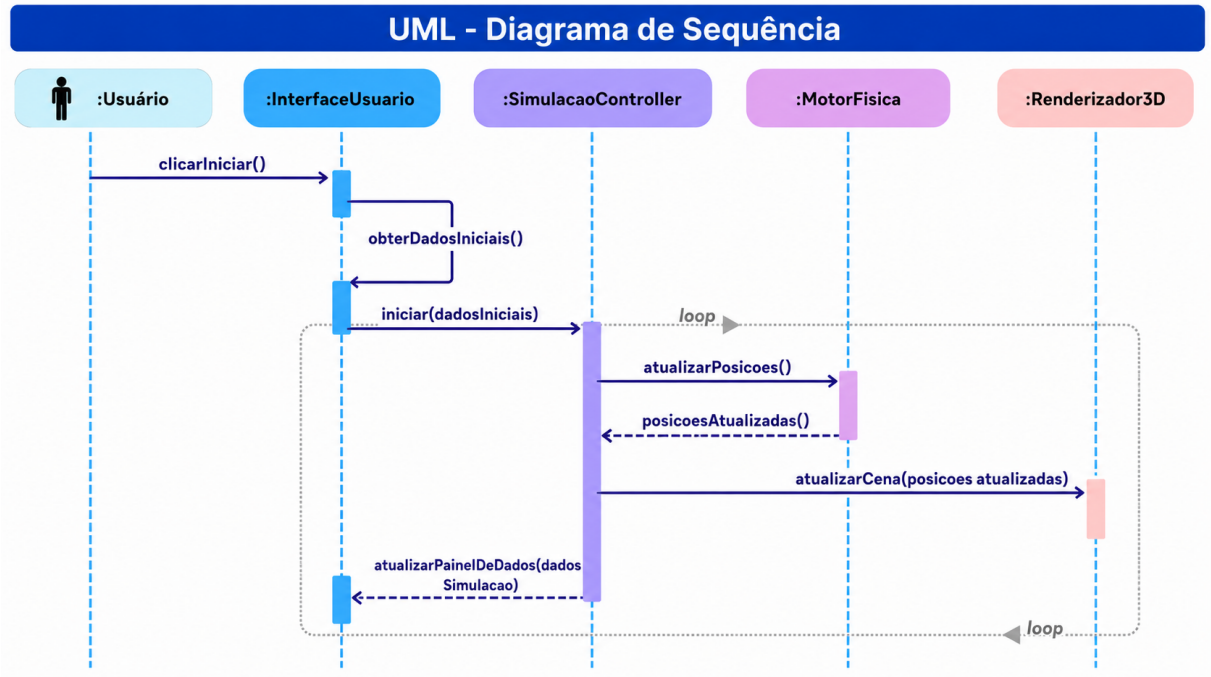


**Figura 4.4:** Diagrama de Sequência demonstrando o fluxo de interação para o início e execução do *loop* da simulação.

A sequência de interações lógicas, traduzida no fluxo de controle do software, está sintetizada na Fig. 4.5. Este esquema evidencia como as requisições transitam entre as camadas do padrão MVC.

A modelagem da sequência é fundamental para assegurar o correto fluxo de controle e, primordialmente, o fluxo de dados que garante o desacoplamento entre o *Model* (motor de física) e a *View* (camada de apresentação). Através desta estrutura, garante-se que o renderizador receba apenas as coordenadas finais calculadas, sem a necessidade de conhecer a complexidade interna do algoritmo de integração numérica.

### 4.2.4 Diagrama de estados

O Diagrama de Estados modela o ciclo de vida do componente central do sistema, descrevendo as diversas configurações em que ele pode se encontrar e os eventos[7] que disparam as

[7]Um evento é um gatilho ou estímulo (como o clique de um botão ou a passagem de tempo) que dispara uma transição, resultando em uma mudança de estado no sistema.

transições. No contexto deste projeto, o Diagrama de Estados (Fig. 4.6) detalha o comportamento do objeto `SimulacaoController`, que opera sob três estados lógicos principais:

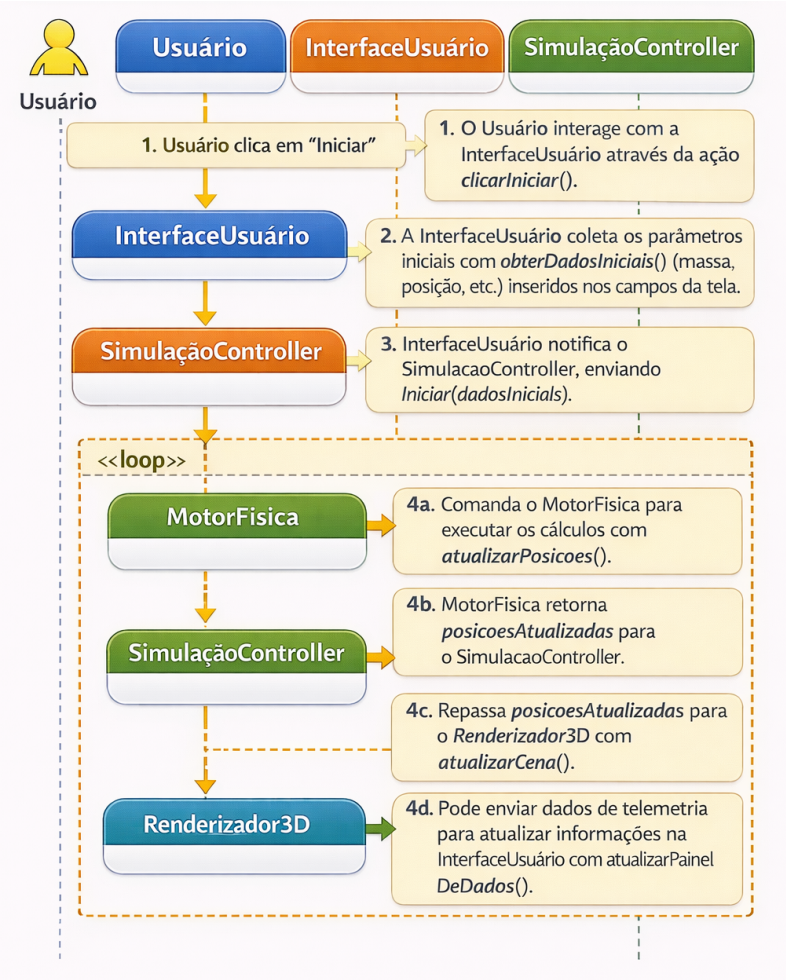


**Figura 4.5:** Representação simplificada da lógica de processamento e atualização de dados quadro a quadro.

Este diagrama demonstra como as interações realizadas na interface do usuário alteram o fluxo de controle interno da aplicação. Os estados principais são definidos como:

- **Parado:** Estado inicial ou de reinicialização. O sistema permanece estático, aguardando a configuração dos parâmetros iniciais e o comando de ativação.
- **Executando:** Estado ativo de processamento. O *loop* de animação é executado continuamente, solicitando ao `MotorFisica` o cálculo das novas coordenadas e ao `Renderizador3D` a atualização da cena.
- **Pausado:** Estado de suspensão temporal. O processamento físico e a atualização visual são interrompidos, preservando o estado atual das variáveis (posição e velocidade) para uma posterior retomada ou reinício.

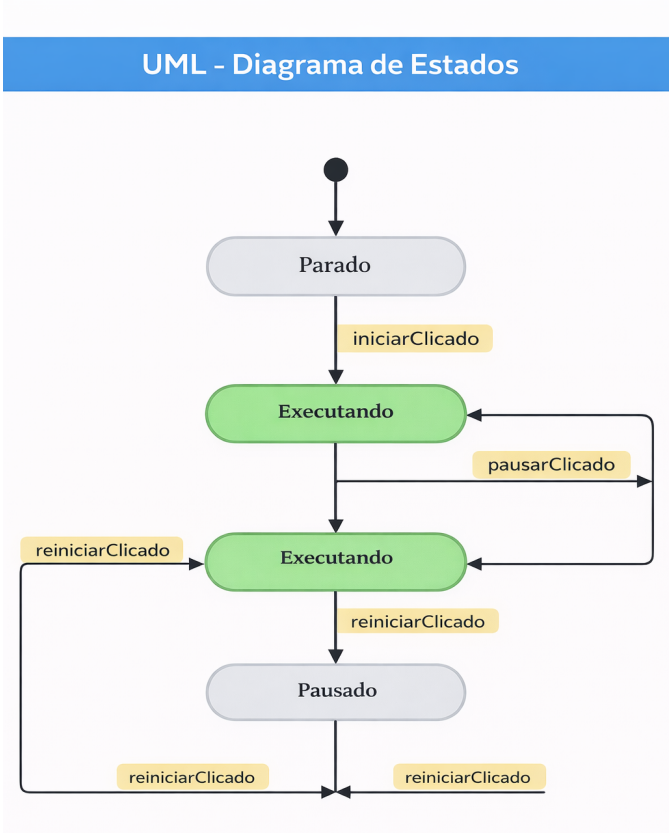


**Figura 4.6:** O Diagrama de Estados do Controlador detalha o comportamento reativo do `SimulacaoController` perante as interações do usuário.

As transições entre esses estados são mediadas por eventos de interface capturados pela `InterfaceUsuario`, tais como `iniciarClicado`, `pausarClicado` e `reiniciarClicado`. A formalização desta máquina de estados assegura que a lógica de controle seja imune a comportamentos inesperados, garantindo a integridade e a previsibilidade da simulação física.

## 4.3 FERRAMENTAS DE DESENVOLVIMENTO

### 4.3.1 *Three.js*: a biblioteca para gráficos 3D na WEB

O desenvolvimento de uma simulação interativa, tridimensional e acessível, como a proposta neste trabalho, exige uma tecnologia robusta que seja capaz de renderizar gráficos diretamente em um navegador web, garantindo acessibilidade e dispensando a instalação de softwares adicionais pelo usuário final. A biblioteca escolhida para esta tarefa foi a *Three.js*, um projeto de código aberto que se tornou um padrão de mercado para a criação de experiências 3D na web [119].

*Three.js* é uma biblioteca *JavaScript*[8] de alto nível que simplifica o processo de criação de gráficos 3D. Ela funciona como uma camada de abstração sobre a *WebGL* (*Web Graphics*

[8] *JavaScript* (JS) é uma linguagem de programação de alto nível, interpretada e com tipagem dinâmica, padroni-

*Library*), uma API de baixo nível que permite a renderização de gráficos 2D e 3D acelerada por hardware (GPU) dentro de um elemento `<canvas>` do HTML5. Enquanto o WebGL oferece grande poder, sua programação é complexa e verbosa. O Three.js encapsula essa complexidade, oferecendo uma API intuitiva e estruturada para a manipulação de cenas tridimensionais [120].

**a) Arquitetura e componentes fundamentais**

A criação de uma aplicação com Three.js baseia-se na combinação de alguns componentes essenciais que, juntos, formam a cena a ser renderizada. A documentação oficial [120] descreve a a estrutura fundamental apresentada na Tab. 4.2.

**b) Aplicação no desenvolvimento da simulação Gravitação 3D**

No contexto deste TCC, a biblioteca *Three.js* é a espinha dorsal da interface visual da simulação de gravitação. A sua arquitetura foi mapeada diretamente para os elementos do sistema a ser simulado:

- Os **corpos celestes** (estrelas, planetas) são representados por objetos do tipo `Mesh`. A geometria utilizada é a `SphereGeometry`, e seus materiais são do tipo `Mesh Standard Material`, permitindo que cada corpo tenha uma cor distinta e reaja à iluminação da cena, conforme mostrado na Fig. 4.7.
- O **espaço sideral** é a própria `Scene` (Apêndice A), que contém os *meshes* dos corpos celestes e as fontes de luz, conforme mostrado na Fig. 4.8.
- A **interação do usuário** é facilitada por componentes adicionais da biblioteca, como o `OrbitControls` (Apêndice B), que permite ao observador girar, aproximar (*zoom*) e mover (*pan*) a `PerspectiveCamera` para analisar o sistema gravitacional de qualquer ângulo, conforme mostrado na Fig. 5.1.

O aspecto mais crucial da aplicação do *Three.js* é o *loop* de animação (*animation loop*), implementado com a função `requestAnimationFrame`[9]. A cada quadro (tipicamente 60 vezes por segundo), o motor de física do simulador (responsável por resolver o sistema de EDO's baseado na 2ª Lei de Newton) atualiza as posições de cada corpo. Em seguida, essas novas coordenadas $(x, y, z)$ são atribuídas à propriedade `.position` de cada objeto `Mesh` correspondente na cena. Finalmente, o `WebGLRenderer` é invocado para renderizar a cena com os corpos em suas novas posições. Esse ciclo contínuo de **cálculo físico** $\rightarrow$ **atualização da cena** $\rightarrow$ **renderização** é o que cria a ilusão de movimento fluido e realista.

---

zada pela especificação *ECMAScript*. Embora seja mais conhecida por ser a linguagem de *script* para páginas web, a sua utilização expandiu-se para o lado do servidor (com *Node.js*), aplicações móveis e outras áreas, tornando-se uma das tecnologias mais versáteis e amplamente utilizadas no desenvolvimento de software.

[9] `requestAnimationFrame` é um método padrão em *JavaScript* para a criação de animações em navegadores. Ele informa ao navegador que você deseja realizar uma animação e solicita que ele agende uma nova pintura da janela antes do próximo ciclo de renderização, otimizando o desempenho.

**Tabela 4.2:** Principais Componentes da Biblioteca Three.js e suas Funções.

| **Componente** | **Classe (`Three.js`)** | **Descrição e Aplicação no Projeto** |
|---|---|---|
| Cena | `Scene` (Apêndice A) | É o contêiner principal onde todos os outros elementos são posicionados. A cena funciona como um universo ou um palco que abriga os objetos, as luzes e as câmeras. |
| Câmera | `PerspectiveCamera` (Apêndice B) | Define o ponto de vista a partir do qual a cena é observada. Existem diferentes tipos de câmeras, sendo a `PerspectiveCamera` (Câmera de Perspectiva) a mais comum para simulações 3D, pois imita a forma como o olho humano percebe a profundidade. |
| Renderizador | `WebGLRenderer` | É o motor responsável por processar as informações da cena e da câmera e desenhar o resultado em um elemento `<canvas>` na página web. Ele "tira a foto" da cena a partir da perspectiva da câmera a cada quadro. |
| Geometria | `SphereGeometry` | Define a forma do objeto, ou seja, o conjunto de vértices que compõem sua estrutura. O *Three.js* oferece geometrias primitivas como esferas (`SphereGeometry`), cubos (`BoxGeometry`) e planos (`PlaneGeometry`). Na simulação, é utilizada para dar forma esférica aos astros. |
| Material | `MeshStandardMaterial` | Define a aparência da superfície da geometria, como cor, textura, opacidade e como ela reage à luz (por exemplo, `Material` para cores chapadas ou `MeshStandardMaterial` para superfícies fotorrealistas). |
| Malha | `Mesh` | A união da Geometria com o Material, formando o objeto visível (ex: o planeta renderizado). |
| Luzes | `Light` | Para que materiais realistas sejam visíveis, a cena precisa ser iluminada. O *Three.js* fornece vários tipos de luz, como `AmbientLight` (luz ambiente) e `PointLight` (luz pontual, como uma lâmpada). A `PointLight` é usada para simular a emissão de luz do Sol. |

A escolha do *Three.js* se justifica, portanto, não apenas por sua capacidade de renderização, mas também por sua flexibilidade em se integrar com a lógica de simulação, permitindo que os resultados de um modelo matemático sejam traduzidos em uma experiência visual intuitiva e pedagogicamente rica. O uso desta biblioteca já foi validado em outros trabalhos acadêmicos voltados para a educação, como no desenvolvimento de laboratórios virtuais para o estudo de fenômenos ópticos [121], na visualização de estruturas moleculares [122] e o Motor elétrico – *SimuFísica*®: um aplicativo para o ensino de eletromagnetismo [113].

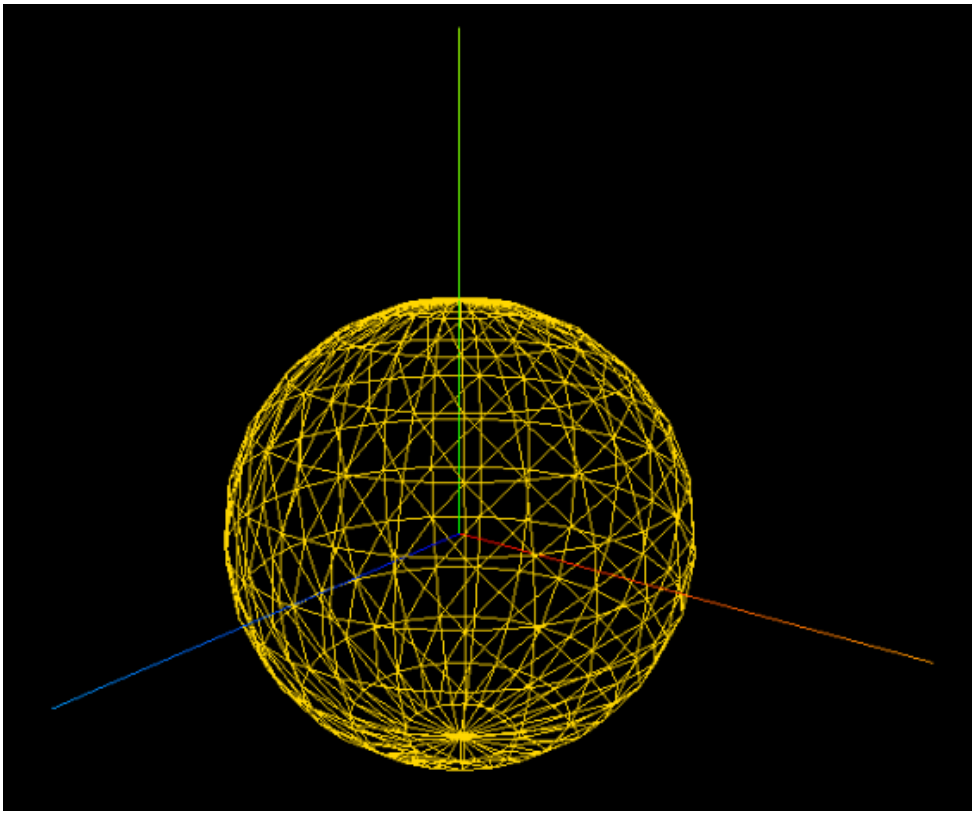

**Figura 4.7:** Representação de um objeto `Mesh` utilizando geometria esférica (`SphereGeometry`) e material padrão (`MeshStandardMaterial`). O material permite cores distintas e reação à iluminação da cena. As linhas indicam o sistema de coordenadas $(x, y, z)$.

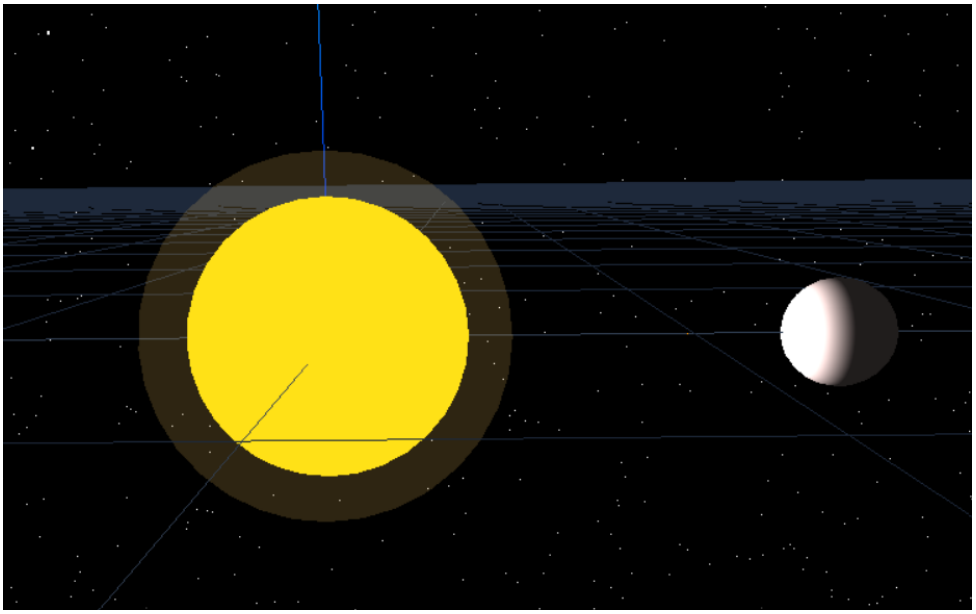

**Figura 4.8:** Representação da iluminação em um sistema tridimensional utilizando a biblioteca `Three.js`. A estrela central é modelada como uma esfera e recebe um efeito de brilho (*glow*) por meio de um objeto `Mesh` com material básico transparente (`MeshBasicMaterial`) e baixa opacidade, simulando a difusão luminosa ao redor do corpo celeste. Adicionalmente, uma fonte de luz pontual (`PointLight`) é posicionada na origem do sistema de coordenadas $(0, 0, 0)$, emitindo luz em todas as direções e iluminando os demais objetos da cena, como o planeta visível à direita. As linhas no plano representam a grade do espaço tridimensional e auxiliam na percepção da profundidade e orientação espacial.

### 4.3.2 Implementação no contexto da simulação

Na prática, o algoritmo RK4 foi encapsulado dentro do "motor de física" do software (Apêndice C). A cada quadro da animação, regido pelo `requestAnimationFrame` do navegador, esta rotina numérica é executada. Ela recebe como entrada o estado atual de todos os corpos (suas posições $\vec{r}_n$ e velocidades $\vec{v}_n$) e o passo de tempo $h$. Após executar os cálculos descritos, ela retorna o novo estado ($\vec{r}_{n+1}$, $\vec{v}_{n+1}$), que é então utilizado para atualizar as propriedades dos objetos `Mesh` na cena do *Three.js*, criando a animação do movimento orbital. A escolha de um $h$ pequeno é crucial para a precisão e estabilidade da simulação.

### 4.3.3 Repositório e acesso ao simulador

O projeto do simulador Gravitação 3D está disponível publicamente em repositório online, contendo o código-fonte completo, bem como os arquivos necessários para sua execução e manutenção. O acesso ao repositório pode ser realizado por meio do seguinte endereço: `https://github.com/jucianemaia/simu3d`.

Adicionalmente, uma versão funcional em estágio alfa pode ser acessada diretamente em `https://simu3d.com.br/`, enquanto demonstrações e exemplos das simulações estão publicados no canal `https://www.youtube.com/@simu3d-g`.

# 5 O SIMULADOR GRAVITAÇÃO 3D: APLICAÇÃO E ANÁLISE DE RESULTADOS

O principal resultado prático deste trabalho é um simulador interativo voltado ao estudo do problema de três corpos, desenvolvido em *JavaScript* com a biblioteca *Three.js*. O sistema foi projetado para integração futura ao ecossistema da plataforma *SimuFísica*®, embora o estágio atual demande ajustes de compatibilização com o *layout* e a identidade visual da plataforma, previstos para etapas subsequentes sob supervisão do professor orientador.

Atualmente, o simulador encontra-se na fase *alfa* de seu ciclo de vida, caracterizada como a fase inicial de testes internos de *software*. Nesta etapa, o desenvolvimento focou em técnicas de teste de caixa branca[1], fundamentais para garantir que a lógica do motor de física e a integração numérica pelo método de Runge-Kutta (RK4) operem conforme as especificações teóricas. Essa transição para testes que avaliam a perspectiva interna do sistema é o que define o lançamento *alfa*[2].

Além do refinamento estético, o cronograma de desenvolvimento contempla a implementação de recursos aprimorados de interatividade, seguindo o padrão de interface do usuário e telemetria da plataforma *SimuFísica*®, como a geração dinâmica de gráficos e o suporte para *download* de dados em formatos compatíveis com ferramentas de análise externa. Essas melhorias visam alinhar o protótipo atual aos padrões de excelência técnica e usabilidade já consolidados nas simulações em uso na plataforma. A ferramenta materializa os objetivos delineados no planejamento do projeto (Apêndice D) e oferece um ambiente imersivo para a exploração da dinâmica gravitacional tridimensional.

## 5.1 ARQUITETURA E INTERFACE DO USUÁRIO (UI/UX)

A arquitetura do simulador prioriza os conceitos de *User Interface* (UI)[3] e *User Experience* (UX)[4]. Enquanto a UI foca na clareza visual dos elementos interativos — como menus de configuração e botões de controle —, a UX abrange a eficiência e a satisfação do usuário durante a exploração pedagógica. O projeto busca otimizar ambos os aspectos, garantindo que a complexidade matemática do problema de três corpos seja traduzida em uma experiência de aprendizado intuitiva e fluida.

As funcionalidades centrais implementadas estão descritas nas subseções abaixo.

[1] O teste de caixa branca (também conhecido como teste estrutural ou de caixa transparente) é uma técnica que avalia a estrutura interna, o fluxo de dados e a lógica do código-fonte da aplicação. Diferente do teste de caixa preta, o testador utiliza o conhecimento da arquitetura do sistema para criar entradas que percorram caminhos específicos no código, sendo análogo aos testes de nós em circuitos integrados (ICT).

[2] O lançamento alfa (ou versão alfa) corresponde ao estágio inicial de desenvolvimento de um *software*, em que as funcionalidades principais e a lógica do motor de física (neste caso, as integrações numéricas via RK4) já estão implementadas. Esta fase é voltada para testes internos de depuração (*debugging*) e validação técnica da interface, precedendo a versão beta, que é destinada a testes com o público-alvo.

[3] UI (User Interface) refere-se à parte visual e interativa de um software com a qual o usuário interage (botões, menus, layout).

[4] UX (User Experience) é um conceito mais amplo que abrange todos os aspectos da interação do usuário final com a aplicação, incluindo a facilidade de uso, a eficiência e a satisfação geral. Um bom projeto de software busca otimizar ambos.

### 5.1.1 Motor de física

O núcleo da simulação utiliza o método de RK4, conforme detalhado na seção 4.1, para garantir a precisão e estabilidade numérica na integração das equações de movimento. Este é o ponto essencial deste trabalho, sem o qual não poderiam ser obtidos os resultados que serão discutidos na Sec. 5.2.

### 5.1.2 Visualização tridimensional interativa

A renderização em 3D proporciona uma percepção espacial precisa das órbitas, permitindo que o usuário manipule a cena livremente por meio de comandos de rotação, translação (pan) e zoom. Essa versatilidade de perspectiva é fundamental para a compreensão da geometria do sistema sob diferentes ângulos. Complementarmente, implementou-se a funcionalidade de rastro de trajetória, um recurso didático que facilita a análise histórica do movimento e a visualização das leis orbitais, conforme ilustrado na Fig. 5.1.

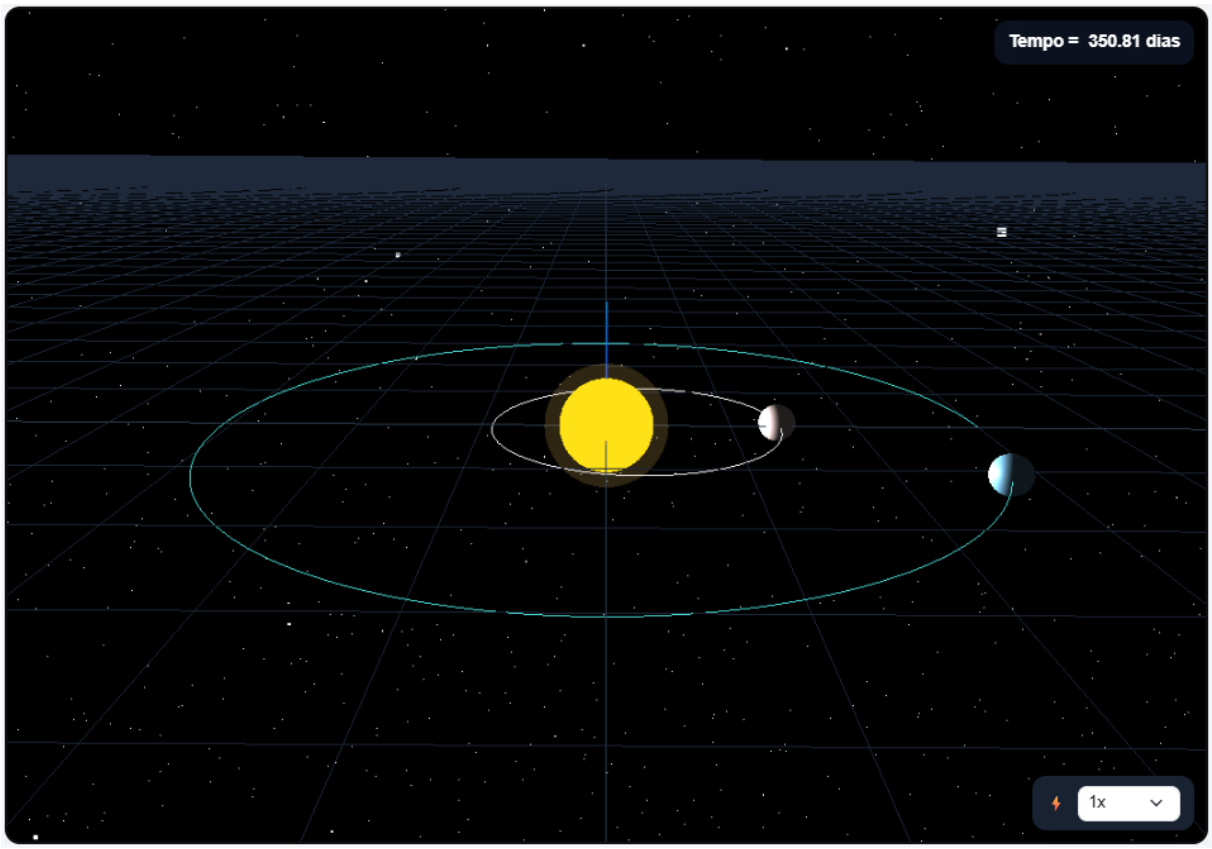


**Figura 5.1:** Renderização dinâmica da cena evidenciando a atualização da posição dos objetos em tempo real. A interface permite interatividade, possibilitando ao usuário rotacionar, aproximar (*zoom*) e transladar a perspectiva da câmera para analisar a dinâmica gravitacional sob diversos ângulos.

### 5.1.3 Interface de usuário intuitiva

A ferramenta dispõe de uma interface gráfica (GUI), como mostrado na Fig. 5.2, projetada para permitir o controle dos parâmetros e das condições iniciais do sistema. Por meio de painéis interativos, o usuário pode configurar a massa, o vetor posição tridimensional $(x, y, z)$ e os componentes do vetor velocidade inicial $(v_x, v_y, v_z)$ de cada um dos três corpos. Essa flexibilidade é essencial para a exploração de diferentes cenários dinâmicos, desde órbitas estáveis até sistemas caóticos.

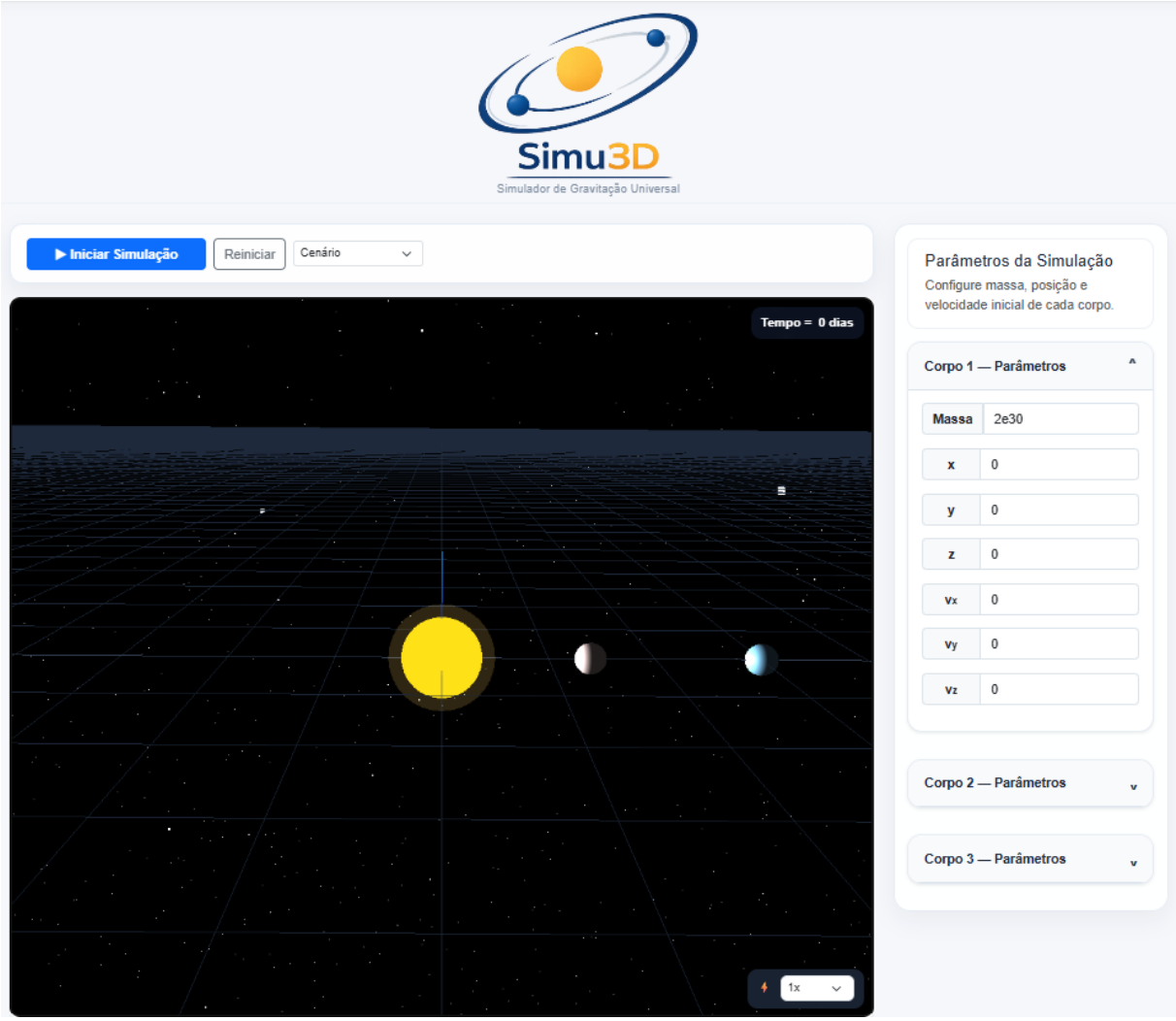


**Figura 5.2:** Interface do simulador de gravitação 3D (Simu3D), evidenciando a integração entre visualização computacional e controle interativo dos parâmetros físicos. À esquerda, observa-se o ambiente de simulação tridimensional, contendo uma estrela central e corpos orbitais representados em um espaço com grade cartesiana, permitindo percepção de profundidade e orientação espacial. Na parte superior, destacam-se os botões de controle da simulação, como iniciar e reiniciar, além da seleção de cenários predefinidos (*presets*). À direita, encontra-se o painel de configuração dos corpos, no qual o usuário pode definir massa, posição e componentes de velocidade nos eixos $x$, $y$ e $z$, possibilitando a manipulação direta das condições iniciais do sistema. Essa organização favorece a experimentação e a análise de fenômenos gravitacionais em tempo real, aproximando o usuário de um ambiente de laboratório virtual.

## 5.2 VALIDAÇÃO E ANÁLISE DOS RESULTADOS

Esta seção apresenta a análise crítica do simulador desenvolvido, com ênfase na avaliação de sua precisão física e de sua capacidade de representar fenômenos gravitacionais de forma consistente. Para isso, foram realizados testes de validação utilizando parâmetros astronômicos reais do Sistema Solar, permitindo comparar os resultados obtidos pelo simulador com dados de referência amplamente aceitos pela comunidade científica.

Para a configuração das condições iniciais de cada corpo celeste no ambiente virtual, foram utilizados como parâmetros de entrada a massa (kg), a distância ao Sol no periélio (m) e a velocidade orbital no periélio (m/s). Esses valores foram selecionados por representarem condições reais de movimento orbital e por possibilitarem a reprodução de trajetórias próximas às observadas no Sistema Solar.

A partir dessas configurações, foram realizadas medições experimentais dos períodos orbitais simulados, permitindo a comparação entre os resultados produzidos pelo modelo computacional e os valores observados na realidade. As Figuras 5.3, 5.4 e 5.5 apresentam exemplos das coletas realizadas para Mercúrio, Vênus e Terra, utilizando as condições iniciais descritas na Tabela 5.1.

A Tabela 5.1 reúne os principais parâmetros físicos empregados na simulação, incluindo massa, distâncias orbital média, periélica e afélica, além das velocidades orbitais média, no periélio e no afélio. A utilização desses dados permitiu reproduzir, de maneira aproximada, as características dinâmicas dos corpos celestes analisados.

**Tabela 5.1:** Parâmetros físicos detalhados do Sistema Solar.

| **Astro** | **Massa (kg)** | **Dist. Média (m)** | **Dist. Periélio (m)** | **Dist. Afélio (m)** | $V_{\text{média}}$ **(m/s)** | $V_{\text{periélio}}$ **(m/s)** | $V_{\text{afélio}}$ **(m/s)** |
|---|---|---|---|---|---|---|---|
| Sol | $1,989 \times 10^{30}$ | - | - | - | - | - | - |
| Mercúrio | $3,301 \times 10^{23}$ | $5,791 \times 10^{10}$ | $4,600 \times 10^{10}$ | $6,982 \times 10^{10}$ | 47.360 | 58.980 | 38.860 |
| Vênus | $4,867 \times 10^{24}$ | $1,082 \times 10^{11}$ | $1,075 \times 10^{11}$ | $1,089 \times 10^{11}$ | 35.020 | 35.260 | 34.780 |
| Terra | $5,972 \times 10^{24}$ | $1,496 \times 10^{11}$ | $1,471 \times 10^{11}$ | $1,521 \times 10^{11}$ | 29.780 | 30.290 | 29.290 |
| Marte | $6,417 \times 10^{23}$ | $2,279 \times 10^{11}$ | $2,066 \times 10^{11}$ | $2,492 \times 10^{11}$ | 24.130 | 26.500 | 21.970 |
| Júpiter | $1,898 \times 10^{27}$ | $7,785 \times 10^{11}$ | $7,405 \times 10^{11}$ | $8,166 \times 10^{11}$ | 13.060 | 13.720 | 12.440 |
| Saturno | $5,683 \times 10^{26}$ | $1,433 \times 10^{12}$ | $1,353 \times 10^{12}$ | $1,514 \times 10^{12}$ | 9.640 | 10.180 | 9.090 |
| Urano | $8,681 \times 10^{25}$ | $2,871 \times 10^{12}$ | $2,741 \times 10^{12}$ | $3,004 \times 10^{12}$ | 6.790 | 7.110 | 6.490 |
| Netuno | $1,024 \times 10^{26}$ | $4,495 \times 10^{12}$ | $4,445 \times 10^{12}$ | $4,546 \times 10^{12}$ | 5.430 | 5.480 | 5.370 |

Baseado em NASA Planetary Fact Sheets (2026) [123].

Os registros visuais apresentados a seguir correspondem aos testes de validação temporal realizados com os planetas internos do Sistema Solar. Em cada experimento, o cronômetro interno do simulador foi iniciado no instante da passagem do planeta pelo periélio e interrompido após a conclusão de uma órbita completa ($360°$) em torno do Sol. Os períodos orbitais obtidos foram então comparados com os valores de referência, possibilitando avaliar a precisão do modelo gravitacional implementado e a sua adequação para fins educacionais.

Para a simulação de Mercúrio, as condições iniciais foram configuradas com a massa de $3,301 \times 10^{23}$ kg, posicionando o astro a uma distância de periélio de $4,600 \times 10^{10}$ m e aplicando uma velocidade tangencial de $58.980$ m/s. O resultado obtido foi um período orbital de 87,38 dias, apresentando uma convergência com os dados reais e um erro relativo de apenas 0,67%.

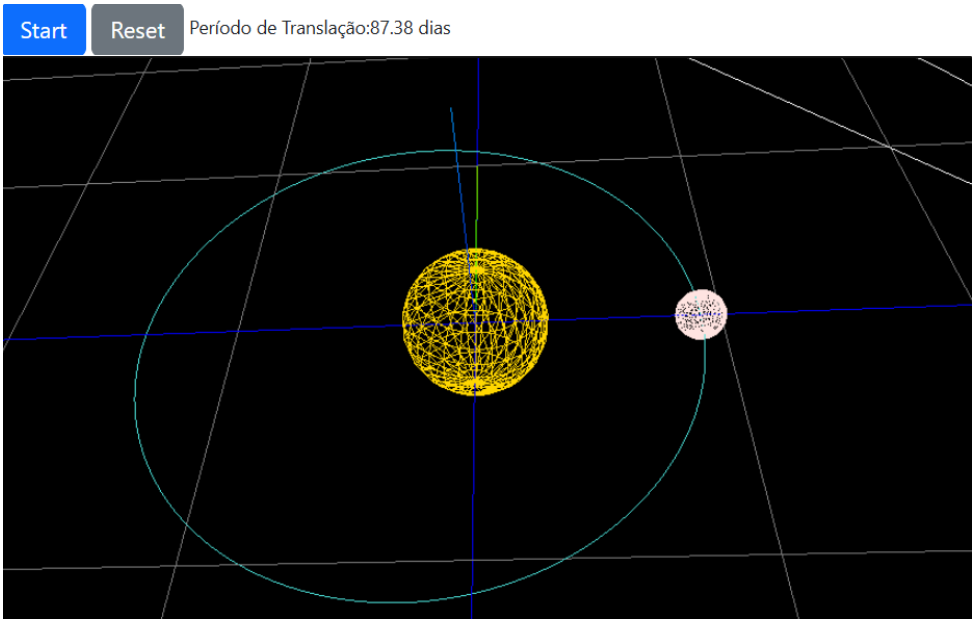


**Figura 5.3:** Registro cronométrico da revolução de Mercúrio, validando a precisão do algoritmo RK4 em altas velocidades.

No teste correspondente a Vênus, utilizou-se a massa de $4,867 \times 10^{24}$ kg, com raio de periélio de $1,075 \times 10^{11}$ m e velocidade inicial de $35.260$ m/s. A simulação registrou um período de 224,88 dias, o que representa a maior precisão alcançada entre os planetas internos, com um erro relativo de apenas 0,08%.

Finalmente, para a validação do sistema Terra-Sol, foram inseridos os parâmetros de $5,972 \times 10^{24}$ kg para a massa, $1,471 \times 10^{11}$ m para a distância inicial e $30.290$ m/s para a velocidade de periélio. O motor de física cronometrou um período de 365,63 dias, mantendo a consistência do sistema com um erro relativo de 0,10%.

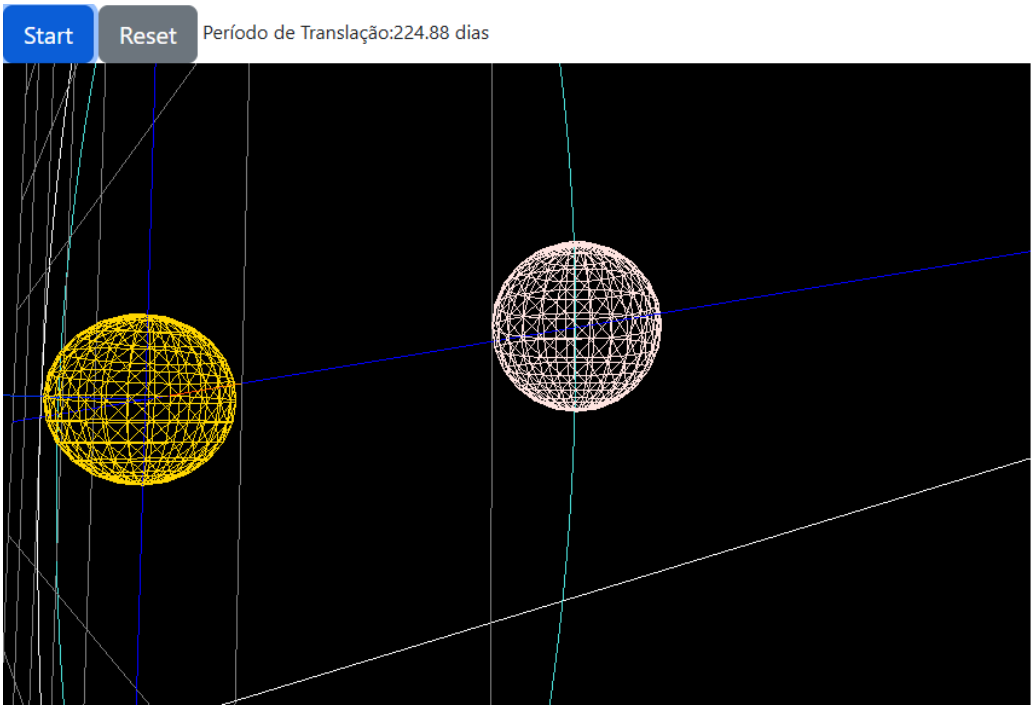


**Figura 5.4:** Validação da periodicidade orbital de Vênus, evidenciando a estabilidade da trajetória circularizada.

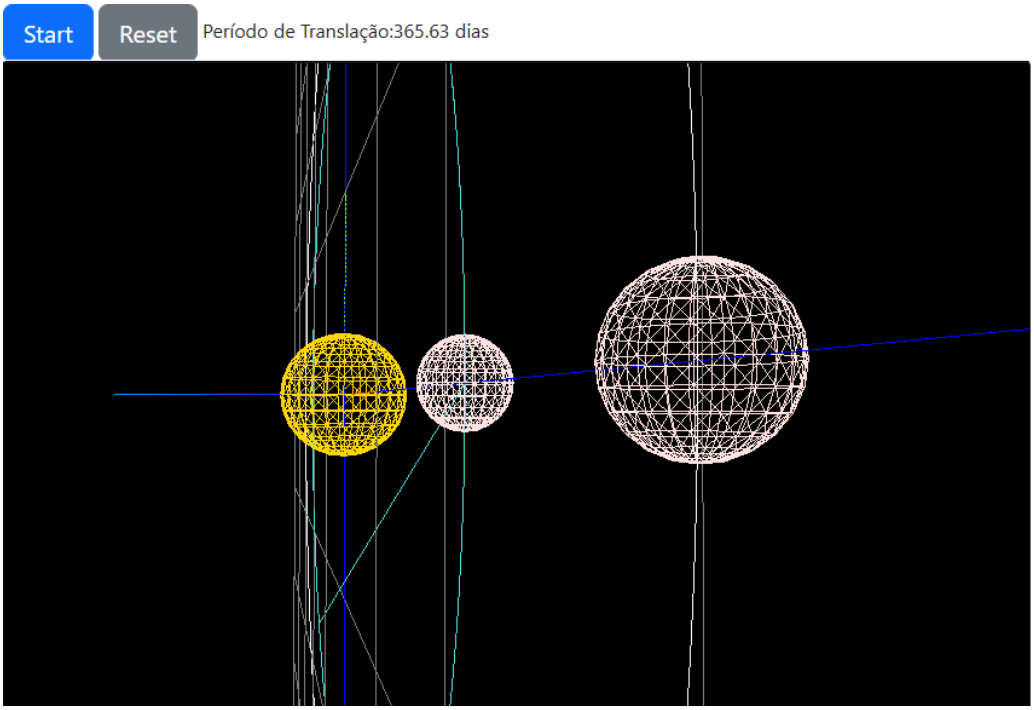


**Figura 5.5:** Interface do simulador apresentando a medição do período orbital terrestre após um ciclo completo.

A eficácia do motor de física foi quantificada por meio do cálculo do erro relativo para o período orbital ($T$), comparando os resultados simulados ($T_{\text{sim}}$) com os valores reais de referência ($T_{\text{real}}$):

$$E_{\text{rel}}(\%) = \left| \frac{T_{\text{sim}} - T_{\text{real}}}{T_{\text{real}}} \right| \times 100\% \tag{5.1}$$

A Tabela 5.2 resume essa análise. Nota-se que o sistema apresenta uma boa precisão para todos os corpos celestes, com erros relativos iguais a 0,12% ou menos. O desempenho em Mercúrio (0,67%) é particularmente notável, demonstrando que a calibração dos parâmetros iniciais no periélio permitiu ao algoritmo RK4 lidar com a excentricidade orbital de forma robusta e fidedigna.

**Tabela 5.2:** Comparação entre períodos orbitais simulados e reais.

| **Planeta** | **Período Simulado (dias)** | **Período Real (dias)** | **Erro Relativo (%)** |
|---|---|---|---|
| Mercúrio | 87,38 | 87,97 | 0,67% |
| Vênus | 224,88 | 224,70 | 0,08% |
| Terra | 365,63 | 365,25 | 0,10% |
| Marte | 687,80 | 686,98 | 0,12% |
| Júpiter | 4.334,75 | 4.332,59 | 0,05% |
| Saturno | 10.767,82 | 10.759,22 | 0,08% |
| Urano | 30.703,89 | 30.685,48 | 0,06% |
| Netuno | 60.231,37 | 60.189,24 | 0,07% |

Valores reais baseados na Ref. [123].

### 5.2.1 Mais exemplos

#### Estrelas binárias

Trata-se de um sistema composto por duas estrelas que orbitam o centro de massa comum. Na Fig. 5.6(a), foram adotados os seguintes parâmetros e condições iniciais: $m_2 = m_3 = 2 \times 10^{30}$kg (equivalente à massa do Sol), $x_2(0) = -50 \times 10^9$ m, $x_3(0) = 50 \times 10^9$ m, $v_2^y(0) = 20$ km/s e $v_3^y(0) = -20$ km/s, sendo as demais condições iniciais consideradas nulas. Nessas condições, as estrelas descrevem órbitas em torno do centro de massa do sistema, que permanece em repouso.

Na Fig. 5.6(b), foram mantidos os mesmos parâmetros, alterando-se apenas a velocidade inicial da estrela 2 para $v_2^y(0) = 25$ km/s. Como essa modificação produz um momento linear total diferente de zero e não atuam forças externas sobre o sistema, o centro de massa passa a se deslocar com velocidade constante na direção do eixo $y$, conforme previsto pela conservação do momento linear.

#### Dinâmica caótica

Neste exemplo, foi considerado um sistema composto por três estrelas, no qual as massas satisfazem a relação $m_1 = 2m_2 = 2m_3$, sendo $m_1 = 4 \times 10^{30}$ kg. As condições iniciais adotadas foram: $x_1(0) = -100 \times 10^9$ m, $x_2(0) = 0$, $x_3(0) = 100 \times 10^9$ m, $v_2^y(0) = 38,9$ km/s e $v_3^y(0) = -30$ km/s, enquanto as demais condições iniciais foram consideradas nulas.

A evolução temporal do sistema, apresentada na Fig. 5.7, evidencia a complexidade das interações gravitacionais entre os três corpos. Observa-se que as trajetórias não seguem padrões orbitais simples ou periódicos, apresentando comportamento característico de sistemas caóticos, nos quais pequenas variações nas condições iniciais podem resultar em evoluções significativamente distintas ao longo do tempo.

#### Órbitas elípticas

A condição para que uma órbita de um planeta (massa $m_2$) ao redor de uma estrela (massa $m_1 \gg m_2$) seja circular é que o módulo da força centrípeta $F_c$ agindo no planeta seja constante durante todo o tempo. Então, igualando $F_c$ à força gravitacional $F_g$, pode-se escrever

$$\frac{Gm_1m_2}{d^2} = \frac{m_2v^2}{d}, \tag{5.2}$$

onde $v$ é o módulo da velocidade do planeta (consideramos a estrela com velocidade zero) e $d$ é a distância do planeta à estrela. A solução da equação 5.2 para $v$ é

$$v = \sqrt{\frac{Gm_1}{d}}. \tag{5.3}$$

Usando $m_1 = 2 \times 10^{30}$ kg e $d = 150 \times 10^9$ m, chegamos em $v = 29,8$ km/s [Fig. 5.8(a)]. Uma órbita elíptica não circular é alcançada com qualquer outra velocidade não nula menor do que a velocidade necessária para uma órbita circular [Fig. 5.8(b), onde $v = 14,9$ km/s]. Nesse último caso, pode-se observar a estrela ocupando um dos focos da elipse, de acordo com a primeira lei de Kepler (Sec. 2.1.1).

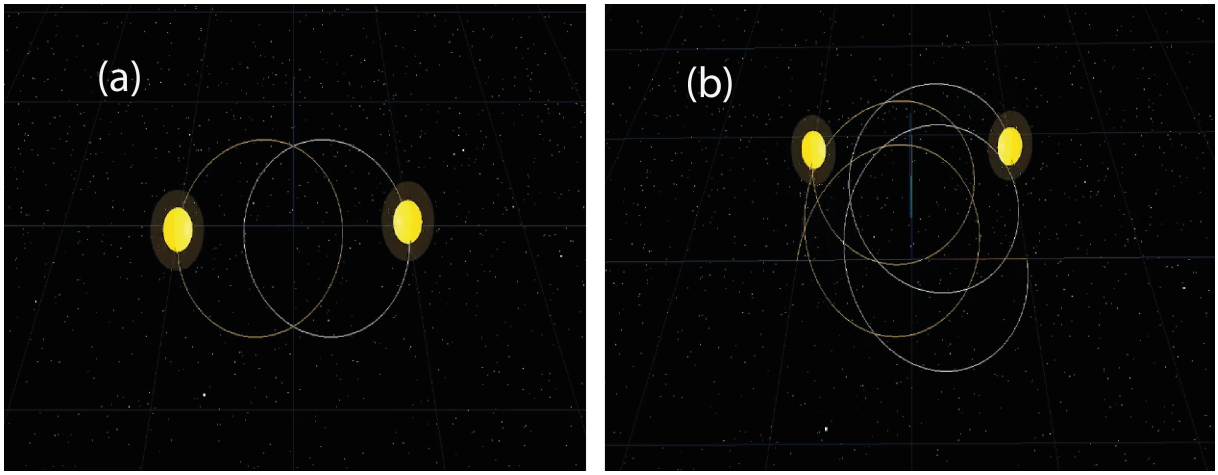


**Figura 5.6:** Simulador configurado com dois corpos com massas iguais a do Sol. Em (a), o momento linear inicial do sistema é zero. Em (b), o momento linear inicial do sistema aponta na direção $y$.

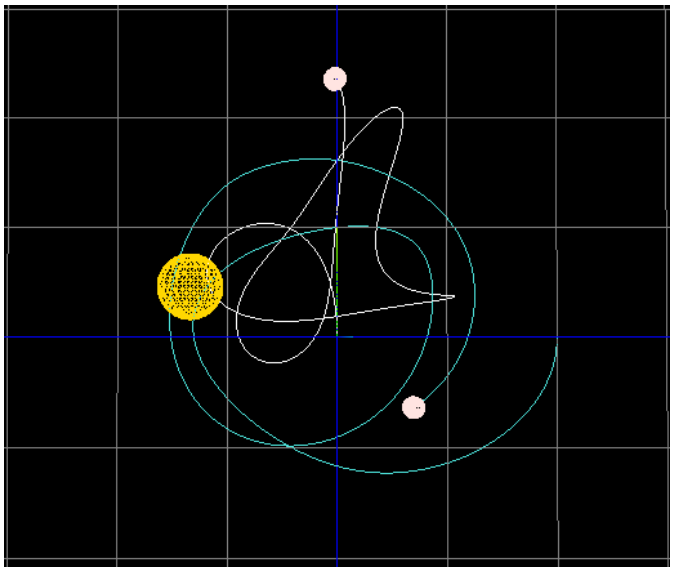

**Figura 5.7:** Simulador representando a dinâmica do problema dos três corpos. As massas são $m_1 = 2m_2 = 2m_3$, com $m_1 = 4 \times 10^{30}$ kg.

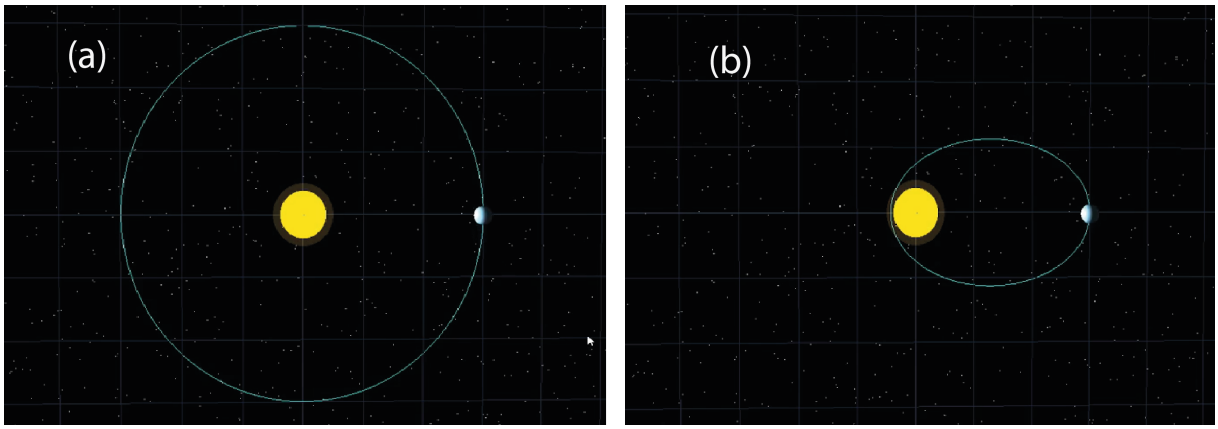


**Figura 5.8:** Simulador ilustrando uma órbita circular (a) e uma não circular (b).

### Órbitas não elípticas

Sabe-se que as órbitas elípticas não constituem as únicas soluções possíveis para a dinâmica gravitacional de um corpo de massa reduzida sob a influência de um corpo muito mais massivo [7]. Dependendo das condições iniciais, também podem ocorrer órbitas parabólicas e hiperbólicas. Estas últimas surgem quando a energia cinética do corpo é superior ao módulo de sua energia potencial gravitacional. Dessa forma, a condição para a ocorrência de uma órbita hiperbólica pode ser expressa por

$$\frac{m_2 v^2}{2} > \frac{G m_1 m_2}{d}, \tag{5.4}$$

em que $m_1$ representa a massa do corpo central, $m_2$ a massa do objeto em movimento e $d$ a distância entre eles. Resolvendo a inequação (5.4) em relação à velocidade, obtém-se

$$v > \sqrt{\frac{2Gm_1}{d}}. \tag{5.5}$$

Essa expressão corresponde à velocidade de escape do sistema gravitacional. Assim, um objeto que passe pelo Sol a uma distância $d$ com velocidade superior a esse valor não permanecerá gravitacionalmente ligado à estrela, afastando-se indefinidamente sem descrever uma órbita fechada.

Considerando novamente $m_1 = 2 \times 10^{30}, \text{kg}$ e $d = 150 \times 10^{9}, \text{m}$, obtém-se $v \approx 42{,}2, \text{km/s}$ como valor limiar entre órbitas elípticas e hiperbólicas. Quando a velocidade é exatamente igual a esse valor, a trajetória resultante é parabólica.

Para verificar esse comportamento, foram realizadas simulações com velocidades iniciais próximas ao valor limite. A Fig. 5.9(a) apresenta a trajetória obtida para $v = 40$ km/s, caracterizada por uma órbita fechada. Já a Fig. 5.9(b) mostra o resultado para $v = 43$ km/s, no qual o corpo adquire velocidade suficiente para escapar da atração gravitacional do Sol, seguindo uma trajetória hiperbólica.

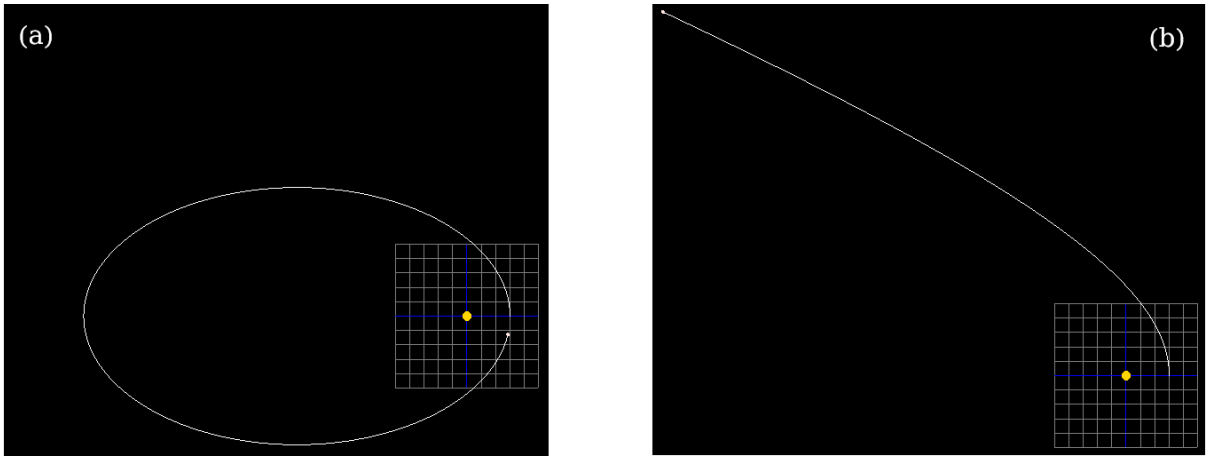


**Figura 5.9:** Simulador ilustrando o limiar entre uma órbita elíptica (a) e uma hiperbólica (b).

### 5.2.2 Justificativa para refinamentos e melhorias

Apesar do cumprimento satisfatório dos itens fundamentais do motor de física e da interface de visualização, a análise do *status* final do *backlog* (Tabela D.4) revela lacunas críticas que impedem a integração imediata do simulador à plataforma *SimuFísica*®. A transição de um protótipo funcional para uma ferramenta de ampla escala exige o atendimento de critérios de robustez e suporte pedagógico que ainda não foram plenamente atingidos.

As justificativas para a necessidade de melhorias substanciais estão fundamentadas nos três pilares descritos na Tabela 5.3:

**Tabela 5.3:** Desafios de Interoperabilidade e Impactos Esperados

| **Pilar Crítico** | **Descrição da Pendência** | **Impacto na Plataforma** |
|---|---|---|
| **Interatividade Científica** | Incompletude da Telemetria (PBI-09). | Impede que o usuário realize investigações quantitativas e coleta de dados orbitais. |
| **Interoperabilidade** | Pendência na Integração de Sistemas (PBI-12). | Risco de instabilidade em ambiente de produção e falhas de renderização em diferentes navegadores. |
| **Mediação Didática** | Ausência do Guia Pedagógico (PBI-13). | Sem o roteiro, a simulação reduz-se a uma animação ilustrativa, perdendo sua função como instrumento de ensino. |

Dessa forma, a homologação final na plataforma *SimuFísica*® fica condicionada a um ciclo de refinamento técnico focado em:

1. **Estabilidade e responsividade:** Ajustar a `InterfaceUsuario` para garantir a usabilidade em dispositivos móveis (essencial para o contexto escolar), bem como ao padrão dos simuladores do SimuFísica.

2. **Rigor numérico de longo prazo:** Validar a conservação de energia em simulações extensas para evitar o "efeito de deriva" nas trajetórias calculadas pelo RK4.

3. **Padronização de saída:** Implementar o componente de telemetria para exportação de dados em formatos compatíveis com ferramentas de análise gráfica.

Em síntese, o projeto atingiu o estado de "Prova de Conceito de Alta Fidelidade"[5], restando o refinamento dos débitos identificados para alcançar o padrão de excelência exigido para o ecossistema da plataforma de destino.

[5]Uma Prova de Conceito (PoC) de Alta Fidelidade é uma representação interativa e detalhada de um produto, simulando quase perfeitamente o design final, a funcionalidade e a experiência do usuário (UX).

# 6 CONCLUSÃO

## 6.1 CONSIDERAÇÕES FINAIS

Ao concluir este percurso de pesquisa e desenvolvimento, pode-se afirmar que o objetivo geral do trabalho foi plenamente alcançado. O desenvolvimento do simulador gravitacional 3D não apenas resultou em uma ferramenta funcional, mas também validou a premissa de que a convergência entre a física computacional e as metodologias ágeis de gestão é um caminho viável e eficiente para a criação de tecnologias educacionais de qualidade.

Até o presente estágio, o projeto consolidou marcos técnicos e científicos fundamentais. No âmbito da programação, a implementação do algoritmo de *Runge-Kutta* de 4ª Ordem (RK4) garantiu um motor de física estável e capaz de processar interações gravitacionais em tempo real com fluidez. A adoção da arquitetura MVC (*Model-View-Controller*) permitiu que o sistema fosse construído de forma modular, separando a lógica de cálculo da representação visual em *Three.js*, o que assegura a manutenção e a escalabilidade do *software*.

Do ponto de vista científico, a validação realizada com parâmetros astronômicos reais demonstrou a alta fidelidade do simulador. A obtenção de erros relativos extremamente reduzidos — destacando-se a Terra ($0,10\%$), Vênus ($0,08\%$) e o excelente ajuste em Mercúrio ($0,67\%$) — atesta que a ferramenta é fidedigna e plenamente apta para o uso em contextos de ensino médio e superior. Esses resultados comprovam que a calibração do motor de física a partir das condições iniciais de periélio permitiu ao algoritmo lidar com excentricidades orbitais de forma robusta, superando os desafios de convergência numérica inicialmente previstos.

Apesar do sucesso técnico do motor físico, a transição para a incorporação definitiva na plataforma *SimuFísica*® exige agora o refinamento de lacunas de maturação funcional e usabilidade. Para que a ferramenta atinja o padrão de excelência exigido para disponibilização pública, os seguintes requisitos deverão ser priorizados em ciclos futuros:

1. **Módulo de telemetria:** Conclusão do sistema de exibição e exportação de dados numéricos em tempo real, permitindo que a simulação funcione como um laboratório de coleta de dados simulados.

2. **Mediação didática:** Elaboração de roteiros pedagógicos que orientem a aplicação prática do simulador em sala de aula, transformando a ferramenta em um objeto de aprendizagem.

3. **Interoperabilidade e UX:** Finalização da API de integração com a plataforma *SimuFísica*® e otimização da interface para garantir responsividade e desempenho em dispositivos móveis.

Em síntese, apresentamos um aplicativo capaz de simular órbitas de planetas e estrelas, reproduzindo com realismo as trajetórias observadas no Sistema Solar. O simulador demonstra maturidade técnica tanto em seu motor físico quanto na componente visual, evidenciando robustez numérica, consistência conceitual e potencial didático.

As etapas futuras concentram-se no aperfeiçoamento de sua maturidade como produto educacional, ampliando processos de validação em sala de aula e consolidando-o como ferramenta transformadora no ensino de Física.

## 6.2 PROJEÇÕES FUTURAS E EXPANSÕES POTENCIAIS

O simulador Gravitação 3D serve como alicerce para futuras expansões que visam transformar a ferramenta em uma suíte completa para o ensino de física clássica e moderna.

**A) Módulo de simulação galáctica (Sgr A*)** Uma expansão de alto impacto será a criação de um cenário específico para o buraco negro supermassivo no centro da Via Láctea.

- Implementação: Fixação de um corpo central com massa equivalente a $\approx 4,3 \times 10^6$ massas solares. O usuário poderá configurar a órbita de estrelas de teste (como a estrela S2) para observar dinâmicas keplerianas em campos gravitacionais extremos.
- Recurso pedagógico: Introdução de *presets* astronômicos reais, facilitando a exploração de dados observacionais modernos.

**B) Inclusão de efeitos relativísticos** Para aumentar a precisão em regimes de campo forte, propõe-se a incorporação de correções relativísticas fundamentadas na Relatividade Geral.

- Implementação: Utilização da aproximação pós-newtoniana [124], adicionando termos de correção que dependem de potências superiores da distância ($1/r^4$). Isso permitirá simular fenômenos como a precessão do periélio de Mercúrio, corrigindo os desvios observados na mecânica puramente clássica.
- Interatividade: Inclusão de um controle de ativação (*checkbox*) para permitir ao estudante comparar, visual e numericamente, as previsões de Newton frente às de Einstein.

**C) Melhorias de engine e *gamificação***

- Detecção de colisão e fusão: Implementação de tratamento para colisões inelásticas e fusão de corpos celestes com conservação de momento linear.
- *Gamificação*: Criação de "Módulos de Desafio", onde o usuário deve realizar tarefas específicas, como estabelecer uma órbita estável em formato de "8" ou ejetar um corpo do sistema devido à interação gravitacional.
- Análise de energia: Plotagem em tempo real de gráficos de energia cinética, potencial e total, permitindo a verificação empírica da conservação de energia do sistema.

Com estas projeções, este trabalho não se encerra como um ponto final, mas sim como o início de um projeto escalável, capaz de evoluir continuamente para atender às demandas do ensino e da divulgação científica contemporânea.

# REFERÊNCIAS


[1] R. S. Westfall. *The Life of Isaac Newton*. Cambridge: Cambridge University Press, 1994.

[2] A. Einstein. *Como vejo o mundo*. Tradução de H. P. de Andrade. Rio de Janeiro: Nova Fronteira, 2017.

[3] M. P. M. de Souza, J. G. Maia e L. N. de Andrade. "Pendulum Tracker—SimuFísica®: a web-based tool for real-time measurement of oscillatory motion". Em: *Physics Education* 60.5 (2025), p. 055024. DOI: 10.1088/1361-6552/adf406.

[4] SimuFísica. *Simuladores para o ensino de física*. URL: https://simufisica.com/ (acesso em 15/03/2026).

[5] UNIR - Fundação Universidade Federal de Rondônia. *SimuFísica da UNIR alcança 10 mil usuários mensais*. Notícia institucional. 2026. URL: https://www.unir.br/cartao/exibir/245.

[6] Three.js. *Three.js – JavaScript 3D Library*. URL: https://threejs.org/ (acesso em 28/03/2025).

[7] S. T. Thornton e J. B. Marion. *Dinâmica clássica de partículas e sistemas*. 5ª ed. São Paulo: Cengage Learning, 2021.

[8] H. Poincaré. *Les méthodes nouvelles de la mécanique céleste*. Paris: Gauthier-Villars, 1892.

[9] M. Prince. "Does Active Learning Work? A Review of the Research". Em: *Journal of Engineering Education* 93.3 (2004), pp. 223–231.

[10] A. M. P. d. Carvalho. *Ensino de Ciências por Investigação: Condições para o Ensino em Sala de Aula*. São Paulo: Cengage Learning, 2013. ISBN: 9788522112333.

[11] J. Piaget. *Biologia e Conhecimento*. Petrópolis: Vozes, 1973.

[12] D. C. Aguay-Saquicaray et al. "Effectiveness of computer simulation in teaching physics in higher education: a systematic review". Em: *Revista Invecom* (2025). Efectividad de la simulación computacional en la enseñanza de la física en educación superior: una revisión sistemática. ISSN: 2739-0063.

[13] L. Bacich e J. Moran. *Metodologias ativas para uma educação inovadora: uma abordagem téorico-prática*. Recurso eletrônico (e-PUB). Porto Alegre: Penso, 2018. ISBN: 9788584291152.

[14] C. M. Oliveira. *Ensino da Temática Energia Com Três Aplicativos da Plataforma SimuFísica no Novo Ensino Médio*. Dissertação de Mestrado. Ji-Paraná, RO, Fevereiro de 2023.

[15] H. M. Nussenzveig. *Curso de Física Básica: Mecânica*. 5ª ed. Vol. 1. São Paulo: Blucher, 2013.

[16] H. Banda e J. Nzabahimana. "The Impact of Physics Education Technology (PhET) Interactive Simulation-Based Learning on Motivation and Academic Achievement Among Malawian Physics Students". Em: *Journal of Science Education and Technology* 32.1 (2023), pp. 127–141. DOI: 10.1007/s10956-022-10010-3.

[17] J. A. Valente. *Computadores e conhecimento: repensando a educação*. Organizador. Campinas, SP: UNICAMP/NIED, 1998.

[18] J. Peixoto e N. C. d. Oliveira. “Concepções de ciência, tecnologia e educação na produção acadêmica do ensino de Ciências da Natureza”. Em: *Cadernos de Pesquisa* 30 (nov. de 2023). Artigo que investiga as concepções de ciência e tecnologia no ensino de Ciências. URL: https://periodicoseletronicos.ufma.br/index.php/cadernosdepesquisa/article/view/17674.

[19] B. A. Moura, W. L. P. d. Santos e M. Pietrocola. “Alfabetização científica e tecnológica: discussões sobre a natureza da ciência e da tecnologia no ensino de física”. Em: *Revista Brasileira de Ensino de Física* 44 (2022), e20220215. DOI: 10.1590/1806-9126-RBEF-2022-0215. URL: https://doi.org/10.1590/1806-9126-RBEF-2022-0215.

[20] M. A. Moreira. “Desafios no ensino da física”. Em: *Revista Brasileira de Ensino de Física* 43.suppl 1 (2021), e20200451. DOI: 10.1590/1806-9126-RBEF-2020-0451. URL: https://doi.org/10.1590/1806-9126-RBEF-2020-0451.

[21] P. M. C. Dias, W. M. S. Santos e M. T. M. d. Souza. “A Gravitação Universal (Um texto para o Ensino Médio)”. Em: *Revista Brasileira de Ensino de Física* 26.3 (2004). Discute a utilização da História da Física e da Aprendizagem Significativa de Ausubel., pp. 257–271. URL: https://www.sbfisica.org.br/rbef/pdf/040503.pdf.

[22] J. Trindade. “Dificuldades na Aprendizagem de Física – Algumas notas”. Em: *Ciência e Tecnologia* (jan. de 1998), pp. 221–229. URL: https://www.researchgate.net/publication/234026609_Dificuldades_na_Aprendizagem_de_Fisica_-_Alguma_notas.

[23] G. L. N. Macedo e L. C. Gomes. “Análise do conceito de força gravitacional nos Principia e a sua transposição didática do saber sábio ao saber a ensinar nos livros de Ciências da Natureza do PNLD 2021”. Em: *Amazônia | Revista de Educação em Ciências e Matemática* 20.45 (2024). Discute a Transposição Didática e a História da Ciência no ensino de Gravitação., pp. 67–91. URL: https://periodicos.ufpa.br/index.php/amazonia/article/view/17498.

[24] J. F. Custódio Filho e M. P. d. Oliveira. “Princípios Físicos e a Construção de Modelos”. Em: *Atas eletrônicas do VII Encontro de Pesquisa em Ensino de Física*. Pós-Graduação em Educação-CED-UFSC e Dept de Física-CFM-UFSC. Florianópolis, SC, 2000.

[25] S. Lins. *Transferindo conhecimento tácito: uma abordagem construtivista*. Rio de Janeiro: E-papers, 2003.

[26] C. J. B. Lattari e R. H. Trevisan. “Metodologia para o ensino de astronomia: uma abordagem construtivista”. Em: *Atas do II Encontro Nacional de Pesquisa em Educação em Ciências*. Valinhos, SP: ABRAPEC, 1999.

[27] Brasil. Ministério da Educação. *Base Nacional Comum Curricular: Educação é a Base*. Brasília: MEC/CONSED/UNDIME. Brasília, DF: Ministério da Educação, 2018. URL: https://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf.

[28] D. Scalzitti et al. “Interactive 3D visualizations in the browser: Leveraging WebGL and Three.js for STEM education”. Em: *Journal of Science Education and Technology* 33 (2024), pp. 145–158. DOI: 10.1007/s10956-023-10082-x.

[29] W. Hunziker e I. M. Sigal. “The quantum N-body problem”. Em: *Journal of Mathematical Physics* 41.6 (2000), pp. 3448–3510. DOI: 10.1063/1.533319.

[30] G. I. Depetri. “Coreografias no Problema de N corpos”. Tese de dout. São Paulo: Universidade de São Paulo, 2011.

[31] J. C. L. d. Souza. *Problema de N Corpos e sua Relação com o Caos*. Trabalho de Conclusão de Curso (Graduação em Física). Sorocaba, SP, 2022. URL: https://repositorio.ufscar.br/handle/20.500.14289/16265.

[32] Universe Sandbox. *Universe Sandbox – Create & Destroy on an Astronomical Scale*. 2025. URL: ....

[33] NASA. *Eyes on the Solar System*. URL: https://eyes.nasa.gov/apps/solar-system/ (acesso em 30/11/2024).

[34] Z. E. Musielak e B. Quarles. “The three-body problem”. Em: *Reports on Progress in Physics* 77.6 (2014), p. 065901. DOI: 10.1088/0034-4885/77/6/065901.

[35] M. J. Valtonen e H. Karttunen. *The three-body problem*. Cambridge: Cambridge University Press, 2006.

[36] C. A. d. S. Batista e M. d. O. Souza. “Um Mergulho na História Conceitual da Astronomia, da Cosmologia e da Física à Luz da Solução de Problemas Laudanianos: dos babilônios à gravitação newtoniana”. Em: *Revista Brasileira de Ensino de Física* 42 (2020), e20190184. DOI: 10.1590/1806-9126-RBEF-2019-0184.

[37] S. M. Carroll. *Spacetime and geometry: an introduction to general relativity*. Cambridge: Cambridge University Press, 2019.

[38] P. Duhem. “Salvar os fenômenos: Ensaio sobre a noção de teoria física de Platão a Galileu”. Em: *Cadernos de História e Filosofia da Ciência* 3 (1984).

[39] M. A. G. Santos et al. “Sobre o Problema de Platão, o Movimento Retrógrado dos Planetas e os Sistemas de Mundo, Geocêntrico e Heliocêntrico”. Em: *Sitientibus Série Ciências Físicas* 20 (2024), pp. 1–22.

[40] A. M. Velásquez-Toribio e M. V. Oliveira. “Primeiro modelo matemático da cosmologia: as esferas concêntricas de eudoxo”. Em: *Revista Brasileira de Ensino de Física* 41.2 (2019), e20180096. DOI: 10.1590/1806-9126-RBEF-2018-0096.

[41] J. E. Steiner. “A gênese do Universo”. Em: *Estudos Avançados* 20.58 (2006), pp. 7–21. DOI: 10.1590/S0103-40142006000300002. URL: https://doi.org/10.1590/S0103-40142006000300002.

[42] A. M. Velásquez-Toribio e M. V. Oliveira. “Primeiro modelo matemático da cosmologia: as esferas concêntricas de Eudoxo”. Em: *Revista Brasileira de Ensino de Física* 41.2 (2019), e20180096. DOI: 10.1590/1806-9126-RBEF-2018-0096.

[43] N. R. Hanson. “Aristotle”. Em: *Constellations and Conjectures*. Ed. por W. C. Humphreys. Vol. 48. Synthese Library. Dordrecht: Springer, 1973, ??–?? DOI: 10.1007/978-94-010-2498-3_7.

[44] C. Ptolemaeus. *Almagestu[m] Cl. Ptolemei Pheludiensis Alexandrini Astronomo[rum] principis*. Tradução de Gerardo de Cremona. Disponível digitalmente pelo Deutsches Museum, München. Venice: Petrus Liechtenstein, 1515. URL: http://dx.doi.org/10.5079/dmm-25.

[45] O. Neugebauer. *A History of Ancient Mathematical Astronomy*. Berlin: Springer-Verlag, 1975.

[46] J. Evans. *The History and Practice of Ancient Astronomy*. New York: Oxford University Press, 1998.

[47] K. d. S. Oliveira Filho e M. d. F. O. Saraiva. *Astronomia e Astrofísica*. 4ª ed. São Paulo: Livraria da Física, 2017.

[48] M. A. G. Santos. "De Platão à Kepler: um recorte sobre o movimento dos planetas no Sistema Solar". Dissertação de Mestrado Profissional. Feira de Santana: Universidade Estadual de Feira de Santana, 2023.

[49] N. Copernicus. *De revolutionibus orbium coelestium*. Trata-se da edição original (editio princeps) de 1543, impressa em Nuremberga. Norimbergae: Iohannes Petreius, 1543.

[50] C. M. Porto. "A Revolução Copernicana: aspectos históricos e epistemológicos". Em: *Revista Brasileira de Ensino de Física* 42 (2020), e20190190. DOI: 10.1590/1806-9126-RBEF-2019-0190. URL: https://doi.org/10.1590/1806-9126-RBEF-2019-0190.

[51] F. Damasio. "Início da revolução científica: questões acerca de Copérnico e os epiciclos, Kepler e as órbitas elípticas". Em: *Revista Brasileira de Ensino de Física* 33.3 (2011), p. 3602. DOI: 10.1590/S1806-11172011000300012. URL: https://www.scielo.br/j/rbef/a/vS8XyX9yX9yX9yX9yX9yX9y/.

[52] T. S. Kuhn. *A Revolução Copernicana: a astronomia planetária no desenvolvimento do pensamento ocidental*. Lisboa: Edições 70, 1990.

[53] V. E. Thoren. *The Lord of Uraniborg: A Biography of Tycho Brahe*. Cambridge: Cambridge University Press, 2007.

[54] C. R. Tossato. "Tycho Brahe e a precisão das observações astronômicas". Em: *Intelligere* 13 (2022), pp. 92–112. URL: http://revistas.usp.br/revistaintelligere.

[55] L. Motz e J. H. Weaver. *The Story of Astronomy*. Cambridge, Massachusetts: Perseus Publishing, 1995.

[56] S. M. Cohen, P. Curd e C. D. C. Reeve. *Readings in Ancient Greek Philosophy: From Thales to Aristotle*. Indianapolis/Cambridge: Hackett Publishing Company, Inc., 2011.

[57] V. E. Thoren. "The comet of 1577 and Tycho Brahe's system of the world". Em: *Archives Internationales d'Histoire des Sciences* 29 (1979), pp. 53–67.

[58] N. d. Fer. *L'ordre des sphères célestes selon l'opinion de Tycho Brahé*. Parte da obra L'Atlas Curieux. Disponível em: Gallica/Bibliothèque nationale de France. Paris, 1705. URL: https://gallica.bnf.fr/.

[59] N. R. Hanson. *Constelaciones y conjeturas*. 2ª ed. Madrid: Alianza Universidad, 2025.

[60] J. Kepler. *Prodromus dissertationum cosmographicarum, continens Mysterium Cosmographicum*. Tübingen: Georgius Gruppenbachius, 1597.

[61] C. R. Tossato. "Mysterium Cosmographicum: Os Antecedentes das Duas Primeiras Leis Keplerianas dos Movimentos Planetários". Em: *Cadernos Espinosanos* 5 (1999), pp. 35–63.

[62] C. R. Tossato e P. R. Mariconda. "O método da astronomia segundo Kepler". Em: *Scientiæ Studia* 8.3 (2010), pp. 339–366. DOI: 10.1590/S1678-31662010000300003.

[63] J. Kepler. *Astronomia Nova Aitiologetos, seu physica coelestis, tradita commentariis de motibus stellae Martis*. Heidelberg/Prague: Vögelin, 1609.

[64] J. Kepler. *Harmonices Mundi Libri V*. Lincii Austriæ: Apud Ioannem Plancum, sumptibus Godofredi Tampachii, 1619.

[65] C. C. Pereira e A. M. Velásquez-Toribio. "A Terceira Lei de Kepler em diferente escalas: de luas até galáxias". Em: *Revista Brasileira de Ensino de Física* 46 (2024), e20230329. DOI: 10.1590/1806-9126-RBEF-2023-0329. URL: https://doi.org/10.1590/1806-9126-RBEF-2023-0329.

[66] M. A. Garms e I. L. Caldas. "Síntese das Leis de Kepler". Em: *Revista Brasileira de Ensino de Física* 40.2 (2018), e2316. DOI: 10.1590/1806-9126-RBEF-2017-0253. URL: https://doi.org/10.1590/1806-9126-RBEF-2017-0253.

[67] O. Gingerich. *The Eye of Heaven: Ptolemy, Copernicus, Kepler*. New York: American Institute of Physics, 1993.

[68] I. Newton. *Philosophiæ Naturalis Principia Mathematica*. Londini: Jussu Societatis Regiæ ac Typis Josephi Streater, 1687.

[69] R. d. A. Martins. "A maçã de Newton: história, lendas e tolices". Em: *Estudos de História e Filosofia das Ciências*. Ed. por C. C. Silva. Capítulo de livro. São Paulo: Livraria da Física, 2006, pp. 167–189. URL: https://www.researchgate.net/publication/275833050.

[70] R. Descartes. *Principia Philosophiae*. Obra onde Descartes formula as leis do movimento e o conceito de inércia linear. Amsterdam: Ludovicum Elzevirium, 1644.

[71] E. S. Teixeira, L. O. d. Q. Peduzzi e O. Freire Junior. "Os caminhos de Newton para a gravitação universal: uma revisão do debate historiográfico entre Cohen e Westfall". Em: *Caderno Brasileiro de Ensino de Física* 27.2 (2010), pp. 215–254. DOI: 10.5007/2175-7941.2010v27n2p215. URL: https://periodicos.ufsc.br/index.php/fisica/article/view/2175-7941.2010v27n2p215.

[72] I. B. Cohen e G. E. Smith, ed. *The Cambridge Companion to Newton*. Cambridge, UK: Cambridge University Press, 2002. DOI: 10.1017/CCOL0521651778.

[73] P. M. C. Dias e T. J. Stuchi. "Isaac Newton, Robert Hooke and the mystery of the orbit". Em: *Brazilian Studies in Philosophy and History of Science: An account of recent works*. Ed. por D. Krause e A. A. P. Videira. Dordrecht: Springer Netherlands, 2011, pp. 77–94. DOI: 10.1007/978-90-481-9422-3_6. URL: https://doi.org/10.1007/978-90-481-9422-3_6.

[74] G. Wanner. "Kepler, Newton e análise numérica". Em: *Acta Numerica* 19 (2010), pp. 561–598. DOI: 10.1017/S0962492910000073. URL: https://doi.org/10.1017/S0962492910000073.

[75] E. Tiesinga et al. "CODATA recommended values of the fundamental physical constants: 2018". Em: *Reviews of Modern Physics* 93 (2 2021), p. 025010. DOI: 10.1103/RevModPhys.93.025010.

[76] M. Jammer. *Concepts of Force: A Study in the Foundations of Dynamics*. New York: Dover Publications, 1999.

[77] I. B. Cohen. *The Birth of a New Physics*. Revised and updated. New York: W. W. Norton & Company, 1985.

[78] M. Grosser. *The Discovery of Neptune*. Cambridge: Harvard University Press, 1962.

[79] B. Peirce. “The Orbit of Neptune, Computed by Sears C. Walker, and Formulae in the Theory of Neptune”. Em: *Proceedings of the American Academy of Arts and Sciences* 1 (1847), pp. 65–66.

[80] H. D. Curtis. *Orbital Mechanics for Engineering Students*. 3ª ed. Oxford: Butterworth-Heinemann, 2014.

[81] N. T. Roseveare. *Mercury's Perihelion from Le Verrier to Einstein*. Oxford: Oxford University Press, 1982.

[82] A. Einstein. “Zur Elektrodynamik bewegter Körper”. Em: *Annalen der Physik* 322.10 (1905), pp. 891–921. DOI: 10.1002/andp.19053221004.

[83] J. Renn. “A física clássica de cabeça para baixo: como Einstein descobriu a teoria da relatividade especial”. Em: *Revista Brasileira de Ensino de Física* 27.1 (2005), pp. 27–36. DOI: 10.1590/S1806-11172005000100004. URL: https://doi.org/10.1590/S1806-11172005000100004.

[84] A. Einstein. “Die Grundlage der allgemeinen Relativitätstheorie”. Em: *Annalen der Physik* 49 (1916), pp. 769–822.

[85] J. A. Wheeler. *A Journey into Gravity and Spacetimes*. New York: Scientific American Library, 1990.

[86] F. W. Dyson, A. S. Eddington e C. R. Davidson. “A determination of the deflection of light by the Sun's gravitational field, from observations made at the total eclipse of May 29, 1919”. Em: *Philosophical Transactions of the Royal Society of London. Series A* 220 (1920), pp. 291–333. DOI: 10.1098/rsta.1920.0009.

[87] L. C. B. Crispino. “Expeditions for the observation in Sobral, Brazil, of the May 29, 1919 total solar eclipse”. Em: *International Journal of Modern Physics D* 27.11 (2018), p. 1843004. DOI: 10.1142/S021827181843004X. URL: https://doi.org/10.1142/S021827181843004X.

[88] K. S. Thorne. *Black Holes & Time Warps: Einstein's Outrageous Legacy*. Prefácio de Stephen Hawking. New York: W. W. Norton & Company, 2014.

[89] N. Ashby. “Relativity in the Global Positioning System”. Em: *Living Reviews in Relativity* 6.1 (2003), p. 1. DOI: 10.12942/lrr-2003-1.

[90] B. P. Abbott, others (LIGO Scientific Collaboration e V. Collaboration). “Observation of Gravitational Waves from a Binary Black Hole Merger”. Em: *Physical Review Letters* 116.6 (2016), p. 061102. DOI: 10.1103/PhysRevLett.116.061102.

[91] Event Horizon Telescope Collaboration. “First M87 Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole”. Em: *The Astrophysical Journal Letters* 875.L1 (2019), 17pp. DOI: 10.3847/2041-8213/ab0ec7. URL: https://doi.org/10.3847/2041-8213/ab0ec7.

[92] C. Rovelli. *Quantum Gravity*. Cambridge: Cambridge University Press, 2004.

[93] H. M. Nussenzveig. *Curso de Física Básica: Volume 2*. São Paulo: Editora Edgard Blucher, 2015.

[94] H. Goldstein, C. Poole e J. Safko. *Classical Mechanics*. 3ª ed. San Francisco: Addison Wesley, 2002.

[95] J. Barrow-Green. *Poincaré and the Three-Body Problem*. Vol. 11. History of Mathematics. Providence, RI: American Mathematical Society, 1997. ISBN: 0-8218-0367-0.

[96] S. H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. 2ª ed. Boca Raton: CRC Press, 2018.

[97] S. J. Aarseth. *Gravitational N-Body Simulations: Tools and Algorithms*. Cambridge: Cambridge University Press, 2003.

[98] S. Freeman et al. "Active learning increases student performance in science, engineering, and mathematics". Em: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8410–8415. DOI: 10.1073/pnas.1319030111.

[99] C. E. Wieman et al. "Teaching Physics Using PhET Interactive Simulations". Em: *The Physics Teacher* 48.4 (2010), pp. 225–227. DOI: 10.1119/1.3361987.

[100] J. Piaget. *To Understand Is to Invent: The Future of Education*. New York: Grossman Publishers, 1973.

[101] K. Perkins et al. "PhET: Interactive simulations for teaching and learning". Em: *The Physics Teacher* 44.1 (2006), pp. 18–23. DOI: 10.1119/1.2150754.

[102] N. D. Finkelstein et al. "When learning about the real world is better done with a virtual circuit". Em: *Physical Review Special Topics-Physics Education Research* 1.1 (2005), p. 010103. DOI: 10.1103/PhysRevSTPER.1.010103.

[103] T. De Jong e W. R. Van Joolingen. "Scientific discovery learning with computer simulations of conceptual domains". Em: *Review of Educational Research* 68.2 (1998), pp. 179–201. DOI: 10.3102/00346543068002179.

[104] PhET Interactive Simulations. *Simulações interativas para o ensino de ciências e matemática*. URL: https://phet.colorado.edu/ (acesso em 30/11/2024).

[105] Geogebra. *Interactive Mathematics and Science Simulations*. URL: https://www.geogebra.org/ (acesso em 30/11/2024).

[106] Algodoo. *Physics for curious minds*. URL: https://www.algodoo.com/ (acesso em 30/11/2024).

[107] The Physics Classroom. *The Physics Classroom*. 2024. URL: https://www.physicsclassroo
com (acesso em 17/08/2025).

[108] Open Source Physics (OSP). *Physics Simulations and Tools*. URL: https://www.compadre.org/osp/ (acesso em 30/11/2024).

[109] Laboratório Virtual de Física – UFC. *Simulações interativas de Física*. URL: https://www.laboratoriovirtual.fisica.ufc.br/simulacoes (acesso em 30/11/2024).

[110] M. F. L. Alves. *Ensino de Física utilizando Tecnologias Digitais: Recursos Multimídia aplicados ao ensino de Ondas Gravitacionais no Ensino Médio*. Monografia (Especialização em Tecnologias Digitais Aplicadas à Educação). Orientador: Prof. Dr. Júlio César Mota Silva. Petrolina - PE: Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano, Campus Petrolina, 2024, p. 48.

[111] H.-C. Wang e C.-Y. Chang. "The comparative efficacy of 2D- versus 3D-based media design for influencing spatial visualization skills". Em: *Computers in Human Behavior* 23.4 (2007), pp. 1943–1957. DOI: 10.1016/j.chb.2006.02.004.

[112] National Aeronautics and Space Administration. *Planet Compare*. NASA. 2026. URL: https://solarsystem.nasa.gov/planet-compare/ (acesso em 08/04/2026).

[113] M. P. M. de Souza, S. P. O. Oliveira e V. L. Luiz. “Motor elétrico – SimuFísica®: um aplicativo para o ensino de eletromagnetismo”. Em: *Revista Brasileira de Ensino de Física* 46 (2024), e20230219. ISSN: 1806-9126. DOI: 10.1590/1806-9126-RBEF-2023-0219.

[114] M. P. M. de Souza, C. M. de Oliveira e R. P. P. Araújo. “O simulador conservação de energia mecânica – SimuFísica”. Em: *A Física na Escola* 22 (2024), p. 240173. URL: https://doi.org/10.59727/fne.v22i1.173.

[115] M. P. M. de Souza, C. M. de Oliveira e R. P. P. Araújo. “The Conservation of Mechanical Energy simulator – SimuFísica”. Em: *arXiv:2410.17951 [physics.ed-ph]* (2024). URL: https://doi.org/10.48550/arXiv.2410.17951.

[116] M. P. M. de Souza e A. B. Siqueira. “Explorando trocas de calor com o simulador Calorimetria – SimuFísica”. Em: *Revista de Enseñanza de la Física* 37 (2025), pp. 153–161. URL: https://doi.org/10.55767/2451.6007.v37.n2.51181.

[117] R. L. Burden e J. D. Faires. *Numerical Analysis*. 9ª ed. Boston, MA: Brooks/Cole, 2010. ISBN: 978-0538733519.

[118] Object Management Group. *Unified Modeling Language (UML) Specification*. Standard formal/2017-12-05. Versão 2.5.1. Needham, MA: Object Management Group (OMG), 2017. URL: https://www.omg.org/spec/UML/2.5.1/.

[119] Mr.doob. *Three.js – JavaScript 3D Library*. Originalmente lançado em 2010. 2024. URL: https://threejs.org/ (acesso em 17/08/2025).

[120] Three.js Authors. *Three.js Documentation*. 2024. URL: https://threejs.org/docs/ (acesso em 17/08/2025).

[121] P. I. Chopyk, V. P. Oleksiuk e O. P. Chukhrai. “Using the Three.js library to develop remote physical laboratory to investigate diffraction”. Em: *Proceedings of the 6th Workshop for Young Scientists in Computer Science & Software Engineering (CS&SE@SW 2023)*. Vol. 3662. Kryvyi Rih, Ukraine: CEUR-WS.org, 2024, pp. 246–259. URL: https://ceur-ws.org/Vol-3662/paper23.pdf.

[122] N. Rego e D. Koes. “3Dmol.js: molecular visualization with WebGL”. Em: *Bioinformatics* 31.8 (2015), pp. 1322–1324. DOI: 10.1093/bioinformatics/btu829.

[123] NASA - National Aeronautics and Space Administration. *Planetary Fact Sheets*. Dados consolidados para parâmetros físicos e orbitais. 2026. URL: https://nssdc.gsfc.nasa.gov/planetary/factsheet/ (acesso em 10/01/2026).

[124] J. B. Hartle. *Gravity: An Introduction to Einstein’s General Relativity*. San Francisco, CA: Addison Wesley, 2003. ISBN: 978-0805386622.

[125] K. Schwaber e J. Sutherland. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. 2020. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf (acesso em 17/08/2025).

[126] D. T. d. Oliveira e P. N. Silva. “Framework Scrum na pesquisa científica: um diálogo possível na relação orientador-orientando”. Em: *REBECIN - Revista Brasileira de Educação em Ciência da Informação* 12.1 (2025), pp. 1–25. DOI: 10.24208/rebecin.v12.379. URL: https://portal.abecin.org.br/rebecin/article/view/379.

[127] S. Guardia, M. Guardia e L. M. Filho. “Proposta de framework para desenvolvimento de trabalhos científicos baseada na metodologia Scrum”. Em: *Revista Metodologias e Aprendizado* 5 (2022), pp. 15–32. DOI: 10.21166/metapre.v5i.2368. URL: https://metodologiaseaprendizado.com.br/revista/index.php/metapre/article/view/2368.

## A CÓDIGO: CENA

```
const gridHelper = new THREE.GridHelper(10, 10, 0x0000ff, 0x808080);
gridHelper.rotation.x = Math.PI / 2;
scene.add(gridHelper);

const geometryBody1 = new THREE.SphereGeometry(0.3, 25, 12); //Sol
const materialBody1 = new THREE.MeshBasicMaterial({
    color: 0xffd700,
    wireframe: true,
});

const geometryBody2 = new THREE.SphereGeometry(0.1, 25, 12);
const materialBody2 = new THREE.MeshBasicMaterial({
    color: 0xFFE4E1,
    wireframe: true,
});

const geometryBody3 = new THREE.SphereGeometry(0.1, 25, 12);
const materialBody3 = new THREE.MeshBasicMaterial({
    color: 0xFFE4E1,
    wireframe: true,
});

const body1 = new THREE.Mesh(geometryBody1, materialBody1);
scene.add(body1);
body1.position.x = 0;
body1.position.y = 0;
body1.position.z = 0;

const body2 = new THREE.Mesh(geometryBody2, materialBody2);
scene.add(body2);
body2.position.x = 1.496;  //cada px equivale a 1x10^11
body2.position.y = 1.496;  //cada px equivale a 1x10^11
body2.position.z = 0;

const body3 = new THREE.Mesh(geometryBody2, materialBody2);
scene.add(body3);
body3.position.x = -1.496;  //cada px equivale a 1x10^11
body3.position.y = -1.496;  //cada px equivale a 1x10^11
body3.position.z = 0;


let pointsTrajetorias1 = [];
let pointsTrajetorias2 = [];


const fator = 50e9;


```

```
49 let pausou = true;
50 function pausar(){
51     pausou = !pausou;
52 }
53
54 renderer.setAnimationLoop(() => {
55     console: 'loop';
56
57     //if(pausou)
58         //parametros()
59
60     if (!pausou)
61         gravitacaoUniversal()
62
63     body1.position.x = x1/fator;
64     body1.position.y = y1/fator;
65     body1.position.z = z1/fator;
66     body2.position.x = x2/fator;
67     body2.position.y = y2/fator;
68     body2.position.z = z2/fator;
69     body3.position.x = x3/fator;
70     body3.position.y = y3/fator;
71     body3.position.z = z3/fator;
72
73     // ----------- trajetoria body1 ---------------
74     pointsTrajetorias1.push( new THREE.Vector3( x2/fator, y2/fator,
          z2/fator ) );
75     const geometryLinha1 = new THREE.BufferGeometry().setFromPoints(
          pointsTrajetorias1);
76     const materialLinha1 = new THREE.LineBasicMaterial({
77         color: 0xffffff
78     });
79     const line1 = new THREE.Line( geometryLinha1, materialLinha1 );
80     scene.add( line1 )
81     // ---------------------------------------------
82
83     // ----------- trajetoria body2 ---------------
84     pointsTrajetorias2.push( new THREE.Vector3( x3/fator, y3/fator,
          z3/fator ) );
85     const geometryLinha2 = new THREE.BufferGeometry().setFromPoints(
          pointsTrajetorias2 );
86     const materialLinha2 = new THREE.LineBasicMaterial({
87         color: 0x4de3db
88     });
89     const line2 = new THREE.Line( geometryLinha2, materialLinha2 );
90     scene.add( line2 )
91     // ---------------------------------------------
92
93     renderer.render(scene, camera);
94     controls.update()
95 });
```

# B CÓDIGO: CONSTRUTOR DA CENA

```
const options = {
    targetSelector: '#scene',
    width: 850, height: 550,
    backgroundColor: 0x4169E1,
}

const renderer = new THREE.WebGL1Renderer();

renderer.setSize(
    options.width, options.height
);

document.querySelector(
    options.targetSelector
).appendChild(renderer.domElement);

/*-------- Cena Principal ---------*/
const scene = new THREE.Scene();
scene.backgroundColor = new THREE.Color(
    options.backgroundColor,
);

const loaderSolTerra = new THREE.TextureLoader();
loaderSolTerra.load("./images/universe-8k.jpg", function (texture){
    scene.background = texture;
});

/*-------- Camera Principal ---------*/
const camera = new THREE.PerspectiveCamera(
    50, options.width / options.height,
);
camera.position.z = 7;

/*-------- Coordenadas ---------*/
scene.add( new THREE.AxesHelper() );

/*------ Controls -------*/
const controls = new THREE.OrbitControls(camera, renderer.domElement)
controls.enableDamping = true
controls.dampingFactor = 0.1


/*-------- Resize ---------*/
function resize()
{
    camera.aspect = window.innerWidth / window.innerHeight
    camera.updateProjectionMatrix()
    renderer.setSize( window.innerWidth, window.innerHeight )
```

```
49 }
50
51 window.addEventListener('resize', resize)
```

## C CÓDIGO: MOTOR FÍSICA

```
1
2  G = 6.6743e-11;
3  m1// = 1.98892e30;  //massa do Sol no SI
4  m2// = 5.9742e24;   //massa da Terra no SI
5  m3// = 4.86900e24;  //massa de Venus no SI
6  t = 0;
7  h = 10000;
8  //h = 3600; //Deve funcionar
9  //h = 60;  //Testar
10 //h = 10000; //SimuFisica
11 let v_x1 = 0, v_y1 = 0, v_z1 = 0, v_x2 = 0, v_y2 = 30e3, v_z2 = 0,
       v_x3 = 0, v_y3 = 38.9e3, v_z3 = 0;
12 let x1 = 0, y1 = 0, z1 = 0, x2 = 150e9, y2 = 0, z2 = 0, x3 = 69.8e9,
       y3 = 0, z3 = 0;
13 let k1_x1 = 0, k1_y1 = 0, k1_z1 = 0, k1_x2 = 0, k1_y2 = 0, k1_z2 = 0,
       k1_x3 = 0, k1_y3 = 0, k1_z3 = 0;
14
15
16 function reset() {
17     t = 0;
18     pausou = true;
19     parametros();
20
21     document.getElementById("tempo").innerHTML = " 0 dias";
22
23     // limpa arrays
24     try {
25         pointsTrajetorias1.length = 0;
26         pointsTrajetorias2.length = 0;
27     } catch (e) {
28         // ainda nao foram inicializados -- ignora
29     }
30
31     // REMOVE TODAS AS LINHAS DA CENA
32     scene.children = scene.children.filter(obj => {
33         if (obj.type === "Line") {
34             obj.geometry.dispose();
35             obj.material.dispose();
36             return false;
37         }
38         return true;
39     });
40 }
41
42 reset()
43
44
45 function parametros() {
```

```
46
47      m1 = Number(document.getElementById('m1').value);
48      x1 = Number(document.getElementById('x1').value);
49      y1 = Number(document.getElementById('y1').value);
50      z1 = Number(document.getElementById('z1').value);
51      v_x1 = Number(document.getElementById('vx1').value);
52      v_y1 = Number(document.getElementById('vy1').value);
53      v_z1 = Number(document.getElementById('vz1').value);
54
55      m2 = Number(document.getElementById('m2').value);
56      x2 = Number(document.getElementById('x2').value);
57      y2 = Number(document.getElementById('y2').value);
58      z2 = Number(document.getElementById('z2').value);
59      v_x2 = Number(document.getElementById('vx2').value);
60      v_y2 = Number(document.getElementById('vy2').value);
61      v_z2 = Number(document.getElementById('vz2').value);
62
63      m3 = Number(document.getElementById('m3').value);
64      x3 = Number(document.getElementById('x3').value);
65      y3 = Number(document.getElementById('y3').value);
66      z3 = Number(document.getElementById('z3').value);
67      v_x3 = Number(document.getElementById('vx3').value);
68      v_y3 = Number(document.getElementById('vy3').value);
69      v_z3 = Number(document.getElementById('vz3').value);
70  }
71
72  const containerSol = document.getElementById('controls_variables_sol'
        );
73  containerSol.addEventListener('input', function (e) {
74    if (e.target.matches('input')) {
75      reset();
76    }
77  });
78
79  const containerPlaneta1 = document.getElementById('
        controls_variables_plan1');
80  containerPlaneta1.addEventListener('input', function (e) {
81    if (e.target.matches('input')) {
82      reset();
83    }
84  });
85
86  const containerPlaneta2 = document.getElementById('
        controls_variables_plan2');
87  containerPlaneta2.addEventListener('input', function (e) {
88      if (e.target.matches('input')) {
89          reset();
90      }
91  });
92
93
```

```
94  function gravitacaoUniversal() {
95
96      //h = Number(document.getElementById("acelerar").value);
97      document.getElementById("tempo").innerHTML = " " + (t
            /(3600*24)).toFixed(2) + " dias";
98
99      //k1
100     k1_x1 = v_x1;
101     k1_y1 = v_y1;
102     k1_z1 = v_z1;
103     k1_x2 = v_x2;
104     k1_y2 = v_y2;
105     k1_z2 = v_z2;
106     k1_x3 = v_x3;
107     k1_y3 = v_y3;
108     k1_z3 = v_z3;
109
110
111     k1_vx1 = - (G * m2 * (x1 - x2)) / (Math.pow(Math.pow(x2 - x1, 2)
            + Math.pow(y2 - y1, 2) + Math.pow(z2 - z1, 2), 1.5))
112                 - (G * m3 * (x1 - x3)) / (Math.pow(Math.pow(x3 - x1, 2)
                        + Math.pow(y3 - y1, 2) + Math.pow(z3 - z1, 2), 1.5));
113     k1_vy1 = - (G * m2 * (y1 - y2)) / (Math.pow(Math.pow(x2 - x1, 2)
            + Math.pow(y2 - y1, 2) + Math.pow(z2 - z1, 2), 1.5))
114                 - (G * m3 * (y1 - y3)) / (Math.pow(Math.pow(x3 - x1, 2)
                        + Math.pow(y3 - y1, 2) + Math.pow(z3 - z1, 2), 1.5));
115     k1_vz1 = - (G * m2 * (z1 - z2)) / (Math.pow(Math.pow(x2 - x1, 2)
            + Math.pow(y2 - y1, 2) + Math.pow(z2 - z1, 2), 1.5))
116                 - (G * m3 * (z1 - z3)) / (Math.pow(Math.pow(x3 - x1, 2)
                        + Math.pow(y3 - y1, 2) + Math.pow(z3 - z1, 2), 1.5));
117     k1_vx2 = - (G * m1 * (x2 - x1)) / (Math.pow(Math.pow(x2 - x1, 2)
            + Math.pow(y2 - y1, 2) + Math.pow(z2 - z1, 2), 1.5))
118                 - (G * m3 * (x2 - x3)) / (Math.pow(Math.pow(x3 - x2, 2)
                        + Math.pow(y3 - y2, 2) + Math.pow(z3 - z2, 2), 1.5));
119     k1_vy2 = - (G * m1 * (y2 - y1)) / (Math.pow(Math.pow(x2 - x1, 2)
            + Math.pow(y2 - y1, 2) + Math.pow(z2 - z1, 2), 1.5))
120                 - (G * m3 * (y2 - y3)) / (Math.pow(Math.pow(x3 - x2, 2)
                        + Math.pow(y3 - y2, 2) + Math.pow(z3 - z2, 2), 1.5));
121     k1_vz2 = - (G * m1 * (z2 - z1)) / (Math.pow(Math.pow(x2 - x1, 2)
            + Math.pow(y2 - y1, 2) + Math.pow(z2 - z1, 2), 1.5))
122                 - (G * m3 * (z2 - z3)) / (Math.pow(Math.pow(x3 - x2, 2)
                        + Math.pow(y3 - y2, 2) + Math.pow(z3 - z2, 2), 1.5));
123     k1_vx3 = - (G * m1 * (x3 - x1)) / (Math.pow(Math.pow(x3 - x1, 2)
            + Math.pow(y3 - y1, 2) + Math.pow(z3 - z1, 2), 1.5))
124                 - (G * m2 * (x3 - x2)) / (Math.pow(Math.pow(x3 - x2, 2)
                        + Math.pow(y3 - y2, 2) + Math.pow(z3 - z2, 2), 1.5));
125     k1_vy3 = - (G * m1 * (y3 - y1)) / (Math.pow(Math.pow(x3 - x1, 2)
            + Math.pow(y3 - y1, 2) + Math.pow(z3 - z1, 2), 1.5))
126                 - (G * m2 * (y3 - y2)) / (Math.pow(Math.pow(x3 - x2, 2)
                        + Math.pow(y3 - y2, 2) + Math.pow(z3 - z2, 2), 1.5));
127     k1_vz3 = - (G * m1 * (z3 - z1)) / (Math.pow(Math.pow(x3 - x1, 2)
```

```
        + Math.pow(y3 - y1, 2) + Math.pow(z3 - z1, 2), 1.5))
128           - (G * m2 * (z3 - z2)) / (Math.pow(Math.pow(x3 - x2, 2)
                 + Math.pow(y3 - y2, 2) + Math.pow(z3 - z2, 2), 1.5));
129
130     //k2
131     k2_x1 = v_x1 + (0.5 * h * k1_vx1);
132     k2_y1 = v_y1 + (0.5 * h * k1_vy1);
133     k2_z1 = v_z1 + (0.5 * h * k1_vz1);
134     k2_x2 = v_x2 + (0.5 * h * k1_vx2);
135     k2_y2 = v_y2 + (0.5 * h * k1_vy2);
136     k2_z2 = v_z2 + (0.5 * h * k1_vz2);
137     k2_x3 = v_x3 + (0.5 * h * k1_vx3);
138     k2_y3 = v_y3 + (0.5 * h * k1_vy3);
139     k2_z3 = v_z3 + (0.5 * h * k1_vz3);
140
141
142     k2_vx1 = - (G * m2 * ((x1 + 0.5 * h * k1_x1) - (x2 + 0.5 * h *
           k1_x2))) / (Math.pow(Math.pow((x2 + 0.5 * h * k1_x2) - (x1 +
           0.5 * h * k1_x1), 2) + Math.pow((y2 + 0.5 * h * k1_y2) - (y1 +
            0.5 * h * k1_y1), 2) + Math.pow((z2 + 0.5 * h * k1_z2) - (z1
           + 0.5 * h * k1_z1), 2), 1.5))
143           - (G * m3 * ((x1 + 0.5 * h * k1_x1) - (x3 + 0.5 * h *
                 k1_x3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3)
                 - (x1 + 0.5 * h * k1_x1), 2) + Math.pow((y3 + 0.5 * h
                  * k1_y3) - (y1 + 0.5 * h * k1_y1), 2) + Math.pow((z3
                  + 0.5 * h * k1_z3) - (z1 + 0.5 * h * k1_z1), 2),
                 1.5));
144     k2_vy1 = - (G * m2 * ((y1 + 0.5 * h * k1_y1) - (y2 + 0.5 * h *
           k1_y2))) / (Math.pow(Math.pow((x2 + 0.5 * h * k1_x2) - (x1 +
           0.5 * h * k1_x1), 2) + Math.pow((y2 + 0.5 * h * k1_y2) - (y1 +
            0.5 * h * k1_y1), 2) + Math.pow((z2 + 0.5 * h * k1_z2) - (z1
           + 0.5 * h * k1_z1), 2), 1.5))
145           - (G * m3 * ((y1 + 0.5 * h * k1_y1) - (y3 + 0.5 * h *
                 k1_y3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3)
                 - (x1 + 0.5 * h * k1_x1), 2) + Math.pow((y3 + 0.5 * h
                  * k1_y3) - (y1 + 0.5 * h * k1_y1), 2) + Math.pow((z3
                  + 0.5 * h * k1_z3) - (z1 + 0.5 * h * k1_z1), 2),
                 1.5));
146     k2_vz1 = - (G * m2 * ((z1 + 0.5 * h * k1_z1) - (z2 + 0.5 * h *
           k1_z2))) / (Math.pow(Math.pow((x2 + 0.5 * h * k1_x2) - (x1 +
           0.5 * h * k1_x1), 2) + Math.pow((y2 + 0.5 * h * k1_y2) - (y1 +
            0.5 * h * k1_y1), 2) + Math.pow((z2 + 0.5 * h * k1_z2) - (z1
           + 0.5 * h * k1_z1), 2), 1.5))
147           - (G * m3 * ((z1 + 0.5 * h * k1_z1) - (z3 + 0.5 * h *
                 k1_z3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3)
                 - (x1 + 0.5 * h * k1_x1), 2) + Math.pow((y3 + 0.5 * h
                  * k1_y3) - (y1 + 0.5 * h * k1_y1), 2) + Math.pow((z3
                  + 0.5 * h * k1_z3) - (z1 + 0.5 * h * k1_z1), 2),
                 1.5));
148     k2_vx2 = - (G * m1 * ((x2 + 0.5 * h * k1_x2) - (x1 + 0.5 * h *
           k1_x1))) / (Math.pow(Math.pow((x2 + 0.5 * h * k1_x2) - (x1 +
```

```
        0.5 * h * k1_x1), 2) + Math.pow((y2 + 0.5 * h * k1_y2) - (y1 +
         0.5 * h * k1_y1), 2) + Math.pow((z2 + 0.5 * h * k1_z2) - (z1
        + 0.5 * h * k1_z1), 2), 1.5))
149             - (G * m3 * ((x2 + 0.5 * h * k1_x2) - (x3 + 0.5 * h *
                   k1_x3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3)
                   - (x2 + 0.5 * h * k1_x2), 2) + Math.pow((y3 + 0.5 * h
                    * k1_y3) - (y2 + 0.5 * h * k1_y2), 2) + Math.pow((z3
                    + 0.5 * h * k1_z3) - (z2 + 0.5 * h * k1_z2), 2),
                   1.5));
150     k2_vy2 = - (G * m1 * ((y2 + 0.5 * h * k1_y2) - (y1 + 0.5 * h *
        k1_y1))) / (Math.pow(Math.pow((x2 + 0.5 * h * k1_x2) - (x1 +
        0.5 * h * k1_x1), 2) + Math.pow((y2 + 0.5 * h * k1_y2) - (y1 +
         0.5 * h * k1_y1), 2) + Math.pow((z2 + 0.5 * h * k1_z2) - (z1
        + 0.5 * h * k1_z1), 2), 1.5))
151             - (G * m3 * ((y2 + 0.5 * h * k1_y2) - (y3 + 0.5 * h *
                   k1_y3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3)
                   - (x2 + 0.5 * h * k1_x2), 2) + Math.pow((y3 + 0.5 * h
                    * k1_y3) - (y2 + 0.5 * h * k1_y2), 2) + Math.pow((z3
                    + 0.5 * h * k1_z3) - (z2 + 0.5 * h * k1_z2), 2),
                   1.5));
152     k2_vz2 = - (G * m1 * ((z2 + 0.5 * h * k1_z2) - (z1 + 0.5 * h *
        k1_z1))) / (Math.pow(Math.pow((x2 + 0.5 * h * k1_x2) - (x1 +
        0.5 * h * k1_x1), 2) + Math.pow((y2 + 0.5 * h * k1_y2) - (y1 +
         0.5 * h * k1_y1), 2) + Math.pow((z2 + 0.5 * h * k1_z2) - (z1
        + 0.5 * h * k1_z1), 2), 1.5))
153             - (G * m3 * ((z2 + 0.5 * h * k1_z2) - (z3 + 0.5 * h *
                   k1_z3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3)
                   - (x2 + 0.5 * h * k1_x2), 2) + Math.pow((y3 + 0.5 * h
                    * k1_y3) - (y2 + 0.5 * h * k1_y2), 2) + Math.pow((z3
                    + 0.5 * h * k1_z3) - (z2 + 0.5 * h * k1_z2), 2),
                   1.5));
154     k2_vx3 = - (G * m1 * ((x3 + 0.5 * h * k1_x3) - (x1 + 0.5 * h *
        k1_x1))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3) - (x1 +
        0.5 * h * k1_x1), 2) + Math.pow((y3 + 0.5 * h * k1_y3) - (y1 +
         0.5 * h * k1_y1), 2) + Math.pow((z3 + 0.5 * h * k1_z3) - (z1
        + 0.5 * h * k1_z1), 2), 1.5))
155             - (G * m2 * ((x3 + 0.5 * h * k1_x3) - (x2 + 0.5 * h *
                   k1_x2))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3)
                   - (x2 + 0.5 * h * k1_x2), 2) + Math.pow((y3 + 0.5 * h
                    * k1_y3) - (y2 + 0.5 * h * k1_y2), 2) + Math.pow((z3
                    + 0.5 * h * k1_z3) - (z2 + 0.5 * h * k1_z2), 2),
                   1.5));
156     k2_vy3 = - (G * m1 * ((y3 + 0.5 * h * k1_y3) - (y1 + 0.5 * h *
        k1_y1))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3) - (x1 +
        0.5 * h * k1_x1), 2) + Math.pow((y3 + 0.5 * h * k1_y3) - (y1 +
         0.5 * h * k1_y1), 2) + Math.pow((z3 + 0.5 * h * k1_z3) - (z1
        + 0.5 * h * k1_z1), 2), 1.5))
157             - (G * m2 * ((y3 + 0.5 * h * k1_y3) - (y2 + 0.5 * h *
                   k1_y2))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3)
                   - (x2 + 0.5 * h * k1_x2), 2) + Math.pow((y3 + 0.5 * h
                    * k1_y3) - (y2 + 0.5 * h * k1_y2), 2) + Math.pow((z3
```

```
                + 0.5 * h * k1_z3) - (z2 + 0.5 * h * k1_z2), 2),
                1.5));
158     k2_vz3 = - (G * m1 * ((z3 + 0.5 * h * k1_z3) - (z1 + 0.5 * h *
            k1_z1))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3) - (x1 +
            0.5 * h * k1_x1), 2) + Math.pow((y3 + 0.5 * h * k1_y3) - (y1 +
             0.5 * h * k1_y1), 2) + Math.pow((z3 + 0.5 * h * k1_z3) - (z1
            + 0.5 * h * k1_z1), 2), 1.5))
159             - (G * m2 * ((z3 + 0.5 * h * k1_z3) - (z2 + 0.5 * h *
                k1_z2))) / (Math.pow(Math.pow((x3 + 0.5 * h * k1_x3)
                - (x2 + 0.5 * h * k1_x2), 2) + Math.pow((y3 + 0.5 * h
                 * k1_y3) - (y2 + 0.5 * h * k1_y2), 2) + Math.pow((z3
                 + 0.5 * h * k1_z3) - (z2 + 0.5 * h * k1_z2), 2),
                1.5));
160
161     //k3
162     k3_x1 = v_x1 + 0.5 * h * k2_vx1;
163     k3_y1 = v_y1 + 0.5 * h * k2_vy1;
164     k3_z1 = v_z1 + 0.5 * h * k2_vz1;
165     k3_x2 = v_x2 + 0.5 * h * k2_vx2;
166     k3_y2 = v_y2 + 0.5 * h * k2_vy2;
167     k3_z2 = v_z2 + 0.5 * h * k2_vz2;
168     k3_x3 = v_x3 + 0.5 * h * k2_vx3;
169     k3_y3 = v_y3 + 0.5 * h * k2_vy3;
170     k3_z3 = v_z3 + 0.5 * h * k2_vz3;
171
172     k3_vx1 = - (G * m2 * ((x1 + 0.5 * h * k2_x1) - (x2 + 0.5 * h *
            k2_x2))) / (Math.pow(Math.pow((x2 + 0.5 * h * k2_x2) - (x1 +
            0.5 * h * k2_x1), 2) + Math.pow((y2 + 0.5 * h * k2_y2) - (y1 +
             0.5 * h * k2_y1), 2) + Math.pow((z2 + 0.5 * h * k2_z2) - (z1
            + 0.5 * h * k2_z1), 2), 1.5))
173             - (G * m3 * ((x1 + 0.5 * h * k2_x1) - (x3 + 0.5 * h *
                k2_x3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3)
                - (x1 + 0.5 * h * k2_x1), 2) + Math.pow((y3 + 0.5 * h
                 * k2_y3) - (y1 + 0.5 * h * k2_y1), 2) + Math.pow((z3
                 + 0.5 * h * k2_z3) - (z1 + 0.5 * h * k2_z1), 2),
                1.5));
174     k3_vy1 = - (G * m2 * ((y1 + 0.5 * h * k2_y1) - (y2 + 0.5 * h *
            k2_y2))) / (Math.pow(Math.pow((x2 + 0.5 * h * k2_x2) - (x1 +
            0.5 * h * k2_x1), 2) + Math.pow((y2 + 0.5 * h * k2_y2) - (y1 +
             0.5 * h * k2_y1), 2) + Math.pow((z2 + 0.5 * h * k2_z2) - (z1
            + 0.5 * h * k2_z1), 2), 1.5))
175             - (G * m3 * ((y1 + 0.5 * h * k2_y1) - (y3 + 0.5 * h *
                k2_y3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3)
                - (x1 + 0.5 * h * k2_x1), 2) + Math.pow((y3 + 0.5 * h
                 * k2_y3) - (y1 + 0.5 * h * k2_y1), 2) + Math.pow((z3
                 + 0.5 * h * k2_z3) - (z1 + 0.5 * h * k2_z1), 2),
                1.5));
176     k3_vz1 = - (G * m2 * ((z1 + 0.5 * h * k2_z1) - (z2 + 0.5 * h *
            k2_z2))) / (Math.pow(Math.pow((x2 + 0.5 * h * k2_x2) - (x1 +
            0.5 * h * k2_x1), 2) + Math.pow((y2 + 0.5 * h * k2_y2) - (y1 +
             0.5 * h * k2_y1), 2) + Math.pow((z2 + 0.5 * h * k2_z2) - (z1
```

```
        + 0.5 * h * k2_z1), 2), 1.5))
177           - (G * m3 * ((z1 + 0.5 * h * k2_z1) - (z3 + 0.5 * h *
                 k2_z3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3)
                 - (x1 + 0.5 * h * k2_x1), 2) + Math.pow((y3 + 0.5 * h
                  * k2_y3) - (y1 + 0.5 * h * k2_y1), 2) + Math.pow((z3
                  + 0.5 * h * k2_z3) - (z1 + 0.5 * h * k2_z1), 2),
                 1.5));
178     k3_vx2 = - (G * m1 * ((x2 + 0.5 * h * k2_x2) - (x1 + 0.5 * h *
        k2_x1))) / (Math.pow(Math.pow((x2 + 0.5 * h * k2_x2) - (x1 +
        0.5 * h * k2_x1), 2) + Math.pow((y2 + 0.5 * h * k2_y2) - (y1 +
         0.5 * h * k2_y1), 2) + Math.pow((z2 + 0.5 * h * k2_z2) - (z1
        + 0.5 * h * k2_z1), 2), 1.5))
179           - (G * m3 * ((x2 + 0.5 * h * k2_x2) - (x3 + 0.5 * h *
                 k2_x3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3)
                 - (x2 + 0.5 * h * k2_x2), 2) + Math.pow((y3 + 0.5 * h
                  * k2_y3) - (y2 + 0.5 * h * k2_y2), 2) + Math.pow((z3
                  + 0.5 * h * k2_z3) - (z2 + 0.5 * h * k2_z2), 2),
                 1.5));
180     k3_vy2 = - (G * m1 * ((y2 + 0.5 * h * k2_y2) - (y1 + 0.5 * h *
        k2_y1))) / (Math.pow(Math.pow((x2 + 0.5 * h * k2_x2) - (x1 +
        0.5 * h * k2_x1), 2) + Math.pow((y2 + 0.5 * h * k2_y2) - (y1 +
         0.5 * h * k2_y1), 2) + Math.pow((z2 + 0.5 * h * k2_z2) - (z1
        + 0.5 * h * k2_z1), 2), 1.5))
181           - (G * m3 * ((y2 + 0.5 * h * k2_y2) - (y3 + 0.5 * h *
                 k2_y3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3)
                 - (x2 + 0.5 * h * k2_x2), 2) + Math.pow((y3 + 0.5 * h
                  * k2_y3) - (y2 + 0.5 * h * k2_y2), 2) + Math.pow((z3
                  + 0.5 * h * k2_z3) - (z2 + 0.5 * h * k2_z2), 2),
                 1.5));
182     k3_vz2 = - (G * m1 * ((z2 + 0.5 * h * k2_z2) - (z1 + 0.5 * h *
        k2_z1))) / (Math.pow(Math.pow((x2 + 0.5 * h * k2_x2) - (x1 +
        0.5 * h * k2_x1), 2) + Math.pow((y2 + 0.5 * h * k2_y2) - (y1 +
         0.5 * h * k2_y1), 2) + Math.pow((z2 + 0.5 * h * k2_z2) - (z1
        + 0.5 * h * k2_z1), 2), 1.5))
183           - (G * m3 * ((z2 + 0.5 * h * k2_z2) - (z3 + 0.5 * h *
                 k2_z3))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3)
                 - (x2 + 0.5 * h * k2_x2), 2) + Math.pow((y3 + 0.5 * h
                  * k2_y3) - (y2 + 0.5 * h * k2_y2), 2) + Math.pow((z3
                  + 0.5 * h * k2_z3) - (z2 + 0.5 * h * k2_z2), 2),
                 1.5));
184     k3_vx3 = - (G * m1 * ((x3 + 0.5 * h * k2_x3) - (x1 + 0.5 * h *
        k2_x1))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3) - (x1 +
        0.5 * h * k2_x1), 2) + Math.pow((y3 + 0.5 * h * k2_y3) - (y1 +
         0.5 * h * k2_y1), 2) + Math.pow((z3 + 0.5 * h * k2_z3) - (z1
        + 0.5 * h * k2_z1), 2), 1.5))
185           - (G * m2 * ((x3 + 0.5 * h * k2_x3) - (x2 + 0.5 * h *
                 k2_x2))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3)
                 - (x2 + 0.5 * h * k2_x2), 2) + Math.pow((y3 + 0.5 * h
                  * k2_y3) - (y2 + 0.5 * h * k2_y2), 2) + Math.pow((z3
                  + 0.5 * h * k2_z3) - (z2 + 0.5 * h * k2_z2), 2),
                 1.5));
```

```
186     k3_vy3 = - (G * m1 * ((y3 + 0.5 * h * k2_y3) - (y1 + 0.5 * h *
            k2_y1))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3) - (x1 +
            0.5 * h * k2_x1), 2) + Math.pow((y3 + 0.5 * h * k2_y3) - (y1 +
             0.5 * h * k2_y1), 2) + Math.pow((z3 + 0.5 * h * k2_z3) - (z1
            + 0.5 * h * k2_z1), 2), 1.5))
187                 - (G * m2 * ((y3 + 0.5 * h * k2_y3) - (y2 + 0.5 * h *
                        k2_y2))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3)
                        - (x2 + 0.5 * h * k2_x2), 2) + Math.pow((y3 + 0.5 * h
                         * k2_y3) - (y2 + 0.5 * h * k2_y2), 2) + Math.pow((z3
                         + 0.5 * h * k2_z3) - (z2 + 0.5 * h * k2_z2), 2),
                        1.5));
188     k3_vz3 = - (G * m1 * ((z3 + 0.5 * h * k2_z3) - (z1 + 0.5 * h *
            k2_z1))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3) - (x1 +
            0.5 * h * k2_x1), 2) + Math.pow((y3 + 0.5 * h * k2_y3) - (y1 +
             0.5 * h * k2_y1), 2) + Math.pow((z3 + 0.5 * h * k2_z3) - (z1
            + 0.5 * h * k2_z1), 2), 1.5))
189                 - (G * m2 * ((z3 + 0.5 * h * k2_z3) - (z2 + 0.5 * h *
                        k2_z2))) / (Math.pow(Math.pow((x3 + 0.5 * h * k2_x3)
                        - (x2 + 0.5 * h * k2_x2), 2) + Math.pow((y3 + 0.5 * h
                         * k2_y3) - (y2 + 0.5 * h * k2_y2), 2) + Math.pow((z3
                         + 0.5 * h * k2_z3) - (z2 + 0.5 * h * k2_z2), 2),
                        1.5));
190
191     //k4
192     k4_x1 = v_x1 + h * k3_vx1;
193     k4_y1 = v_y1 + h * k3_vy1;
194     k4_z1 = v_z1 + h * k3_vz1;
195     k4_x2 = v_x2 + h * k3_vx2;
196     k4_y2 = v_y2 + h * k3_vy2;
197     k4_z2 = v_z2 + h * k3_vz2;
198     k4_x3 = v_x3 + h * k3_vx3;
199     k4_y3 = v_y3 + h * k3_vy3;
200     k4_z3 = v_z3 + h * k3_vz3;
201
202     k4_vx1 = - (G * m2 * ((x1 + h * k3_x1) - (x2 + h * k3_x2))) / (
            Math.pow(Math.pow((x2 + h * k3_x2) - (x1 + h * k3_x1), 2) +
            Math.pow((y2 + h * k3_y2) - (y1 + h * k3_y1), 2) + Math.pow((
            z2 + h * k3_z2) - (z1 + h * k3_z1), 2), 1.5))
203                 - (G * m3 * ((x1 + h * k3_x1) - (x3 + h * k3_x3))) / (
                        Math.pow(Math.pow((x3 + h * k3_x3) - (x1 + h * k3_x1)
                        , 2) + Math.pow((y3 + h * k3_y3) - (y1 + h * k3_y1),
                        2) + Math.pow((z3 + h * k3_z3) - (z1 + h * k3_z1), 2)
                        , 1.5));
204     k4_vy1 = - (G * m2 * ((y1 + h * k3_y1) - (y2 + h * k3_y2))) / (
            Math.pow(Math.pow((x2 + h * k3_x2) - (x1 + h * k3_x1), 2) +
            Math.pow((y2 + h * k3_y2) - (y1 + h * k3_y1), 2) + Math.pow((
            z2 + h * k3_z2) - (z1 + h * k3_z1), 2), 1.5))
205                 - (G * m3 * ((y1 + h * k3_y1) - (y3 + h * k3_y3))) / (
                        Math.pow(Math.pow((x3 + h * k3_x3) - (x1 + h * k3_x1)
                        , 2) + Math.pow((y3 + h * k3_y3) - (y1 + h * k3_y1),
                        2) + Math.pow((z3 + h * k3_z3) - (z1 + h * k3_z1), 2)
```

```
                , 1.5));
206     k4_vz1 = - (G * m2 * ((z1 + h * k3_z1) - (z2 + h * k3_z2))) / (
            Math.pow(Math.pow((x2 + h * k3_x2) - (x1 + h * k3_x1), 2) +
            Math.pow((y2 + h * k3_y2) - (y1 + h * k3_y1), 2) + Math.pow((
            z2 + h * k3_z2) - (z1 + h * k3_z1), 2), 1.5))
207               - (G * m3 * ((z1 + h * k3_z1) - (z3 + h * k3_z3))) / (
                      Math.pow(Math.pow((x3 + h * k3_x3) - (x1 + h * k3_x1)
                      , 2) + Math.pow((y3 + h * k3_y3) - (y1 + h * k3_y1),
                      2) + Math.pow((z3 + h * k3_z3) - (z1 + h * k3_z1), 2)
                      , 1.5));
208     k4_vx2 = - (G * m1 * ((x2 + h * k3_x2) - (x1 + h * k3_x1))) / (
            Math.pow(Math.pow((x2 + h * k3_x2) - (x1 + h * k3_x1), 2) +
            Math.pow((y2 + h * k3_y2) - (y1 + h * k3_y1), 2) + Math.pow((
            z2 + h * k3_z2) - (z1 + h * k3_z1), 2), 1.5))
209               - (G * m3 * ((x2 + h * k3_x2) - (x3 + h * k3_x3))) / (
                      Math.pow(Math.pow((x3 + h * k3_x3) - (x2 + h * k3_x2)
                      , 2) + Math.pow((y3 + h * k3_y3) - (y2 + h * k3_y2),
                      2) + Math.pow((z3 + h * k3_z3) - (z2 + h * k3_z2), 2)
                      , 1.5));
210     k4_vy2 = - (G * m1 * ((y2 + h * k3_y2) - (y1 + h * k3_y1))) / (
            Math.pow(Math.pow((x2 + h * k3_x2) - (x1 + h * k3_x1), 2) +
            Math.pow((y2 + h * k3_y2) - (y1 + h * k3_y1), 2) + Math.pow((
            z2 + h * k3_z2) - (z1 + h * k3_z1), 2), 1.5))
211               - (G * m3 * ((y2 + h * k3_y2) - (y3 + h * k3_y3))) / (
                      Math.pow(Math.pow((x3 + h * k3_x3) - (x2 + h * k3_x2)
                      , 2) + Math.pow((y3 + h * k3_y3) - (y2 + h * k3_y2),
                      2) + Math.pow((z3 + h * k3_z3) - (z2 + h * k3_z2), 2)
                      , 1.5));
212     k4_vz2 = - (G * m1 * ((z2 + h * k3_z2) - (z1 + h * k3_z1))) / (
            Math.pow(Math.pow((x2 + h * k3_x2) - (x1 + h * k3_x1), 2) +
            Math.pow((y2 + h * k3_y2) - (y1 + h * k3_y1), 2) + Math.pow((
            z2 + h * k3_z2) - (z1 + h * k3_z1), 2), 1.5))
213               - (G * m3 * ((z2 + h * k3_z2) - (z3 + h * k3_z3))) / (
                      Math.pow(Math.pow((x3 + h * k3_x3) - (x2 + h * k3_x2)
                      , 2) + Math.pow((y3 + h * k3_y3) - (y2 + h * k3_y2),
                      2) + Math.pow((z3 + h * k3_z3) - (z2 + h * k3_z2), 2)
                      , 1.5));
214     k4_vx3 = - (G * m1 * ((x3 + h * k3_x3) - (x1 + h * k3_x1))) / (
            Math.pow(Math.pow((x3 + h * k3_x3) - (x1 + h * k3_x1), 2) +
            Math.pow((y3 + h * k3_y3) - (y1 + h * k3_y1), 2) + Math.pow((
            z3 + h * k3_z3) - (z1 + h * k3_z1), 2), 1.5))
215               - (G * m2 * ((x3 + h * k3_x3) - (x2 + h * k3_x2))) / (
                      Math.pow(Math.pow((x3 + h * k3_x3) - (x2 + h * k3_x2)
                      , 2) + Math.pow((y3 + h * k3_y3) - (y2 + h * k3_y2),
                      2) + Math.pow((z3 + h * k3_z3) - (z2 + h * k3_z2), 2)
                      , 1.5));
216     k4_vy3 = - (G * m1 * ((y3 + h * k3_y3) - (y1 + h * k3_y1))) / (
            Math.pow(Math.pow((x3 + h * k3_x3) - (x1 + h * k3_x1), 2) +
            Math.pow((y3 + h * k3_y3) - (y1 + h * k3_y1), 2) + Math.pow((
            z3 + h * k3_z3) - (z1 + h * k3_z1), 2), 1.5))
217               - (G * m2 * ((y3 + h * k3_y3) - (y2 + h * k3_y2))) / (
```

```
                Math.pow(Math.pow((x3 + h * k3_x3) - (x2 + h * k3_x2)
                , 2) + Math.pow((y3 + h * k3_y3) - (y2 + h * k3_y2),
                2) + Math.pow((z3 + h * k3_z3) - (z2 + h * k3_z2), 2)
                , 1.5));
218     k4_vz3 = - (G * m1 * ((z3 + h * k3_z3) - (z1 + h * k3_z1))) / (
           Math.pow(Math.pow((x3 + h * k3_x3) - (x1 + h * k3_x1), 2) +
           Math.pow((y3 + h * k3_y3) - (y1 + h * k3_y1), 2) + Math.pow((
           z3 + h * k3_z3) - (z1 + h * k3_z1), 2), 1.5))
219             - (G * m2 * ((z3 + h * k3_z3) - (z2 + h * k3_z2))) / (
                Math.pow(Math.pow((x3 + h * k3_x3) - (x2 + h * k3_x2)
                , 2) + Math.pow((y3 + h * k3_y3) - (y2 + h * k3_y2),
                2) + Math.pow((z3 + h * k3_z3) - (z2 + h * k3_z2), 2)
                , 1.5));
220
221
222     //incremento
223
224     x1 = x1 + (h / 6) * (k1_x1 + (2 * k2_x1) + (2 * k3_x1) + k4_x1);
225     y1 = y1 + (h / 6) * (k1_y1 + (2 * k2_y1) + (2 * k3_y1) + k4_y1);
226     z1 = z1 + (h / 6) * (k1_z1 + (2 * k2_z1) + (2 * k3_z1) + k4_z1);
227     x2 = x2 + (h / 6) * (k1_x2 + (2 * k2_x2) + (2 * k3_x2) + k4_x2);
228     y2 = y2 + (h / 6) * (k1_y2 + (2 * k2_y2) + (2 * k3_y2) + k4_y2);
229     z2 = z2 + (h / 6) * (k1_z2 + (2 * k2_z2) + (2 * k3_z2) + k4_z2);
230     x3 = x3 + (h / 6) * (k1_x3 + (2 * k2_x3) + (2 * k3_x3) + k4_x3);
231     y3 = y3 + (h / 6) * (k1_y3 + (2 * k2_y3) + (2 * k3_y3) + k4_y3);
232     z3 = z3 + (h / 6) * (k1_z3 + (2 * k2_z3) + (2 * k3_z3) + k4_z3);
233
234     v_x1 = v_x1 + (h / 6) * (k1_vx1 + (2 * k2_vx1) + (2 * k3_vx1) +
           k4_vx1);
235     v_y1 = v_y1 + (h / 6) * (k1_vy1 + (2 * k2_vy1) + (2 * k3_vy1) +
           k4_vy1);
236     v_z1 = v_z1 + (h / 6) * (k1_vz1 + (2 * k2_vz1) + (2 * k3_vz1) +
           k4_vz1);
237     v_x2 = v_x2 + (h / 6) * (k1_vx2 + (2 * k2_vx2) + (2 * k3_vx2) +
           k4_vx2);
238     v_y2 = v_y2 + (h / 6) * (k1_vy2 + (2 * k2_vy2) + (2 * k3_vy2) +
           k4_vy2);
239     v_z2 = v_z2 + (h / 6) * (k1_vz2 + (2 * k2_vz2) + (2 * k3_vz2) +
           k4_vz2);
240     v_x3 = v_x3 + (h / 6) * (k1_vx3 + (2 * k2_vx3) + (2 * k3_vx3) +
           k4_vx3);
241     v_y3 = v_y3 + (h / 6) * (k1_vy3 + (2 * k2_vy3) + (2 * k3_vy3) +
           k4_vy3);
242     v_z3 = v_z3 + (h / 6) * (k1_vz3 + (2 * k2_vz3) + (2 * k3_vz3) +
           k4_vz3);
243
244     t = t + h;
245 }
```

## D PROJETO COM *SCRUM*

O desenvolvimento de software, mesmo em contexto acadêmico, beneficia-se de uma abordagem estruturada que permita flexibilidade, gestão de incertezas e entregas incrementais de valor. Para a gestão deste trabalho, foi adotado o *framework Scrum*[1], uma metodologia ágil voltada ao desenvolvimento iterativo e incremental de produtos complexos.

O *Scrum* fundamenta-se nos pilares do empirismo — transparência, inspeção e adaptação — o que o torna particularmente adequado para projetos de pesquisa e desenvolvimento, nos quais os requisitos podem evoluir ao longo do processo.

De acordo com o *Scrum Guide* [125], o *framework* estabelece três papéis fundamentais no desenvolvimento de projetos:

**Product Owner**: responsável por representar os interesses do cliente ou usuário final e maximizar o valor do produto desenvolvido. Atua como elo entre os stakeholders e a equipe de desenvolvimento, sendo responsável por definir a visão do produto, gerenciar e priorizar o *Product Backlog*, planejar as liberações (*releases*) e validar as entregas ao final de cada *sprint*. Entre suas principais atribuições estão a definição das prioridades do projeto, o gerenciamento de novos requisitos e a aceitação ou rejeição dos incrementos desenvolvidos.

**Scrum Master**: atua como facilitador do processo, garantindo que os princípios e práticas do *Scrum* sejam corretamente aplicados. Diferentemente de um gerente tradicional, não exerce autoridade hierárquica sobre a equipe, mas trabalha removendo impedimentos, promovendo a colaboração e apoiando a equipe na melhoria contínua do processo. Também auxilia o *Product Owner* na organização do *Product Backlog* e assegura que o trabalho ocorra em conformidade com os valores da metodologia ágil.

**Development Team**: corresponde à equipe responsável pela construção do produto. Trata-se de um grupo multidisciplinar e auto-organizado, que define a melhor forma de executar as atividades necessárias para cada *sprint*. Seus integrantes são responsáveis por projetar, desenvolver, testar e manter o produto, garantindo a qualidade dos incrementos entregues ao final de cada ciclo de desenvolvimento.

A metodologia *Scrum* organiza o trabalho em ciclos iterativos denominados *sprints*, que geralmente possuem duração entre uma semana e um mês. Cada *sprint* possui data de início e término definidas e resulta na entrega de um incremento funcional do produto, ou seja, uma versão potencialmente utilizável e mais evoluída do sistema. Durante o ciclo de desenvolvimento, mudanças significativas no escopo são evitadas para preservar o foco da equipe e garantir a entrega planejada.

A curta duração dos *sprints* proporciona diversos benefícios ao processo de desenvolvimento, tais como maior facilidade de planejamento, obtenção rápida de *feedback*, melhor controle do progresso do projeto, redução de riscos e maior motivação da equipe por meio de ciclos frequentes de entrega e avaliação.

O ciclo de vida completo do produto consiste na repetição sucessiva desses *sprints* até a conclusão do projeto.

[1] *Scrum* é uma das metodologias ágeis mais difundidas para gestão de projetos complexos. Organiza o trabalho em ciclos curtos (*sprints*) que permitem inspeção e adaptação frequentes do produto, promovendo colaboração, flexibilidade e entrega contínua de valor.

### D.0.1 Adaptação do *Scrum* para um projeto acadêmico

Embora o *Scrum* tenha sido originalmente concebido para equipes de desenvolvimento de *software*, sua natureza iterativa, incremental e adaptativa permite sua aplicação em diversos contextos, incluindo a gestão de projetos de pesquisa científica. Nesse cenário, o *framework* pode ser adaptado para conciliar agilidade no desenvolvimento com o rigor metodológico exigido pela produção acadêmica [126].

No contexto de um TCC, em que o desenvolvimento do projeto é frequentemente realizado por um único estudante, torna-se necessária uma adaptação dos papéis tradicionais do *Scrum*. Nesse modelo, o discente assume simultaneamente o papel de *Developer* (responsável pela execução técnica e científica do projeto) e parte das funções do *Product Owner*, especialmente no que se refere à organização do *backlog* e à definição das tarefas técnicas.

O orientador, por sua vez, atua como facilitador do processo, auxiliando na definição das prioridades, avaliando as entregas e garantindo o rigor científico do trabalho. Assim, o projeto é estruturado em ciclos de desenvolvimento (*sprints*) de curta duração — geralmente entre uma e quatro semanas —, nos quais são produzidos incrementos concretos do trabalho, como capítulos desenvolvidos, funcionalidades implementadas ou experimentos concluídos.

Essa abordagem permite que o estudante mantenha foco em metas menores e progressivas, favorecendo o acompanhamento do progresso do projeto e possibilitando ajustes rápidos no planejamento por meio de revisões periódicas [127].

Dessa forma, a adaptação do *Scrum* para este trabalho foi organizada da seguinte maneira:

- **Developer (Desenvolvedor):** papel desempenhado pela autora deste trabalho, responsável pela execução das atividades técnicas, implementação do sistema, desenvolvimento das simulações e redação do documento científico.
- **Product Owner (Dono do Produto):** papel compartilhado entre a autora, responsável pela organização técnica do *backlog*, e o professor orientador, que define a visão geral do projeto, os requisitos pedagógicos e valida as entregas realizadas.
- **Scrum Master:** função assumida pela própria autora, com foco na autogestão do processo de desenvolvimento, identificação e resolução de impedimentos técnicos ou conceituais e garantia da aplicação adequada das práticas do *Scrum*.

A adaptação do *Scrum* para equipes reduzidas ou individuais tem sido discutida na literatura como uma estratégia eficaz para organização e acompanhamento de projetos acadêmicos, contribuindo para a manutenção do foco, da produtividade e do progresso contínuo durante o desenvolvimento da pesquisa.

### D.0.2 Artefatos do *Scrum* no projeto

Os artefatos do *Scrum* foram criados para garantir a transparência sobre o trabalho a ser feito e o progresso alcançado a cada novo encontro com o orientador que revisava os resultados e orientava quais funcionalidades a simulação deveria ter. Assim foram criadas as orientações e desenvolvimento do *Product Backlog*.

### a) Product Backlog

O *Product Backlog*[2] é a lista ordenada de tudo o que é necessário para o produto final. Os itens foram priorizados com base no valor que agregam ao objetivo do projeto: criar uma simulação funcional e pedagogicamente útil (Tab. D.1).

### b) Sprints e Sprint Backlogs

O desenvolvimento do simulador foi estruturado em quatro *Sprints*[3], cada uma com duração de três semanas. Esta cadência permitiu a inspeção regular do progresso e a adaptação dos requisitos técnicos e pedagógicos conforme a evolução do projeto.

A Tabela D.2 apresenta o cronograma consolidado, detalhando as metas de cada ciclo e os respectivos itens do *Product Backlog* (PBIs) contemplados.

### c) Definition of Done (DoD)

Para assegurar o rigor técnico e a consistência acadêmica em cada incremento, estabeleceu-se uma Definição de Pronto (*Definition of Done* — DoD). Esta atua como um contrato de qualidade: um item do *Backlog* só é considerado oficialmente concluído ao satisfazer integralmente os critérios descritos na Tabela D.3.

[2] Artefato do *scrum* que consiste em uma lista ordenada e priorizada de todas as funcionalidades, requisitos e melhorias desejadas para um produto. É a única fonte de trabalho para o time de desenvolvimento.

[3] Evento central do *Scrum*. Trata-se de um ciclo de tempo fixo (*time-box*), geralmente de uma a quatro semanas, destinado à criação de um incremento de produto utilizável e potencialmente liberável.

**Tabela D.1:** Product Backlog ordenado e fundamentado.

| ID | Item (PBI) | Descrição Técnica e Valor Agregado | Prioridade |
|---|---|---|---|
| PBI-01 | Ambiente Three.js | Configuração do *boilerplate* inicial e cena 3D básica. Essencial para a estabilidade técnica. | Muito Alta |
| PBI-02 | Motor de Física | Implementação do motor para o problema de três corpos. Garante o rigor científico da pesquisa. | Crítica |
| PBI-03 | Renderização (Mesh) | Criação das esferas para representação física. Visualização das entidades no espaço. | Alta |
| PBI-04 | Integração de Loop | Sincronização do motor de física com o ciclo de animação para movimento em tempo real. | Alta |
| PBI-05 | Interface de *Inputs* | Desenvolvimento de UI para inserção de massas, posições e velocidades. Foco na autonomia do usuário. | Alta |
| PBI-06 | Controles de Fluxo | Botões de Iniciar, Pausar e Reiniciar. Essencial para o controle pedagógico do ritmo. | Média |
| PBI-07 | Controles de Câmera | Implementação do *OrbitControls* para exploração espacial e desenvolvimento da intuição 3D. | Média |
| PBI-08 | Rastro (Órbita) | Funcionalidade de desenho de trajetórias. Recurso pedagógico para visualização de padrões. | Média |
| PBI-09 | Telemetria | Exibição de vetores e dados numéricos em tempo real. Apoio à análise científica de dados. | Baixa |
| PBI-10 | Estética Visual | Refinamento de texturas, luz e fundo estrelado. Foco no engajamento e imersão do estudante. | Baixa |
| PBI-11 | Testes e Performance | Testes de caixa branca e otimização. Garante a confiabilidade do *software*. | Alta |
| PBI-12 | Integração SimuFísica | Compatibilização final com a plataforma *SimuFísica*®. Entrega do valor final. | Muito Alta |
| PBI-13 | Documentação | Elaboração do guia técnico e pedagógico. Formalização acadêmica do projeto. | Alta |

**Tabela D.2:** Detalhamento do cronograma e metas das *Sprints*.

| Sprint | Meta / Objetivo | Itens do *Sprint Backlog* (Tarefas) | PBIs | Duração |
|---|---|---|---|---|
| 1 | Base Tecnológica e Física | Estudo de *JavaScript* e *Three.js*; implementação do motor de física (RK4); renderização básica dos corpos celestes. | 01 a 04 | 3 sem. |
| 2 | Interatividade e Interface | Desenvolvimento da UI; configuração de *inputs* de dados (massa, posição, velocidade) e botões de controle de fluxo. | 05 a 07 | 3 sem. |
| 3 | Refinamento e Visualização | Implementação de rastros orbitais, inserção de grades de referência e ajustes de iluminação e texturas. | 08 a 10 | 3 sem. |
| 4 | Qualidade e Integração | Testes de caixa branca; comparação com dados reais; integração à plataforma *SimuFísica*® e documentação final. | 11 a 13 | 3 sem. |

**Tabela D.3:** Critérios da Definição de Pronto (DoD).

| Categoria | Critério de Aceitação / Qualidade |
|---|---|
| **Desenvolvimento** | Código-fonte implementado, devidamente comentado e seguindo padrões de legibilidade e boas práticas de programação. |
| **Qualidade** | Funcionalidade validada tecnicamente, operando sem erros (*bugs*[4]) que comprometam o motor de física. |
| **Validação** | Incremento revisado e aprovado pelo orientador em reunião de *Sprint Review*, garantindo o rigor científico. |
| **Integração** | Código testado e integrado com sucesso ao repositório principal (*Main Branch*) do projeto. |

### D.0.3 O ciclo de desenvolvimento iterativo

O desenvolvimento seguiu os eventos prescritos pelo *Scrum*. Ao início de cada *Sprint*, ocorria o *Planejamento do Sprint (Sprint Planning)*, onde a meta era definida e os itens do *backlog* eram selecionados. Semanalmente, uma autoavaliação (simulando um *Daily Scrum*) era realizada para verificar o progresso em relação à meta.

Ao final de cada Sprint, um incremento funcional do software era apresentado ao professor orientador na Revisão do Sprint (*Sprint Review*). Este evento era crucial para a inspeção do produto e adaptação do *Product Backlog* com base no *feedback* recebido. Em seguida, a Retrospectiva do *Sprint* (*Sprint Retrospective*) era um momento de reflexão sobre o processo de trabalho, identificando pontos a serem melhorados no *Sprint* seguinte.

A adoção do *Scrum* proporcionou uma estrutura auxiliar para o desenvolvimento deste TCC, permitindo a gestão de um projeto complexo de forma organizada, a mitigação de riscos através de ciclos curtos de *feedback* e a garantia de que o produto final estivesse alinhado com seus objetivos pedagógicos e técnicos.

### D.0.4 Avaliação do cumprimento do *Backlog* e das *Sprints*

Ao término do ciclo de desenvolvimento, constatou-se que as quatro *Sprints* planejadas foram executadas dentro do *time-box* estabelecido. A adoção do *framework Scrum* conferiu ao projeto uma gestão adaptativa, permitindo que a complexidade inerente à implementação do motor de física (RK4) na *Sprint* 1 fosse absorvida sem paralisar o desenvolvimento das interfaces nas etapas posteriores.

A Tabela D.4 resume o *status* final de cada item do *Product Backlog* (PBI). Embora o núcleo funcional do simulador tenha atingido a "Definição de Pronto" (*DoD*), a análise revela que a entrega final concentrou-se na estabilidade do motor e da renderização básica, em detrimento de recursos de telemetria e documentação pedagógica.

A conformidade dos itens planejados demonstra que a estruturação do trabalho em pequenos incrementos funcionais diminuiu os riscos de não conclusão. Particularmente, a antecipação dos testes de validação com dados reais (conforme discutido na Tabela 5.2) permitiu que eventuais desvios numéricos fossem corrigidos ainda durante a fase de desenvolvimento do motor de física, garantindo a entrega de uma ferramenta pedagogicamente fidedigna.

**Tabela D.4:** Status final do cumprimento do *Product Backlog*.

| ID | Item do Backlog (PBI) | Sprint | Status | DoD Atendida |
|---|---|---|---|---|
| PBI-01 | Ambiente *Three.js* e Cena 3D | 1 | Concluído | Sim |
| PBI-02 | Motor de Física (RK4) | 1 | Concluído | Sim |
| PBI-03 | Renderização dos Corpos | 1 | Concluído | Sim |
| PBI-04 | Integração Física-Animação | 1 | Concluído | Sim |
| PBI-05 | Interface de *Inputs* | 2 | Concluído | Sim |
| PBI-06 | Controles de Simulação | 2 | Concluído | Sim |
| PBI-07 | Controles de Câmera | 2 | Concluído | Sim |
| PBI-08 | Sistema de Rastros (Trajetórias) | 3 | Concluído | Sim |
| PBI-09 | Telemetria e Dados Numéricos | 3 | Não concluído | Não |
| PBI-10 | Estética e Iluminação | 3 | Concluído | Sim |
| PBI-11 | Testes de Caixa Branca e Desempenho | 4 | Concluído | Sim |
| PBI-12 | Integração à *SimuFísica*® | 4 | Não concluído | Não |
| PBI-13 | Documentação e Guia Pedagógico | 4 | Não concluído | Não |

**TERMO DE AUTORIZAÇÃO**

Eu, JUCIANE GONÇALVES MAIA, abaixo assinado, aluna regularmente matriculada no Curso de Licenciatura em Física, portador(a) do RA: 201611640, RG: ***.709.***-**-SSP-RO, CPF:_***.709.***-**, venho por meio deste autorizar a disponibilização pelo DEFIJI do meu Trabalho de Conclusão de Curso em meios eletrônicos existentes ou que venham a ser criados.

Ji-Paraná, 15 de junho de 2026.

gov.br
Documento assinado digitalmente
JUCIANE GONCALVES MAIA
Data: 15/06/2026 19:08:42-0300
Verifique em https://validar.iti.gov.br

JUCIANE GONÇALVES MAIA