

**UNIVERSIDADE TUIUTI DO PARANÁ**

**HEITOR FRANCISCO SANCHES WIRMOND**

**QUAL A MELHOR ENGINE PARA SE COMEÇAR NO  
DESENVOLVIMENTO DE JOGOS UNITY X UNREAL**

**CURITIBA  
2023**

**HEITOR FRANCISCO SANCHES WIRMOND**

**QUAL A MELHOR ENGINE PARA SE COMEÇAR NO  
DESENVOLVIMENTO DE JOGOS UNITY X UNREAL**

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Faculdade de Ciências Exatas e de Tecnologia da Universidade Tuiuti do Paraná, como requisito à obtenção ao grau de Bacharel.

Orientador: Prof. Diógenes Cogo Furlan

**CURITIBA  
2023**

## RESUMO

Os jogos desempenham um papel essencial ao longo da história humana, remontando a pelo menos 5 mil anos. Exemplos notáveis incluem o jogo de tabuleiro Senet, utilizado como método de aprendizado e estratégia no antigo Egito, proporcionando entretenimento e promovendo interações sociais.

A evolução das game engines surge como uma resposta à necessidade de facilitar o processo de criação e desenvolvimento de jogos. Ao explorar a história dos jogos, comprehende-se a importância de selecionar a game engine adequada para aqueles que desejam se tornar desenvolvedores de jogos. São analisadas algumas das game engines mais icônicas, desde as pioneiras até as mais modernas, examinando suas características, evolução e os jogos famosos desenvolvidos com cada uma delas.

Os resultados dessa pesquisa fornecem uma base sólida para que os interessados na área de desenvolvimento de jogos possam escolher a game engine mais adequada às suas necessidades e objetivos, servindo como um guia útil para aqueles que desejam iniciar sua jornada como desenvolvedores de jogos e alcançar sucesso nessa indústria em constante evolução.

Como parte prática desse estudo, é proposta a criação de um projeto de jogo utilizando duas game engines analisadas: a Unreal Engine e a Unity. O objetivo desse projeto é demonstrar as diferenças de desempenho, funcionalidades e recursos oferecidos por cada game engine, explorando características únicas, como velocidade de renderização, capacidade de processamento, qualidade gráfica e interações com o jogador.

Essa abordagem prática permite aos leitores compreender melhor as características distintas de cada game engine e tomar decisões mais embasadas ao escolher a plataforma mais adequada para o desenvolvimento de seus próprios jogos, aprofundando a compreensão sobre o desenvolvimento de jogos.

**Palavras-chave:** Game engine, Unreal, jogo, Unity.

## **LISTA DE FIGURAS**

FIGURA 1 – Tabuleiro de Senet.	11
FIGURA 2 – Gameplay do jogo Doom 2: Hell on Earth.	16
FIGURA 3 – Spacewar! no Museu da História do Computador, em 2007.	16
FIGURA 4 – Gameplay do jogo Doom 2: Hell on Earth.	19
FIGURA 5 – Interface Unreal.	22
FIGURA 6 – Interface Unity.	25
FIGURA 7 – Tabela Comparativa.	31
FIGURA 8 – Teste 1.	36
FIGURA 9 – Teste 2.	37
FIGURA 10 – Teste 3.	37
FIGURA 11 – Teste 4.	38
FIGURA 12 – Teste 5.	38
FIGURA 13 – Teste 6.	39
FIGURA 14 – Teste 7.	40
FIGURA 15 – Teste 8.	40

## **LISTA DE TABELAS**

TABELA 1 – Computador Usado para os testes. . . . .	33
TABELA 2 – Lista de testes. . . . .	35
TABELA 3 – Teste 1 - Unity Starter Assets - ThirdPerson   Updates in new CharacterController package. . . . .	36
TABELA 4 – Teste 2 - Unreal Engine Third Person Project. . . . .	36
TABELA 5 – Teste 3- Unreal Engine Vehicle Project. . . . .	37
TABELA 6 – Teste 4- Unreal Engine Jogo Próprio. . . . .	38
TABELA 7 – Teste 5- Unreal Engine Archivs Project. . . . .	38
TABELA 8 – Teste 6 - Unity HDRP Demo Scene Test. . . . .	39
TABELA 9 – Teste 7 - Unity URP Demo Scene Test. . . . .	39
TABELA 10 – Teste 8 jogo próprio Unity . . . . .	40
TABELA 11 – Teste 8 jogo próprio Unity . . . . .	40

## **LISTA DE SIGLAS E ACRÔNIMOS**

ABNT	Associação Brasileira de Normas Técnicas
OTEE	Over the Edge Entertainment
TMRC	Tech Model Railroad Club
AMD	Advanced Micro Devices
HDRP	High Definition Render Pipeline
URP	Universal Render Pipeline

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>8</b>
1.1	CONTEXTO . . . . .	8
1.2	RELEVÂNCIA DO TRABALHO . . . . .	8
1.3	OBJETIVOS . . . . .	8
1.4	ESTRUTURA DO DOCUMENTO . . . . .	9
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA . . . . .</b>	<b>11</b>
2.1	O QUE É UM JOGO? . . . . .	11
2.2	JOGOS ELETRÔNICOS . . . . .	12
2.3	GAME ENGINES . . . . .	13
2.4	O QUE DIFERENÇA UMA GAME ENGINE DE UM SOFTWARE COMUM	14
2.5	HISTÓRIA DAS GAME ENGINES . . . . .	15
2.6	ANOS 50 . . . . .	15
2.7	ANOS 60 . . . . .	15
2.7.1	Spacewar! . . . . .	15
2.8	ANOS 70 E O PONG . . . . .	17
2.9	ANOS 80 . . . . .	17
2.9.1	AGI . . . . .	17
2.9.2	SCUMM - Lucas Arts . . . . .	18
2.10	ANOS 90 . . . . .	18
2.10.1	DOOM Engine . . . . .	18
2.11	UNREAL ENGINE E UNITY . . . . .	19
2.11.1	Unreal Engine . . . . .	19
2.11.2	Unity . . . . .	23
<b>3</b>	<b>REVISÃO DA LITERATURA . . . . .</b>	<b>27</b>
3.1	TRABALHO 1: ESTUDO COMPARATIVO ENTRE ENGINES DE DESENVOLOVIMENTO DE JOGOS 2D . . . . .	27
3.2	TRABALHO 2: ESTUDO COMPARATIVO DE MOTORES DE JOGOS NO DESENVOLVIMENTO DE UM JOGO 2D PARA WEB . . . . .	27
3.3	TRABALHO 3: ANÁLISE DA EVOLUÇÃO DE ENGINES DE JOGOS . . . . .	29
3.4	TRABALHO 4: ANÁLISE DE GAME ENGINES PARA PLATAFORMAS MÓVEIS . . . . .	30
3.5	TRABALHO 5: ANÁLISE DE FERRAMENTAS E DESENVOLVIMENTO DE JOGO PARA TREINAMENTO DE PARATLETAS . . . . .	31
<b>4</b>	<b>METODOLOGIA . . . . .</b>	<b>33</b>
4.1	DEFINIÇÃO DO ESCOPO DA PESQUISA . . . . .	33
4.2	PREPARAÇÃO DO AMBIENTE DE TESTE . . . . .	33
4.3	DEFINIÇÕES DAS MÉTRICAS DE DESEMPENHOS . . . . .	33

4.3.1	Métricas: . . . . .	34
4.4	ESCOLHA DAS DEMOS PARA OS TESTES . . . . .	35
4.5	REALIZAÇÃO DOS TESTES . . . . .	35
<b>5</b>	<b>RESULTADOS EXEPEMENTAIS . . . . .</b>	<b>36</b>
5.1	TESTES 1 E 2 THIRD PERSON CONTROLLERS UNITY E UNREAL ENGINE . . . . .	36
5.2	TESTE 3 VEHICLE PROJECT UNREAL ENGINE . . . . .	37
5.3	TESTE 4 JOGO PRÓPRIO UNREAL ENGINE . . . . .	37
5.4	TESTE 5 UNREAL ENGINE ARCHIVIS PROJECT . . . . .	38
5.5	TESTES 6 E 7 UNITY URP DEMO SCENE E UNITY HDP DEMO SCENE	39
5.6	TESTE 8 JOGO PRÓPRIO UNITY . . . . .	39
5.7	COMPARAÇÃO DE RECURSOS . . . . .	40
5.8	REFLEXÕES SOBRE OS TESTES . . . . .	41
<b>6</b>	<b>CONCLUSÃO . . . . .</b>	<b>42</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>44</b>

## 1 INTRODUÇÃO

### 1.1 CONTEXTO

Os jogos têm desempenhado um papel significativo na história da humanidade, remontando a tempos antigos. Desde os primórdios da civilização, as pessoas se envolvem em atividades lúdicas, proporcionando entretenimento, desafio intelectual e interações sociais. Johan Huizinga definiu o jogo como uma atividade voluntária praticada dentro de certos limites de tempo e espaço, com regras obrigatórias, mas livremente aceitas, acompanhada de um sentimento de tensão e alegria, e com a consciência de ser diferente da vida cotidiana.

### 1.2 RELEVÂNCIA DO TRABALHO

A análise dos jogos e das game engines é relevante para a sociedade, pois os jogos têm um papel importante no entretenimento, na socialização e no desafio intelectual. Compreender o conceito científico dos jogos e sua evolução histórica nos permite apreciar sua importância cultural ao longo do tempo. Além disso, a análise das game engines é crucial para programadores iniciantes, pois ajuda na escolha adequada das ferramentas para o desenvolvimento de jogos de qualidade. Isso resulta em jogos mais acessíveis e envolventes, promovendo a diversão e a interação social. Assim, essa análise contribui para impulsionar a indústria de jogos, estimulando a inovação e a criatividade, e proporciona oportunidades de aprendizado e crescimento para os desenvolvedores,

### 1.3 OBJETIVOS

Objetivos Específicos:

Introduzir o conceito de jogo, destacando suas características, importância cultural e histórica.

Apresentar o conceito de game engine e explicar sua função no desenvolvimento de jogos.

Estudar as game engines de forma geral, analisando suas características, funcionalidades e evolução ao longo do tempo.

Investigar as primeiras game engines desenvolvidas, explorando sua influência e impacto no desenvolvimento de jogos.

Identificar e analisar as game engines mais utilizadas atualmente, destacando suas principais características, popularidade e aplicação em diferentes tipos de jogos.

## Objetivos Gerais:

Compreender a importância dos jogos e das game engines na indústria de entretenimento.

Proporcionar conhecimentos fundamentais sobre o conceito de jogo e sua relação com as game engines.

Fornecer um panorama histórico das game engines, desde as pioneiras até as mais modernas.

Auxiliar programadores iniciantes na escolha adequada de uma game engine para o desenvolvimento de jogos.

Contribuir para o avanço e crescimento da indústria de jogos, incentivando a inovação e o uso eficiente das game engines.

Ajudar na escolha de um desenvolvedor iniciante no mundo dos jogos em qual game engine escolher.

## 1.4 ESTRUTURA DO DOCUMENTO

Essa monografia está organizada da seguinte forma. No capítulo 2 é apresentada toda a fundamentação teórica utilizando conceitos inter-relacionados com o tema proposto provendo uma justificativa bem fundamentada para conduzir o estudo desta pesquisa, contemplando: informações importantes para o entendimento do estudo, a introdução de o que é um jogo e uma game engine, como as primeiras game engines foram feitas, e algumas das principais game engines ao longo da história além de uma explicação mais detalhada das duas engines principais que serão submetidas aos testes O capítulo 3 é apresentada a revisão da literatura correlata, pesquisas que abordam temas como: jogos 2d e 3d, algumas das principais tecnologias do mercado de jogos mobile como a Unity, e construct 3 além de contexto histórico sobre cada tecnologia e uma análise mais aprofundada sobre as características e funcionalidades de cada uma como motor de redenização, animação, Áudio, Colisão e Física, Dispositivos de entrada, todas essas que são ferramentas feitas para auxiliar o usuário mas também aumentam a complexidade do desenvolvimento, o que é fundamental quando a missão é indicar uma engine ideal para novos desenvolvedores na área.

No capítulo 4 apresenta a metodologia a ser aplicada no procedimento de testes e validações de resultados, algo que é necessário ser feito com bastante cuidado pois as engines afetam diretamente o desempenho do jogo a ser desenvolvido, são definidas também as métricas a serem medidas em ambas as engines além de considerar a usabilidade delas pensando em novos desenvolvedores. Ao mesmo tempo são

definidas as preparações para o ambiente de teste e como será a coleta de dados durante os testes e de que forma será feita a análise comparativa dos dados.

## 2 FUNDAMENTAÇÃO TEÓRICA

### 2.1 O QUE É UM JOGO?

Dentro da definição de jogo proposta por Johan Huizinga, que descreve o jogo como uma atividade voluntária praticada dentro de certos limites de tempo e espaço, com regras livremente aceitas, mas obrigatórias, e dotada de um fim em si mesma, o jogo do Senet, um dos jogos analógicos mais antigos conhecidos, exemplifica essas características.

O Senet, originado no Antigo Egito por volta de 3100 a.C., era jogado em um tabuleiro retangular com 30 casas e envolvia a movimentação de peças de acordo com o lançamento de dados. Além de fornecer entretenimento, desafio intelectual e oportunidades de socialização, o Senet tinha um significado simbólico e espiritual para os egípcios antigos. Acredita-se que o resultado do jogo afetava a vida após a morte, adicionando uma dimensão espiritual e cerimonial ao jogo.

Portanto, o Senet exemplifica a prática de um jogo analógico que se enquadra na definição de jogo de acordo com Johan Huizinga. Ele era jogado de forma voluntária, dentro de limites de tempo e espaço determinados, com regras estabelecidas que eram obrigatórias para os jogadores. O Senet tinha seu próprio objetivo intrínseco, proporcionava uma sensação de tensão e alegria durante o jogo e era reconhecido como algo distinto da vida cotidiana, tanto pela sua natureza lúdica quanto pelo seu significado espiritual.

FIGURA 1 – Tabuleiro de Senet.



Outro exemplo icônico de jogo analógico é o xadrez, que se originou na Índia por volta do século VI d.C. O xadrez é jogado em um tabuleiro quadrado com peças distintas, cada uma com movimentos e habilidades específicas. Esse jogo estratégico

requer pensamento tático, planejamento antecipado e tomada de decisões, tornando-se uma forma popular de entretenimento intelectual em várias culturas ao longo da história.

Além disso, jogos de cartas, como o pôquer, também têm uma longa história nos jogos analógicos. O pôquer se desenvolveu no século XIX nos Estados Unidos, mas suas raízes podem ser rastreadas a jogos de cartas mais antigos na Europa e na Ásia. O pôquer envolve habilidades de apostas, blefes e estratégias de jogo, tornando-se uma forma popular de jogo social e competitivo.

Através dos séculos, diferentes culturas criaram uma variedade de jogos que se encaixam na definição de Huizinga. Esses jogos proporcionavam momentos de diversão, competição amigável, exercício mental e conexão social.

## 2.2 JOGOS ELETRÔNICOS

Os jogos eletrônicos são formas de entretenimento interativo baseadas em software, que envolvem a participação ativa dos jogadores em um ambiente virtual. Eles são projetados para proporcionar uma experiência lúdica por meio da combinação de elementos como regras, desafios, objetivos, interatividade e feedback. Os jogos eletrônicos são executados em dispositivos eletrônicos, como computadores, consoles de videogame, smartphones e tablets, e apresentam gráficos, sons e outros estímulos sensoriais que contribuem para a imersão do jogador, eles são caracterizados por uma variedade de elementos essenciais. Eles geralmente possuem um conjunto de regras estabelecidas, que definem os limites e a estrutura do jogo. Essas regras podem incluir desde mecânicas básicas, como movimentação e combate, até sistemas complexos de interação e progressão.

Outro elemento fundamental dos jogos eletrônicos são as metas e objetivos. Os jogadores são apresentados a desafios e tarefas específicas que devem ser alcançados para progredir no jogo. Essas metas podem variar desde simplesmente chegar ao fim de um nível até completar missões complexas ou atingir pontuações elevadas.

De acordo com Juul (2005), um jogo eletrônico é caracterizado por ser uma atividade que possui regras, metas claras, desafios e uma interação sistemática entre o jogador e o sistema virtual. Essa interação permite ao jogador explorar e manipular um ambiente virtual, enfrentar obstáculos e alcançar objetivos específicos.

Os jogos eletrônicos evoluíram ao longo do tempo, passando de formas simples e primitivas para experiências altamente complexas e imersivas. Com o avanço da tecnologia, os gráficos, o áudio e a jogabilidade foram aprimorados, proporcionando aos jogadores uma experiência cada vez mais realista e envolvente.

Além disso, os jogos eletrônicos têm sido objeto de estudos acadêmicos em diversas áreas, como psicologia, sociologia, educação e ciência da computação. A pesquisa científica sobre jogos eletrônicos tem explorado temas como os efeitos dos

jogos na cognição, emoções, habilidades sociais e aprendizado (Granic, Lobel, Engels, 2014). Os jogos eletrônicos também têm sido usados como ferramentas de ensino e treinamento em diferentes contextos, como educação, saúde e negócios (Connolly et al., 2012).

## 2.3 GAME ENGINES

Uma game engine, ou motor de jogo, é um conjunto de ferramentas e sistemas de software que permitem o desenvolvimento, a criação e a execução de jogos digitais. Ela fornece uma estrutura abstrata e um conjunto de funcionalidades essenciais para facilitar o processo de desenvolvimento de jogos, permitindo que os desenvolvedores se concentrem na lógica do jogo, na criação de conteúdo e na experiência do usuário.

A separação entre os componentes principais do software, como renderização gráfica, detecção de colisões, áudio e os elementos artísticos, mundos do jogo e regras de jogo, é uma característica fundamental de uma game engine. Essa separação permite que os desenvolvedores licenciem e modifiquem jogos existentes, criando novos conteúdos e experiências de jogo com alterações mínimas no software do motor. O surgimento da "comunidade de modding" foi impulsionado por essa capacidade de modificar e personalizar jogos usando as ferramentas fornecidas pelos desenvolvedores originais.

Durante a década de 1990, a reutilização e a modificação de motores de jogo foram consideradas, e alguns jogos, como Quake III Arena e Unreal, foram projetados com esse objetivo em mente. Os motores se tornaram altamente personalizáveis, permitindo que os desenvolvedores modifiquessem a funcionalidade do motor por meio de linguagens de script, como o Quake C. A licença de motores também se tornou uma fonte de receita para os desenvolvedores que criaram os motores, à medida que os desenvolvedores de jogos começaram a licenciar motores existentes em vez de desenvolver todos os componentes do motor internamente.

No entanto, a distinção entre um jogo e seu motor nem sempre é clara. Alguns motores estabelecem uma separação mais evidente, enquanto outros não fazem uma distinção clara entre os dois. Em alguns jogos, o código de renderização pode ser específico para desenhar elementos particulares do jogo, como personagens ou objetos, enquanto em outros jogos, o motor de renderização fornece recursos mais gerais para materiais e sombreamento, permitindo que as características do jogo sejam definidas principalmente por meio de dados. Argumentavelmente, uma arquitetura baseada em dados é o que distingue um motor de jogo de um software que é um jogo, mas não um motor. Quando um jogo possui lógica rígida ou regras de jogo codificadas diretamente, ou utiliza código específico para renderizar tipos específicos de objetos de jogo, torna-se difícil ou até impossível reutilizar esse software para criar um jogo

diferente. Portanto, é aconselhável reservar o termo "motor de jogo" para se referir a um software que seja flexível e possa ser utilizado como base para diversos jogos diferentes, sem a necessidade de modificações significativas. Uma game engine é uma plataforma de desenvolvimento de jogos que oferece uma base sólida de ferramentas e sistemas de software para criar jogos digitais. Ela permite que os desenvolvedores se concentrem na criação de conteúdo e na experiência do usuário, ao mesmo tempo em que facilita a reutilização e a modificação de jogos existentes. A linha entre um jogo e seu motor pode ser tênue, com a definição dos dois componentes frequentemente mudando à medida que o design do jogo se solidifica.

## 2.4 O QUE DIFERENCIA UMA GAME ENGINE DE UM SOFTWARE COMUM

Uma Game Engine se diferencia de um software comum devido a várias características e funcionalidades específicas que são essenciais para a criação de jogos interativos.

1 . Componentes especializados: Uma game engine é composta por diversos componentes especializados que atendem às necessidades específicas dos jogos. Isso inclui sistemas de renderização gráfica em 2D ou 3D, detecção de colisões, física simulada, gerenciamento de áudio, entrada de usuário, inteligência artificial, entre outros. Esses componentes são otimizados para oferecer desempenho e funcionalidades adequados aos requisitos dos jogos.

2 . Ferramentas de desenvolvimento: Uma game engine fornece um conjunto abrangente de ferramentas de desenvolvimento que facilitam a criação e a edição de conteúdo para jogos. Isso pode incluir editores visuais para criação de níveis, animação, modelagem 3D, programação visual, entre outras ferramentas que permitem aos desenvolvedores construir e modificar o mundo do jogo de forma eficiente.

3 . Suporte a múltiplas plataformas: As game engines são projetadas para serem portáteis e executar jogos em diferentes plataformas, como PCs, consoles de videogame, dispositivos móveis e até mesmo realidade virtual. Elas lidam com a abstração das diferenças de hardware e sistemas operacionais, permitindo que os desenvolvedores criem jogos uma vez e os executem em várias plataformas.

4 . Flexibilidade e reutilização: Uma característica fundamental de uma game engine é a capacidade de reutilizar o código e o conteúdo entre diferentes projetos de jogos. Uma engine bem projetada oferece uma arquitetura extensível e modular, permitindo que os desenvolvedores adicionem funcionalidades personalizadas e adaptem o motor para atender às necessidades específicas de cada jogo.

5 . Integração no fluxo de produção: As game engines são projetadas para integrar-se aos fluxos de produção de jogos, o que inclui a importação de ativos de arte, animações, áudio e outros recursos criados em ferramentas especializadas, como

software de modelagem 3D ou editores de áudio. Isso facilita o fluxo de trabalho colaborativo e eficiente entre artistas, designers e programadores.

## 2.5 HISTÓRIA DAS GAME ENGINES

Esse capítulo será dedicado a uma introdução a história das game engines desde os anos 40 até os dias atuais. Por mais que a palavra Game Engine seja datada da década de 90 esse conceito existe desde a década de 80 e é baseado em tecnologias anteriores desde os anos 50.

## 2.6 ANOS 50

Na década de 1950, foram desenvolvidos alguns dos primeiros jogos eletrônicos interativos. O primeiro deles foi o Cathode Ray Tube Amusement Device, criado por Thomas T. Goldsmith Jr. e Estle Ray Mann em 1947. Este jogo utilizava componentes analógicos e exibia gráficos em uma tela de tubo de raios catódicos. Os jogadores controlavam um ponto de luz que representava um avião e tinham que destruir alvos na tela. Embora tenha sido um experimento interessante, o CRT Amusement Device não foi disponibilizado ao público em geral.

Em 1952, A.S. Douglas, um estudante de doutorado na Universidade de Cambridge, desenvolveu o jogo Naughts and Crosses, também conhecido como OXO. Ele era exibido em uma tela de tubo de raios catódicos de baixa resolução, conectada a um grande computador mainframe da universidade. O OXO permitia que os jogadores jogassem uma partida de jogo da velha contra o computador.

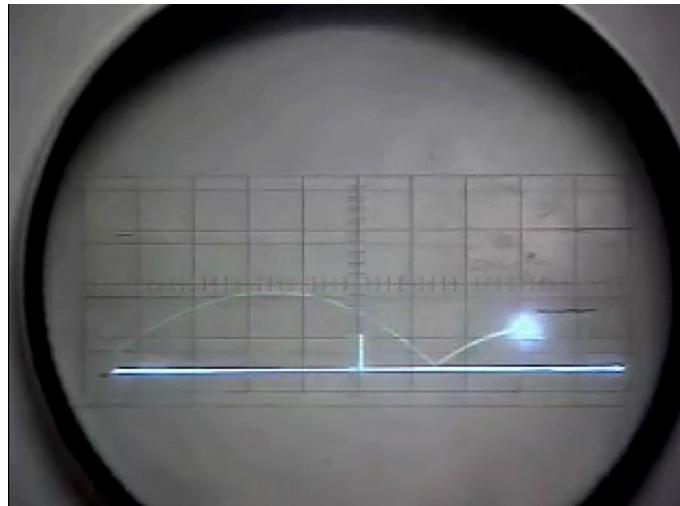
Em 1958, William Higinbotham, um físico que trabalhava no Brookhaven National Laboratory, criou o jogo Tennis for Two. Ele utilizava um computador analógico e um osciloscópio para exibir gráficos simples de uma partida de tênis. O Tennis for Two foi apresentado ao público durante um evento no laboratório, permitindo que os visitantes jogassem e experimentassem uma forma primitiva de jogo eletrônico. Esses jogos pioneiros da década de 1950 são considerados precursores dos videogames modernos. Embora não tenham alcançado a mesma popularidade e comercialização dos jogos subsequentes, eles estabeleceram os fundamentos iniciais da interação eletrônica e dos gráficos em tela, abrindo caminho para o desenvolvimento dos jogos eletrônicos que conhecemos hoje.

## 2.7 ANOS 60

### 2.7.1 Spacewar!

Spacewar! foi desenvolvido por um grupo de programadores do Tech Model Railroad Club (TMRC) do MIT, incluindo Steve Russell, Wayne Witanen e J. Martin

FIGURA 2 – Gameplay do jogo Doom 2: Hell on Earth.



Graetz. Durante os meses de verão de 1961, eles dedicaram cerca de 200 horas de trabalho ao projeto, programando o jogo diretamente em linguagem assembly para o computador PDP-1 da Digital Equipment Corporation (DEC).

O PDP-1 era um computador mainframe pioneiro que utilizava válvulas de vácuo e transistores para processamento de dados. A programação em assembly envolvia escrever instruções de baixo nível específicas para o hardware do PDP-1, o que permitia um controle mais direto sobre as operações do computador.

Ao trabalhar com o hardware do PDP-1, os programadores do TMRC aproveitaram as capacidades do computador para criar os elementos visuais e interativos de Spacewar!. Eles utilizaram o monitor CRT do PDP-1 para exibir os gráficos vetoriais do jogo, que permitiam um movimento suave das naves espaciais e uma experiência visual mais imersiva.

Os controles do jogo foram implementados nos painéis frontais do PDP-1, com switches e botões que os jogadores utilizavam para girar, acelerar, atirar torpedos e usar o recurso de hiperespaço. Essa abordagem direta permitiu aos jogadores uma interação mais imediata com o jogo e uma sensação de controle sobre as naves espaciais.

FIGURA 3 – Spacewar! no Museu da História do Computador, em 2007.



Spacewar! introduziu conceitos inovadores que se tornaram fundamentais nos

jogos de videogame posteriores, como gráficos vetoriais, jogabilidade multiplayer em tempo real e a combinação de ação e estratégia. O jogo foi extremamente popular entre os estudantes do MIT e em outras instituições acadêmicas que tiveram acesso ao PDP-1.

## 2.8 ANOS 70 E O PONG

O jogo Pong, desenvolvido pela Atari, é considerado um marco na história dos videogames. Lançado em 1972, foi concebido por Nolan Bushnell, fundador da Atari, e Allan Alcorn, um engenheiro da empresa. A ideia para o jogo surgiu quando Bushnell jogou o "Tennis for Two", criado por William Higinbotham em 1958.

O desenvolvimento técnico do Pong foi realizado por Allan Alcorn, que utilizou componentes eletrônicos básicos, como transistores e resistores, para criar o jogo. Ao contrário dos jogos modernos que utilizam microprocessadores, o Pong foi desenvolvido sem a utilização desses componentes. O jogo consistia em um monitor preto e branco conectado a uma placa de circuito, com dois controles giratórios que simulavam uma partida de tênis virtual. Alcorn fez várias melhorias no projeto original, incluindo a adição de efeitos sonoros simples.

O primeiro protótipo do Pong foi instalado no bar Andy Capp's Tavern, em Sunnyvale, Califórnia. Rapidamente, o jogo atraiu uma grande quantidade de jogadores e se tornou extremamente popular. Impressionados com o sucesso, a Atari decidiu lançar o Pong como uma máquina de arcade autônoma, o que resultou em um grande sucesso comercial.

O Pong foi um dos primeiros jogos de arcade amplamente reconhecidos e teve um impacto significativo na indústria dos videogames. Sua jogabilidade simples, mas viciante, estabeleceu as bases para muitos outros jogos que viriam depois.

## 2.9 ANOS 80

### 2.9.1 AGI

No final de 1982, a IBM iniciou o desenvolvimento do PCjr, um computador doméstico de menor custo que possuía melhorias nos gráficos e no som em relação ao IBM Personal Computer. A Sierra foi contratada pela IBM para criar um jogo que pudesse aproveitar essas novas capacidades. O resultado foi o desenvolvimento da Adventure Game Interpreter (AGI), uma engine de jogos projetada especificamente para o jogo King's Quest.

O King's Quest, lançado em 1984, foi o jogo que marcou a estreia da AGI. Ele se tornou um dos primeiros jogos de aventura animados e coloridos, com música e efeitos sonoros, a serem executados em computadores domésticos. Embora o PCjr

não tenha sido um sucesso de vendas, o King's Quest foi muito bem recebido pelo mercado de computadores compatíveis com o PC, impulsionando as vendas da Sierra. Isso levou a empresa a continuar o desenvolvimento da AGI e utilizá-la em mais 14 jogos entre 1984 e 1989.

### 2.9.2 SCUMM - Lucas Arts

O Script Creation Utility for Maniac Mansion (SCUMM) é uma engine de jogos desenvolvida pela Lucasfilm Games, posteriormente renomeada LucasArts, para facilitar o desenvolvimento de seu jogo de aventura gráfica Maniac Mansion (1987). Posteriormente, ela foi utilizada como a engine para outros jogos de aventura da LucasArts.

O SCUMM é uma espécie de intermediário entre uma engine de jogo e uma linguagem de programação, permitindo que os designers criem locais, itens e sequências de diálogo sem precisar escrever código na linguagem na qual o código-fonte do jogo é escrito. Isso também possibilitou que o script do jogo e os arquivos de dados fossem multiplataforma, ou seja, reutilizados em várias plataformas diferentes. O SCUMM também serviu como hospedeiro para outras engines de jogo incorporadas.

## 2.10 ANOS 90

### 2.10.1 DOOM Engine

A Doom Engine é uma engine de jogo revolucionária desenvolvida pela id Software para o icônico jogo de tiro em primeira pessoa, Doom, lançado em 1993. Projetada por John Carmack e sua equipe, essa engine teve um impacto significativo na indústria de jogos, popularizando o gênero FPS (First-Person Shooter) e estabelecendo os fundamentos técnicos para muitos jogos futuros.

A Doom Engine introduziu várias inovações que tornaram o jogo único na época. Ela permitia a criação de mapas em ambientes 3D simulados, com espaços tridimensionais, onde os jogadores podiam explorar corredores, salas e ambientes complexos. Além disso, a engine era capaz de renderizar texturas mapeadas em paredes e objetos, proporcionando detalhes visuais realistas.

Outro destaque da Doom Engine era o seu sistema de iluminação simulado em tempo real, que permitia áreas do jogo serem iluminadas e sombreadas de maneira dinâmica. Além disso, a engine suportava efeitos sonoros digitais de alta qualidade e trilhas sonoras de fundo, contribuindo para a imersão do jogador.

Acima um exemplo de um dos principais jogos feitos na doom engine Doom 2: Hell on Earth. A Doom Engine também foi projetada para suportar o modo multiplayer, permitindo que os jogadores se conectassem e competissem ou colaborassem em am-

FIGURA 4 – Gameplay do jogo Doom 2: Hell on Earth.



bientes compartilhados. Sua capacidade de multiplayer contribuiu para a popularidade duradoura do jogo.

A Doom Engine foi licenciada para outros desenvolvedores, levando à criação de uma série de jogos similares ao Doom. Exemplos de jogos baseados na engine incluem Heretic, Hexen, Strife e a sequência direta de Doom, Doom II: Hell on Earth.

## 2.11 UNREAL ENGINE E UNITY

Essas duas engines são descritas com maior detalhe pois serão nosso objeto de pesquisa, pois serem duas das mais populares engines não só para iniciantes como são as mais pedidas no mercado atualmente.

### 2.11.1 Unreal Engine

O Unreal Engine, criado por Tim Sweeney, fundador da Epic Games, passou por um processo de desenvolvimento significativo em sua primeira geração. A jornada começou em 1995, quando Sweeney decidiu construir um motor para apoiar a produção de um jogo revolucionário que eventualmente se tornaria o famoso Unreal, um FPS em primeira pessoa lançado em 1998. Antes disso, Sweeney já havia desenvolvido ferramentas de edição para seus jogos shareware ZZT e Jill of the Jungle.

Durante o desenvolvimento, Sweeney assumiu a tarefa árdua de programar o motor quase sozinho. Ele escreveu cerca de 90 por cento do código, cobrindo todos os aspectos, desde os sistemas gráficos até as ferramentas e o sistema de rede. No início, o Unreal Engine dependia exclusivamente de renderização por software, com todas as operações gráficas sendo executadas pela CPU. Com o tempo, o motor evoluiu para aproveitar os recursos oferecidos pelas placas gráficas dedicadas, com destaque para a API Glide, desenvolvida especialmente para aceleradores 3dfx. Embora o Unreal Engine também suportasse OpenGL e Direct3D, a API Glide era preferível devido às suas vantagens em termos de desempenho e gerenciamento de texturas.

O desenvolvimento do motor trouxe desafios significativos, com a criação do

renderizador sendo a parte mais difícil. Sweeney precisou reescrever o algoritmo central várias vezes até alcançar os resultados desejados. Além disso, a criação da infraestrutura que conectava todos os subsistemas do motor exigiu um esforço considerável.

A primeira geração do Unreal Engine trouxe várias inovações gráficas, incluindo detecção de colisão, iluminação colorida e um sistema básico de filtragem de texturas. Além disso, o motor incluía o UnrealEd, um editor de níveis que permitia que os designers alterassem o layout dos níveis em tempo real. Essa funcionalidade flexível proporcionou uma experiência de criação de mapas avançada e contribuiu para o sucesso do motor.

Embora o Unreal fosse desenvolvido para competir com Doom, criados pela id Software, ele recebeu elogios de John Carmack, co-fundador da id Software. Carmack reconheceu os avanços visuais do Unreal, como a implementação de efeitos especiais, e afirmou que o Unreal Engine estabeleceu novos padrões para jogos de ação.

À medida que a popularidade do Unreal crescia, a Epic Games aprimorou o motor para lançar o Unreal Tournament. Essa versão melhorada do Unreal Engine foi projetada para otimizar o desempenho em máquinas de menor potência, aprimorar o código de rede e melhorar a inteligência artificial dos bots em modos de jogo em equipe.

Epic Games lançou o Unreal Engine 2 em 2002, oferecendo suporte para o desenvolvimento de jogos tanto em PCs quanto em consoles populares da época, como PlayStation 2, GameCube e Xbox. O Unreal Engine 2 se destacou por sua versatilidade e recursos avançados. Introduziu o motor de física Karma e um sistema de partículas, que permitiram a criação de simulações físicas realistas, como colisões de personagens e dinâmicas de objetos rígidos.

O motor de jogo recebeu grande reconhecimento e foi adotado como um padrão da indústria, impulsionado pelo sucesso de jogos aclamados como BioShock e Thief: Deadly Shadows. O Unreal Engine 2 também trouxe melhorias significativas na renderização, permitindo níveis mais detalhados e visuais impressionantes.

Além disso, o Unreal Engine 2 expandiu as possibilidades de criação ao oferecer recursos como uma ferramenta de edição cinematográfica, suporte para exportação de modelos 3D de softwares populares e um sistema de animação esquelética. Essas adições ampliaram o alcance do motor de jogo, permitindo o desenvolvimento de jogos em diferentes estilos e gêneros.

Além de sua ampla utilização nos consoles mencionados, o Unreal Engine 2 teve uma versão especializada chamada UE2X, otimizada especificamente para o Xbox original. Também foi surpreendente ver o Unreal Engine 2 sendo executado no console portátil Nintendo 3DS, como demonstrado pelo jogo Tom Clancy's Splinter Cell 3D, desenvolvido pela Ubisoft Montreal.

Esses avanços e recursos do Unreal Engine 2 tornaram-no uma escolha popular para desenvolvedores de jogos, impulsionando seu sucesso e influência na indústria de jogos.

O Unreal Engine 3 (UE3) foi lançado em 2005 e teve um impacto significativo no avanço dos gráficos e recursos para jogos. Ele foi amplamente adotado em plataformas populares da época, como PlayStation 3, Xbox 360 e Wii. Um aspecto notável do UE3 foi a colaboração com a NVIDIA, que possibilitou a implementação do suporte ao recurso PhysX para simulações físicas avançadas nos jogos. Essa parceria resultou na criação de efeitos realistas, como explosões dinâmicas e interações convincentes com objetos no ambiente do jogo, melhorando a imersão e a qualidade das experiências de jogo.

O Unreal Engine 4 (UE4), anunciado em 2012 e lançado em 2014, trouxe uma série de melhorias técnicas e recursos para a criação de jogos. Com suporte para uma ampla variedade de plataformas, como Microsoft Windows, Linux, Mac OS X, Xbox One, PlayStation 4, iOS, Android e Nintendo Switch, o UE4 ofereceu flexibilidade aos desenvolvedores.

Uma das principais melhorias do UE4 foi a introdução da iluminação global aprimorada, que utilizava técnicas como o voxel cone tracing para renderizar a iluminação em tempo real, eliminando a necessidade de iluminação pré-renderizada. Isso resultou em ambientes mais realistas e imersivos nos jogos desenvolvidos com a engine.

Além disso, o UE4 apresentou o sistema visual de scripting chamado "Blueprint", que permitia um desenvolvimento mais rápido e intuitivo da lógica dos jogos. Com uma interface gráfica, o Blueprint facilitou a colaboração entre artistas, designers e programadores, reduzindo o tempo necessário para implementar e testar alterações no jogo. Outra vantagem do UE4 foi a capacidade de atualizar o código C++ em tempo de execução, possibilitando ajustes mais rápidos e eficientes durante o desenvolvimento.

Essas melhorias e recursos do Unreal Engine 4 contribuíram para a criação de jogos de alta qualidade, proporcionando aos desenvolvedores uma plataforma poderosa e flexível para expressar sua criatividade e entregar experiências interativas impressionantes.

O Unreal Engine 5 (UE5) foi anunciado em 2020 e lançado em 2022, trazendo avanços significativos em termos de gráficos e tecnologia para jogos. Essa versão introduziu recursos impressionantes, como o "Nanite", que permite o uso de modelos altamente detalhados, com milhões ou bilhões de polígonos, sem a necessidade de otimizações manuais. Isso possibilitou a criação de ambientes detalhados e realistas, reduzindo o esforço de desenvolvimento necessário.

Outro destaque do UE5 é o "Lumen", um sistema de iluminação global em tempo real que oferece uma iluminação mais realista e dinâmica, com efeitos de sombra e reflexos aprimorados. Isso contribui para a criação de ambientes visualmente

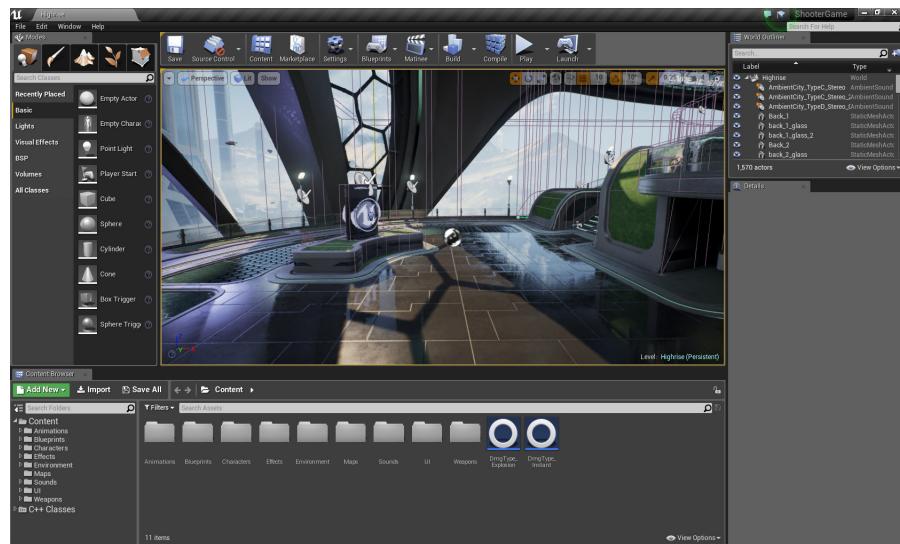
impressionantes e imersivos.

A versão mais recente do Unreal Engine é o UE5.1, lançado em novembro de 2022. Essa atualização trouxe aprimoramentos adicionais, como suporte para tecnologias de ponta, incluindo ray tracing em tempo real e recursos aprimorados de física e simulação.

O Unreal Engine 5 continua sendo amplamente utilizado na indústria de jogos e fornece aos desenvolvedores as ferramentas necessárias para criar experiências imersivas e visualmente impressionantes. Com seus recursos avançados e contínuas atualizações, o UE5 se mantém como uma das principais opções para desenvolvedores na criação de jogos de alta qualidade.

O Unreal Engine hoje permite o desenvolvimento para 21 plataformas diferentes, incluindo Playstation 4, Nintendo Switch, Xbox One, Mac OS X, Steam, Oculus Rift, Playstation VR, Samsung Gear VR, ARCore, Daydream, Magic Leap, SteamVR, HTC Vive, Linux, HTML5, Windows, Android, iOS, ARKit, Facebook Gameroom e PsVita. Essa ampla variedade de suporte de plataforma oferece aos desenvolvedores a flexibilidade de criar jogos e experiências interativas para uma ampla gama de dispositivos e sistemas operacionais, alcançando uma grande audiência de jogadores.

FIGURA 5 – Interface Unreal.



A tela de projeto na Unreal Engine é o centro de criação e gerenciamento do jogo. É uma interface visual rica em recursos que permite importar ativos, criar e organizar níveis, ajustar propriedades de objetos e criar lógica de jogo.

Na tela de projeto da Unreal Engine, os desenvolvedores podem importar ativos, como modelos 3D, texturas, áudios e animações, e organizá-los em pastas para facilitar o acesso e o uso.

Os desenvolvedores podem criar e organizar níveis do jogo, conhecidos como "mapas", onde podem construir ambientes, adicionar objetos, definir a iluminação e

aplicar efeitos especiais. Eles podem acessar esses mapas através de um editor de níveis, que oferece uma visualização interativa em 3D.

A tela de projeto também oferece painéis e janelas personalizáveis, como o painel de detalhes, onde os desenvolvedores podem ajustar as propriedades de objetos selecionados, e o painel de conteúdo, onde podem navegar e pesquisar por ativos disponíveis.

Um recurso poderoso na tela de projeto da Unreal Engine é o Blueprint Editor, que permite aos desenvolvedores criar lógica de jogo visualmente, utilizando blocos de código chamados "Blueprints". Isso facilita a programação e a prototipagem rápida de funcionalidades interativas. Com as novas versões da Unreal engine e epic começou a dar um suporte mais longo a elas, durante anos até o lançamento da próxima versão para que os desenvolvedores sempre tenham a versão mais recente.

### 2.11.2 Unity

A história do Unity começa com a formação da OTEE (Over the Edge Entertainment), composta por Nicholas Francis, Joachim Ante e David Helgason. Em 2002, Nicholas postou em um fórum pedindo ajuda para implementar um sistema de shaders em seu motor de jogo. Joachim respondeu e eles decidiram colaborar para criar um sistema de shaders que funcionasse em ambos os seus motores separados. No entanto, eles logo perceberam que era mais divertido trabalhar juntos em um único motor e decidiram criar um novo engine de jogos. Embora inicialmente planejassem desenvolver jogos, eles viram a necessidade de uma tecnologia de base mais avançada e decidiram criar uma ferramenta para fazer jogos, em vez de fazer jogos em si.

Para financiar o desenvolvimento do Unity, eles decidiram criar um jogo comercial completo usando o novo motor. Assim, o jogo Gooball foi lançado em março de 2005. Além de testar o motor, o objetivo era gerar receita para sustentar a equipe e futuros desenvolvimentos.

Com os lucros do Gooball, a OTEE contratou mais desenvolvedores para aprimorar o Unity antes de seu lançamento oficial em junho de 2005. Eles se esforçaram para tornar o Unity o mais polido possível, corrigindo bugs, melhorando a interface e fornecendo documentação e suporte abrangentes para os usuários.

Inicialmente, a maioria dos usuários do Unity eram entusiastas e desenvolvedores independentes. No entanto, a OTEE percebeu a importância de atrair desenvolvedores profissionais e grandes empresas de jogos para usar o Unity. Isso levou a dois anos de suporte e atualizações constantes para provar que o Unity era uma plataforma estável e bem suportada.

As primeiras versões do Unity foram lançadas com suporte apenas para Mac OS X. No entanto, o suporte para Windows e navegadores web foi adicionado nas versões

posteriores. O Unity trouxe gráficos 3D acelerados por hardware para a indústria de jogos em navegadores, substituindo opções como o Flash. A capacidade de utilizar plugins externos em C/C++ também foi adicionada, permitindo que os desenvolvedores estendessem o Unity para usar hardware e software não suportados nativamente.

O Unity continuou a evoluir rapidamente, e em 2007 foi lançada a versão 2.0, que trouxe suporte aprimorado para Windows, além de recursos como streaming de web, sombras em tempo real, networking, engine de terreno e um novo sistema de interface gráfica em código.

Com o lançamento do iPhone e a popularidade crescente de aplicativos móveis, a Unity Technologies decidiu desenvolver o Unity iPhone, uma versão do Unity que permitia a publicação de jogos para o iPhone. A Unity também expandiu seu suporte para Windows e lançou o Unity 2.5 na Game Developers Conference em 2009.

Em 2010, o Unity 3.0 foi lançado, trazendo recursos aguardados, como unificação do editor, iluminação Beast, renderização diferida, ocultação Umbra, depuração de baixo nível e filtros de áudio FMOD. A versão 3.5 adicionou suporte ao Flash, ampliando ainda mais as opções de publicação.

O lançamento do Unity 4.0 trouxe recursos importantes, incluindo a tecnologia de animação Mecanim, suporte a Linux, sistema de partículas Shuriken e suporte nativo para desenvolvimento de jogos 2D, acabando com a necessidade de técnicas improvisadas para criar jogos 2D no Unity.

O lançamento do Unity 5 em 2015 marcou um passo importante para tornar o desenvolvimento de jogos acessível a todos. Ele trouxe melhorias na iluminação e áudio, além do suporte ao WebGL, permitindo que os jogos fossem adicionados a navegadores compatíveis na web sem a necessidade de plug-ins.

O Unity 5.0 introduziu recursos como iluminação global em tempo real, pré-visualizações de mapeamento de luz, Unity Cloud, um novo sistema de áudio e o motor de física Nvidia PhysX 3.3. Além disso, os Efeitos de Imagem Cinemáticos foram introduzidos para tornar os jogos do Unity mais distintos.

A versão 5.6 do Unity trouxe novos efeitos de iluminação e partículas, melhorias de desempenho e suporte nativo para plataformas como Nintendo Switch, Facebook Gameroom, Google Daydream e a API de gráficos Vulkan. Um reproduutor de vídeo 4K capaz de reproduzir vídeos em 360 graus para realidade virtual também foi adicionado.

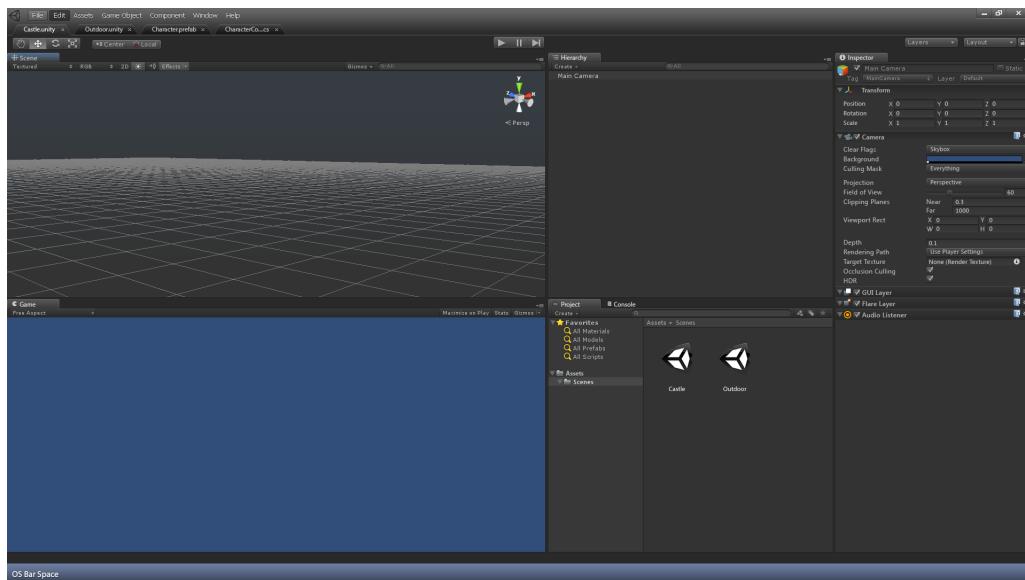
Apesar do sucesso do Unity em democratizar o desenvolvimento de jogos, alguns jogadores criticaram a acessibilidade da plataforma devido à grande quantidade de jogos produzidos rapidamente por desenvolvedores inexperientes.

A Unity Technologies continuou a lançar versões atualizadas, como o Unity 2017 e o Unity 2018. O Unity 2017 introduziu ferramentas como a Timeline, que permitia aos desenvolvedores criar animações arrastando e soltando elementos nos jogos, e o Cinemachine, um sistema de câmera inteligente. O Unity 2018 trouxe o Scriptable

Render Pipeline para criar gráficos de alta qualidade, ferramentas de aprendizado de máquina e suporte para o Magic Leap.

A partir de 2020, aplicativos feitos com o Unity estavam sendo executados em mais de 1,5 bilhão de dispositivos, representando 50 porcento de todos os jogos para dispositivos móveis. A Unity também expandiu suas atividades além dos jogos de videogame, adquirindo a Finger Food Advanced Technology Group e lançando o Unity Forma, uma ferramenta de solução automotiva e de varejo. A Unity Engine é bastante flexível, permitindo importar artes e assets em diversos formatos, suportando modelos 3D em uma variedade de formatos, como .fbx, .dae (Collada), .3ds, .dxf, .obj, .skp, 3D Studio Max, Maya, Blender, Cinema4D, Modo, LightWave e Cheetah3D. Além disso, oferece suporte direto para áudio nos formatos .mp3, .ogg, .wav, .aiff, .aif, .mod, .it, .s3m e .xm, e imagens nos formatos BMP, TIF, TGA, JPG e PSD. De acordo com o manual da ´ ultima versão da Unity disponível (2022.3), Atualmente a Unity suporta as seguintes plataformas para desenvolvimento: Windows, ios, android, macos, linux, xbox one, xbox series, playstation 4, playstation 5, stadia, nintendo switch, ios, desktop VR, mobile VR, Hololens, WebGL.

FIGURA 6 – Interface Unity.



A tela de projeto no Unity é onde os desenvolvedores podem criar, organizar e gerenciar seu jogo. É uma interface visual que permite importar e organizar ativos, criar cenas, definir configurações do projeto e acessar ferramentas de desenvolvimento.

Na tela de projeto do Unity, os desenvolvedores podem importar ativos como modelos 3D, texturas, áudios e animações. Eles podem organizar esses ativos em pastas para facilitar o acesso e o uso em seu jogo.

Além disso, os desenvolvedores podem criar cenas, que são os ambientes do jogo, arrastando e soltando objetos na hierarquia. Eles podem ajustar as propriedades dos objetos selecionados e definir sua posição, rotação e escala na cena.

A tela de projeto também permite configurar as preferências e as configurações do projeto, como resolução, plataforma de destino e qualidade gráfica. Os desenvolvedores também podem acessar ferramentas de desenvolvimento, como o editor de scripts, onde podem escrever e editar o código do jogo.

A história do Unity é marcada pelo objetivo de tornar o desenvolvimento de jogos mais acessível e pela constante evolução do motor de jogo para atender às necessidades dos desenvolvedores, ela se destacou por sua flexibilidade na importação de diferentes formatos de arquivos, além de oferecer suporte para uma ampla variedade de plataformas, desde dispositivos móveis até consoles de jogos e realidade virtual.

### **3 REVISÃO DA LITERATURA**

Neste capítulo serão apresentados os trabalhos relacionados, os quais seguem uma linha de pesquisa semelhante ao proposto nesta pesquisa, visando assim aprofundar o projeto de pesquisa e analisar diferentes visões e desenvolvimentos da mesma área acadêmica.

#### **3.1 TRABALHO 1: ESTUDO COMPARATIVO ENTRE ENGINES DE DESENVOLVIMENTO DE JOGOS 2D**

Autor: Gelderson Bezerra Alves

O trabalho compara duas das engines mais populares para o desenvolvimento de jogos independentes 2D, Unity e Construct 3 . O objetivo principal é comparar diversos aspectos em relação à fase de desenvolvimento do jogo escolhido em cada engine . Os parâmetros de comparação são de natureza unicamente técnicas, para que o desenvolvedor consiga entender exatamente as características que cada engine apresenta . A criação do mesmo jogo em ambas ferramentas é o passo crucial para realizar essa comparação e conseguir obter resultados sólidos . Os resultados da comparação evidenciam que, enquanto a Unity dá mais poder de desenvolvimento para o desenvolvedor ao custo da complexidade da ferramenta, a Construct 3 é mais simples de desenvolver um jogo 2D .

O ambiente atual da internet permite que a criação de jogos independentes se torne uma realidade no mercado de desenvolvimento de games. Com isso, diversas ferramentas para desenvolver jogos de forma ágil surgiram na comunidade. Escolher a melhor engine para desenvolver um jogo dessa natureza nem sempre é fácil. Este trabalho visa a comparação entre duas das engines mais populares atualmente para o desenvolvimento de jogos independentes 2D, Unity e Construct 3. O objetivo principal é comparar diversos aspectos em relação à fase de desenvolvimento do jogo escolhido em cada engine. Os parâmetros de comparação são de natureza unicamente técnicas, para que o desenvolvedor consiga entender exatamente as características que cada engine apresenta. A criação do mesmo jogo em ambas ferramentas é o passo crucial para realizar essa comparação e conseguir obter resultados sólidos. Os resultados da comparação evidenciam que, enquanto a Unity dá mais poder de desenvolvimento para o desenvolvedor ao custo da complexidade da ferramenta, a Construct 3 é mais simples de desenvolver um jogo 2D .

#### **3.2 TRABALHO 2: ESTUDO COMPARATIVO DE MOTORES DE JOGOS NO DESENVOLVIMENTO DE UM JOGO 2D PARA WEB**

Autor: JEAN IGOR DE QUEIROZ PANTOJA

O trabalho 2 tem como objetivo principal realizar um estudo comparativo entre diferentes tecnologias de desenvolvimento de jogos por meio da implementação de um jogo 2D para a web. O estudo busca analisar e avaliar os motores de jogo, auxiliando usuários iniciantes na escolha do melhor motor para atender às suas necessidades específicas.

No contexto da globalização e do avanço tecnológico, as indústrias de jogos digitais têm sido impulsionadas, acompanhadas por políticas públicas que incentivam a cultura e o desenvolvimento econômico nesse setor. Os jogos digitais são considerados bens informacionais com um enorme potencial de geração de empregos, renda e também de mobilização social e cognitiva. Além disso, esses jogos têm conquistado popularidade em diversas faixas etárias e vêm sendo amplamente utilizados como ferramentas educativas.

No decorrer do trabalho, são abordadas as características distintas dos jogos 2D e 3D, enfatizando que os jogos 2D são mais leves e casuais, enquanto os jogos 3D proporcionam uma experiência imersiva e realista. Também são apresentados diversos gêneros de jogos, como ação, simulação, aventura, estratégia, quebra-cabeça, RPG e luta, sendo fornecidos exemplos de jogos representativos de cada categoria.

A pesquisa se concentra nas tecnologias de desenvolvimento de jogos, desde os primórdios dos jogos desenvolvidos em editores de texto até os modernos motores de jogo. Os motores de jogo são explorados como conjuntos de bibliotecas que oferecem funcionalidades pré-programadas, destinadas a auxiliar no desenvolvimento de jogos. Além disso, são mencionadas ferramentas comuns utilizadas nesse processo, como a documentação oficial do motor de jogo, ambiente de desenvolvimento integrado, interface de usuário, personagens não jogáveis, objetos do jogo, sistema de entrada, scripts e física.

Destaca-se a relevância do documento de design de jogo, elaborado pelo game designer, que descreve diversos aspectos do jogo, tais como ideias fundamentais, enredo, personagens, cenários e design de níveis. Esse documento é considerado uma ferramenta de suma importância para orientar a equipe de desenvolvimento e facilitar a compreensão do jogo por parte dos investidores.

No âmbito da pesquisa empírica, foram selecionados os motores de jogo mais procurados no Google no Brasil durante os últimos 12 meses. Os motores selecionados foram o Unity, o Godot, o Game Maker, o RPG Maker, o Construct e o Blender. Após aplicar critérios de inclusão e exclusão, foram descartados os motores Unreal e Blender devido a restrições de licença e requisitos mínimos de hardware e software. Os motores Unity e Godot foram escolhidos para a fase de desenvolvimento do jogo "Os mortos-vivos Quixadá", e métricas foram estabelecidas para a comparação dos motores, levando em consideração o código fonte gerado durante o desenvolvimento e os cenários de teste.

Os resultados obtidos revelaram diferenças significativas entre os motores Unity e Godot em relação aos requisitos mínimos, tamanho final do jogo, complexidade do código e desempenho em termos de uso da CPU e memória RAM. Com base nesses achados, conclui-se que o motor Godot é recomendado para usuários iniciantes no desenvolvimento de jogos, enquanto o Unity é mais indicado para jogos com funcionalidades complexas e exigências mais elevadas, buscando proporcionar uma experiência imersiva e envolvente para o jogador.

Portanto, este trabalho científico contribui para a compreensão das tecnologias de desenvolvimento de jogos, oferecendo insights valiosos para os desenvolvedores iniciantes na escolha do motor de jogo mais adequado às suas necessidades, além de fornecer uma análise comparativa detalhada entre os motores Unity e Godot, embasada em dados empíricos coletados durante a implementação do jogo "Os mortos-vivos Quixadá".

### 3.3 TRABALHO 3: ANÁLISE DA EVOLUÇÃO DE ENGINES DE JOGOS

Autores: Daniel Scherer, Daniele Ventura Batista, Aline de Cantalice Mendes

Este trabalho científico tem como objetivo investigar a evolução das Game Engines na indústria de jogos, fornecendo uma análise comparativa das principais plataformas utilizadas atualmente. O estudo inicia com uma breve contextualização histórica sobre o desenvolvimento de jogos, desde os primórdios até os dias atuais, destacando os marcos importantes que moldaram a indústria, como o advento do Spacewar, DOOM e Quake.

Em seguida, é abordada a transição do desenvolvimento manual de jogos para o uso de Game Engines, ressaltando as vantagens e a redução de complexidade proporcionada por essas ferramentas para os desenvolvedores. São discutidos os benefícios trazidos pelas Game Engines, como a facilitação do processo de criação, a incorporação de recursos gráficos avançados, a simplificação da implementação de física e colisões, além da capacidade de suportar múltiplas plataformas.

O estudo analisa, de forma detalhada, as principais Game Engines utilizadas atualmente, incluindo o RPG Maker, Unreal, M.U.G.E.N, Game Maker e CryEngine. Cada plataforma é examinada em termos de suas características, funcionalidades, linguagens de programação suportadas e capacidades de exportação para diferentes plataformas, como PC, consoles e dispositivos móveis. São destacadas as vantagens e desvantagens de cada Engine, proporcionando uma visão abrangente das opções disponíveis para os desenvolvedores.

Além disso, é explorado como a evolução contínua das Game Engines tem impulsionado o desenvolvimento de jogos de alta qualidade e impactado positivamente a indústria de jogos. O estudo enfatiza como essas plataformas têm contribuído

para aprimorar a experiência do jogador, possibilitando a criação de mundos virtuais imersivos, gráficos realistas e mecânicas de jogo inovadoras. Também é discutido como as Game Engines têm sido utilizadas em outras áreas além dos jogos, como medicina e arquitetura, para simulações e visualizações interativas.

Por fim, são apresentadas conclusões sobre a evolução das Game Engines na indústria de jogos, evidenciando o impacto positivo dessas ferramentas no processo de desenvolvimento de jogos e destacando a importância de escolher a plataforma adequada para atender às necessidades e objetivos de cada projeto.

### 3.4 TRABALHO 4: ANÁLISE DE GAME ENGINES PARA PLATAFORMAS MÓVEIS

Autor:Rennan Araújo Barbosa

Este trabalho científico aborda a importância do desenvolvimento de jogos e a crescente demanda por soluções que facilitem esse processo, com foco nas engines de desenvolvimento de jogos para dispositivos móveis. O texto destaca as principais dificuldades enfrentadas pelos desenvolvedores, como a implementação de funções realistas (gravidade, colisões, iluminação, áudio e animação) e a necessidade de otimização de código.

Para lidar com esses desafios, as game engines surgiram como uma solução. Elas são softwares que oferecem uma série de rotinas e funcionalidades comuns, permitindo que os desenvolvedores se concentrem em aspectos mais específicos do jogo. Dessa forma, as engines aumentam a produtividade e aceleram o processo de desenvolvimento.

No contexto das plataformas móveis, onde o mercado de jogos está em constante crescimento, o uso de game engines se torna ainda mais crucial. A escolha adequada da engine pode garantir um desenvolvimento mais eficiente e rápido, atendendo às demandas do mercado.

O texto discute três game engines específicas para dispositivos móveis: AndEngine, Cocos2D e Unity. Cada uma delas apresenta características e funcionalidades únicas. A AndEngine é uma biblioteca gratuita voltada para o desenvolvimento de jogos 2D no Android. Ela utiliza a API OpenGL para renderização gráfica e possui suporte para animação, áudio e física, por meio da integração com a engine Box2D.

A Cocos2D, por sua vez, é uma biblioteca para o desenvolvimento de jogos 2D no iOS e MacOS. Ela também utiliza a API OpenGL ES para renderização gráfica e oferece suporte para animação e áudio por meio da engine CocosDenshion. Assim como a AndEngine, a Cocos2D utiliza a Box2D para lidar com física e colisões.

A Unity, por sua vez, é uma engine multiplataforma amplamente utilizada no desenvolvimento de jogos móveis. Ela oferece uma gama completa de recursos, incluindo suporte gráfico avançado, animação, áudio, física, colisões, redes, multijogador

e inteligência artificial. A Unity possui uma interface de usuário intuitiva, um conjunto abrangente de ferramentas de análise e depuração, além de uma grande comunidade de desenvolvedores.

FIGURA 7 – Tabela Comparativa.

	<i>Cocos2D</i>	<i>AndEngine</i>	<i>Unity</i>
Motor de Renderização			
Animação			
Áudio			
Colisão e Física			
Sistemas de núcleo			
Dispositivos de Entrada			
Análise e Depuração			
Independência de Plataforma			
Gerente de Recursos			
Redes e Multijogador			
IA			
Otimização			

No decorrer do trabalho, são discutidos também os conceitos de jogo, suas classificações com base em características, gênero, faixa etária recomendada e gráficos. O planejamento de um jogo, que envolve a definição das características, o Game Design Document e a seleção da plataforma de desenvolvimento, também é abordado.

Destaca-se a evolução do desenvolvimento de jogos ao longo do tempo, desde a programação em Assembly até as engines modernas. São mencionadas as diferentes metodologias de desenvolvimento de jogos, como o processo cascata e os métodos ágeis adaptados para a indústria de jogos.

Além disso, são apresentadas soluções como as APIs multimídia, ferramentas sem programação e as próprias game engines para o desenvolvimento de jogos digitais. As APIs multimídia fornecem acesso simplificado ao hardware do dispositivo, abstraindo detalhes de baixo nível e melhorando o desempenho do jogo. As ferramentas sem programação permitem a criação visual de jogos, sem a necessidade de conhecimento em programação, embora possam ter limitações em termos de recursos e flexibilidade. Já as game engines oferecem uma abordagem mais abrangente, fornecendo uma camada de abstração para funcionalidades comuns do desenvolvimento de jogos, como renderização, física e colisões.

### 3.5 TRABALHO 5: ANÁLISE DE FERRAMENTAS E DESENVOLVIMENTO DE JOGO PARA TREINAMENTO DE PARATLETAS

Autor:Deyvson Lazaro da Silva

Este trabalho aborda a análise de game engines para o desenvolvimento de um jogo de treinamento de paratletas no basquete em cadeira de rodas. A importância da escolha adequada de ferramentas, incluindo a game engine, é ressaltada. O objetivo principal é realizar a análise comparativa das game engines e desenvolver um jogo adequado aos requisitos propostos.

A estrutura do trabalho é descrita, começando pelo referencial teórico no Capítulo 2, onde são apresentados conceitos relacionados ao desenvolvimento de jogos e game engines. Os conceitos de jogos, incluindo a definição utilizada neste trabalho, são abordados, assim como a categoria de "serious games". Também é discutida a importância da diversão e entretenimento nos jogos, bem como a evolução da indústria.

No Capítulo seguinte, são apresentados os conceitos relacionados às game engines, explicando sua função de gerenciar o ciclo de vida dos jogos. A arquitetura de uma game engine é descrita, assim como algumas das principais opções disponíveis no mercado, como Unity, CryEngine, Unreal Engine e Lumberyard. A Unity recebe uma descrição mais detalhada, destacando seus recursos e suporte multiplataforma.

A análise comparativa das game engines é realizada no Capítulo seguinte. Parâmetros como sistema de script, curva de aprendizado, licenciamento e custo, documentação e suporte são considerados. Os resultados da comparação indicam que a Unity é a game engine mais adequada devido à sua comunidade ativa, vasta documentação e suporte multiplataforma.

O desenvolvimento do jogo de treinamento de paratletas é proposto no Capítulo seguinte. O contexto e a motivação para o desenvolvimento são apresentados, destacando a importância de superar as barreiras existentes no treinamento de atletas paralímpicos. A proposta inclui um simulador que utiliza a Unity engine, o sensor Kinect v2 e imagens projetadas em uma parede. Restrições, premissas e requisitos do sistema são descritos.

No capítulo final, são apresentadas as considerações finais do trabalho. Destaca-se a contribuição para a área de treinamento de paratletas ao propor uma solução tecnológica inovadora e explorar o potencial dos jogos digitais nesse contexto. Sugere-se como trabalhos futuros a implementação e avaliação do jogo desenvolvido, a expansão do sistema para outras modalidades esportivas e a realização de estudos adicionais sobre a eficácia do treinamento com o uso de jogos.

## 4 METODOLOGIA

### 4.1 DEFINIÇÃO DO ESCOPO DA PESQUISA

Foram definidos 4 ambientes de teste em cada engine para poder mostrar ambas game engines Unity e Unreal em diferentes situações, sendo elas 6 demos disponibilizadas pelas próprias empresas para teste, que estão disponíveis para download gratuitamente e 2 jogos feitos pelo autor, um para cada game engine.

### 4.2 PREPARAÇÃO DO AMBIENTE DE TESTE

Foram verificadas as especificações de cada engine previamente para a definição do ambiente de teste para que se encaixe com as especificações do computador do autor. Durante os Testes todos os programas externos ao teste foram fechados, para simular um ambiente igual para todos. Para os testes foram utilizados a versão da Unity 2022.8.3 e a versão 5.3 da Unreal engine. Para a coleta dos dados foi utilizado o próprio software de rastreamento de informações da AMD. AMD Software: Adrenalin edition versão 23.11.1.

TABELA 1 – Computador Usado para os testes.

Tipo do Componente	Modelo do Componente
CPU	Ryzen 7 2700
Placa de Vídeo	Rx 570
Fonte	500w 80plus Bronze
Placa mãe	b450m
Memória ram	16gb 2400mhz ddr4

FONTE: O próprio Autor, 2023.

### 4.3 DEFINIÇÕES DAS MÉTRICAS DE DESEMPENHO

Foram definidos 5 métricas relevantes para a ajuda da discussão proposta principalmente por que uma das principais dificuldades da definição de uma engine é o computador da determinada pessoa que quer iniciar na área, como game engines não

são programas leves, nem todos principalmente no brasil tem a condição monetária para ter um computador de ponta. Os itens definidos durante os testes foram, FPS Máximo, FPS Mínimo, porcentagem de uso da CPU, porcentagem de uso da GPU e quantidade de uso da memória ram durante o teste. A seguir uma explicação de cada métrica escolhida para o teste

#### 4.3.1 Métricas:

- Fps Minimo e Máximo: Fps (Quadros por segundo) é a unidade de cadência de fotogramas feitos por segundo em um dispositivo audiovisual, em nossos testes o mínimo considerado aceitável são 30 quadros por segundo, sendo o ideal acima de 60 quadros.
- Porcentagem de uso da CPU: CPU (Unidade central de processamento) é o componente principal de um computador responsável por executar instruções de programas armazenados na memória. A CPU realiza operações lógicas, aritméticas, de controle e outras funções necessárias para executar tarefas específicas. Em nossos testes quanto maior o "Mundo" proposto pelo jogo em teste, mais a CPU é utilizada, quanto menos utilizada a CPU melhor.
- Porcentagem de uso da GPU: GPU (Unidade de processamento Gráfico) componente de hardware especializado projetado para acelerar o processamento de gráficos e imagens em um computador. Também é o componente mais exigido quando falamos de game engines visto que é o componente mais utilizado percentualmente. Quanto menor o uso dela, melhor o resultado.
- 'Uso da memória RAM

A Memória de Acesso Aleatório (RAM), é um tipo de memória volátil empregada em sistemas computacionais e demais dispositivos eletrônicos. Denominada "aleatória" por possibilitar o acesso direto a qualquer local de armazenamento, independentemente da sequência de dados previamente armazenados. RAM desempenha um papel essencial ao armazenar temporariamente dados e instruções em uso ativo pelo sistema operacional e pelos programas em execução. Ao iniciar o computador, o sistema operacional e os aplicativos são carregados na RAM, proporcionando acesso rápido aos dados necessários durante a operação. Vale ressaltar que a RAM é volátil, ou seja, perde seus dados quando a energia é desligada. Também é fundamental para o desenvolvimento de jogos, já que são softwares com várias janelas abertas ao mesmo tempo, assim consumindo uma grande quantidade de memória. Quanto menor o consumo da memória melhor o resultado dos testes.

#### 4.4 ESCOLHA DAS DEMOS PARA OS TESTES

- Na tabela abaixo está a lista dos 8 testes (4 para cada engine) eles foram escolhidos pelos seguintes motivos:
- Disponibilizar testes nos mais variados ambientes desde o mais simples até o mais complexo.
- Um teste para cada engine no mínimo, feito pelo próprio autor, a fim de experenciar como é o desenvolvimento na plataforma na prática.
- todos os testes precisam ser abertos para serem baixados gratuitamente.

TABELA 2 – Lista de testes.

Plataforma	Nome do teste
Unity	Unity Starter Assets - ThirdPerson   Updates in new CharacterController package
Unity	Unity Terrain - URP Demo Scene
Unity	Unity Terrain - HDRP Demo Scene
Unity	Jogo Próprio Unity
Unreal Engine	Jogo Próprio Unreal Engine
Unreal Engine	Third Person Project
Unreal Engine	Vehicle Project
Unreal Engine	Archivis Project

FONTE: O próprio autor, 2023.

#### 4.5 REALIZAÇÃO DOS TESTES

Cada teste foi rodado por cerca de 5 minutos, utilizando o AMD SOFTWARE: Adrenalin Edition 23.11.1 para coleta de dados, o primeiro minuto de cada teste foi desconsiderado pois, demora algum segundos pra placa gráfica e os outros componentes se estabilizarem no teste. Todos os testes foram feitos em 4 opções gráficas diferentes a exceto se na qualidade gráfica o desempenho ja for mais do que o suficiente para rodar o conteúdo com tranquilidade, não há motivo para rodar em qualidades gráficas inferiores para o tema proposto.

## 5 RESULTADOS EXPERIMENTAIS

A seguir os resultados dos testes com uma tabela para cada um dos testes e seus resultados, com uma breve descrição de cada teste, e uma reflexão final após todos eles.

### 5.1 TESTES 1 E 2 THIRD PERSON CONTROLLERS UNITY E UNREAL ENGINE

- Primeiro e segundo testes ambos feitos com demos obtidas dos fabricantes das engines, com controles básicos de um personagem em terceira pessoa, testando o que seria o mínimo necessário para rodar a engine, como nesse caso o teste na Unity era muito leve mesmo na qualidade gráfica mais alta ele foi testado apenas nesta qualidade.

TABELA 3 – Teste 1 - Unity Starter Assets - ThirdPerson | Updates in new Character-Controller package.

Qualidade Gráfica	Fps Máximo	FPS Mínimo	Utilização do CPU	Utilização da Memória Ram	Utilização da GPU
Muito Baixo	-	-	-	-	-
Médio	-	-	-	-	-
Alto	-	-	-	-	-
Ultra	320	250	entre 15 e 25 porcento	8.6 GB/s	1200 MB/s

FONTE: O próprio Autor, 2023.

FIGURA 8 – Teste 1.

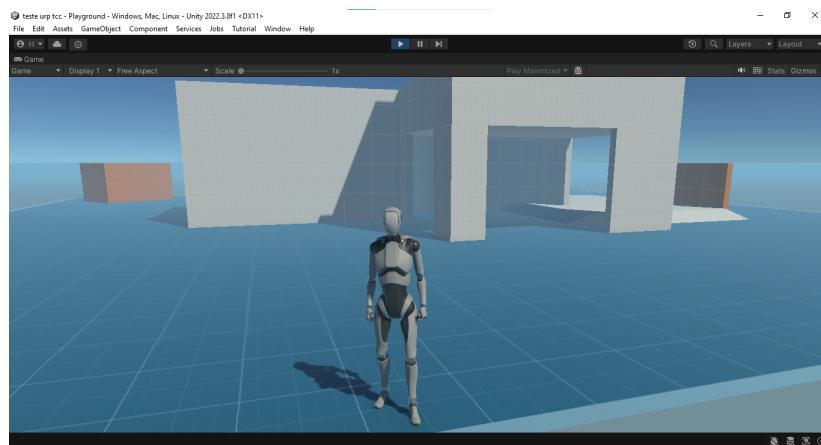
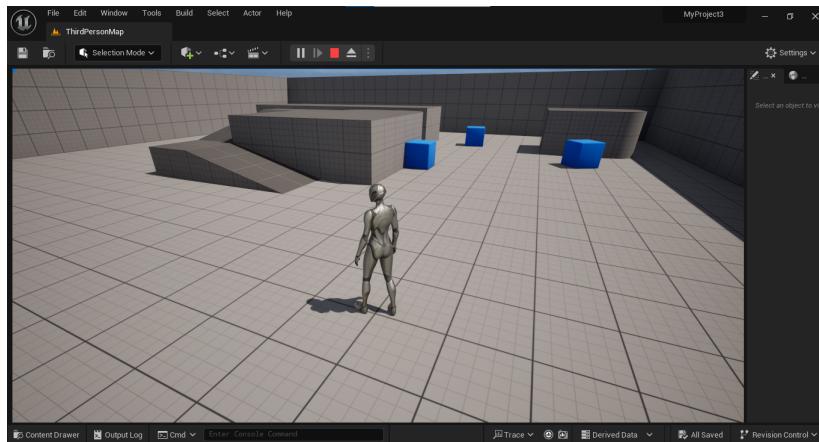


TABELA 4 – Teste 2 - Unreal Engine Third Person Project.

Qualidade Gráfica	Fps Máximo	FPS Mínimo	Utilização do CPU	Utilização da Memória Ram	Utilização da GPU
Muito Baixo	117	108	entre 25 e 30 porcento	7.8 GB/s	1570 MB/s
Médio	114	98	entre 25 e 30 porcento	7.8 GB/s	1811 MB/s
Alto	65	63	entre 26 e 30 porcento	8.5 GB/s	3490 MB/s
Ultra	53	48	entre 25 e 30 porcento	10 GB/s	3800 MB/s

FONTE: O próprio Autor, 2023.

FIGURA 9 – Teste 2.



## 5.2 TESTE 3 VEHICLE PROJECT UNREAL ENGINE

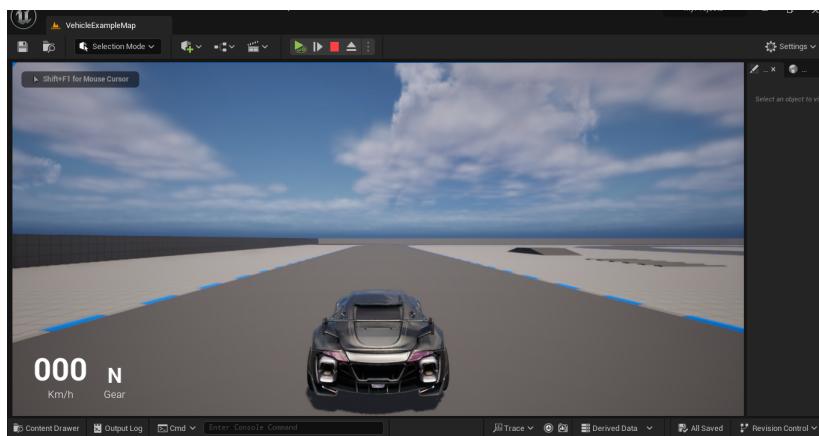
Demo de jogo de corrida disponível para download pela própria Unreal, escolhido para simular um ambiente não tão grande quanto um game de mundo aberto mas bem maior do que os 2 primeiros testes

TABELA 5 – Teste 3- Unreal Engine Vehicle Project.

Qualidade Gráfica	Fps Máximo	FPS Mínimo	Utilização do CPU	Utilização da Memória Ram	Utilização da GPU
Muito Baixo	120	120	entre 28 e 30 porcento	8.4 GB/s	1500 MB/s
Médio	116	90	entre 30 e 35 porcento	7.8 GB/s	31000 MB/s
Alto	105	70	entre 30 e 35 porcento	7.8 GB/s	3.800 MB/s
Ultra	83	58	entre 30 e 35 porcento	9 GB/s	3800 MB/s aviso de falta de memoria

FONTE: O próprio Autor, 2023.

FIGURA 10 – Teste 3.



## 5.3 TESTE 4 JOGO PRÓPRIO UNREAL ENGINE

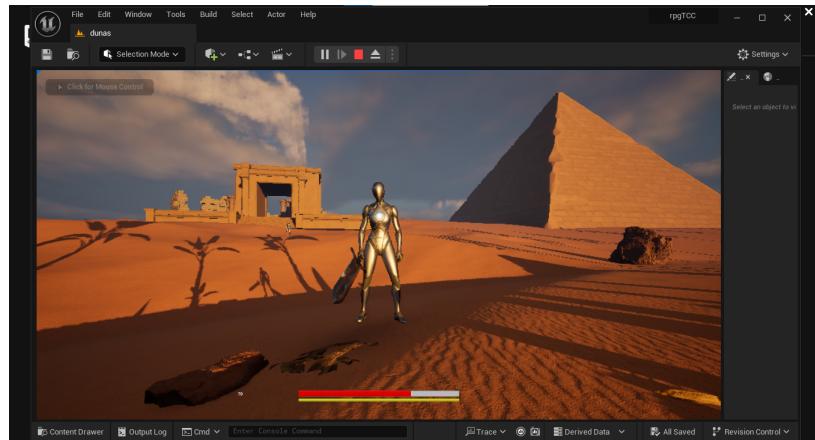
Demo de jogo próprio feito para simular um pequeno mundo aberto inspirado no egito antigo, texturas de alta resolução escolhidas para tentar chegar ao limite de desempenho do computador

TABELA 6 – Teste 4- Unreal Engine Jogo Próprio.

Qualidade Gráfica	Fps Máximo	FPS Mínimo	Utilização do CPU	Utilização da Memória Ram	Utilização da GPU
Muito Baixo	118	109	entre 28 e 30 porcento	7.8 GB/s	3000 MB/s
Médio	116	90	entre 30 e 35 porcento	7.8 GB/s	3100 MB/s
Alto	105	70	entre 30 e 35 porcento	7.8GB/s	3.800mb MB/s
Ultra	83	58	entre 30 e 35 porcento	9 GB/s	3800 MB/s aviso de falta de memoria

FONTE: O próprio Autor, 2023.

FIGURA 11 – Teste 4.



#### 5.4 TESTE 5 UNREAL ENGINE ARCHIVIS PROJECT

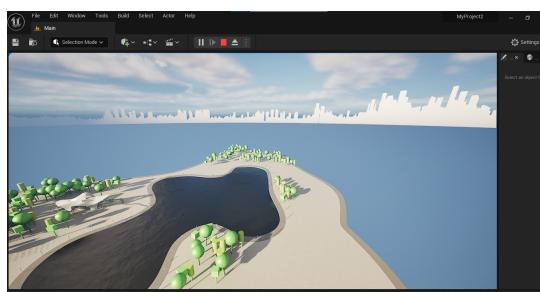
Demo de arquitetura da Unreal engine escolhido para simular um mundo aberto, não é um jogo mas como as demos de jogos abertos da Unreal são extremamente exigentes esse projeto serve como uma simulação de um mundo aberto em renderização para termos uma noção de como seria a performance porém com texturas não tão exigentes quanto o teste anterior, apesar de ser maior em tamanho.

TABELA 7 – Teste 5- Unreal Engine Archivs Project.

Qualidade Gráfica	Fps Máximo	FPS Mínimo	Utilização do CPU	Utilização da Memória Ram	Utilização da GPU
Muito Baixo	120	118	entre 16 e 35 porcento	8 GB/s	3200 MB/s
Médio	110	94	entre 15 e 35 porcento	8.2 GB/s	3570 MB/s
Alto	40	35	entre 18 e 20 porcento	9 GB/s	3870 MB/s
Ultra	30	28	entre 15 e 20 porcento	10 GB/s	3900 MB/s aviso de falta de memoria

FONTE: O próprio Autor, 2023.

FIGURA 12 – Teste 5.



## 5.5 TESTES 6 E 7 UNITY URP DEMO SCENE E UNITY HDP DEMO SCENE

- Duas demos identicas simulando mundo aberto porém com técnicas de renderização diferentes da Unity, enquanto uma está em HDRP que prioriza os melhores gráficos a outra é feita em URP que tem um maior suporte de plataformas, sendo esta segunda a única com suporte para dispositivos móveis. Nos testes se constata que a demo em URP é um tanto mais leve em quantidade de memória ram utilizada e com a diferença gráfica não sendo grande porém não é necessariamente mais otimizada.

TABELA 8 – Teste 6 - Unity HDRP Demo Scene Test.

Qualidade Gráfica	Fps Máximo	FPS Mínimo	Utilização do CPU	Utilização da Memória Ram	Utilização da GPU
Muito Baixo	82	43	entre 20 e 30 porcento	11 GB/s	3800 MB/s
Médio	80	42	entre 20 e 30 porcento	12 GB/s	3900 MB/s
Alto	82	35	entre 20 e 30 porcento	12 GB/s	3900 MB/s
Ultra	80	34	entre 20 e 30 porcento	12 GB/s	3900 MB/s

FONTE: O próprio Autor, 2023.

FIGURA 13 – Teste 6.

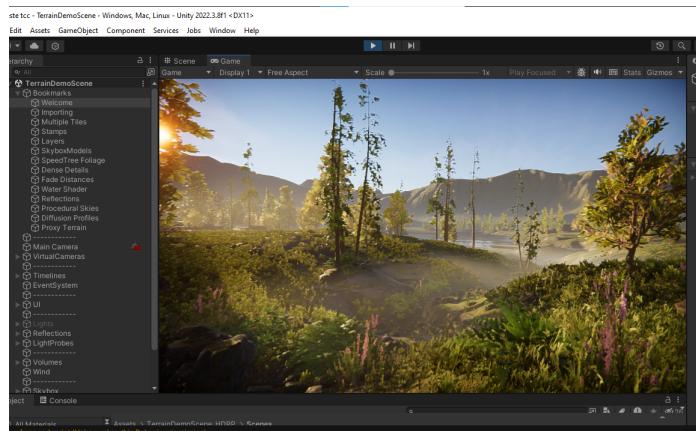


TABELA 9 – Teste 7 - Unity URP Demo Scene Test.

Qualidade Gráfica	Fps Máximo	FPS Mínimo	Utilização do CPU	Utilização da Memória Ram	Utilização da GPU
Muito Baixo	45	15	entre 10 e 20 porcento	9.2 GB/s	3400 MB/s
Médio	38	15	entre 10 e 20 porcento	9.5 GB/s	3600 MB/s
Alto	35	13	entre 10 e 20 porcento	10.5 GB/s	3800 MB/s
Ultra	30	13	entre 10 e 20 porcento	10.2 GB/s	3800 MB/s

FONTE: O próprio Autor, 2023.

## 5.6 TESTE 8 JOGO PRÓPRIO UNITY

Jogo próprio feito na Unity, escolhido para simular como seria a performance de um jogo 3d feito por um iniciante, visto que foi um dos primeiros projetos 3d do autor.

FIGURA 14 – Teste 7.

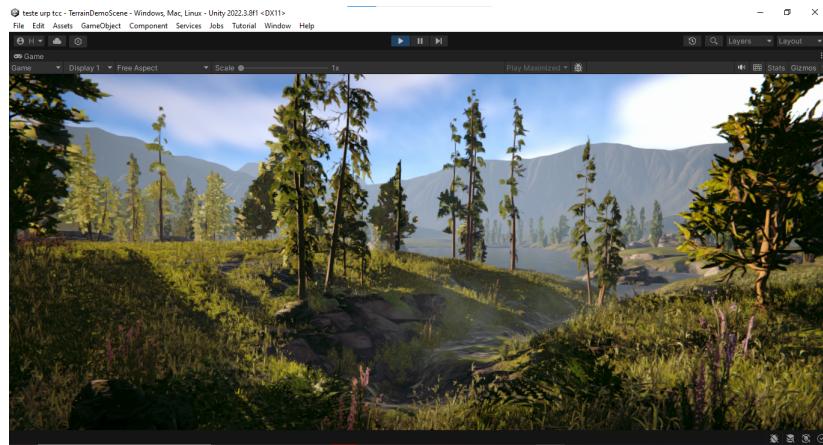
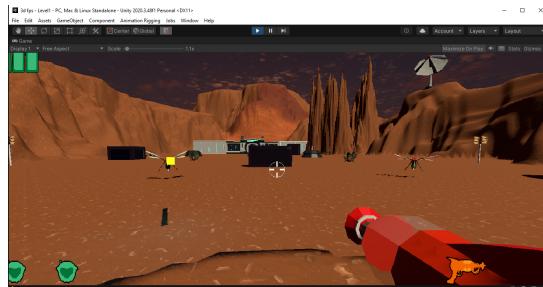


TABELA 10 – Teste 8 jogo próprio Unity .

Qualidade Gráfica	Fps Máximo	FPS Mínimo	Utilização do CPU	Utilização da Memória Ram	Utilização da GPUS
Muito Baixo	-	-	-	-	-
Médio	-	-	-	-	-
Alto	-	-	-	-	-
Ultra	135	80	entre 10 e 20 porcento	7.6 GB/s	760mb MB/s

FONTE: O próprio Autor, 2023.

FIGURA 15 – Teste 8.



## 5.7 COMPARAÇÃO DE RECURSOS

Abaixo uma tabela com os principais recursos de cada engine analisada, assim possibilita uma análise mais direta entre as engines, comparando os recursos entre elas.

TABELA 11 – Teste 8 jogo próprio Unity .

Recurso	Unity	Unreal
Plataformas Suportadas	Xbox, Playstation, Nintendo Switch, Windows, Linux, VR, Mac, android, ios, WebGL	Xbox, Playstation, Nintendo Switch, Windows, Linux, VR, Mac, android, ios
Linguagens suportadas	C	C++, Blueprints
Illuminação Global nativa	Não	Sim
Sombras Dinâmicas nativas	Não	Sim
Malhas de Alta resolução, nativas e gratuitas	Não	Sim
Maior facilidade para desenvolvimento de jogos 2d	Sim	Não
Maior otimização em pequena escala	Sim	Não
Maior otimização em larga escala	Sim	Não
Gratuito	Sim	Sim
Tutoriais gratuitos para aprendizagem na própria plataforma	Sim	Sim

FONTE: O próprio Autor, 2023.

## 5.8 REFLEXÕES SOBRE OS TESTES

Após todas as baterias de testes foi verificado que, apesar de ser mais exigente computacionalmente, não necessariamente a Unity engine é menos otimizada, pelo contrário, entre as configurações de qualidade gráfica foram notadas diferenças muito maiores entre a performance nos testes na Unreal, do que na Unity, o qual é um ponto positivo, pois pode atender desde quem tem um computador mais básico e está começando, até um desenvolvedor experiente que precisa rodar tudo no máximo. Já a Unity não é recomendada para jogos de alta qualidade gráfica, principalmente realista, além de possuir assets de menor resolução, principalmente os gratuitos, ela escala muito mal em performance, é ótima para jogos leves, especialmente 2d, porém se é necessário uma qualidade gráfica maior, a definição e os gráficos são piores na Unity, com um desempenho pior que a Unreal engine também. Uma observação, os resultados a cima são dos testes em modo janela em ambas as engines, mas também foram testados em modo full screen, neste modo todos os testes na Unreal engine exigiram cerca de 2 GB de ram a mais, enquanto os na Unity exigiram praticamente a mesma quantidade. Os testes foram feitos com apenas os testes rodando na engine, porém numa situação real de desenvolvimento variás abas estariam abertas, além de ambientes de código e outros programas então para um desenvolvimento prático necessita-se de mais memória ram do que os testes apontam. Por fim, entre 4 testes na Unreal 3 deram aviso de falta de memória de vídeo na placa gráfica utilizada, porém colocado no alto já melhorou bastante o resultado, sem perca significativa de qualidade gráfica.

## 6 CONCLUSÃO

Ao longo da trajetória histórica, os jogos têm desempenhado uma função de relevância significativa na experiência humana. Exemplos paradigmáticos, como Senet, xadrez e pôquer, ilustram as características inerentes aos jogos analógicos, conforme conceituadas por Johan Huizinga. Com a emergência dos jogos eletrônicos, esta modalidade de entretenimento evoluiu substancialmente, proporcionando experiências imersivas e desafios complexos. Além de seu valor recreativo, os jogos eletrônicos têm sido objeto de investigação acadêmica em diversas disciplinas e têm sido empregados como ferramentas educacionais e de treinamento, promovendo o estímulo cognitivo e a interação social.

No contexto do desenvolvimento de jogos eletrônicos, as game engines assumem uma posição de crucial importância. Estas oferecem um conjunto especializado de recursos, ferramentas de desenvolvimento, suporte multiplataforma, flexibilidade para a reutilização de código e integração eficiente no fluxo de produção. Tais atributos conferem às game engines uma relevância fundamental no desenvolvimento de jogos interativos, permitindo que os desenvolvedores concentrem seus esforços na criação de conteúdo, proporcionando experiências envolventes aos jogadores.

Ao longo do tempo, o desenvolvimento e a adoção de engines de jogos têm transformado substancialmente a indústria de jogos. Exemplos notáveis incluem a Adventure Game Interpreter (AGI), que catalisou o êxito de títulos como King's Quest, viabilizando jogos de aventura animados e coloridos. A Doom Engine, por sua vez, exerceu um impacto ainda mais notório ao popularizar o gênero de tiro em primeira pessoa (FPS) e estabelecer padrões técnicos para jogos futuros, apresentando recursos avançados, como ambientes 3D simulados, renderização de texturas mapeadas, iluminação em tempo real e suporte a multiplayer.

Estas engines não apenas impulsionaram o sucesso de jogos individuais, mas também pavimentaram o caminho para o desenvolvimento de outros títulos com base em suas estruturas, evidenciando o papel crucial das engines como ferramentas essenciais para fomentar a criatividade e inovação na indústria de jogos, influenciando o panorama do desenvolvimento de jogos até os dias atuais.

Dentro deste contexto, duas das game engines mais proeminentes são o Unity e a Unreal Engine, cada qual com suas características distintivas. O Unity destaca-se por sua abordagem em tornar o desenvolvimento de jogos mais acessível e democrático, sendo amplamente adotado por desenvolvedores independentes e estúdios de pequeno e médio porte. Dotado de uma interface intuitiva e suporte multiplataforma, possibilita que os desenvolvedores iniciem a criação de jogos rapidamente, mesmo sem um conhecimento aprofundado em programação.

Por outro lado, a Unreal Engine, desenvolvida pela Epic Games, é reconhe-

cida por sua qualidade visual excepcional e recursos gráficos poderosos. Sendo a preferência para estúdios de grande porte e projetos que demandam um alto padrão de fidelidade visual, a Unreal Engine oferece ferramentas avançadas, incluindo o Blueprints, um sistema de programação visual que permite criar lógica de jogo sem a necessidade de codificação direta.

Uma diferença notável entre o Unity e a Unreal Engine reside no modelo de negócios adotado por cada uma. O Unity oferece um modelo de licenciamento flexível, com opções gratuitas e pagas, enquanto a Unreal Engine adota uma estrutura de royalties, na qual os desenvolvedores remuneram a Epic Games com uma porcentagem dos lucros obtidos com o jogo.

A escolha entre a Unreal Engine e a Unity no desenvolvimento de jogos está intrinsecamente vinculada às necessidades e objetivos específicos de cada desenvolvedor. A Unreal Engine destaca-se pela excepcional qualidade gráfica e é amplamente empregada por grandes empresas, oferecendo um mercado de trabalho mais abrangente. Empresas renomadas, como 2K, Ninja Theory, Tencent, Devolver, Obsidian e CD Projekt Red, além dos gigantes Xbox e PlayStation, utilizam o motor da Epic Games.

Entretanto, o caminho na Unreal Engine pode apresentar desafios para iniciantes, dado o requisito de maior experiência, especialmente em linguagens como C++ e na gestão dos Blueprints. Embora os Blueprints ofereçam uma abordagem "low code", sua crescente complexidade pode constituir um desafio.

Por outro lado, a Unity revela-se mais acessível para programadores principiantes, proporcionando uma interface mais amigável e menos intrincada. Configura-se como uma escolha ideal para aqueles que estão ingressando no desenvolvimento de jogos ou desejam criar seu primeiro jogo. Ainda que não alcance os mesmos padrões de qualidade gráfica da Unreal Engine, a Unity compensa essa lacuna ao ser mais flexível em relação às plataformas.

Assim, a decisão entre Unreal Engine e Unity no desenvolvimento de jogos repousa no equilíbrio entre a qualidade gráfica almejada, a experiência e familiaridade do desenvolvedor com linguagens de programação, bem como a complexidade e as exigências específicas do projeto. Ambas as engines têm sua posição no cenário do desenvolvimento de jogos, atendendo a diferentes perfis de desenvolvedores e objetivos de projeto.

No futuro, pretende-se implementar um projeto inspirado no Antigo Egito na Unity, permitindo uma comparação mais direta entre as duas engines. Este projeto incluirá a adição de novos níveis, cidades, personagens e missões, visando transformar a experiência em um jogo completo, indo além da categoria de mera demonstração.

## REFERÊNCIAS

- Alves, Gelderson Bezerra. Estudo comparativo entre engines de desenvolvimento de jogos 2D / Gelderson Bezerra Alves. Universidade Federal do Ceará 2021;
- BARBOSA, R. A. Análise de Game Engines para Plataformas Móveis. Departamento de Ciências Exatas, Universidade Federal da Paraíba, 2013
- Bushnell, N. Finding the Next Steve Jobs: How to Find, Hire, Keep and Nurture Creative Talent. 2013
- Carmack, J. Doom source code (<https://github.com/id-Software/DOOM>) 1993
- Gregory, J. Game Engine Architecture (2nd ed.). CRC Press 2014.
- Haas, John et al. A History of the Unity Game Engine. Local: Editora, 2014.
- Hergenhahn, B. R., Olson, M. H. An Introduction to Theories of Learning. Pearson 2013.
- Jensen, K. Thor. 25 Years Later: The History of Unreal and an Epic Dynasty. IGN, [S.I.], May 22, 2023. Disponível em:  
<https://www.ign.com/articles/2010/02/23/history-of-the-Unreal-engine>. Acesso em: [15 de junho de 2023].
- Kent, S. L. The Ultimate History of Video Games: From Pong to Pokémon and Beyond. Three Rivers Press 2001.
- Kushner, D. Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture. 2011.
- Kushner, D. Prepare to Meet Thy Doom: And More True Gaming Stories. 2016.
- Lander, B. Game Programming Patterns. Genever Benning 2019.
- Murray, H. J. R. A History of Board Games Other Than Chess. Hacker Art Books, 1978.
- Pantoja, Jean Igor de Queiroz. Estudo comparativo de Motores de jogos no desenvolvimento de jogo 2D para web. / Jean Igor de Queiroz Pantoja. Universidade Federal do Ceará 2022.
- Parlett, David. The Oxford History of Board Games. Oxford University Press, 1999.
- Parlett, David. The Penguin Book of Card Games. Penguin Books, 2008.

- PV, Satheesh. Unreal Engine 4 Game Development Essentials. Reino Unido: Packt Publishing, 2016.
- Sheff, D., Eddy, B. Game Over: How Nintendo Conquered the World. Vintage.) 1999.
- SCHERER, D., BATISTA, D. V., MENDES, A. C. Análise da Evolução de Engines de Jogos. Departamento de Computação, Universidade Estadual da Paraíba (UEPB) 2020.
- SILVA, D. L. Análise de ferramentas e desenvolvimento de jogo para treinamento de paratletas. Universidade Federal de Pernambuco, Centro de Informática, Recife, Julho de 2018.
- The Dot Eaters <https://thedoteaters.com/?bitstory=bitstory-article-1/tennis-for-two>  
Acesso em: [15 de julho de 2023]
- The Dot Eaters <https://thedoteaters.com/?bitstory=bitstory-article-1/spacewar> Acesso em: [15 de julho de 2023]
- Yu Li et al. Artificial Intelligence in Unity Game Engine. 2017. Acesso em: [15 de julho de 2023]
- Zavarise, Giada. The secret history of underdog game engine RPG Maker and how it got its bad reputation. PC Gamer, [S.I.], Oct. 11, 2017. Disponível em: <https://www.pcgamer.com/the-secret-history-of-underdog-game-engine-rpg-maker-and-how-it-got-its-bad-reputation/>. Acesso em: [15 de junho de 2023].
- Unreal Engine <https://www.Unrealengine.com>
- Unity <https://Unity.com>