PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

ESCOLA POLITÉCNICA E DE ARTES

GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

UNITY 3D: ESTUDO DOS SEUS RECURSOS E APLICAÇÃO NO DESENVOLVIMENTO DE ARTEFATOS DIGITAIS

FILIPE MOREIRA FERRACIOLI

GOIÂNIA – GO

2025

FILIPE MOREIRA FERRACIOLI

UNITY 3D: ESTUDO DOS SEUS RECURSOS E APLICAÇÃO NO DESENVOLVIMENTO DE ARTEFATOS DIGITAIS

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a):

Profª. Drª. Solange da Silva

Banca examinadora:

Prof. Me. Rafael Leal Martins

Prof. Me. Fernando Goncalves Abadia

GOIÂNIA – GO

2025

FILIPE MOREIRA FERRACIOLI

UNITY 3D: ESTUDO DOS SEUS RECURSOS E APLICAÇÃO NO DESENVOLVIMENTO DE ARTEFATOS DIGITAIS

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia da Computação, em: 02/12/2025.

_____

Orientadora: Profª. Drª. Solange da Silva

_____

Prof. Me. Rafael Leal Martins

_____

Prof. Me. Fernando Goncalves Abadia

AGRADECIMENTOS

Primeiramente, agradeço a Deus, por ter me sustentado em cada etapa dessa caminhada, concedendo-me forças nos momentos difíceis e sabedoria para superar os desafios.

À minha mãe, Sênia, e ao meu pai, Carlos, deixo minha sincera e eterna gratidão por tudo que fizeram por mim — pelos sacrifícios, conselhos, apoio, investimento, paciência e, sobretudo, pela fé em mim.

Estendo também meu agradecimento à minha madrinha, Cleide, e ao meu padrinho, Amarildo, por todo o apoio e carinho que me ofereceram ao longo dessa jornada tão importante da minha vida.

Em especial, deixo registrada minha eterna gratidão ao meu primo Lucas, que, mesmo não estando mais entre nós, continua vivo em minha memória e no meu coração, sendo uma inspiração constante.

Aos meus amigos e colegas de faculdade, agradeço por todos os momentos compartilhados, pelas boas risadas, aprendizados e companheirismo, que tornaram essa caminhada muito mais leve e significativa.

À minha orientadora, professora Solange da Silva, expresso minha imensa gratidão por sua paciência, dedicação e orientação. Seu apoio foi crucial para a realização deste trabalho.

Aos professores que me acompanharam ao longo do curso, agradeço não apenas pelo conhecimento transmitido, mas também pelos valiosos ensinamentos que levarei comigo por toda a vida.

RESUMO

Este trabalho apresenta um estudo sobre o funcionamento da *engine Unity 3D* e sua capacidade de gerar artefatos digitais por meio da integração entre componentes físicos, visuais e lógicos. Por meio de revisão bibliográfica, foram explorados conceitos fundamentais relacionados às *engines* de jogos, seus mecanismos internos, estrutura de objetos e principais recursos de desenvolvimento. Em seguida, foi conduzida uma etapa experimental baseada na construção de um artefato técnico, utilizado como instrumento para observar na prática o comportamento da *engine,* o fluxo de execução de scripts, o gerenciamento de cenas, a aplicação de física, o controle de câmera e a manipulação de terrenos tridimensionais. A análise dos resultados demonstrou que a *Unity* opera por meio de uma arquitetura modular e extensível, permitindo que diferentes sistemas trabalhem de forma integrada para gerar aplicações interativas. O estudo confirmou que a *engine* oferece ferramentas robustas e acessíveis para fins acadêmicos, educacionais e profissionais, além de evidenciar a importância de princípios de engenharia de *software*, como organização estrutural e modularidade, na construção de projetos escaláveis. Conclui-se que a *Unity 3D* é uma plataforma eficaz para compreender, demonstrar e desenvolver artefatos digitais, oferecendo suporte amplo para experimentação técnica e evolução contínua de aplicações.

Palavras-chave: *Unity 3D, Engines* de Jogos, Artefatos Digitais, Desenvolvimento Interativo.

ABSTRACT

This work presents a study on the functioning of the Unity 3D engine and its ability to generate digital artifacts through the integration of physical, visual, and logical components. Through a literature review, fundamental concepts related to game engines, their internal mechanisms, object structures, and key development features were explored. An experimental stage was then conducted based on the construction of a technical artifact, used as a tool to observe in practice the engine's behavior, script execution flow, scene management, physics simulation, camera control, and three-dimensional terrain manipulation. The analysis of the results demonstrated that Unity operates through a modular and extensible architecture, allowing different systems to work in an integrated manner to generate interactive applications. The study confirmed that the engine provides robust and accessible tools for academic, educational, and professional purposes, and highlighted the importance of software engineering principles—such as structural organization and modularity—in building scalable projects. It is concluded that Unity 3D is an effective platform for understanding, demonstrating, and developing digital prototypes, offering broad support for technical experimentation and continuous application evolution.

Keywords: Unity 3D; Game Engines; Digital Artifacts; Interactive Development.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AI	*Artificial Intelligence*
API	*Application Programming Interface*
C#	*C-Sharp*
CPU	*Central Processing Unit*
ES	Engenharia de *Software*
GPU	*Graphics Processing Unit*
IA	Inteligência Artificial
LOD	*Level of Detail*
MDA	Mecânica, Dinâmica e Estética
MVP	*Minimum Viable Product*
NPCs	*Non-Playable Character*
RAM	*Random Access Memory*
UI	*User Interface*
YAML	*YAML Ain't Markup Language*

SUMÁRIO

1. INTRODUÇÃO

Os jogos digitais têm experimentado um crescimento contínuo desde o início dos anos 2000, impulsionados pela evolução de *hardware*, *software* e técnicas de desenvolvimento. Nesse período, títulos como *The Sims*, *SimCity 4*, *Gran Turismo*, *Forza Motorsport*, *Arma*, *Euro Truck Simulator* e *Farming Simulator* consolidaram o gênero de simulação como um dos mais expressivos do mercado, ao reproduzir com níveis crescentes de precisão os sistemas e processos do mundo real (Gautron et al., 2022; Hax; Ferreira Filho, 2015). A busca por realismo gráfico, comportamental e interativo motivou o aprimoramento constante das *engines* de jogos, que passaram a oferecer ferramentas mais robustas para criação de ambientes, personagens, interfaces e sistemas dinâmicos.

Nesse contexto, o desenvolvimento de jogos passou a demandar soluções capazes de unificar modelagem, física, lógica, interface e organização de projetos. *Engines* como a *Unity 3D* tornaram-se fundamentais por disponibilizarem um ecossistema completo que integra componentes visuais, módulos de simulação física, ferramentas de edição, sistemas de *script* e estruturas para gerenciamento de *assets*. Tecnologias como inteligência artificial (IA) e física avançada, destacadas em pesquisas recentes, também exercem papel importante na construção de comportamentos complexos, mecanismos responsivos e dinâmicas interativas (Wu et al., 2022).

Além do entretenimento, sistemas digitais desenvolvidos em *engines* como a *Unity* vêm ganhando espaço em cenários de ensino e treinamento. Em simuladores agrícolas, por exemplo, atividades como operação de maquinário, entendimento de processos de cultivo e tomada de decisão podem ser praticadas de maneira segura e acessível, sem os custos e riscos associados a operações reais (Dot Digital Group, 2019). Essa aplicabilidade evidencia que *engines* de desenvolvimento não se limitam à criação de jogos comerciais, mas também atuam como plataformas para construção de artefatos interativos utilizados em diferentes áreas.

Diante desse cenário, compreender o funcionamento interno de uma *engine* e a maneira como ela organiza, processa e produz artefatos digitais é essencial para a formação de profissionais da computação. A *Unity 3D*, especificamente, destaca-se por sua documentação ampla, grande comunidade de suporte e um conjunto consolidado de ferramentas que favorecem tanto iniciantes quanto desenvolvedores

experientes. Assim, explorar seus recursos e demonstrar sua capacidade prática por meio da criação de um artefato representa uma oportunidade relevante de estudo.

Diante deste contexto, este projeto visa responder à seguinte questão de pesquisa: **Como funciona a *Unity 3D* e como ela gera artefatos digitais?**

Este trabalho tem como objetivo geral **apresentar o funcionamento da *engine* e demonstrar alguns de seus recursos por meio de implementações práticas**.

Os objetivos específicos são:

- **Apresentar as principais partes da *Unity***
- **Demonstrar exemplos de artefatos desenvolvidos dentro da ferramenta**, articulando conceitos teóricos com aplicações concretas.

Espera-se que os resultados deste trabalho possam contribuir:

- Na compreensão dos mecanismos internos da *Unity 3D.*
- Apresentando as estruturas de projeto e suas ferramentas fundamentais.
- Oferecendo uma visão clara sobre como a *engine* pode ser utilizada para criar aplicações interativas e artefatos funcionais.
- Auxiliando estudantes e desenvolvedores iniciantes a compreenderem como recursos da *engine* podem ser aplicados na construção de artefatos digitais completos.

Esta monografia está organizada da seguinte forma: o **Capítulo 1** apresenta a introdução, contendo a contextualização do tema, a questão de pesquisa e os objetivos do estudo. O **Capítulo 2** reúne o referencial teórico, abordando conceitos fundamentais sobre *engines* de desenvolvimento digital, aspectos técnicos relacionados à *Unity 3D* e trabalhos acadêmicos que tratam do desenvolvimento de jogos e artefatos utilizando essa ferramenta. O **Capítulo 3** descreve os materiais e métodos empregados na pesquisa, detalhando as etapas bibliográficas e experimentais. O **Capítulo 4** apresenta a ferramenta *Unity 3D,* explorando sua estrutura, seus componentes essenciais e o funcionamento da *engine*. O **Capítulo 5** descreve o desenvolvimento do artefato criado para demonstrar, na prática, os recursos estudados da *Unity*. O **Capítulo 6** aborda técnicas e funcionalidades avançadas da *engine* aplicadas durante a construção do artefato. O **Capítulo 7** discute os resultados obtidos por meio das implementações e experimentos realizados. Por fim, o **Capítulo 8** apresenta a conclusão, destacando as contribuições, aprendizados e possibilidades de desdobramentos futuros.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta os fundamentos teóricos relacionados às *engines* de desenvolvimento de jogos, com foco na *Unity 3D*. São discutidos conceitos essenciais sobre *engines*, sua evolução, seus componentes internos e os recursos técnicos que possibilitam a criação de artefatos digitais interativos. Em seguida, são abordadas tecnologias como IA, física, interface e organização estrutural, que desempenham papéis fundamentais no desenvolvimento dentro da *Unity*. Por fim, apresentam-se trabalhos relacionados que exploram a criação de jogos e artefatos utilizando essa *engine.*

2.1 *ENGINES* DE DESENVOLVIMENTO DIGITAL E SUAS APLICAÇÕES

2.1.1 Definição e Características de *Engines* de Jogos

Engines de jogos são plataformas integradas que fornecem os recursos necessários para criar ambientes digitais interativos, unificando renderização, física, *scripts*, gerenciamento de *assets* e ferramentas de edição. Fortuna (2000) destaca que tais plataformas buscam representar comportamentos e sistemas do mundo real, equilibrando precisão técnica e facilidade de uso. No contexto da *Unity 3D*, essa abordagem permite ao desenvolvedor criar desde simulações complexas até aplicações educacionais, manipulando elementos lógicos e visuais dentro de uma mesma estrutura.

A utilização de recursos como física, IA e interfaces configura uma base sólida para a criação de artefatos e jogos completos, permitindo que usuários de diferentes perfis possam compreender e manipular elementos técnicos por meio de ferramentas acessíveis e amplamente documentadas (Gautron et al., 2022).

2.1.2 Evolução das *Engines* e o Papel da *Unity 3D*

Ao longo dos anos 2000, o crescimento dos jogos digitais impulsionou significativamente o desenvolvimento de *engines* mais robustas e acessíveis. Ferramentas como *Unity*, *Unreal Engine* e *Godot* possibilitaram a criação de títulos de diferentes gêneros, desde simulações sociais (*The Sims*), cidades (*SimCity*),

condução (*Gran Turismo, Forza Motorsport*) até simulações profissionais como *Euro Truck Simulator* e *Farming Simulator*.

A evolução constante de técnicas de renderização, sistemas físicos e ferramentas de criação contribuiu para ampliar as possibilidades de desenvolvimento, permitindo que *engines* como a *Unity* fossem adotadas tanto por iniciantes quanto por estúdios profissionais (Hax; Ferreira Filho, 2015). A física avançada e o uso crescente de IA influenciam diretamente na construção de experiências mais estruturadas, responsivas e consistentes (Wu et al., 2022).

2.2 ASPECTOS TÉCNICOS DA *UNITY 3D*

2.2.1 Inteligência Artificial e Comportamento de Objetos

A *Unity 3D* oferece ferramentas que permitem modelar comportamentos automatizados para personagens e objetos, utilizando *scripts* em *C#* e componentes próprios da *engine*. A IA aplicada nesses contextos viabiliza a criação de elementos que reagem dinamicamente ao cenário, ao jogador ou a outros objetos, conforme destacam Wu et al. (2022).

Essas funcionalidades permitem simular rotinas, navegação por terrenos, detecção de eventos e execução de ações pré-programadas, compondo sistemas que podem ser utilizados em jogos, aplicações educacionais ou simuladores (Gautron et al., 2022).

2.2.2 Física nos Ambientes Digitais

A física desempenha um papel essencial dentro da *Unity*, possibilitando que corpos rígidos, colisores, juntas e forças representem comportamentos coerentes dentro do ambiente tridimensional. Sistemas como *Rigidbodies*, *Colliders* e *Joints* permitem que objetos reajam a gravidade, colisões e movimentos de forma natural, aproximando a experiência do comportamento esperado no mundo real (Wu et al., 2022).

Para aplicações que demandam movimentação, veículos, animações ou interações ambientais, a física torna-se uma ferramenta fundamental, sendo

amplamente utilizada para validar artefatos, como observado em *engines* empregadas em jogos de direção e simulação (Hax; Ferreira Filho, 2015).

2.2.3 Interface e Usabilidade no Desenvolvimento com *Unity*

A interface do editor *Unity* organiza recursos visuais e lógicos de forma a facilitar o fluxo de trabalho do desenvolvedor. Ferramentas como o *Inspector*, *Hierarchy*, *Project* e *Console* estruturam o processo de criação, permitindo rápida navegação entre objetos, componentes e mensagens de execução. Segundo Falzon (2004), a organização e ergonomia são essenciais para evitar sobrecarga cognitiva e permitir que sistemas complexos sejam manipulados de maneira clara e objetiva.

Além disso, a construção de interfaces dentro da própria aplicação como menus, *HUDs* e interações segue princípios semelhantes, equilibrando clareza, acessibilidade e consistência para garantir que usuários finais compreendam e utilizem adequadamente os artefatos produzidos.

2.3 *UNITY 3D* COMO FERRAMENTA EDUCACIONAL E DE TREINAMENTO

O uso da *Unity 3D* vai além do entretenimento, sendo aplicada em ambientes educacionais e de treinamento profissional, refletindo o que ocorre com simuladores digitais utilizados em áreas como agricultura, saúde e engenharia. A possibilidade de criar cenários interativos e controlados permite que estudantes testem conceitos, explorem ferramentas e compreendam estruturas sem a necessidade de ambientes físicos reais (Dot Digital Group, 2019).

Assim como ocorre em simuladores reais, artefatos criados na Unity possibilitam experimentação segura, repetição de processos e construção prática de conhecimento por meio de modelos digitais. Essa capacidade torna a *engine* uma alternativa acessível para instituições e pesquisadores.

2.4 TRABALHOS RELACIONADOS

Esta seção apresenta trabalhos acadêmicos associados ao desenvolvimento de jogos digitais e ao uso da *Unity 3D* no ensino e na aplicação prática de técnicas de engenharia de software (ES).

2.4.1 Desenvolvendo Jogos Utilizando a Engenharia de *Software* e a *Unity 3D*

O trabalho de Marques (2024) apresentou o desenvolvimento de um jogo de *videogame* aplicando conhecimentos da Engenharia de *Software* e utilizando a *engine Unity 3D. A pesqui*sa realizada consistiu em uma revisão bibliográfica sobre os conceitos de ES e desenvolvimento de jogos, além da criação prática de um artefato de jogo digital. Durante o processo, foram aplicadas práticas como documentação de requisitos, modularização do sistema, versionamento com *GitHub* e desenvolvimento incremental-iterativo, evidenciando a aplicação de técnicas de ES em um ambiente de produção de jogos.

A conclusão do trabalho destacou que o uso da Engenharia de *Software*, mesmo de forma informal, foi essencial para orientar o desenvolvimento do jogo, proporcionando um panorama mais claro das soluções, tarefas e formas de implementação. A abordagem baseada em ES facilitou o planejamento e a execução incremental das funcionalidades, favorecendo resultados satisfatórios tanto na organização do projeto quanto na qualidade do produto.

2.4.2 Desenvolvendo um Jogo para Ensinar Física com *Unity 3D*

O trabalho de Torres (2016) propôs o desenvolvimento de um jogo digital com o objetivo de auxiliar no ensino da disciplina de Física. Utilizando a *engine Unity 3D e ferrame*ntas de modelagem 3D da *Autodesk*, o projeto integrou técnicas de computação gráfica e *design* de jogos para criar um ambiente interativo e educativo. A metodologia envolveu a aplicação de teorias de *design* de *games* e computação gráfica, resultando em um artefato funcional.

A conclusão indicou que jogos digitais podem ser ferramentas eficazes no processo de ensino-aprendizagem, proporcionando uma experiência lúdica que facilita a compreensão de conceitos complexos. O trabalho também sugeriu a

ampliação do roteiro e a realização de testes em ambientes educacionais para validar a eficácia do jogo desenvolvido.

2.4.3 Projetando Mecânicas de Jogos com Base em uma Abordagem Iterativa

Matos (2020) desenvolveu um estudo focado na aplicação de uma metodologia iterativa no desenvolvimento de mecânicas de jogos, utilizando a estrutura conceitual do modelo Mecânica, Dinâmica e Estética (MDA). O objetivo foi demonstrar, de forma prática, como a aplicação de ciclos iterativos pode influenciar positivamente o processo de *design* de jogos.

A pesquisa evidenciou que a utilização de abordagens iterativas permite uma maior flexibilidade e adaptabilidade no desenvolvimento de jogos, facilitando ajustes e melhorias contínuas. A conclusão ressaltou a importância de metodologias bem definidas para o sucesso no desenvolvimento de jogos digitais, especialmente em ambientes acadêmicos e profissionais.

2.4.4 Da Teoria à Prática em Desenvolvimento de Jogos Digitais

O estudo de Andrade (2016) investigou os modelos de processo utilizados no desenvolvimento de jogos digitais no mercado paraibano. A pesquisa combinou uma revisão bibliográfica com a aplicação de questionários a empresas locais, visando identificar as metodologias adotadas e as competências requeridas dos profissionais da área.

Os resultados destacaram a diversidade de abordagens utilizadas pelas empresas e a necessidade de alinhamento entre a formação acadêmica e as demandas do mercado. A conclusão sugeriu a inclusão de conteúdos práticos nos currículos acadêmicos para preparar melhor os estudantes para o mercado de desenvolvimento de jogos digitais.

3. MÉTODO

Esta pesquisa, segundo sua natureza é um resumo de assunto, buscando a compreensão acerca do tema e a partir da investigação e observação das informações contiguas compreender suas causas e explicações (Wazlawick, 2014).

Segundo seus objetivos é de caráter exploratório, segundo Gil (2021): “As pesquisas exploratórias têm como propósito proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito ou a construir hipóteses” (Gil, 2021, p. 27).

Segundo seus procedimentos técnicos, esta pesquisa é bibliográfica e experimental. De acordo com Gil (2021) a pesquisa bibliográfica é elaborada com base em material já publicado, inclui materiais impressos, como livros, revistas, jornais, teses, dissertações e anais de eventos científicos, bem como mídias digitais.

Toda pesquisa científica e considerada bibliográfica em sua concepção devido a necessidade de levantamento teórico para fundamentação do trabalho científico. (GIL, 2021)

Conforme Gil (2021) pesquisas bibliográficas seguem as seguintes etapas:

a) **Escolha do tema:** Desenvolvimento de *games.*
b) **Levantamento bibliográfico preliminar:** Levantamento bibliográfico de acordo com área de interesse do pesquisador com intuito de familiarização com o tema e facilitação na definição do problema de pesquisa.
c) **Formulação do problema:** Como funciona a *Unity 3D* e como ela gera artefatos digitais?
d) **Busca das fontes:** Fontes em artigos científicos disponíveis na plataforma Capes e *Google Academy,* Livros sobre o tema obtidos em acervo pessoal. Tais fontes propiciaram a formulação do problema de pesquisa e a elucidação dos processos envolvidos acerca do tema.
e) **Leitura do material:** Identificação de pontos chaves correlacionadas ao tema de pesquisa.
f) **Fichamento:** Fichamento de citação produzido a partir do material bibliográfico selecionado afim guardar citações importantes acerca do tema.
g) **Organização lógica:** Foram organizadas as ideias com o propósito de realizar os objetivos da pesquisa.
h) **Redação do texto:** Escrita do TCC1.

Pesquisas experimentais consistem em determinar o objeto de estudo, selecionar variáveis capazes de influenciá-lo e definir as formas de controle e de observação dos efeitos que a variável produz no objeto de estudo. (Gil, 2021). Logo, de acordo com Gil (2021) esta pesquisa experimental apresenta as seguintes etapas:

a) **Formulação do problema:** Como funciona a *Unity 3D* e como ela gera artefatos digitais?
b) **Construção das hipóteses:** Através da *Unity 3D* é possível compreender o funcionamento de seus principais recursos e demonstrar, por meio da criação de um artefato, como a *engine* organiza componentes, processa informações e gera artefatos digitais.
c) **Definição do plano experimental:** O plano experimental consistiu em desenvolver um artefato simples com o objetivo de demonstrar, na prática, alguns recursos fundamentais da *Unity 3D.* Para isso, foram implementados e testados componentes como sistemas físicos (*Wheel Colliders* e *Joints*), estruturas de terreno, *scripts* em *C#*, gerenciamento básico de cenas e elementos visuais. Cada implementação serviu como demonstração do funcionamento da *engine*, permitindo observar sua estrutura de componentes, seu processamento e a geração de artefatos digitais.
d) **Determinação do ambiente:** O ambiente utilizado foi virtual. Utilizando um computador ***Desktop:***
    a. *Intel i7 13700f (CPU).*
    b. *Kingston Fury Beast RGB 32GB (2x16GB) (RAM).*
    c. *Asus TUF B660m D4* (*Motherboard*).
    d. *Nvidia GTX 970 4GB Gigabyte G1 Gaming (GPU).*
    e. *Windows 11 Pro* (Sistema Operacional).
    f. *Unity3D 2022.3.4*9f1 (*Game Engine*).
    g. *GitHub* (Controle de Versão).
e) **Coleta de dados:** A coleta de dados ocorreu durante o desenvolvimento e teste do artefato, registrando o comportamento dos componentes da *Unity* utilizados, como sistemas físicos, organização de objetos, execução de scripts e manipulação de terreno. Essas observações foram realizadas diretamente no editor e durante a execução da aplicação, permitindo analisar como a *engine* processa e gera artefatos digitais.

f) **Análise e interpretação dos dados:** A análise consistiu em verificar se os recursos utilizados da *Unity 3D* funcionaram conforme o esperado e se permitiram demonstrar de forma clara suas capacidades técnicas. Foram avaliados aspectos como estabilidade dos componentes, execução dos *scripts*, comportamento do sistema de física e organização da *engine* no desenvolvimento de artefatos. A interpretação dos dados permitiu identificar o funcionamento efetivo de cada recurso e sua relação com a geração de artefatos digitais.
g) **Redação do relatório:** Escrita do TCC.

4. A *UNITY 3D*

Este capítulo tem como objetivo apresentar a ferramenta *Unity, abordando suas f*uncionalidades, características principais e justificativas para sua escolha no desenvolvimento de projetos digitais, dados estes retirados de *UNITY TECHNOLOGIES* (2025). A *Unity* é amplamente reconhecida como uma das principais *game engines* disponíveis no mercado, destacando-se pela versatilidade e pelo suporte a múltiplas plataformas.

A interface da ferramenta é composta por diversos painéis essenciais para o fluxo de trabalho no desenvolvimento de jogos. O ***Hierarchy***, por exemplo, permite organizar e visualizar os objetos presentes na cena, funcionando como uma árvore estrutural do projeto em tempo real. Já o ***Scene View*** oferece ao desenvolvedor uma representação visual do ambiente de jogo, possibilitando manipulações em tempo real dos elementos da cena. Complementando essa visualização, o ***Game View*** simula como o jogo será exibido ao jogador, sendo essencial para testes de jogabilidade e ajustes de câmera.

Outro componente central é o ***Inspector***, utilizado para configurar propriedades dos objetos selecionados, desde transformações espaciais até atributos específicos de componentes atrelados ao objeto. A ***Project View*** organiza os arquivos do projeto, enquanto o painel ***Console*** reporta erros, mensagens de depuração e *logs* durante a execução do projeto.

A arquitetura dos objetos da *Unity é centrada em* ***GameObjects*** e seus ***Components***. Cada objeto da cena é uma instância de *GameObject*, que pode conter múltiplos componentes responsáveis por suas funcionalidades, como renderização, física ou *scripts*. Componentes como *Transform* são obrigatórios e armazenam as posições e rotações dos objetos. Outros componentes comuns incluem *Rigidbody*, que aplica física e gravidade, *BoxCollider*, *SphereCollider*, *CapsuleCollider*, e *MeshCollider*, que definem colisões em diferentes formatos.

Além disso, a *Unity* fornece renderizadores como *MeshRenderer* e *SkinnedMeshRenderer* para exibir objetos tridimensionais, bem como ferramentas especializadas para manipulação de terrenos (*Terrain*) e iluminação (*Lighting*). Para interações mecânicas entre objetos, a *engine* oferece sistemas de juntas físicas como *HingeJoint*, *FixedJoint* e *ConfigurableJoint*.

Do ponto de vista da programação, o principal elemento é o *MonoBehaviour*, classe base para a criação de *scripts* em *C#* que controlam comportamentos dos objetos, eventos de entrada, física, animação, entre outros aspectos do jogo. O modelo de *script* é orientado a eventos do ciclo de vida do jogo, como *Start()*, *Update()* e *FixedUpdate()*.

Por fim, a *Unity também oferece d*iretrizes para organização eficiente de projetos, como estruturação de pastas, controle de dependências e práticas recomendadas para colaboração em equipe.

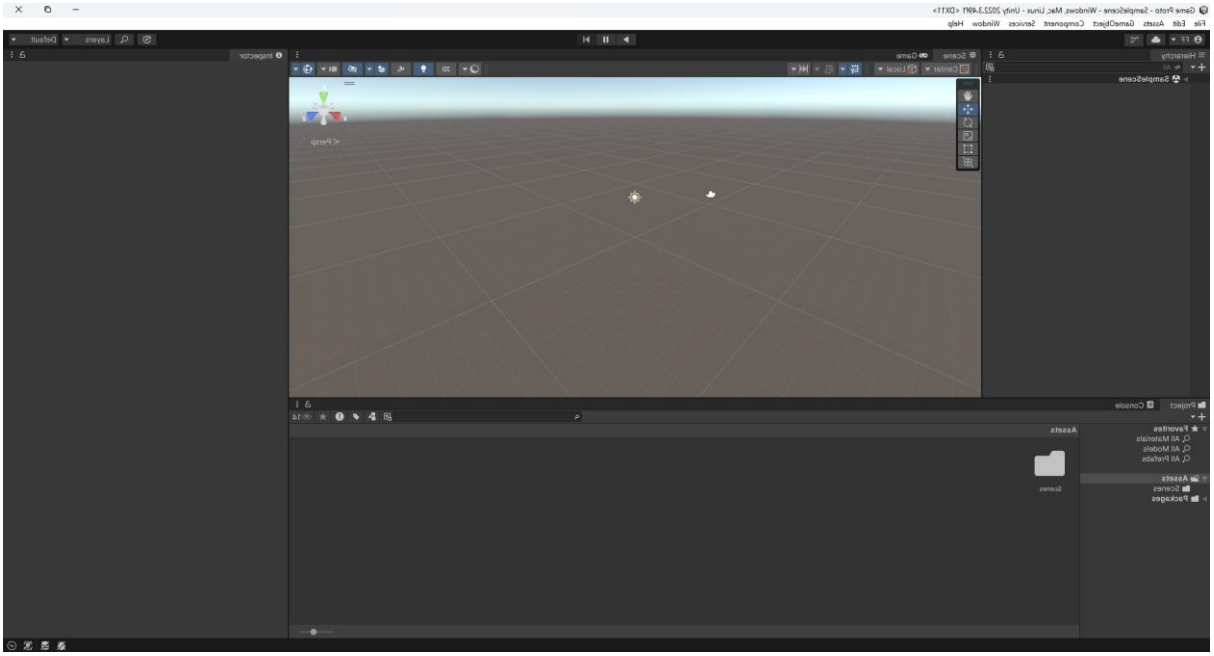
Essa gama de recursos justifica a escolha da *Unity* como ferramenta principal para o desenvolvimento deste projeto, por sua robustez, documentação completa e ampla comunidade de suporte.

4.1 CONHECENDO O EDITOR

Como já mencionado anteriormente, a *Unity é uma das plataf*ormas mais reconhecidas e robustas entre os motores de jogos disponíveis no mercado. Ao abrir o editor pela primeira vez, percebe-se que sua interface é composta por múltiplas janelas, cada uma com funções específicas que facilitam o processo de desenvolvimento.

Na Figura 1, é possível observar o layout padrão da interface do editor, que inclui janelas fundamentais como ***Hierarchy***, ***Scene***, ***Game***, ***Inspector***, ***Project*** e ***Console***. A seguir, são detalhadas as funcionalidades dessas janelas e sua relevância para o projeto.

Figura 1 - Interface padrão do editor



Fonte: elaborado pelo autor.

4.1.1 A janela *Hierarchy*

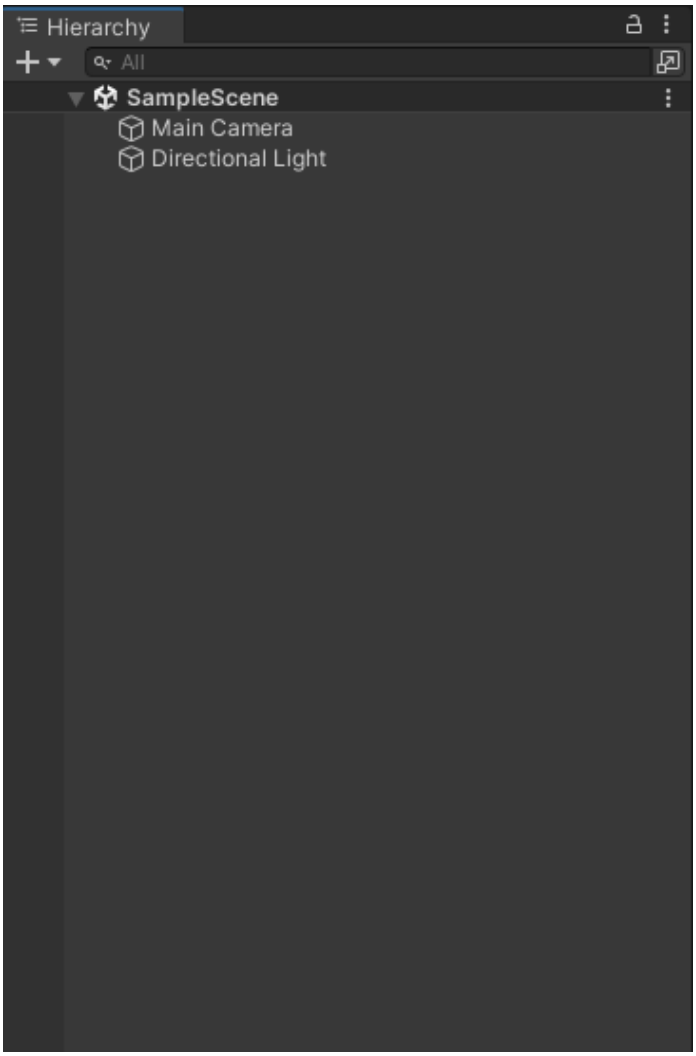
A janela *Hierarchy* exibe todos os *GameObjects* contidos em uma cena, como modelos 3D, câmeras ou *prefabs*. Ela permite a organização estrutural da cena, possibilitando a classificação e o agrupamento de objetos. Quando um *GameObject* é adicionado ou removido da cena, essa modificação também é refletida automaticamente na *Hierarchy*, e vice-versa.

A estrutura de parentalidade é um dos recursos essenciais dessa janela, permitindo a criação de hierarquias entre objetos. Um objeto pai pode conter múltiplos filhos que herdam transformações como posição e rotação. Na Figura 2, observa-se, por exemplo, que *Main Camera* e *Directional Light* são filhos do objeto *SampleScene*.

Outras funcionalidades importantes incluem:

- **Criação de *GameObjects***: clicando com o botão direito em uma área livre da janela, ou utilizando o atalho ***Ctrl+Shift+N*** *(Windows)* / ***Command+Shift+N*** *(macOS)*.
- **Criação de objetos filhos**: arrastando um objeto para dentro de outro, ou clicando com o botão direito sobre o objeto pai e selecionando a opção de criação.

Figura 2 - Janela *Hierarchy*



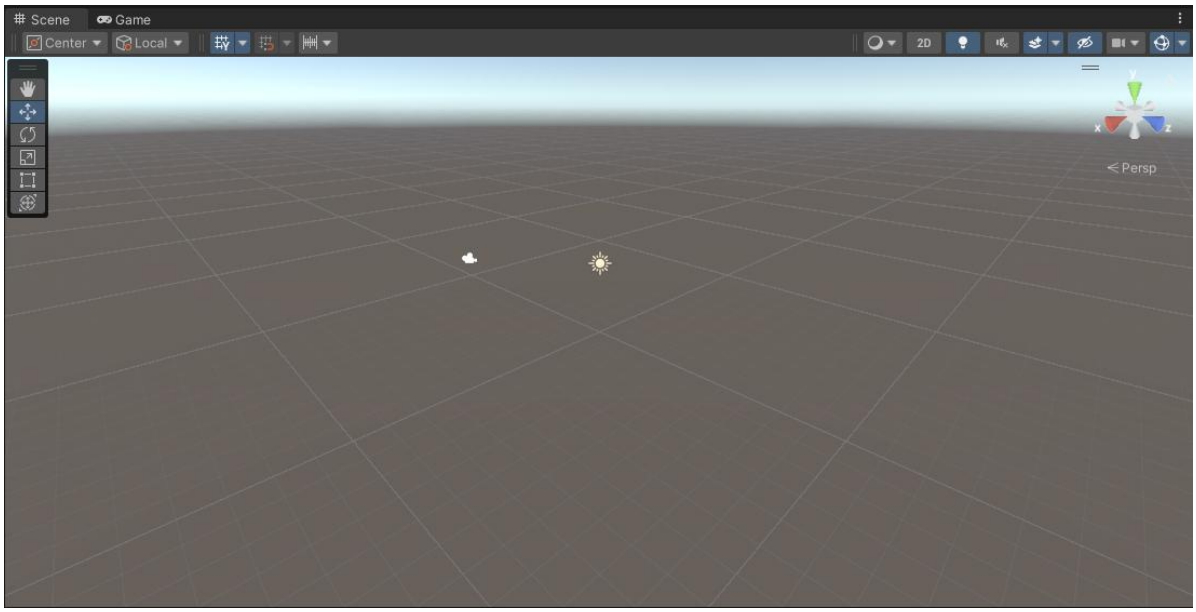
Fonte: elaborado pelo autor.

4.1.2 A janela *Scene*

A janela *Scene* exibe visualmente o mundo em construção, permitindo interações diretas com os *GameObjects* posicionados na cena. É possível selecionar, movimentar e ajustar elementos como personagens, câmeras, luzes e objetos do ambiente.

Como mostrado na Figura 3, essa visualização é essencial para organizar espacialmente os elementos do projeto e realizar ajustes precisos durante o desenvolvimento.

Figura 3 - A janela *Scene*



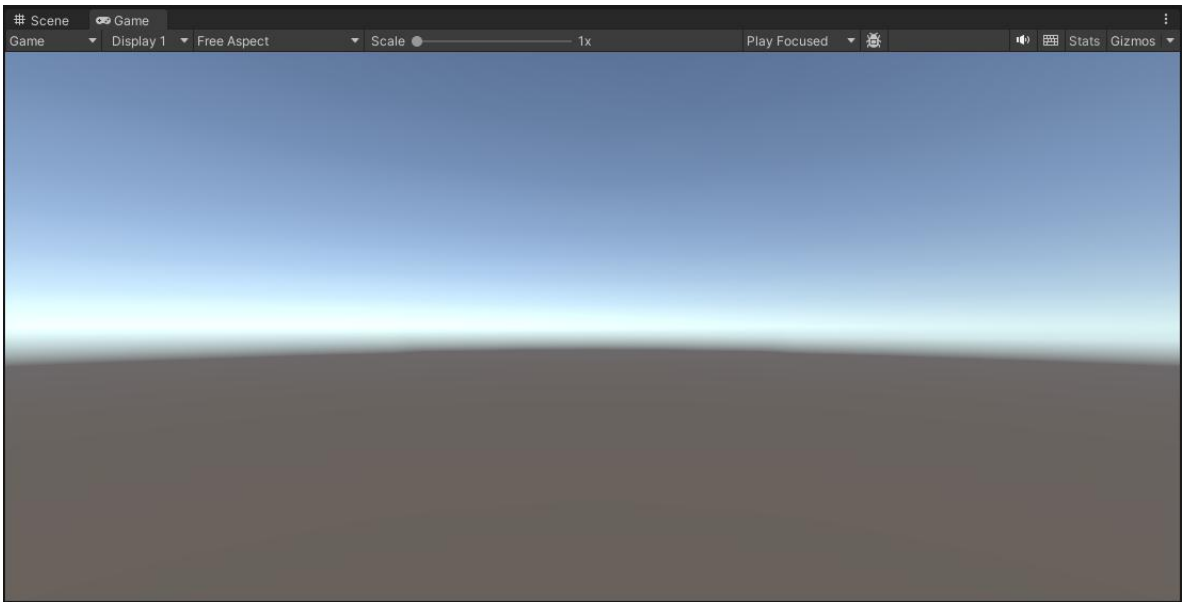
Fonte: elaborado pelo autor.

4.1.3 A janela *Game*

A janela *Game* representa a visualização final do jogo, sendo renderizada a partir das câmeras posicionadas na cena. É por meio dela que o desenvolvedor compreende o que será efetivamente exibido ao jogador.

Conforme se vê na Figura 4, essa janela inclui uma barra de controle com funções como ajuste de escala, visualização de quadros por segundo, estatísticas de desempenho e controle de áudio. É também por meio dessa janela que se realizam testes interativos durante o desenvolvimento.

Figura 4 - A janela *Game*



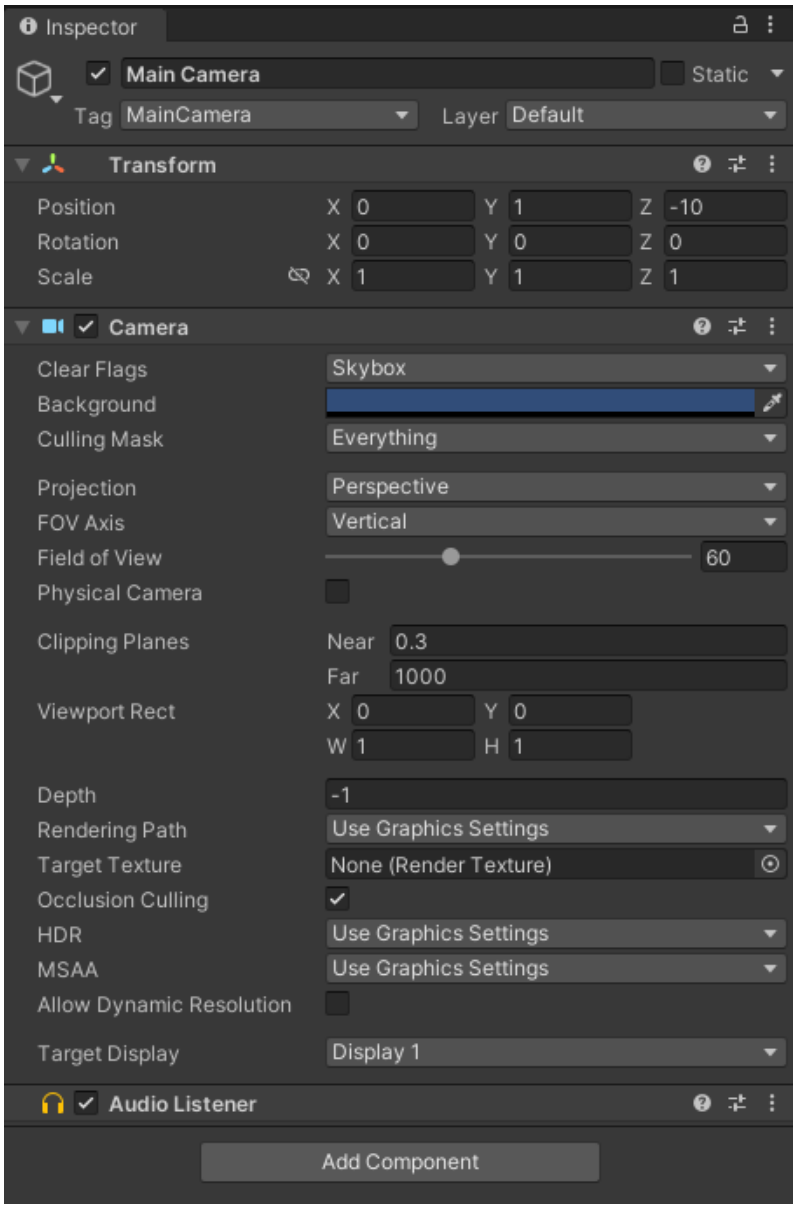
Fonte: elaborado pelo autor.

4.1.4 A janela *Inspector*

O *Inspector* permite visualizar e modificar as propriedades de qualquer elemento selecionado no projeto — sejam *GameObjects*, componentes, materiais ou configurações do próprio editor.

Ao selecionar um objeto na cena ou na *Hierarchy*, suas propriedades são automaticamente exibidas no *Inspector*, como ilustrado na Figura 5. Essa janela é fundamental para a edição de atributos e aplicação de comportamentos específicos via componentes e *scripts*.

Figura 5 - A janela *Inspector*



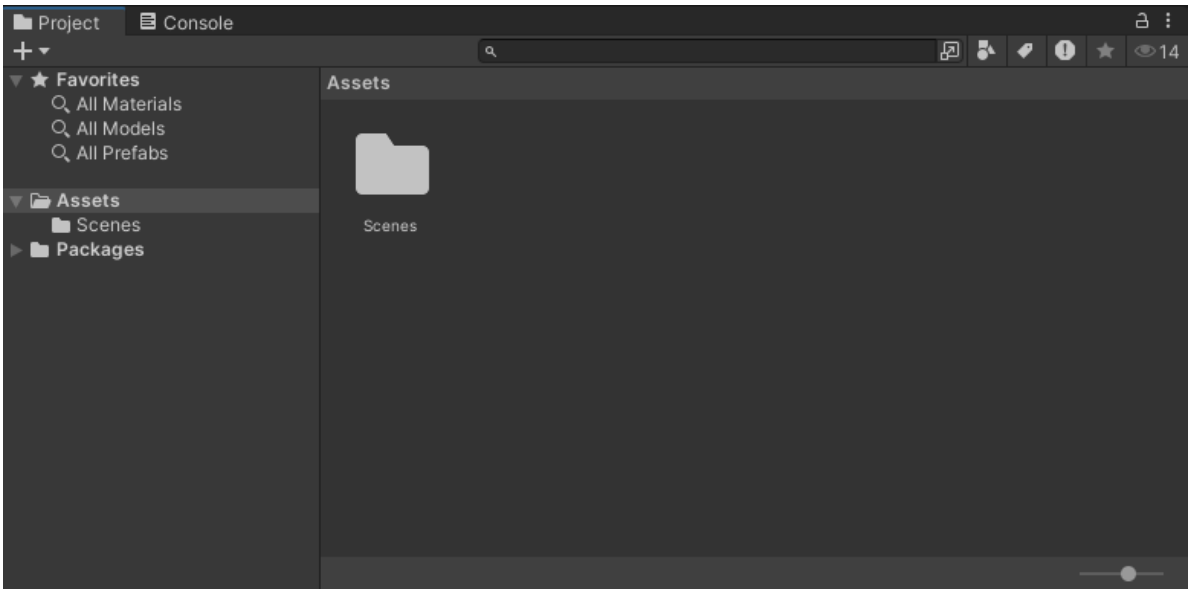
Fonte: elaborado pelo autor.

4.1.5 A janela *Project*

A janela *Project* exibe todos os arquivos e pastas que compõem o projeto. Sua estrutura hierárquica está dividida em dois painéis: o da esquerda apresenta a organização das pastas, enquanto o da direita exibe os conteúdos da pasta selecionada, geralmente em forma de ícones.

Como mostrado na Figura 6, essa janela facilita o gerenciamento de ativos (*assets*), como texturas, sons, modelos, animações e *scripts* utilizados na aplicação.

Figura 6 - A janela *Project*



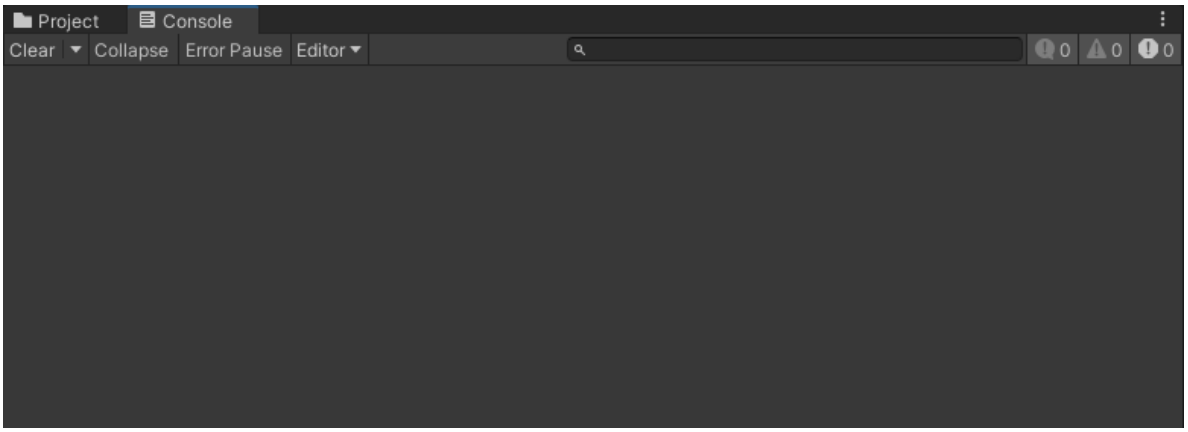
Fonte: elaborado pelo autor.

4.1.6 A janela *Console*

A janela *Console* desempenha um papel essencial no processo de depuração do projeto. Nela são exibidas mensagens, erros e avisos gerados durante a execução, com destaque especial para os scripts.

A Figura 7 mostra como essa janela apresenta essas mensagens, sendo um recurso indispensável para identificação de falhas e ajustes no comportamento da aplicação.

Figura 7 - A janela *Console*



Fonte: elaborado pelo autor.

4.2 ARQUITETURA DA *UNITY*

A arquitetura *org*anizacional de um projeto desenvolvido na *Unity* refere-se à forma como seus componentes, *assets*, cenas, *scripts* e outros recursos são estruturados e inter-relacionados dentro do ambiente de desenvolvimento. Uma arquitetura bem definida é essencial para manter o projeto modular, escalável e fácil de manter à medida que sua complexidade aumenta.

Estabelecer padrões claros desde o início garante que todos os membros da equipe possam localizar e modificar elementos do jogo com facilidade, evitando retrabalho e problemas estruturais difíceis de corrigir no futuro. Mesmo que a *Unity* não imponha uma *e*strutura de pastas fixa, as melhores práticas destacam a importância de definir e seguir uma convenção clara e documentada.

Essa padronização inclui a criação de uma estrutura de diretórios bem definida, com nomes descritivos e organização lógica, promovendo escalabilidade (ao facilitar a adição de novos recursos) e manutenibilidade (ao simplificar a localização de arquivos).

4.2.1 Estrutura do Projeto

Esta subseção apresenta diretrizes práticas para estruturar projetos na *Unity* de forma organizada, especialmente em jogos 3D, que envolvem uma grande variedade de recursos, como modelos, áudios e *scripts*.

A recomendação principal é manter uma hierarquia lógica dentro da pasta *Assets*/, separando os arquivos por função. Em projetos *template* oficiais da *Unity,* essa estrutura geralmente inclui diretórios como:

- ***Art***: armazena os recursos gráficos, podendo ser subdividido em *Materials*, Models e Textures.
- ***Audio***: contém subpastas como *Music* (trilhas) e *Sound* (efeitos sonoros).
- ***Code***: engloba *Scripts*, *Shaders* e outros elementos de código. Pode-se subdividir ainda por módulos como *AI*, *UI*, ou *Gameplay*.
- ***Scenes* ou *Levels***: abriga os arquivos *.unity*, podendo incluir subpastas como *Prefabs* e *UI* associadas às cenas.
- ***UI***: alternativa à subpasta anterior, concentra os elementos de interface em um diretório dedicado.

- ***Plugins* ou *ThirdParty***: guarda *assets* externos importados da *Asset Store* ou de fontes de terceiros, mantendo-os separados do restante do projeto.
- ***Docs***: usado para documentação técnica, arquivos de *design*, rascunhos e outros materiais auxiliares, conforme sugerido pela *Unity.*

A aplicação dessa estrutura modular facilita que desenvolvedores e artistas localizem rapidamente os ativos necessários, tornando o projeto mais eficiente.

Quanto à nomenclatura, a *Unity* recomenda evitar espaços e usar notações como *CamelCase* ou *snake_case*. O uso de sufixos e prefixos descritivos, como *_Mat* para materiais ou *UI_* para *scripts* de interface, também é incentivado. Essa padronização reduz ambiguidades e torna os arquivos mais auto descritivos — por exemplo, evita nomes genéricos como “SemNome (1)” ou “*SpriteFinalFinal*”.

Em projetos colaborativos, a organização é ainda mais importante para evitar conflitos e perda de dados. Sistemas de controle de versão como *Git*, *Perforce* ou *PlasticSCM* devem ser configurados para versionar também os arquivos *.meta* gerados pela *Unity*, uma vez que eles contêm dados essenciais para o gerenciamento de referências entre objetos.

É recomendável ainda ativar a opção de “Exibir arquivos meta” nas configurações do editor e forçar a serialização de *assets* em texto (*YAML*) ao invés de binário. Isso possibilita a inspeção e mesclagem de cenas e *prefabs* usando ferramentas como o ***UnityYAMLMerge***, que permite a resolução de conflitos de forma mais precisa.

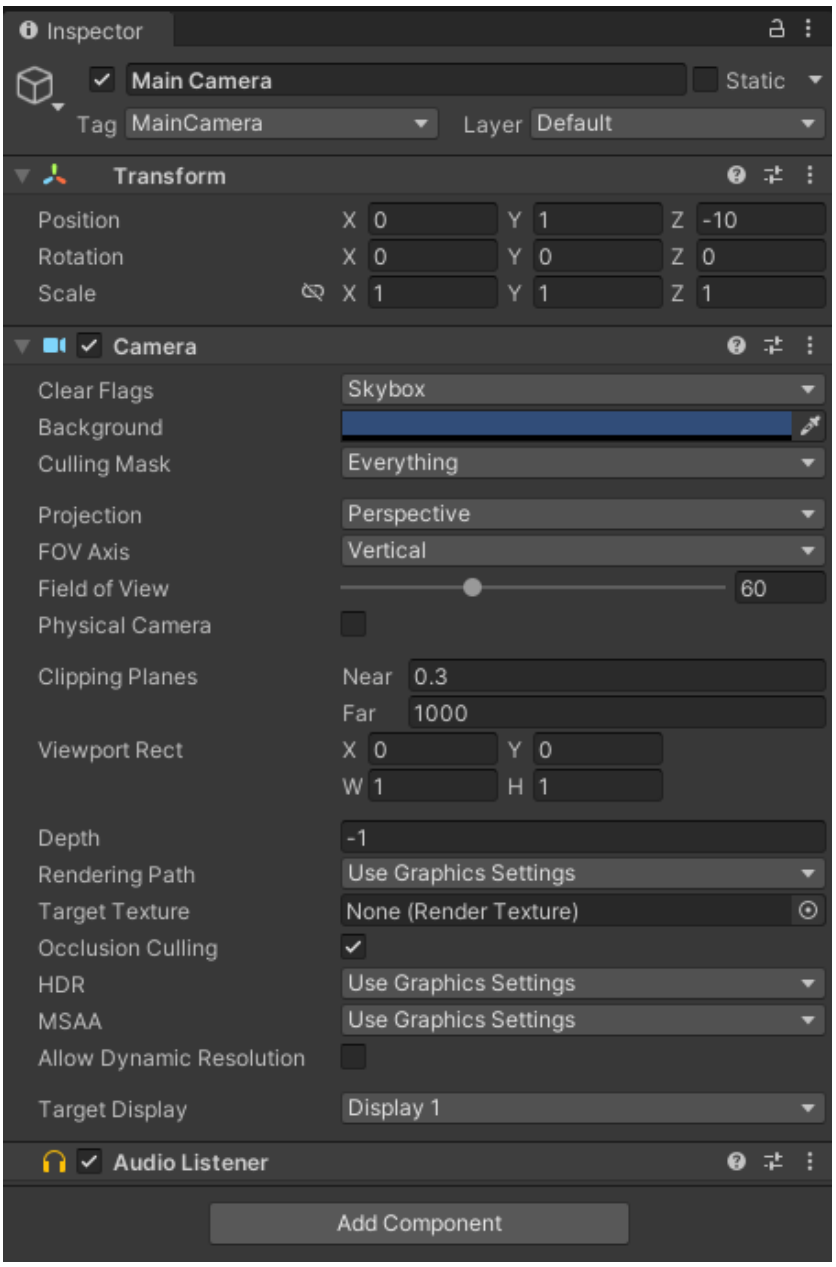
4.2.2 Sistema de Componentes e Objetos

Na *Unity,* cada *GameObject* atua como um contêiner vazio ao qual se adicionam componentes que determinam seu comportamento. Entre esses componentes, o *Transform* é obrigatório, pois define a posição, rotação e escala do objeto.

4.2.2.1 *GameObject*: A unidade Fundamental

O *GameObject* é a base de toda a estrutura de uma cena. Ele representa qualquer entidade presente no jogo e pode conter componentes visuais, físicos e lógicos. Através do *Inspector*, como ilustrado na Figura 8, é possível visualizar propriedades como *Name*, *Tag*, *Layer* e o estado de ativação do objeto.

Figura 8 - *GameObject*



Fonte: elaborado pelo autor.

4.2.2.2 Componentes de Transformação e Movimentação

O *Transform* armazena dados de posição, rotação e escala. Já o *Rigidbody* permite simular física realista, como gravidade, colisões e empurrões, ao aplicar forças físicas. Para movimentos mais controlados, o *CharacterController* é utilizado, especialmente em personagens em primeira ou terceira pessoa, dispensando o uso de física de corpos rígidos.

Esses três componentes são representados na Figura 9, destacando seus usos distintos dentro do sistema de movimentação da *Unity.*

Figura 9 - Apresenta os três componentes apresentado nessa seção.

Fonte: elaborado pelo autor.

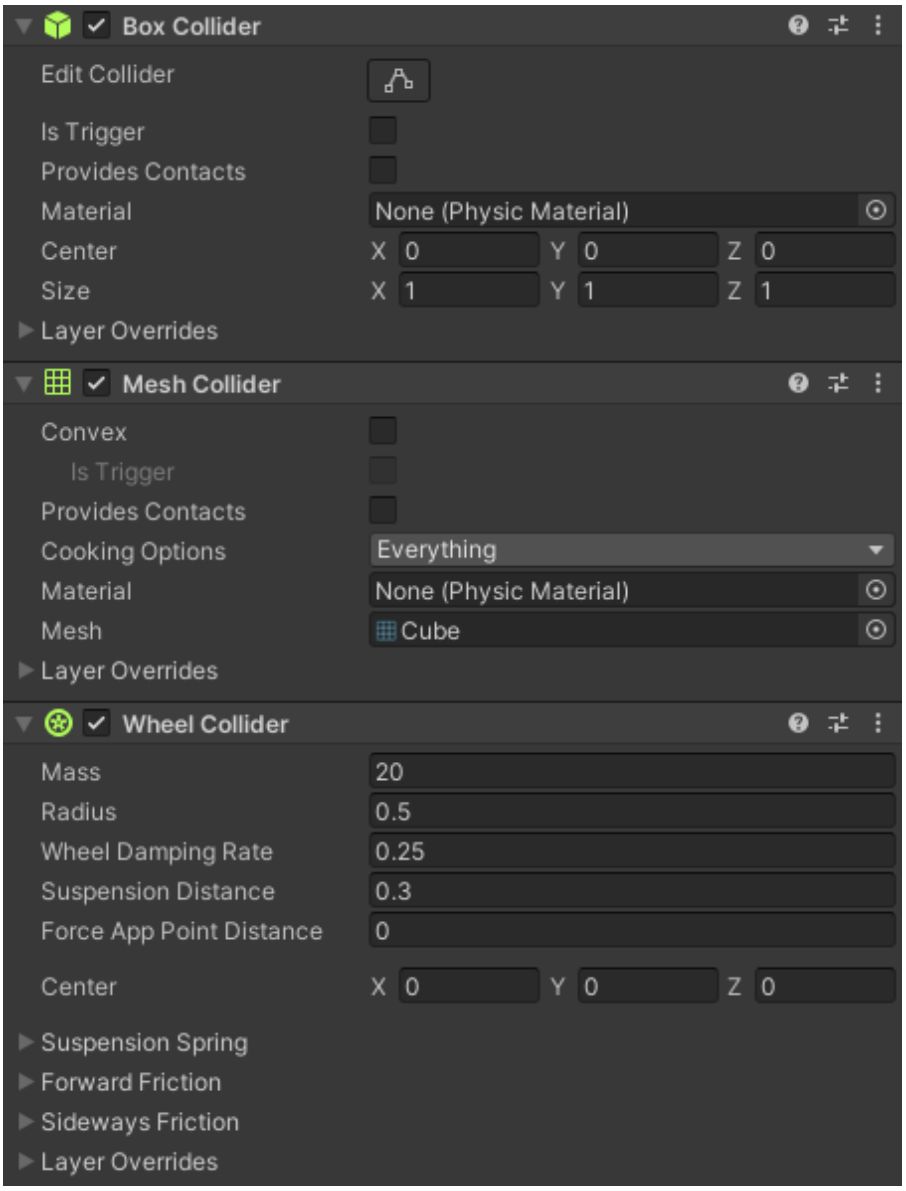
4.2.2.3 Componentes de Colisão e Interação

A *Unity oferece diferente*s tipos de colisores, cada um adequado a um cenário específico:

- *BoxCollider*: forma cúbica, simples e de baixo custo computacional;
- *SphereCollider*: ideal para objetos arredondados e rolantes;
- *CapsuleCollider*: usado frequentemente em personagens, por sua forma cilíndrica com extremidades suaves;
- *MeshCollider*: adapta-se à malha do objeto, sendo mais precisa, porém mais custosa;
- *WheelCollider*: especializado para simulação física de veículos.

A Figura 10 compara colisores primitivos e avançados, como *BoxCollider*, *MeshCollider* e *WheelCollider*.

Figura 10 - Colisores *BoxCollider, MeshCollider* e *WheelCollider*



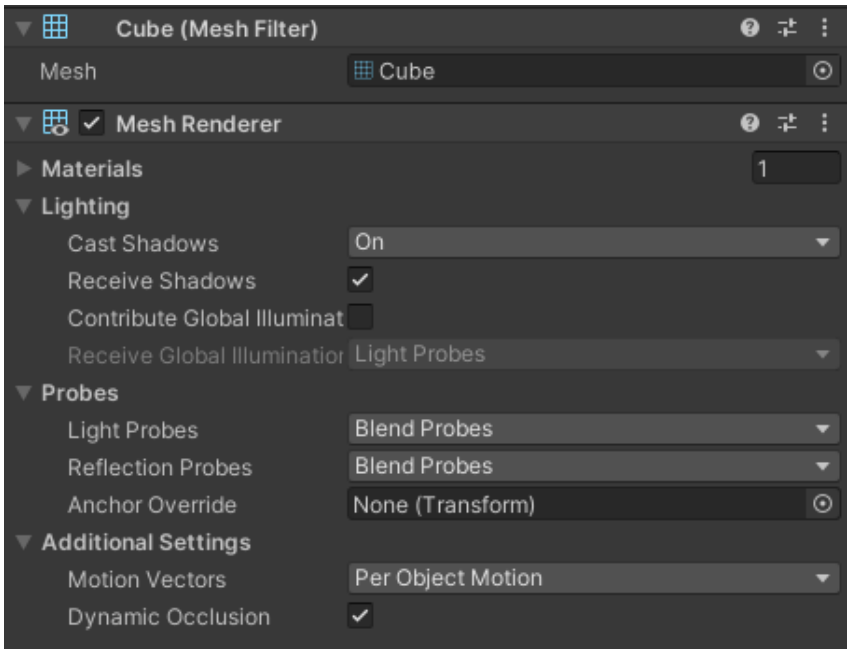
Fonte: elaborado pelo autor.

4.2.2.4 Componentes de Visualização e Renderização

O *MeshRenderer* é responsável por renderizar malhas estáticas, enquanto o *SkinnedMeshRenderer* cuida da exibição de malhas deformáveis comumente usadas em personagens animados com esqueletos ou simulação de tecidos.

Esses dois componentes são destacados na Figura 11, que mostra suas diferenças na aplicação prática.

Figura 11 - Componente *MeshRenderer*



Fonte: elaborado pelo autor.

4.2.2.5 Ambiente e Terreno

O *Terrain* é um tipo especial de *GameObject* que permite a criação de paisagens complexas com altura, textura, árvores e grama. Sua otimização automática em tempo de execução garante bom desempenho, mesmo com grandes extensões de terreno.

4.2.2.6 Componentes de Física Avançada

As *Joints* são utilizadas para conectar dois corpos rígidos, simulando articulações. A *Unity oferece várias op*ções:

- *HingeJoint*: simula dobradiças, como portas;
- *FixedJoint*: une dois objetos de forma fixa, sem parentalidade;
- *ConfigurableJoint*: altamente flexível, podendo simular desde molas até mecanismos complexos.

4.2.2.7 Componentes de Luz e Sombra

A propriedade *Type* dos componentes de luz define o comportamento da iluminação:

- ***Point Light***: emite luz igualmente em todas as direções a partir de um ponto;
- ***Spot Light***: emite luz em forma de cone;
- ***Directional Light***: simula luz solar, vinda de um ponto infinitamente distante;
- ***Area Light***: define uma área emissora de luz, útil para efeitos realistas em superfícies.

4.3 *SCRIPTING* NA *UNITY*

A Unity permite que desenvolvedores expandam e personalizem funcionalidades além do que é oferecido na interface do editor por meio de *scripting*. Esse recurso é fundamental para o controle do comportamento dinâmico de objetos, lógica de jogo, interações do usuário e diversos outros aspectos.

4.3.1 Linguagem de Programação e *API*

A linguagem utilizada para programação na *Unity é o* ***C#*** *(C-Sharp),* uma linguagem orientada a objetos amplamente reconhecida por sua robustez e expressividade. A *Unity disponibiliza uma* extensa ***API (Application Programming Interface)*** que permite aos desenvolvedores acessarem e manipular elementos do jogo por meio de código.

Essa *API* oferece classes e métodos para controle de entrada do jogador, movimentação de objetos, manipulação de áudio, controle de física, animação, acesso a componentes, gerenciamento de cenas, entre outros recursos essenciais.

Ao escrever *scripts* em *C#*, o desenvolvedor consegue criar funcionalidades altamente customizáveis e dinâmicas, que não seriam possíveis apenas com os recursos visuais do editor.

4.3.2 Estrutura de *Scripts*

Os *scripts* na *Unity são criados como* arquivos *.cs*, e cada *script* geralmente corresponde a uma classe pública que herda de *MonoBehaviour*, a classe base para todos os scripts que interagem com o ciclo de vida do Unity3D.

4.3.2.1 MonoBehaviour

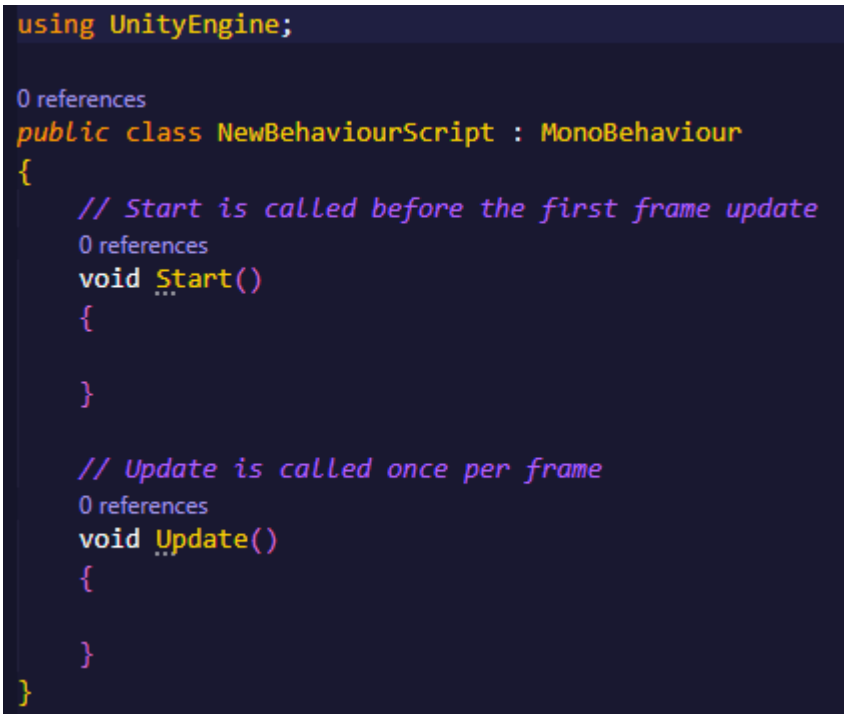
A classe *MonoBehaviour* é o ponto de entrada padrão para *scripts* comportamentais. Ao ser associada a um *GameObject*, ela permite definir comportamentos personalizados por meio de métodos especiais chamados automaticamente pelo motor de jogo.

Entre esses métodos, destacam-se:

- *Start()*: chamado uma única vez no início da execução do *script*;
- *Update()*: chamado a cada quadro, ideal para detectar entradas ou realizar atualizações contínuas.

A Figura 12 ilustra a estrutura básica de um *script MonoBehaviour* recém-criado, destacando sua sintaxe e os principais métodos utilizados no ciclo de vida de um objeto no jogo.

Figura 12 - *MonoBehaviour Script*



Fonte: elaborado pelo autor.

5. DESENVOLVIMENTO DO ARTEFATO: DEMONSTRAÇÃO PRÁTICA DOS RECURSOS DA UNITY 3D

O artefato desenvolvido tem como objetivo demonstrar, na prática, o funcionamento de alguns dos principais recursos da *Unity 3D* estudados neste trabalho. Ele atua como um **Mínimo Produto Viável (MVP) técnico**, servindo para validar o uso de sistemas da *engine*, como física, gerenciamento de terreno, renderização dinâmica e integração de múltiplos componentes.

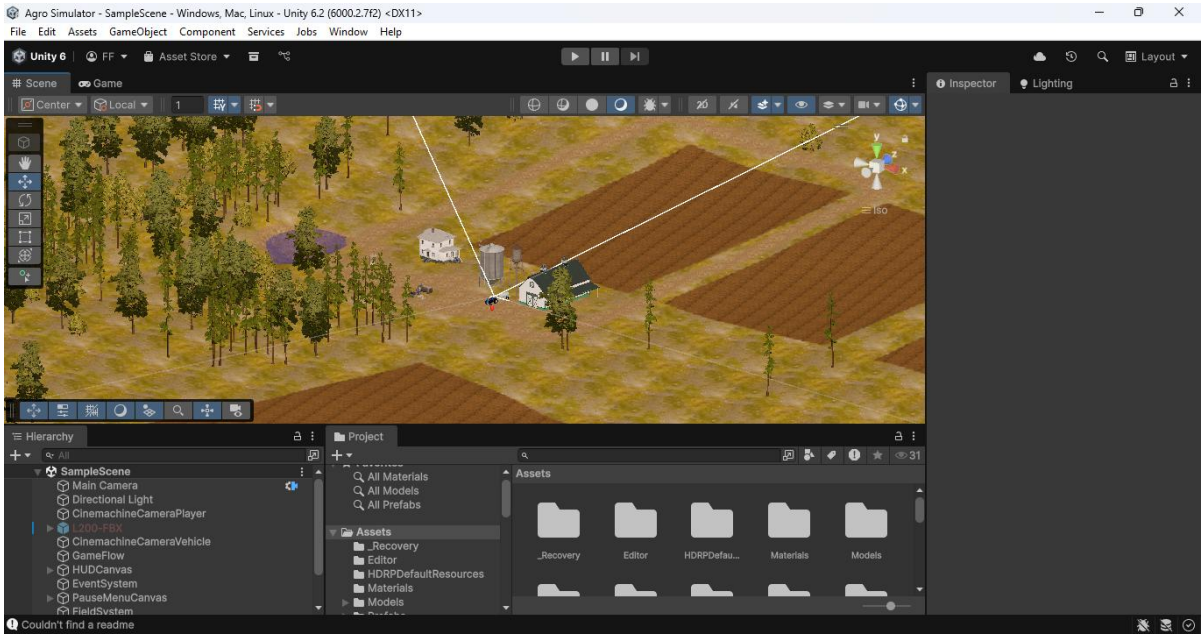
Nesse contexto, a implementação concentra-se na avaliação de recursos essenciais oferecidos pela Unity:

- Sistema de física veicular baseado em *Wheel Colliders.*
- Conexão física via *Configurable Joint.*
- Criação e edição de terrenos utilizando *Heightmaps* e ferramentas da própria *engine.*

Esses sistemas foram escolhidos por representarem áreas fundamentais da *Unity* e por permitirem analisar como a *engine* organiza objetos, processa interações e gera artefatos digitais. O artefato também inclui três cenas principais: *splash screen,* menu e cena jogável demonstrando o fluxo básico de inicialização e navegação de um produto interativo desenvolvido na *Unity.*

Embora simplificado como mostrado na Figura 13, o artefato permite observar o comportamento nativo de diversos subsistemas da *engine* e validar sua adequação para projetos interativos mais complexos.

Figura 13 - *Layout* da janela da *Unity* com o projeto aberto



Fonte: elaborado pelo autor.

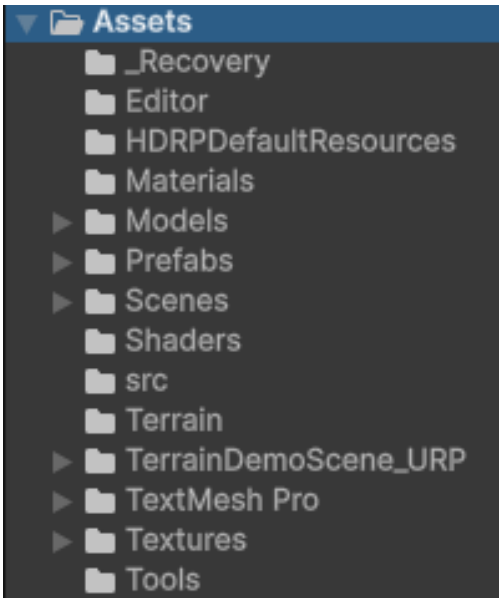
5.1 ARQUITETURA INICIAL DO PROJETO

5.1.1 Organização Geral de Diretórios

A estrutura inicial do projeto segue uma organização modular simples, adequada para experimentação. Pastas específicas foram criadas para *scripts (src)*, *prefabs*, *materials*, *shaders*, *scenes*, *models* e *textures*. Essa separação facilita localizar recursos relacionados e compreender a forma como a *Unity* gerencia *assets* internos.

A Figura 14 apresenta a organização modelo utilizada no projeto do artefato.

Figura 14 - *Screenshot* da estrutura de pastas no *Project Window*



Fonte: elaborado pelo autor.

5.1.2 Fluxo Geral de Aplicação

O fluxo do artefato segue o padrão clássico de jogos comerciais: a aplicação inicia na *splash screen*, transita para o menu principal e, a partir dele, carrega a cena jogável, conforme mostrado na Figura 15. Essa estrutura demonstra o ciclo básico de navegação e prepara terreno para a futura implementação de menus adicionais, sistemas de configurações mais complexas e gerenciamento global de estados.

Figura 15 - Diagrama simples de navegação entre cenas

Fonte: elaborado pelo autor.

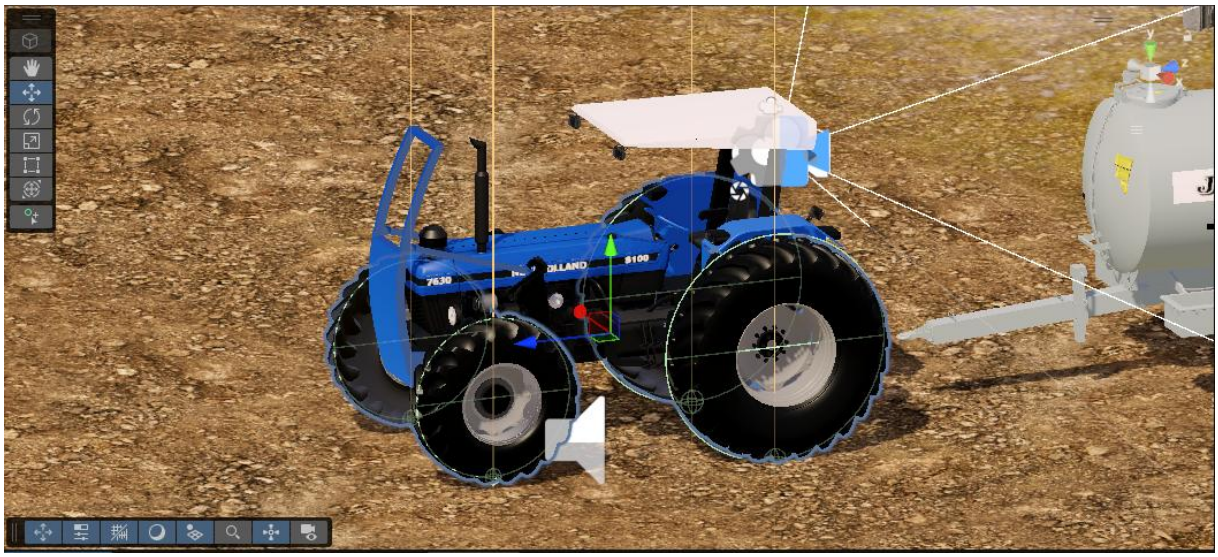
5.2 SISTEMA DE MOVIMENTAÇÃO VEICULAR

5.2.1 Escolha do *Wheel Collider*

O *Wheel Collider* foi selecionado como base da simulação por oferecer um equilíbrio entre realismo e simplicidade, além de ser a solução nativa da *Unity* para o comportamento físico de veículos. Embora não seja o sistema mais avançado para simulações altamente realistas, sua flexibilidade permite criar desde modelos *arcade* até comportamentos mais próximos da física veicular real.

A Figura 16 mostra o componente *Wheel Collider* aplicado a um veículo.

Figura 16 – Componente *Wheel Collider* aplicado a um veículo



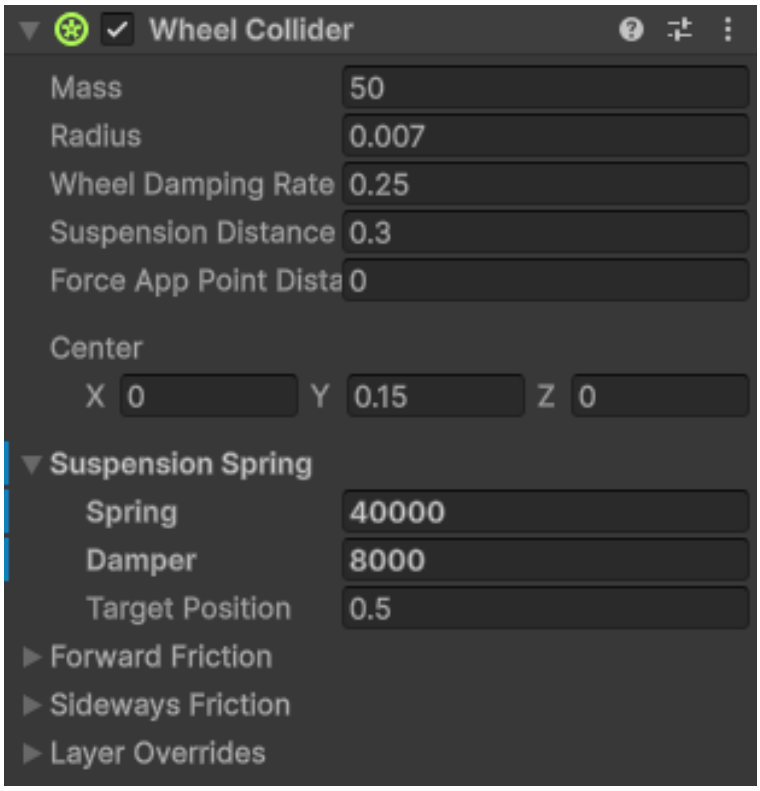
Fonte: elaborado pelo autor.

5.2.2 Configuração dos Parâmetros da Física

A configuração do *Wheel Collider* se mostrou uma das etapas mais desafiadoras do desenvolvimento, especialmente por exigir equilíbrio entre parâmetros como atrito longitudinal e lateral, rigidez e amortecimento da suspensão, massa aparente das rodas e distribuição geral de peso. Apesar de já existir um comportamento funcional, a suspensão pesada típica de tratores ainda não foi completamente validada, representando um ponto fundamental para evolução futura.

A Figura 18 apresenta os principais parâmetros do componente *Wheel Collider.*

Figura 17 - *Inspector* do *Wheel Collider* com os parâmetros usados



Fonte: elaborado pelo autor.

5.2.3 Limitações Atuais

As principais limitações identificadas incluem a necessidade de ajustes mais refinados nos parâmetros da suspensão, além da possível adoção de soluções híbridas ou substituição parcial do sistema padrão da *engine*, caso seja necessário atingir maior fidelidade física. Ainda assim, o artefato estabelece uma base funcional consistente para iterações futuras.

5.3 SISTEMA DE REBOQUE E CONEXÃO

O sistema de reboque foi implementado a partir de um mecanismo de detecção por proximidade, utilizando uma área de *trigger* que identifica quando o jogador está apto a acoplar o implemento. Essa abordagem simplifica o uso e facilita o processo de teste durante o artefato, eliminando a necessidade de alinhamento preciso entre os corpos rígidos.

5.3.1 Mecânica de Engate

Ao pressionar a tecla configurada (**Q**), o reboque é conectado ao veículo por meio de um *Configurable Joint*, que define as restrições físicas e o comportamento de movimentação entre ambos. Embora ainda não exista *feedback* visual indicando a possibilidade de acoplamento, o sistema de engate funciona de forma consistente.

A Figura 18 mostra o colisor de caixa na sua função de *trigger* para identificar a área de engate, assim como a *Configurable Joint* aplicado ao veículo.

Figura 18 – Sistema de reboque utilizando o componente *Configurable Joint*

Fonte: elaborado pelo autor.

5.3.2 Física Independente do Reboque

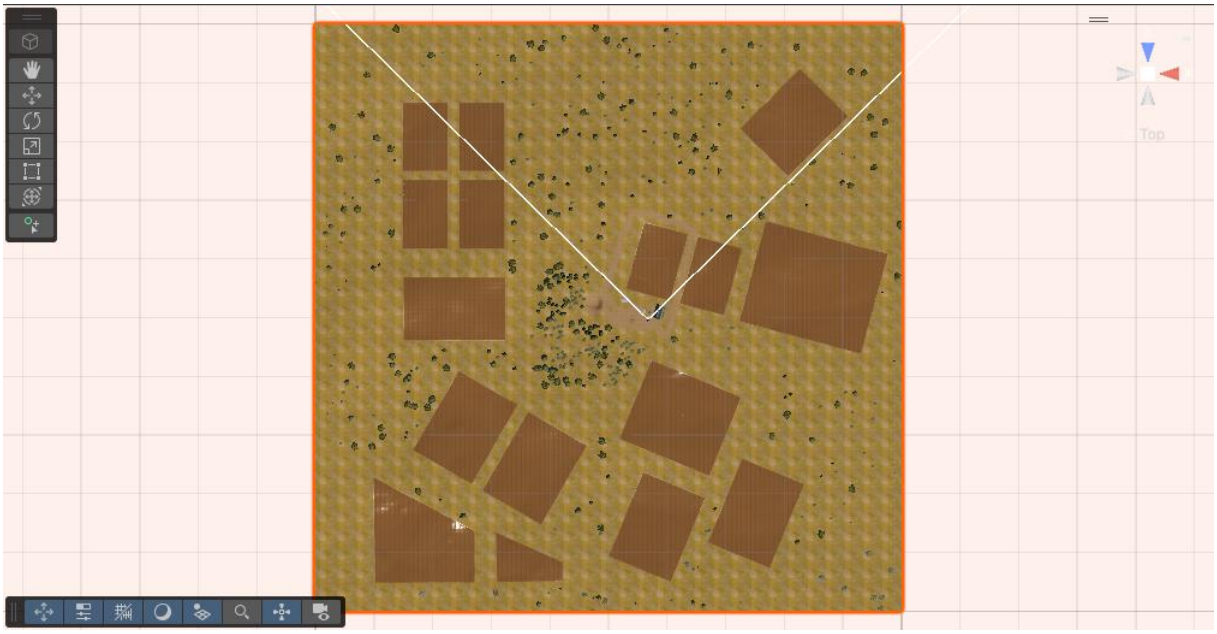
O reboque possui um *Rigidbody* próprio e utiliza *Wheel Colliders* independentes, permitindo que se comporte fisicamente de forma coerente com o veículo, inclusive em velocidades mais elevadas. A conexão mantém estabilidade satisfatória mesmo sob acelerações bruscas, reforçando a viabilidade dessa abordagem para futuras implementações mais complexas.

5.4 TERRENO E AMBIENTE DA CENA PRINCIPAL

O terreno foi criado a partir de um *heightmap* real obtido por meio do *Google Maps* e processado com a ferramenta *open-source Tangrams*. O resultado foi importado para a *Unity* e ajustado com as *Terrain Tools*, permitindo criar relevos realistas e adequados aos testes de dirigibilidade.

A Figura 19 apresenta uma visão do topo do terreno.

Figura 19 – Imagem do topo do terreno



Fonte: elaborado pelo autor.

5.4.1 Características do Mapa

A cena utiliza o tamanho padrão de terreno da *Unity* (1000 x 1000), contendo vegetação distribuída e variações naturais de relevo. O mapa fornece ambiente suficiente para testar movimentação, além de servir como base para futuras expansões.

5.4.2 Ambientação e Renderização

O ambiente inclui céu atmosférico, mas ainda não possui *post-processing* do *URP* ou sistemas sonoros. A ausência desses elementos reflete a natureza inicial do artefato, priorizando sistemas essenciais antes de refinamentos visuais.

5.5 SISTEMA DE CÂMERA E JOGABILIDADE DO PERSONAGEM

A utilização do *Cinemachine* proporcionou controle refinado e eficiente da câmera com esforço reduzido. O sistema oferece transições suaves e comportamentos automáticos úteis para jogos digitais, justificando sua escolha no artefato.

A Figura 20 apresenta a visão de primeira pessoa do *player* utilizando o componente *Cinemachine.*

Figura 20 – *Cinemachine* em ação

Fonte: elaborado pelo autor.

5.5.1 Troca entre Jogador e Veículo

A transição entre caminhar e dirigir é instantânea, realizada por meio da ativação e desativação de objetos, combinada com a movimentação padronizada de câmera fornecida pelo *Cinemachine*. Esse fluxo demonstra uma base funcional para interações complexas entre o jogador e o mundo.

5.5.2 Personagem Controlável

O jogador utiliza um *CharacterController*, com movimentação básica, sem mecânicas avançadas por enquanto. Essa implementação atende plenamente ao estágio inicial do projeto, servindo apenas de meio para testar interações de entrada e saída de veículos. A Figura 21 apresenta os parâmetros desse componente.

Figura 21 – Parâmetros do componente *Character Controller*

Fonte: elaborado pelo autor.

5.6 VALIDAÇÕES TÉCNICAS

Embora testes formais ainda não tenham sido conduzidos, foi possível validar o funcionamento de componentes essenciais: movimentação do jogador e veicular, engate entre veículo e reboque além do terreno. Esses testes informais demonstram que o artefato é funcional e capaz de ser sustentado pelo ecossistema de desenvolvimento fornecido pela *Unity.*

6. TÉCNICAS E RECURSOS DA *UNITY 3D* PARA APRIMORAR EXPERIÊNCIAS INTERATIVAS

A qualidade de experiências digitais criadas em *engines* como a *Unity 3D* depende diretamente da fidelidade das interações, da precisão física dos elementos e da consistência visual dos ambientes. Neste capítulo, são apresentadas técnicas, componentes e ferramentas que ampliam o potencial da *Unity* na construção de artefatos e aplicações interativas, demonstrando recursos que contribuem para realismo, responsividade e profundidade visual.

O foco recai sobre elementos nativos como *Wheel Colliders, Cinemachine, Configurable Joints, Terrain Tools,* que, quando combinados, oferecem uma base robusta para validar comportamentos físicos, construir ambientes detalhados e criar interações dinâmicas dentro da *engine.*

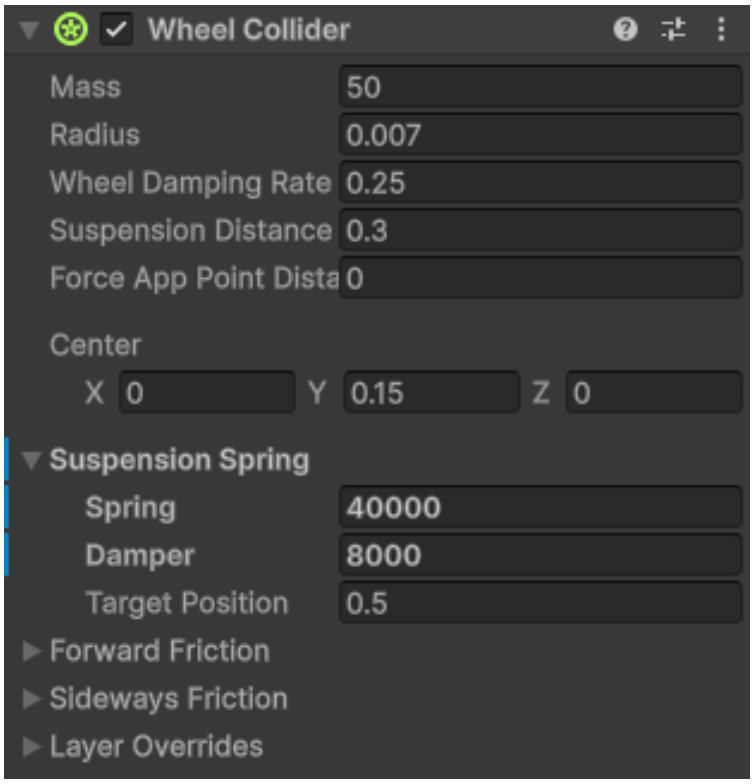
6.1 *WHEEL COLLIDER:* SIMULAÇÃO FÍSICA DE VEÍCULOS

O *Wheel Collider* é a solução nativa da *Unity* para simulação de rodas. Baseado em *raycasts*, ele modela forças de tração, frenagem, suspensão e atrito longitudinal/lateral, conforme discutido por *GamesIndie B* (2017). Dentre seus principais parâmetros estão:

- ***Suspension Distance*** e ***Spring/Damper***: definem o comportamento da suspensão;
- ***Forward* e *Side Friction***: ajustam o atrito longitudinal e lateral;
- ***Motor Torque* e *Brake Torque***: controlam aceleração e frenagem;
- ***Wheel Radius*** e ***Mass***: influenciam o peso e a distribuição de força.

A Figura 22 mostra esses parâmetros.

Figura 22 – Principais parâmetros do componente *Wheel Collider*



Fonte: elaborado pelo autor.

6.1.1 Aplicação no Projeto

No artefato em estudo, o *Wheel Collider* é usado para demonstrar o funcionamento do sistema de física da *Unity* em veículos, permitindo observar o comportamento das rodas, a influência da topografia e a resposta aos parâmetros ajustados na *engine.*

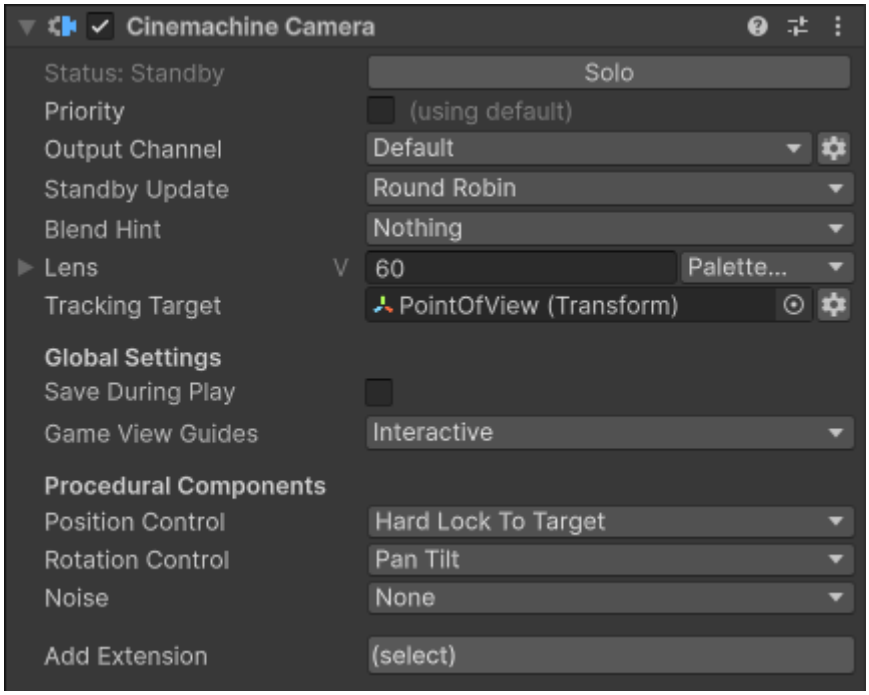
6.2 *CINEMACHINE*: CONTROLE AVANÇADO DE CÂMERAS

Como mostra Desenvolvedor *Unity A* (2022), o *Cinemachine* permite:

- **Transições suaves entre câmeras**, sem necessidade de codificação manual.
- ***Follow* e *LookAt* automáticos**, permitindo acompanhar o veículo em movimento.
- ***Noise Profiles***, que simulam vibrações de câmera, como em veículos em terreno irregular.
- ***Composer* e *Framing Transposer***, que mantêm o foco em elementos principais da cena.

A figura 23 apresenta esses parâmetros.

Figura 23 – Principais parâmetros do componente *Cinemachine Camera*



Fonte: elaborado pelo autor.

6.2.1 Aplicação no Projeto

No artefato, o *Cinemachine* foi empregado para controlar a câmera de forma fluida, alternando entre vistas e acompanhando o personagem e o veículo com suavidade, reforçando compreensão espacial e navegação.

6.3 *CONFIGURABLE JOINTS*: CONEXÕES FÍSICAS AVANÇADAS

O *Configurable Joint*, conforme descrito por *Paridot* (2023), permite configurar restrições de movimento em até seis eixos independentes, diferindo de *joints* mais simples como *Hinge* ou *Fixed Joints GAMESINDIE A (*2017).

É possível ajustar parâmetros como:

- ***Linear Limit****.*
- ***Angular Limit****.*

6.3.1 Aplicação no Projeto

No artefato, o *joint* serve para demonstrar conexões físicas entre objetos, mostrando como a *Unity* lida com inércia, balanço e restrições físicas em tempo real aplicável a sistemas de engate, articulações mecânicas ou estruturas segmentadas.

6.4 *TERRAIN TOOLS* E CONSTRUÇÃO DE TERRENOS REALISTAS

Um *heightmap* é uma imagem em tons de cinza que representa a elevação do terreno, áreas mais claras indicam altitudes maiores. Importando um *heightmap*, é possível **reproduzir relevo real** a partir de dados geográficos reais Desenvolvedor *Unity* B (2023).

6.4.1 Pintura e Modelagem do Terreno

O *Terrain Tools* oferece pincéis configuráveis para **escultura, pintura de texturas e distribuição de vegetação**.
No projeto, essa ferramenta é usada para:

- Criar áreas planas, suavizar e alterar a altura do terreno.
- Aplicar texturas de solo (grama, terra arada, lama).
- Inserir árvores e vegetação.

6.4.2 Ajustes Finais e Desempenho

O uso de ***Level of Detail* (*LOD*)** e ***Occlusion Culling*** é recomendado para otimizar o desempenho sem comprometer a qualidade visual, especialmente em mapas de grandes dimensões.

6.5 INTEGRAÇÃO ENTRE SISTEMAS

A combinação coordenada dos recursos apresentados amplia a capacidade de criação dentro da *Unity*. A física (*Wheel Colliders* + *Joints*) fornece estrutura; *Cinemachine* garante leitura visual; e *Terrain Tools* estabelece o ambiente. Esses sistemas, quando integrados modularmente, permitem construir artefatos que demonstram o funcionamento completo da *engine* e abrem espaço para evoluções incrementais.

7. DISCUSSÃO DOS RESULTADOS OBTIDOS

O desenvolvimento do artefato permitiu observar, na prática, como a *Unity 3D* organiza seus elementos internos, processa dados e gera artefatos digitais a partir de componentes, *scripts* e sistemas de física. A implementação dos recursos escolhidos *Wheel Colliders, Configurable Joint, Terrain Tools, Cinemachine* e estrutura básica de cenas demonstrou o funcionamento integrado da *engine* e sua capacidade de produzir aplicações interativas de forma modular.

Ao longo da fase experimental, foi possível validar que a *Unity* utiliza um modelo de construção baseado em ***GameObjects***, ***Componentes*** e ***Scripts***, o que garante flexibilidade para criar comportamentos físicos, visuais e lógicos. Sistemas como corpos rígidos, colisores, juntas físicas e controladores de câmera responderam conforme esperado, evidenciando a consistência da *API* e a previsibilidade do comportamento interno da *engine.* Esse conjunto de observações contribuiu diretamente para responder à questão de pesquisa sobre como a *Unity* opera e gera artefatos digitais estruturados.

A criação do artefato também permitiu explorar o fluxo de execução utilizado pela *engine,* incluindo a organização de cenas, o uso de eventos do ciclo de vida dos *scripts* (*Start, Update, FixedUpdate*) e o processo de importação, configuração e instanciação de *assets.* Esses elementos evidenciaram como a *Unity* gerencia recursos, atualiza estados e compõe o ambiente de execução final apresentado ao usuário.

Além disso, a etapa prática demonstrou que a produção de artefatos digitais na *Unity* depende da combinação entre:

- **Componentes físicos**, responsáveis pela simulação de forças e colisões.
- **Componentes visuais**, que interpretam malhas, materiais e luzes.
- ***Scripts* em *C#***, que definem comportamentos personalizados.
- **Sistemas de cena**, que agrupam objetos e instruções para compor o ambiente final.
- **Ferramentas de modelagem e edição**, como o *Terrain Tools,* essenciais para estruturar e refinar o ambiente tridimensional.

Outro ponto relevante observado foi a importância da **organização do projeto**, conforme discutido nos capítulos anteriores. A divisão em diretórios específicos para *scripts*, materiais, modelos, cenas e *prefabs* permitiu compreender com mais clareza

como a *Unity* identifica, indexa e serializa arquivos, gerando metadados necessários para manter referências internas. Esse processo está diretamente relacionado à forma como a *engine* cria e mantém artefatos digitais consistentes.

Embora o artefato seja simples, ele demonstrou adequadamente o funcionamento da *engine* e sua capacidade de integrar múltiplos sistemas de maneira coesa. Também evidenciou que mesmo pequenas implementações exigem atenção à modularidade, ao controle de dependências e ao uso adequado dos componentes disponíveis. Esses fatores estão alinhados às boas práticas de engenharia de *software* observadas na literatura e reforçam que a *Unity* é uma plataforma robusta tanto para fins educacionais quanto para desenvolvimento profissional.

Ao comparar os achados deste estudo com trabalhos relacionados, observa-se que pesquisas como a de Marques (2024) enfatizam a aplicação de metodologias formais de engenharia de software no desenvolvimento de jogos, enquanto este trabalho concentrou-se na compreensão técnica da *engine.* Apesar das diferenças de enfoque, ambos convergem na constatação de que a organização estrutural, a modularidade e o uso adequado das ferramentas da *Unity* são elementos fundamentais para a criação de aplicações digitais consistentes.

De forma geral, os resultados obtidos demonstram que a *Unity 3D* oferece um conjunto amplo e integrado de recursos que permitem compreender, de forma prática, como uma *engine* de desenvolvimento processa informações, conecta sistemas internos e gera artefatos digitais completos. A experiência obtida durante o desenvolvimento deste artefato confirma a viabilidade da *engine* para estudos acadêmicos e evidencia sua importância como ferramenta para ensino, experimentação e criação de aplicações interativas.

8. CONCLUSÃO

Este trabalho teve como objetivo responder à questão de pesquisa **“Como funciona a *Unity 3D* e como ela gera artefatos digitais?”**, buscando compreender o funcionamento interno da *engine,* seus recursos fundamentais e sua capacidade de produzir aplicações interativas por meio da combinação entre componentes visuais, físicos e lógicos.

A etapa experimental permitiu aplicar esses conceitos na prática, por meio do desenvolvimento de um artefato técnico voltado à demonstração dos principais recursos da *Unity.* O processo possibilitou observar, em ambiente real, como a *engine* organiza suas estruturas internas, processa eventos, aplica forças físicas, gerencia cenas e atualiza continuamente os estados dos objetos. Recursos como *Wheel Colliders, Configurable Joint, Cinemachine, Terrain Tools* e *scripts* em *C#* foram implementados e testados, revelando a forma como a *Unity* integra múltiplos subsistemas para gerar comportamentos interativos consistentes.

Os resultados obtidos confirmam que a *Unity 3D* fornece um conjunto robusto de ferramentas capaz de sustentar a criação de artefatos funcionais, demonstrando, de forma clara, como a *engine* opera e como seus artefatos digitais são formados. Observou-se também que práticas de organização estrutural, modularidade e separação de responsabilidades princípios fundamentais da engenharia de software contribuem diretamente para a clareza, escalabilidade e manutenção do projeto.

A experimentação prática evidenciou que, mesmo em artefatos simples, o desenvolvimento em *Unity* exige equilíbrio entre domínio técnico, planejamento arquitetural e testes iterativos. Esse conjunto de fatores permitiu não apenas compreender a *engine,* mas também identificar limitações, desafios e possibilidades de aprimoramento, fundamentais para o aprofundamento futuro do projeto.

Assim, conclui-se que o objetivo geral do trabalho foi alcançado: o estudo permitiu compreender o funcionamento da *Unity 3D,* demonstrar seus principais recursos na prática e analisar como a *engine* gera artefatos digitais a partir da integração entre seus sistemas internos. O artefato desenvolvido cumpriu seu papel como instrumento de validação e aprendizagem, fornecendo base sólida para possíveis evoluções e para o desenvolvimento de aplicações mais complexas utilizando a *engine.*

Para a continuidade deste trabalho, recomenda-se:

- Aprofundar o estudo de sistemas avançados da *Unity*, como gerenciamento de animações, interfaces dinâmicas e pipelines de renderização.
- Expandir o artefato com novas funcionalidades que permitam explorar subsistemas ainda não abordados, como eventos, áudio, iluminação e navegação.
- Realizar testes de desempenho e otimização, visando compreender melhor o impacto de diferentes configurações da *engine* sobre a geração de artefatos digitais.

REFERÊNCIAS

ANDRADE, José Raul de Brito. **Da teoria à prática em desenvolvimento de jogos digitais: um estudo sobre os modelos de processo utilizados no mercado paraibano.** 2016. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Federal da Paraíba, João Pessoa, 2016. Disponível em: https://repositorio.ufpb.br/jspui/handle/123456789/2860. Acesso em: 8 maio 2025.

DESENVOLVEDOR UNITY A. Criando Terrenos na Unity Como Você Nunca Viu. [Vídeo]. YouTube, 2023. 14 min. Disponível em: https://www.youtube.com/watch?v=fnmSwcrTPrw. Acesso em: 03 nov. 2025.

DESENVOLVEDOR UNITY B. Movimento e Câmera em Primeira Pessoa na Unity usando *Cinemachine*. [Vídeo]. YouTube, 2022. 17 min. Disponível em: https://www.youtube.com/watch?v=BsmXvXsd9WU. Acesso em: 03 nov. 2025.

DOT DIGITAL GROUP. **Simuladores para a educação digital no agronegócio.** 2019. Disponível em: https://dotgroup.com.br/artigo/simuladores-para-a-educacao-digital-no-agronegocio/. Acesso em: 8 maio 2025.

FALZON, Pierre (ed.). **Ergonomia.** Lisboa: Instituto Piaget, 2004. Disponível em: https://journals.openedition.org/laboreal/pdf/13400. Acesso em: 8 maio 2025.

FORTUNA, J. C. **Simuladores educacionais: definições e apropriações como objetos de aprendizagem.** Educação Gráfica, v. 21, n. 1, p. 267–278, 2017. Disponível em: https://www.educacaografica.inf.br/download-do-artigo?artigo_id=1922. Acesso em: 8 maio 2025.

GAMESINDIE A. Unity 5 – *Hinge Joint* – Configuração Básica e Fazendo uma Carreta. [Vídeo]. YouTube, 2017. 18 min. Disponível em: https://www.youtube.com/watch?v=cRBUDsZ5uJk. Acesso em: 03 nov. 2025.

GAMESINDIE B. Unity 5 – *Wheel Collider* #1 – Configuração Básica e Erros Comuns. [Vídeo]. YouTube, 2017. 33 min. Disponível em: https://www.youtube.com/watch?v=xKXyhGP-jNQ. Acesso em: 03 nov. 2025.

GAMESINDIE B. Unity 5 – *Wheel Collider* #2 – Acelerar, Frear e Dirigir! =]. [Vídeo]. YouTube, 2017. 24 min. Disponível em: https://www.youtube.com/watch?v=FRg3C7x-L40. Acesso em: 03 nov. 2025.

GAUTRON, Romain et al. ***gym-DSSAT: a crop model turned into a Reinforcement Learning environment.*** **arXiv preprint** arXiv:2207.03270, 2022. Disponível em: https://arxiv.org/abs/2207.03270. Acesso em: 8 maio 2025.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa.** 6. ed. São Paulo: Atlas, 2017.

HAX, Fernando; FERREIRA FILHO, Raymundo. **Uso de *games* de simulação de agricultura no ensino técnico agrícola.** ResearchGate, 2015. Disponível em: https://www.researchgate.net/publication/278016180_Uso_de_Games_de_Simulacao_de_Agricultura_no_Ensino_Tecnico_Agricola. Acesso em: 8 maio 2025.

MARQUES, Nael Barreto. **Desenvolvendo jogos utilizando a engenharia de software e a Unity3D.** 2024. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) – Pontifícia Universidade Católica de Goiás, Goiânia, 2024. Disponível em: https://repositorio.pucgoias.edu.br/jspui/handle/123456789/8482. Acesso em: 8 maio 2025.

MATOS, Daniel Andrade de. **Projetando mecânicas de jogos com base em uma abordagem iterativa.** 2020. Trabalho de Conclusão de Curso (Bacharelado em Design) – Universidade de Brasília, Brasília, 2020. Disponível em: https://bdm.unb.br/bitstream/10483/29715/1/2020_DanielAndradeDeMatos_tcc.pdf. Acesso em: 8 maio 2025.

PARIDOT. Using *Configurable Joints* to Make Train Couplers in Unity. [Vídeo]. YouTube, 2023. 7 min. Disponível em: https://www.youtube.com/watch?v=F2CWDjjyadM. Acesso em: 03 nov. 2025.

TORRES, Ricella Delunardo. **Desenvolvendo um jogo para ensinar física com *Unity 3D*.** 2016. Monografia (Bacharelado em Sistemas de Informação) – Universidade Federal de Ouro Preto, João Monlevade, 2016. Disponível em: https://www.monografias.ufop.br/handle/35400000/255. Acesso em: 8 maio 2025.

UNITY TECHNOLOGIES. ***Unity Manual.*** *UNITY TECHNOLOGIES*, 2025. Disponível em: https://docs.unity3d.com/. Acesso em: 27 mar. 2025.

WAZLAWICK, Raul Sidnei. **Metodologia de pesquisa para ciência da computação.** 2. ed. Rio de Janeiro: Elsevier, 2014.

WU, Jing et al. ***Optimizing Crop Management with Reinforcement Learning and Imitation Learning.*** **arXiv preprint** arXiv:2209.09991, 2022. Disponível em: https://arxiv.org/abs/2209.09991. Acesso em: 8 maio 2025.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante FILIPE MOREIRA FERRACIOLI do Curso de Engenharia da Computação, matrícula 20211003300010, telefone: 64 9 9645-9874 e-mail mrffilipe@outlook.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Desenvolvendo um jogo de simulação agrícola com Unity 3D, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, xx de Setembro de 2023.

gov.br Documento assinado digitalmente
FILIPE MOREIRA FERRACIOLI
Data: 16/09/2025 14:41:36-0300
Verifique em https://validar.iti.gov.br

Assinatura do autor: _____

Nome completo do autor: FILIPE MOREIRA FERRACIOLI

Assinatura do professor–orientador: __SOLANGE DA SILVA

Nome completo do professor-orientador: gov.br Documento assinado digitalmente
SOLANGE DA SILVA
Data: 16/09/2025 19:47:58-0300
Verifique em https://validar.iti.gov.br