UM ESTUDO HISTÓRICO SOBRE O DESENVOLVIMENTO DE JOGOS DIGITAIS

Wérik Prado Nascimento; Pedro Henrique Yudi Arakaki, D. E. F; Prof. Dr. Henrique Dezani (Orientador), Profa. Dra. Adriana Alvarenga Dezani (Co-orientadora)

e-mail:
werik.nascimento@fatec.sp.com.br; pedro.arakaki@fatec.sp.com.br.

Resumo: Este Trabalho de Graduação tem como objetivo analisar a evolução das *engines* de jogos digitais, com foco nas linguagens e métodos de programação utilizados desde as primeiras até as mais modernas ferramentas de desenvolvimento. O estudo explora como essas tecnologias influenciam o desempenho, a funcionalidade e a criação de jogos cada vez mais complexos. Além disso, discute-se o impacto do avanço das linguagens de programação no desenvolvimento de jogos imersivos e acessíveis. A pesquisa inclui um estudo de caso comparativo entre as *engines* Unreal Engine e Unity.

Palavras-chave: *Engines* de jogos. Linguagens de programação. Desenvolvimento de jogos. Unreal Engine. Unity.

Abstract: *This Graduation Project aims to analyze the evolution of game engines, focusing on the programming languages and development methods used from the earliest to the most modern tools. The study explores how these technologies influence performance, functionality, and the creation of increasingly complex games. Additionally, the impact of programming languages advancements on immersive and accessible game development is discussed. The research includes a comparative case study between Unreal Engine and Unity, as well as experts interviews to identify future trends.*

Keywords: *Game engines. Programming languages. Game development. Unreal Engine. Unity.*

1 Introdução

A indústria de jogos digitais, nas últimas décadas, experimentou um crescimento vertiginoso, tornando-se uma das maiores áreas de entretenimento e inovação tecnológica. Um dos pilares desse desenvolvimento é o uso de *game engines – softwares* que proporcionam as bases necessárias para a criação de jogos, permitindo que os desenvolvedores se concentrem em aspectos criativos enquanto otimizam os recursos técnicos. Desde as primeiras *engines*, como a DOOM Engine, até ferramentas mais modernas como Unreal Engine e Unity, as linguagens de programação e os métodos de desenvolvimento evoluíram significativamente.

As *game engines* são responsáveis pela execução de tarefas fundamentais, como renderização gráfica, física, inteligência artificial e som, garantindo que um jogo funcione de forma fluida e eficiente. Para realizar essas funções, é crucial o uso de linguagens de programação que se adequem às demandas de performance e complexidade, como C++, C#, Python, entre outras. A escolha da linguagem e do método de programação impacta diretamente no desempenho do jogo, influenciando fatores como o tempo de resposta, o consumo de recursos e a qualidade gráfica (EBERT et al., 2015).

A evolução das *engines* de jogos também reflete o avanço nas técnicas de programação e no próprio *hardware* disponível. Se nos primeiros jogos, as linguagens de baixo nível como Assembly e C eram predominantes, a democratização do desenvolvimento de jogos na última década, especialmente com o surgimento de *engines* acessíveis e multiplataforma, trouxe a adoção de linguagens de alto nível como C# e JavaScript, tornando o processo de desenvolvimento mais ágil e inclusivo.

Dado o contexto de transformação e inovação constantes, o presente trabalho tem como objetivo explorar o desenvolvimento das *engines* de jogos ao longo do tempo, com um foco específico nas linguagens e códigos utilizados pelos desenvolvedores. O estudo abordará como essas ferramentas evoluíram, desde os primeiros motores até as soluções mais modernas e acessíveis, analisando também o impacto das escolhas de linguagem no desempenho e funcionalidade dos jogos.

2 Justificativa

Com o crescimento da indústria de jogos digitais e a expansão das plataformas de desenvolvimento, entender a evolução das *engines* e das linguagens utilizadas é essencial tanto para desenvolvedores quanto para estudiosos da área. A análise detalhada dos códigos, métodos e linguagens fornece uma visão mais clara de como a escolha dessas tecnologias influencia diretamente na criação de jogos eficientes e imersivos. Além disso, compreender essa trajetória ajuda a prever tendências futuras e a identificar áreas onde novas inovações podem surgir, especialmente com o advento de tecnologias como Realidade Virtual (VR), Realidade Aumentada (AR) e inteligência artificial aplicada ao desenvolvimento de jogos.

3 Objetivo(s)

O objetivo deste trabalho é analisar a evolução das *engines* de jogos desde suas origens, com um foco específico nas linguagens e métodos de programação utilizados pelos desenvolvedores. O estudo busca compreender como essas ferramentas influenciaram o desenvolvimento dos jogos ao longo do tempo e de que maneira o avanço das linguagens impactou a criação de jogos cada vez mais complexos e imersivos.

Como objetivos específicos tem-se: explorar as linguagens de programação utilizadas nas principais *engines* de jogos, desde os primeiros motores até os mais modernos; analisar a

evolução dos métodos de desenvolvimento e como esses impactaram a criação de jogos; identificar as tendências futuras no desenvolvimento de *engines* e linguagens de programação aplicadas a jogos digitais; e realizar um estudo de caso comparativo entre duas *engines.*

4 Fundamentação Teórica

O desenvolvimento de jogos digitais envolve uma série de processos complexos que vão desde o *design* conceitual até a implementação técnica. As game *engines* desempenham um papel central nessa estrutura, sendo o conjunto de ferramentas que permitem a criação e execução de jogos. Para compreender a evolução das *engines* de jogos e das linguagens de programação utilizadas, é necessário examinar as principais mudanças tecnológicas, as motivações dos desenvolvedores e os desafios enfrentados ao longo dos anos.

4.1 Definição e Estrutura das Game *Engines*

As *engines* de jogos podem ser definidas como plataformas de *software* que fornecem as funções básicas necessárias para o desenvolvimento de jogos, como simulação física, renderização gráfica, controle de animações e interação com o usuário (EBERT et al., 2015). Elas são responsáveis por abstrair as complexidades do *hardware* subjacente e permitir que os desenvolvedores se concentrem em aspectos mais criativos e específicos do jogo.

Historicamente, as *engines* surgiram como ferramentas internas das empresas de desenvolvimento de jogos. A DOOM Engine (1993), por exemplo, foi uma das primeiras a ser amplamente reconhecida, oferecendo uma base para a criação de jogos de tiro em primeira pessoa, utilizando a linguagem C. Ao longo dos anos, as *engines* foram evoluindo, passando a incluir uma gama mais ampla de recursos, como suporte para gráficos 3D, sistemas de física realistas e motores de som.

4.2 A Evolução das Linguagens de Programação Utilizadas nas Game *Engines*

As linguagens de programação são o coração do desenvolvimento de qualquer *software*, e as *engines* de jogos não são exceção. Cada linguagem escolhida traz consigo vantagens e desafios que afetam diretamente a eficiência do desenvolvimento, o desempenho dos jogos e a capacidade de inovação dos desenvolvedores.

4.2.1 Primeiras Linguagens Utilizadas: Assembly e C

Nos primórdios dos jogos digitais, o desenvolvimento era realizado predominantemente em Assembly ou C, linguagens de baixo nível que ofereciam controle preciso sobre o *hardware*. Isso era essencial em uma época em que os recursos de processamento eram limitados, e cada bit de memória precisava ser utilizado com eficiência (NANDLURKAR; SHINDE, 2017). A linguagem C permitia aos desenvolvedores escreverem códigos relativamente otimizados, enquanto ainda mantinha um nível de abstração que facilitava a criação de jogos mais complexos.

A DOOM *Engine*, por exemplo, foi amplamente codificada em C, uma escolha comum na década de 1990 para *engines* de alto desempenho. Essa linguagem proporcionava velocidade e um gerenciamento eficiente de memória, dois fatores críticos para a criação de jogos em uma época em que os consoles e computadores possuíam capacidades de processamento limitadas (ZINK et al., 2016).

4.2.2 A Ascensão do C++ e o Paradigma Orientado a Objetos

Com o advento da linguagem C++, que introduziu o conceito de programação orientada a objetos (POO), o desenvolvimento de jogos passou por uma transformação significativa. O C++ combina o desempenho de baixo nível do C com recursos de abstração e modularidade que permitem a criação de sistemas mais organizados e fáceis de gerenciar (NIELSON, 2016).

A Unreal Engine, lançada em 1998, foi uma das primeiras *engines* a adotar amplamente o C++, aproveitando a flexibilidade e a eficiência da linguagem para criar jogos de alta performance com gráficos complexos. A orientação a objetos facilitou a implementação de mecânicas de jogo mais sofisticadas, como a interação com personagens e ambientes dinâmicos (GOLDSCHMITT, 2020).

4.2.3 A Popularização de C# com Unity

Com o surgimento da Unity Engine, a linguagem C# começou a ganhar destaque no desenvolvimento de jogos. Lançada em 2005, a Unity foi projetada para ser uma *engine* multiplataforma, permitindo o desenvolvimento para diversas plataformas com um único código-base. O C# oferece uma sintaxe mais simples e recursos de gerenciamento automático de memória (*garbage collection*), o que facilita o desenvolvimento rápido e eficiente, mesmo para desenvolvedores menos experientes (MIRANDA; ALEXANDRE, 2021).

A simplicidade do C# e sua forte integração com a *engine* Unity permitiram que essa plataforma se tornasse extremamente popular entre desenvolvedores independentes e pequenas equipes de desenvolvimento, democratizando o processo de criação de jogos (FRITZ; TROUTMAN, 2018).

4.2.4 Linguagens Scripting e a Flexibilidade do Desenvolvimento

Além das linguagens principais como C++ e C#, muitas *engines* de jogos modernas utilizam linguagens de *script* para facilitar o desenvolvimento de certas funcionalidades. Linguagens como Lua e Python são amplamente usadas para programação de lógica de jogo, inteligência artificial e sistemas dinâmicos, permitindo que os desenvolvedores ajustem e modifiquem o comportamento do jogo sem precisar recompilar o código principal (LEE; ROBERTSON, 2019).

Linguagens de *script* oferecem uma flexibilidade maior ao desenvolvimento, permitindo rápidas iterações e ajustes sem comprometer a performance geral do jogo. A CryEngine, por exemplo, utiliza Lua como sua principal linguagem de *script*, e a Unreal Engine adotou o Blueprint, uma linguagem visual de *script* que oferece uma interface amigável para a criação de lógica de jogo sem a necessidade de codificação direta.

4.3 As Game *Engines* Modernas e a Multiplicidade de Plataformas

As *engines* de jogos modernos são projetadas para atender uma multiplicidade de plataformas, desde consoles até dispositivos móveis e jogos baseados em navegadores. Esse cenário introduziu novos desafios em termos de otimização e uso eficiente de recursos. *Engines* como Unreal Engine 5 e Unity são exemplos de plataformas que oferecem suporte para o desenvolvimento multiplataforma, permitindo que os jogos sejam desenvolvidos uma única vez e exportados para diferentes dispositivos.

Com a crescente demanda por gráficos de alta qualidade e a necessidade de otimizar o desempenho, especialmente em dispositivos móveis, as linguagens de programação e os métodos de desenvolvimento nas *engines* têm evoluído para suportar novas tecnologias como *ray tracing*, inteligência artificial e realidade virtual. O uso de C++ continua a ser dominante para componentes críticos, enquanto linguagens de *script* como Python e C# oferecem flexibilidade e simplicidade para os elementos não essenciais (JEFFERSON et al., 2022).

4.4 O Futuro das *Engines* de Jogos e das Linguagens de Programação

O futuro das *engines* de jogos parece estar intimamente ligado ao avanço da tecnologia e à integração de novos paradigmas de desenvolvimento, como inteligência artificial e machine learning. A introdução de motores de jogos baseados em IA, como o Godot Engine, mostra que a busca por soluções mais ágeis e inteligentes no desenvolvimento de jogos continua em constante evolução (TARASOV et al., 2023).

5 Trabalhos Similares

5.1 Análise da evolução de *engines* de jogos

SCHERER, Daniel; VENTURA, Daniele Batista; MENDES, Aline de Cantalice. **Análise da evolução de *engines* de jogos**. O trabalho analisa o desenvolvimento e a evolução das *engines* de jogos ao longo dos anos, com foco em *frameworks* como *Unity* e *Unreal*. A pesquisa abrange desde as linguagens de programação utilizadas inicialmente, como *Assembly*, até a integração de bibliotecas gráficas modernas como *DirectX* e *OpenGL*. Além disso, o estudo explora como o avanço dessas *engines* tornou o desenvolvimento de jogos mais acessível, permitindo que desenvolvedores se foquem no design e na jogabilidade, em vez de aspectos técnicos complexos. A pesquisa conclui que a evolução das *engines* facilitou o desenvolvimento de jogos mais realistas e complexos, com menores requisitos técnicos e maior suporte a diferentes plataformas.

5.2 Estudo comparativo entre *engines* de desenvolvimento de jogos 2D

ALVES, Gelderson Bezerra. **Estudo comparativo entre *engines* de desenvolvimento de jogos 2D**. Este artigo apresenta uma análise comparativa entre duas das principais *engines* usadas no desenvolvimento de jogos 2D: *Unity* e *Construct 3*. O autor destaca que, enquanto a *Unity* proporciona um maior controle e poder de desenvolvimento, a *Construct 3* se destaca pela simplicidade e acessibilidade para iniciantes. A pesquisa foca nos aspectos técnicos dessas *engines*, incluindo a curva de aprendizado, suporte à comunidade e os recursos de exportação para múltiplas plataformas, como *HTML5*, *Android* e *iOS*. O artigo é especialmente útil para desenvolvedores independentes que desejam escolher a *engine* mais adequada para seus projetos.

5.3 Estudo comparativo de motores de jogos no desenvolvimento de um jogo 2D para *web*

PANTOJA, Jean Igor de Queiroz. **Estudo comparativo de motores de jogos no desenvolvimento de um jogo 2D para *web***. Quixadá: Universidade Federal do Ceará, 2022. O estudo aborda as diferenças entre motores de jogos, focando no desenvolvimento de um jogo 2D para plataformas *web*. Utilizando métricas como tempo de carregamento, uso de *CPU* e *RAM*, o autor compara *Unity* e *Godot* para avaliar qual oferece melhor desempenho e facilidade de uso para desenvolvedores iniciantes. O estudo conclui que a *Unity* se destaca em termos de desempenho, enquanto a *Godot* apresenta uma curva de aprendizado mais suave, sendo indicada para iniciantes.

6 Metodologia

6.1 *Engines* da Década de 1990

A primeira vez que teve indícios de *engine* em 1962 com SpaceWar, embora não utilizasse nenhum tipo de *engine* de jogo, porém ela estabeleceu uma base de conceito de jogo reutilizável. A primeira *engine* surgiu realmente em 1993, com a DOOM Engine (Figura 1) feita pela empresa ID Software e seus criadores John Carmack e John Romero, este sendo um

motor de jogo reutilizável, ou seja, fornece as ferramentas e funcionalidades básicas necessárias para o desenvolvimento de jogos eletrônicos, permitindo que desenvolvedores criem diferentes jogos sem precisar escrever tudo do zero. E vale frisar que essa *engine* foi feita em C e algumas de suas partes em Assembly.

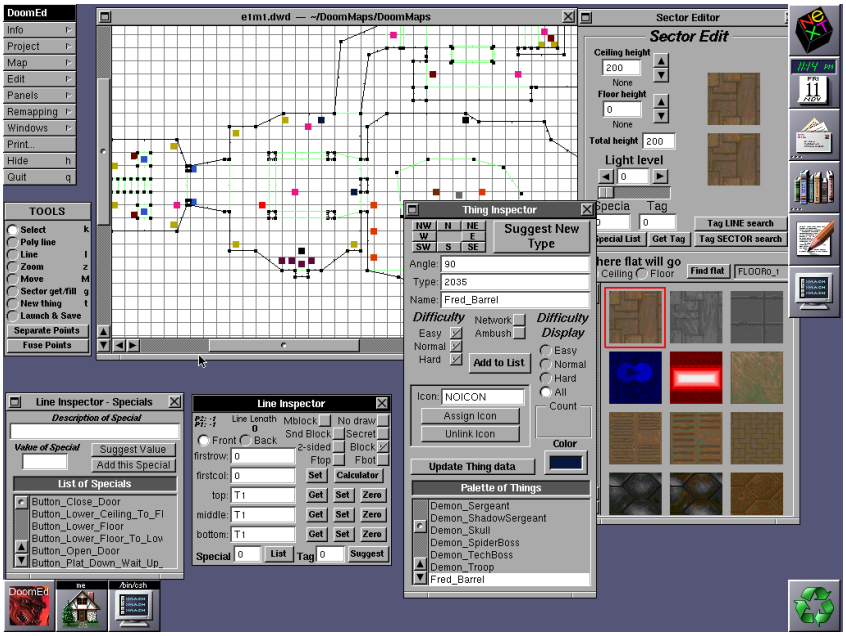
Uma inovação que Carmack fez foi deixar os *players* poderem inserir *mods* no jogo sem quebrar o *game* como citado no livro de KUSHNER:

> "For *Doom*, Carmack organized the data so players could replace sound and graphics in a nondestructive manner. He created a subsystem that separated the media data, called WADs (an acronym, suggested by Tom Hall, it stood for Where's All the Data?), from the main program. Every time someone booted up the game, the program would look for the WAD file of sounds and images to load in. This way, someone could simply point the main program to a different WAD without damaging the original contents." (KUSHNER, 2003, p. 38)

A evolução histórica das *engines* de jogos pode ser dividida em marcos tecnológicos significativos. As primeiras *engines*, como a DOOM Engine e a Quake Engine, revolucionaram o mercado ao introduzir técnicas inovadoras, como a renderização pseudo-3D e total em 3D. Desenvolvidas em linguagens como C e Assembly, essas *engines* permitiram controle detalhado sobre o hardware, estabelecendo uma base sólida para jogos mais complexos. Nos anos 2000, o avanço do C++ impulsionou o surgimento de *engines* como a Unreal Engine (Figura 2) e a CryEngine, que aprimoraram os gráficos 3D e introduziram física realista. O uso do C++ possibilitou a criação de sistemas modulares e mais eficientes, atendendo à crescente demanda por qualidade e performance.

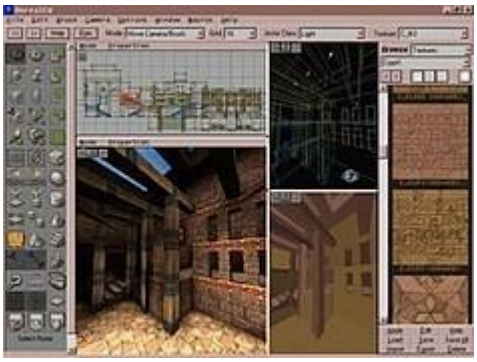
Nas últimas décadas, as engines modernas, como a Unity e a Unreal Engine 4, trouxeram avanços significativos em acessibilidade e funcionalidade. A Unity popularizou o uso do C#, oferecendo uma solução multiplataforma que atraiu especialmente desenvolvedores independentes. Já a Unreal Engine 4 incorporou o sistema Blueprints, uma ferramenta de programação visual que simplificou o desenvolvimento para usuários sem experiência avançada em codificação. Essas inovações democratizaram o desenvolvimento de jogos, tornando-o mais acessível e diversificado, enquanto continuam a definir tendências na indústria.

Figura 1 - DOOM Engine



Fonte: doomwiki.org. (2024)

Figura 2 - Unreal Engine 1



Fonte: en.wikipedia.org. (2024)

6.2 *Engines* da Década de 2000

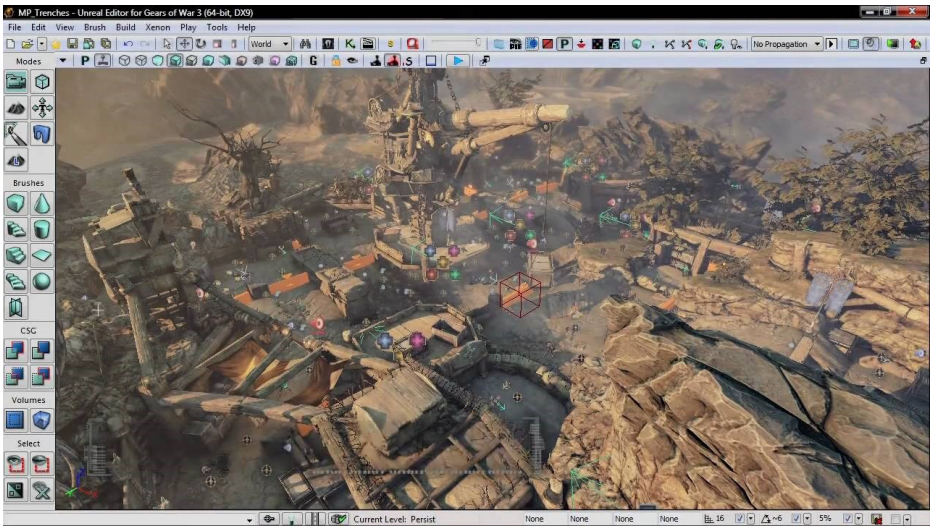
A década de 2000 foi marcada por avanços significativos nas *engines* de jogos, que revolucionaram a experiência de *gameplay*. Os gráficos 3D avançaram de forma expressiva, permitindo a criação de ambientes vastos e realistas em jogos como Grand Theft Auto III e Red Dead Redemption. Essas melhorias foram possibilitadas pelo uso de tecnologias de alta definição, que elevaram o nível de imersão visual. Paralelamente, os jogos em "mundo aberto" ganharam destaque, oferecendo liberdade de exploração e narrativa não linear. GTA III, em particular, foi pioneiro nesta abordagem, influenciando profundamente o design de jogos subsequentes (ACADEMY OF ART UNIVERSITY, 2024).

Outro marco foi a acessibilidade proporcionada pelas *engines* modernas para desenvolvedores independentes. Unity (Figura 4) e Unreal Engine (Figura 3) emergiram como ferramentas essenciais para estúdios menores, democratizando o desenvolvimento de jogos. A

Unity, com sua interface amigável e suporte multiplataforma, permite que desenvolvedores indie criassem títulos de alta qualidade mesmo com recursos limitados. Além disso, a integração de motores de física, como o Havok, trouxe um nível de realismo sem precedentes, permitindo interações mais precisas e naturais com os objetos no ambiente virtual, o que aprimorou ainda mais a imersão nos *games* (GAMEOPEDIA, 2024).

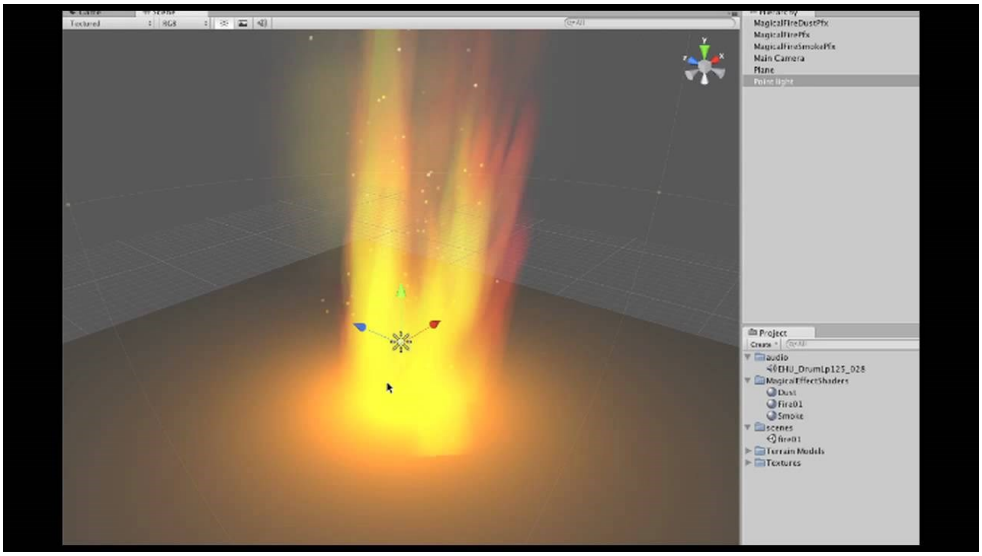
Entre as inovações das *engines*, o UnrealScript, introduzido pela Epic Games na Unreal Engine, destacou-se como uma linguagem de *script* orientada a objetos voltada para facilitar o desenvolvimento de *gameplay*. Com sintaxe semelhante a Java e C++, era acessível a programadores experientes, mas oferecia funcionalidades específicas para jogos, como suporte a herança, referências de objetos e integração robusta entre elementos do jogo. Sua eficiência no desenvolvimento, permitindo alterações rápidas sem recompilar a *engine*, foi essencial para a criação de novas mecânicas. Na Unreal Engine 4, o UnrealScript foi substituído pelo sistema Blueprints, que aprimorou ainda mais a acessibilidade ao adotar uma abordagem visual para a programação (EPIC GAMES, 2024).

Figura 3 - Unreal Engine 3



Fonte: Extraído de Unreal Engine 3 : Features Showcase. (2011, 2min53s)

Figura 4 - Unity Engine 3.0



Fonte: Extraído de #Unity3D Game Engine 3.0 - New Features Part1. (2010, 1s)

6.3 *Engines* da Década de 2010

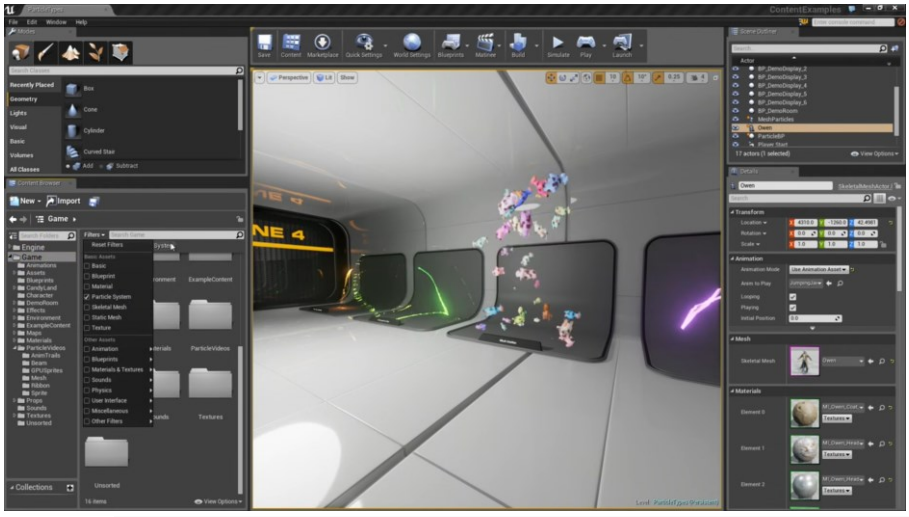
Desde 2010, as *engines* de jogos passaram por inovações que transformaram a forma como os jogos são desenvolvidos. A Unreal Engine 4 (Figura 5), lançada em 2014, introduziu tecnologias avançadas como o *real-time ray tracing*, que proporcionou gráficos mais realistas, com reflexos e sombras dinâmicas. Isso elevou o nível de imersão nos jogos e consolidou a Unreal Engine como uma das plataformas mais poderosas da indústria. Além disso, o sistema Blueprints, uma linguagem de programação visual, tornou o desenvolvimento mais acessível, permitindo que até mesmo equipes com pouca experiência técnica pudessem criar projetos complexos e sofisticados. Esse conjunto de ferramentas colocou a Unreal Engine na vanguarda do desenvolvimento de jogos AAA e produções cinematográficas de alta qualidade. A Unity (Figura 6), por sua vez, se destacou como a *engine* preferida para o desenvolvimento de jogos independentes e para dispositivos móveis, graças à sua interface acessível e ao suporte multiplataforma. Lançada com o objetivo de democratizar o desenvolvimento de jogos, a Unity também integrou ferramentas avançadas para realidade virtual (VR) e realidade aumentada (AR), tornando-se uma escolha ideal para experiências interativas em múltiplas plataformas. Sua flexibilidade e suporte à comunidade ajudaram a estabelecer um ecossistema robusto, permitindo que pequenos desenvolvedores criassem jogos de alta qualidade com recursos limitados (JEFFERSON et al., 2022).

O CryEngine, desenvolvido pela Crytek, destacou-se por seu realismo visual impressionante e não pertence nem à Unity nem à Unreal Engine. Este motor de jogo é reconhecido por seu desempenho gráfico, utilizado em títulos icônicos como Crysis 3. No entanto, sua complexidade e a curva de aprendizado íngreme limitaram seu uso por estúdios menores. Em um esforço para atrair desenvolvedores independentes, a Crytek introduziu um modelo de pagamento baseado em doações, mas o alto nível técnico necessário para operar o CryEngine permaneceu um desafio para muitos desenvolvedores. Apesar disso, ele continuou a ser uma referência em gráficos avançados e física realista.

O Frostbite Engine, criado pela DICE e amplamente utilizado pela Electronic Arts (EA), foi otimizado para gráficos detalhados e física avançada, consolidando-se como uma escolha predominante em franquias como Battlefield e FIFA. Apesar de suas qualidades gráficas, o

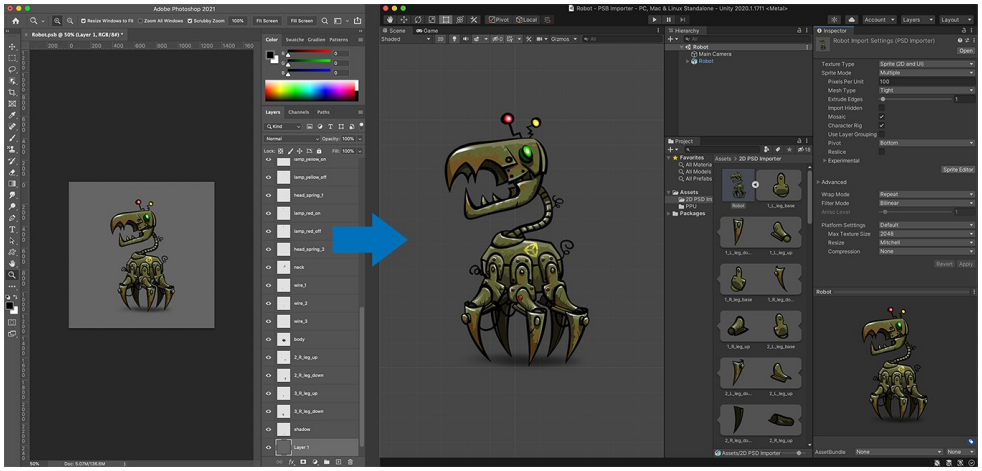
Frostbite enfrentou críticas pela dificuldade de adaptação a outros gêneros, especialmente RPGs, o que impactou negativamente o desenvolvimento de jogos como Mass Effect: Andromeda. Esse desafio reflete as limitações do motor em contextos que requerem maior flexibilidade na criação de mecânicas de jogo complexas. O Anvil Engine, desenvolvido pela Ubisoft, também se destacou durante a década. Inicialmente lançado como Scimitar Engine em 2007 e posteriormente renomeado para Anvil, ele foi projetado especificamente para atender às demandas das franquias da Ubisoft. Utilizado em títulos como Assassin 's Creed Origins e Valhalla, o Anvil permitiu a criação de mundos abertos vastos e detalhados, com sistemas avançados de iluminação e animação. Este motor tornou-se essencial para a franquia, entregando experiências imersivas e narrativas ricas. Embora não tenha a versatilidade de engines como Unity e Unreal, o Anvil é um exemplo de motor especializado que atende às necessidades específicas de seu desenvolvedor. Cada uma dessas engines desempenhou um papel significativo no avanço da indústria durante a década de 2010. A Unreal Engine e a Unity lideraram em termos de acessibilidade e inovação, enquanto o CryEngine, Frostbite e Anvil consolidaram seus lugares como motores de jogo especializados, adaptados às necessidades de grandes estúdios e projetos ambiciosos. Com suporte a tecnologias emergentes, como *real-time ray tracing*, VR e AR, essas *engines* não apenas moldaram o desenvolvimento de jogos durante a década, mas também estabeleceram as bases para o futuro da indústria, oferecendo experiências cada vez mais imersivas e realistas.

Figura 5 - Unreal Engine 4



Fonte: Extraído de Into to Cascade: Particle Terminology | 01 | v4.2 Tutorial Series | Unreal Engine. (2014, 1min48s)

Figura 6 - Unity 5.0



Fonte: Publicado em *2D* PSDImporter 5.0 released for Unity 2021.1. (2021)

6.4 *Engines* da Década de 2020

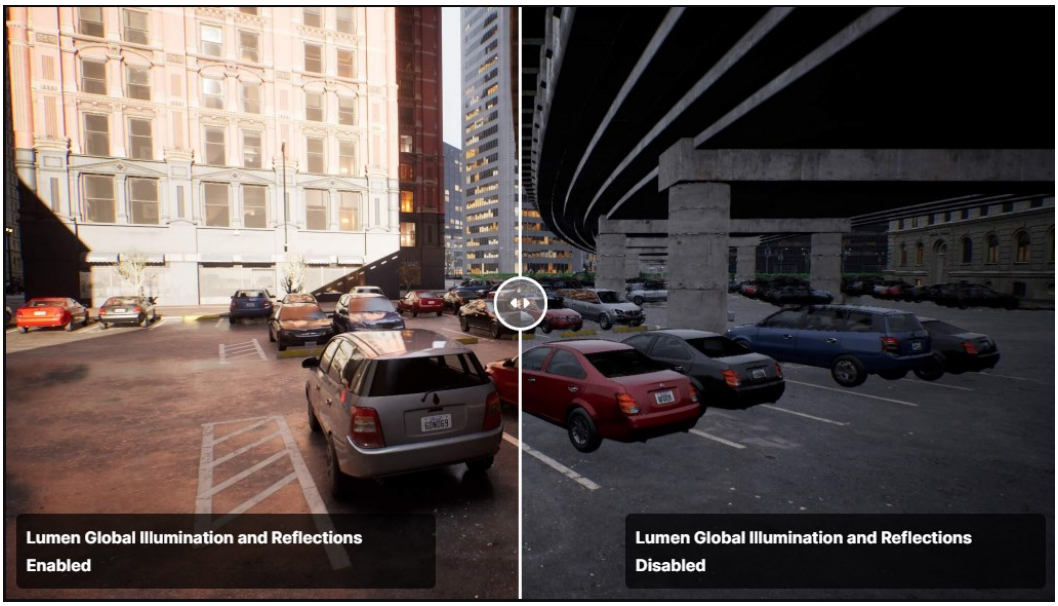
As inovações tecnológicas introduzidas em 2020 marcaram uma evolução significativa nas engines de jogos. A Unreal Engine 5 trouxe a tecnologia Nanite, que permitiu a criação de ambientes com bilhões de polígonos renderizados em tempo real, eliminando a necessidade de simplificação dos modelos e aumentando a qualidade visual. Outra inovação da Unreal Engine 5 foi o Lumen (Figura 7), que trouxe iluminação global em tempo real, permitindo uma iluminação dinâmica e mais precisa sem a necessidade de cálculos pré-processados. Além disso, tanto a Unreal Engine quanto a Unity adotaram o *real-time ray tracing*, permitindo gráficos realistas com reflexos e sombras detalhadas em tempo real, o que elevou a imersão dos ambientes interativos (EPIC GAMES DEVELOPER COMMUNITY, 2024).

No campo da realidade aumentada e virtual, a Unity (Figura 8) se destacou por continuar a expandir seu suporte a AR e VR, oferecendo ferramentas mais integradas para a criação de experiências imersivas, sendo especialmente eficaz para o desenvolvimento de jogos e simulações. Por outro lado, o Frostbite Engine aprimorou sua capacidade de criar ambientes dinâmicos e destrutíveis, uma característica notável em grandes jogos de batalha multijogador como Battlefield 2042, onde a destruição de ambientes em tempo real se tornou um dos principais atrativos do jogo (MIRANDA; ALEXANDRE, 2021).

Entre os destaques da época, a Unreal Engine 5 se tornou a principal escolha para desenvolvedores que buscavam gráficos de nova geração, com títulos como Fortnite migrando para essa versão devido ao impacto das tecnologias Nanite e Lumen. Já a Unity manteve sua posição de liderança no mercado independente, oferecendo flexibilidade e acessibilidade para a criação de títulos inovadores em AR e VR, além de facilitar o desenvolvimento multiplataforma. A Decima Engine, utilizada em Horizon Zero Dawn e Death Stranding, também se destacou por renderizar grandes ambientes abertos com gráficos detalhados, especialmente voltados para a nova geração de consoles. Por fim, o CryEngine, apesar de sua menor popularidade, continuou relevante no desenvolvimento de jogos com foco em gráficos

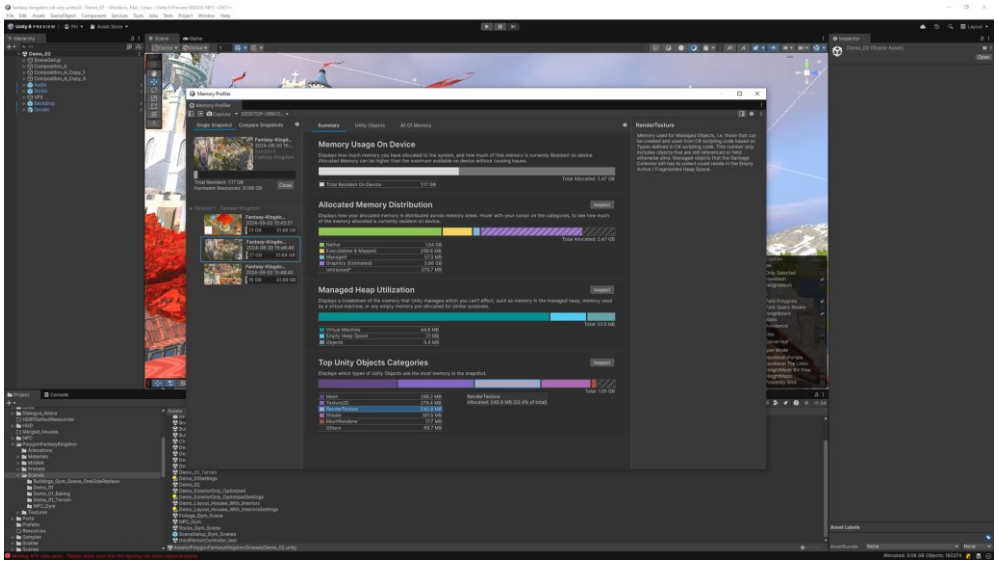
realistas e fidelidade visual, realizando aprimoramentos contínuos para suportar a nova geração de gráficos (TROUTMAN, 2018).

Figura 7 - Lumen na Unreal Engine 5



Fonte: Publicado em Unreal Engine 5.0 Release Notes. (2023)

Figura 8 - Unity 6.0



Fonte: unity.com (2024)

7 Desenvolvimento

A evolução das *engines* de jogos ao longo dos anos reflete um processo de crescente complexidade e sofisticação no desenvolvimento de jogos. As primeiras *engines*, como a DOOM Engine e a Quake Engine, estabeleceram as bases para gráficos e mecânicas de jogo simples, enquanto versões mais recentes, como a Unreal Engine 4 e a Unity, oferecem ferramentas avançadas, como o *real-time ray tracing*, suporte para realidade virtual (VR) e aumentada (AR), e a capacidade de criar jogos multiplataforma. A transição para *engine* que utilizam linguagens como C++ e C# representa um avanço significativo em termos de modularidade, desempenho e acessibilidade, permitindo aos desenvolvedores maior flexibilidade na criação de jogos para diferentes tipos de dispositivos e públicos.

A comparação entre a Unreal Engine e a Unity destaca as vantagens específicas de cada motor. A Unreal Engine se destaca principalmente em grandes estúdios de desenvolvimento, sendo amplamente adotada por sua capacidade de criar gráficos de alta qualidade e proporcionar um desempenho robusto. Títulos como Fortnite e Gears 5 são exemplos de como a Unreal Engine pode levar os gráficos a um nível de realismo impressionante, especialmente com inovações como o Nanite e o Lumen. Em contraste, a Unity é uma escolha popular entre desenvolvedores independentes e pequenas equipes devido à sua flexibilidade, interface acessível e suporte multiplataforma. Com títulos como Monument Valley e Super Mario Run, a Unity se destaca por permitir a criação de jogos inovadores em AR e VR, além de sua capacidade de adaptar jogos para uma variedade de plataformas, incluindo dispositivos móveis e consoles.

A escolha das linguagens de programação tem um impacto direto no desenvolvimento de jogos. Linguagens como C++, utilizadas na Unreal Engine, oferecem um controle preciso sobre o *hardware*, permitindo otimizações essenciais para jogos com altos requisitos de desempenho, mas também exigem um conhecimento mais profundo do sistema, aumentando a complexidade do desenvolvimento. Em contrapartida, linguagens como C#, empregadas na Unity, proporcionam uma curva de aprendizado mais suave, permitindo um desenvolvimento mais rápido. A simplicidade da sintaxe e o gerenciamento automático de memória facilitam o trabalho de desenvolvedores com menos experiência e reduzem a incidência de erros comuns, o que acelera o processo de criação de jogos, especialmente para equipes menores ou desenvolvedores independentes.

As tendências emergentes no desenvolvimento de *engines* de jogos indicam um futuro cada vez mais integrado com novas tecnologias. A crescente popularidade da realidade virtual (VR) e da realidade aumentada (AR) está impulsionando a necessidade de *engines* capazes de oferecer suporte otimizado para essas tecnologias, criando novas possibilidades para experiências imersivas. A Unity, com suas ferramentas integradas para VR e AR, já está se posicionando como uma plataforma importante para o desenvolvimento de jogos imersivos, enquanto a Unreal Engine também está se adaptando rapidamente a essas tendências, garantindo que os desenvolvedores possam explorar novas formas de interação e narrativa nos jogos do futuro.

8 Resultados e Discussões

A pesquisa revelou que as *engines* evoluíram significativamente ao longo das décadas, desde as primeiras versões, como a DOOM Engine, até as mais modernas, como Unreal Engine 4 e Unity, oferecendo maior sofisticação técnica e suporte multiplataforma. A popularização de ferramentas como a Unity permitiu o desenvolvimento mais acessível e democrático de jogos, principalmente por desenvolvedores independentes e pequenas equipes.

Além disso, a análise das linguagens de programação mostrou que elas desempenham um papel fundamental no equilíbrio entre desempenho e facilidade de desenvolvimento, com linguagens como C++ proporcionando otimizações avançadas, enquanto C# e linguagens de *script* oferecem flexibilidade e rapidez no processo de criação. As tendências emergentes, como a integração de IA, machine learning, realidade virtual (VR) e realidade aumentada (AR), indicam que as *engines* de jogos continuarão a evoluir, trazendo novas possibilidades para a criação de experiências imersivas.

9 Conclusões

Com isso, o objetivo do estudo foi plenamente alcançado, oferecendo uma compreensão abrangente das mudanças tecnológicas que moldaram e continuam a moldar a indústria de desenvolvimento de jogos.

Referências

UNITY3D GAME ENGINE 3.0 - NEW FEATURES PART1. Vídeo. 1s. Publicado pelo canal TornadoTwins.7 de jul. de 2010. Disponível em: https://www.youtube.com/watch?v=kcMsyZ91Cgs. Acesso em: 1 nov. 2024.

ACADEMY OF ART UNIVERSITY. *Game development and new media of the 2000s: a decade of innovation*. 2024. Disponível em: https://blog.academyart.edu. Acesso em: 17 out. 2024.

ALVES, Gelderson Bezerra. **Estudo comparativo entre engines de desenvolvimento de jogos 2D**. 2021. Monografia (Graduação em Engenharia de Software) – Universidade Federal do Ceará, Quixadá. Disponível em: https://repositorio.ufc.br/bitstream/riufc/58827/1/2021_tcc_gbalves.pdf. Acesso em: 10 set. 2024.

BEST, Martin. Unity 6 está aqui: Veja o que há de novo. 17 de out. de 2024. Disponível em: https://unity.com/blog/unity-6-features-announcement. Acesso em: 1 nov. 2024.

EBERT, Jane et al. The role of programming languages in game engine development. **Journal of Game Development**, v. 14, n. 2, p. 103-121, 2015.

EPIC GAMES. *UnrealScript Foundations*. **Unreal Engine Documentation**. 2024. Disponível em: https://api.unrealengine.com. Acesso em: 17 out. 2024.

EPIC GAMES DEVELOPER COMMUNITY. *Unreal Engine programming and scripting*. **Epic Games**. 2024. Disponível em: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-programming-and-scripting. Acesso em: 17 out. 2024.

FRITZ, Evan; TROUTMAN, Jackson. **Game development with Unity and C#**. New York: O'Reilly Media, 2018.

GAMEOPEDIA. *Game engines: all you need to know*. 2024. Disponível em: https://www.gameopedia.com. Acesso em: 17 out. 2024.

GOLDSCHMITT, Simon. Advanced game mechanics in Unreal Engine: an overview of C++ in game development. **GameDev Journal**, v. 8, n. 1, p. 88-112, 2020.

INTO TO CASCADE: PARTICLE TERMINOLOGY | 01 | V4.2 TUTORIAL SERIES | UNREAL ENGINE. Vídeo. 1min48s. Publicado pelo canal Unreal Engine. 8 de jul. de 2014. Disponível em: https://www.youtube.com/watch?v=OXK2Xbd7D9w. Acesso em: 1 nov. 2024.

JEFFERSON, Max et al. Game engines and programming trends for 2023. **Tech Innovations in Gaming**, v. 10, n. 4, p. 217-239, 2022.

KUSHNER, David. **Masters of Doom: how two guys created an empire and transformed pop culture**. New York: Random House, 2003.

LEE, Cynthia; ROBERTSON, Mark. The rise of scripting languages in game development. **Game Programming Quarterly**, v. 17, n. 3, p. 145-167, 2019.

MIRANDA, Julia; ALEXANDRE, Paul. Unity and C#: the democratization of game development. **GameDev Today**, v. 6, n. 4, p. 66-90, 2021.

NANDLURKAR, Vishal; SHINDE, Aashish. Assembly and C: foundations of early game engines. **Computer Science Today**, v. 23, n. 3, p. 198-213, 2017.

NIELSON, Jacob. **Introduction to C++ for game development**. Cambridge: MIT Press, 2016.

PANTOJA, Jean Igor de Queiroz. **Estudo comparativo de motores de jogos no desenvolvimento de um jogo 2D para web**. 2022. Monografia (Graduação em Engenharia de Software) – Universidade Federal do Ceará, Quixadá. Disponível em: https://repositorio.ufc.br/bitstream/riufc/70844/1/2022_tcc_jiqpantoja.pdf. Acesso em: 10 set. 2024.

PCGAMESN. *From Unreal Engine to Unity: the most important game engines ever created*. 2024. Disponível em: https://www.pcgamesn.com. Acesso em: 17 out. 2024.

RUSTUM. *2D* PSDImporter 5.0 released for Unity 2021.1. 1 de mai. de 2021. Disponível em: https://discussions.unity.com/t/2d-psdimporter-5-0-released-for-unity-2021-1/840313. Acesso em: 1 nov. 2024.

SCHERER, Daniel; VENTURA, Daniele Batista; MENDES, Aline de Cantalice. **Análise da evolução de engines de jogos**. Campina Grande: Universidade Estadual da Paraíba, 2021. Disponível em: https://repositorio.ufc.br/bitstream/riufc/70844/1/2022_tcc_jiqpantoja.pdf. Acesso em: 10 set. 2024.

TARASOV, Dimitri et al. AI-based game engines: the next frontier. **International Journal of Game Studies**, v. 11, n. 2, p. 101-120, 2023.

UNREAL DOCUMENTATION. Unreal Engine 5.0 Release Notes. Disponível em: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5.0-release-notes?application_version=5.0. Acesso em: 1 nov. 2024.

UNREAL ENGINE 3 : FEATURES SHOWCASE. Vídeo. 2min53s. Publicado pelo canal ACROFAN. 22 de mai. de 2023. Disponível em: https://www.youtube.com/watch?v=p9YDfrPbQWY. Acesso em: 1 nov. 2024.

WIKIPEDIA.Unreal Engine 1.13 de nov. de 2024. Disponível em: https://en.wikipedia.org/wiki/Unreal_Engine_1. Acesso em: 1 nov. 2024.

XYMPHBOT. Level editor. 2 de dez. de 2024. Disponível em: https://doomwiki.org/wiki/Level_editor. Acesso em: 1 nov. 2024.

ZINK, James; CARLSON, Alex; LEWIS, Danielle. The role of C in early game engines. **Journal of Computational Design**, v. 12, n. 1, p. 45-67, 2016.