Curvas de aprendizagem em *game engines*: percepções de um desenvolvedor novato em jogos

Learning curves in game engines: insights from a novice game developer

Arthur de Meira Lins[1], **Fabrizio Honda**[1], **Fernanda Pires** [1], **Marcela Pessoa**[1]

[1] Escola Superior de Tecnologia - Universidade do Estado do Amazonas (EST-UEA)
ThinkTEd Lab - Pesquisa, Desenvolvimento e Inovação em Tecnologias Emergentes

{amadml.snf22,fpires,msppessoa}@uea.edu.br,

fabrizio.honda@icomp.ufam.edu.br

Abstract. Introduction: *Choosing a game engine for game development is challenging, especially for beginners.* ***Objective:*** *This article analyzes the perceptions of a novice developer, an undergraduate computing student, in implementing mechanics in the game engines Unity, Godot, and Pygame, and provides practical tips for beginners in the field.* ***Methodology:*** *We conducted a case study, whose data were collected through an online form.* ***Results:*** *The novice developer considered Godot the simplest and most intuitive game engine, with the shortest development time. Unity proved viable, but had a steep learning curve, while Pygame, although more complex, allowed greater practical application of computing knowledge.*
Keywords *game engines, game development, novice, perceptions, case study*

Resumo. Introdução: *A escolha de uma game engine para o desenvolvimento de jogos é um desafio, sobretudo para iniciantes.* ***Objetivo:*** *Este artigo analisa as percepções de um desenvolvedor novato, graduando em computação, na implementação de mecânicas nas game engines Unity, Godot e Pygame, e fornece dicas práticas para iniciantes na área.* ***Metodologia:*** *Conduziu-se um estudo de caso, cujos dados foram coletados por meio de um formulário online.* ***Resultados:*** *A Godot foi tida como a game engine mais simples e intuitiva, com menor tempo de desenvolvimento. A Unity mostrou-se viável, mas com uma alta curva de aprendizagem, enquanto o Pygame, embora mais complexo, proporcionou maior aplicação prática dos conhecimentos de computação.*
Palavras-Chave *game engines, desenvolvimento de jogos, novato, percepções, estudo de caso*

1. Introdução

Os jogos digitais ultrapassam o papel de entretenimento, contribuindo para o desenvolvimento de habilidades cognitivas como raciocínio lógico, resolução de problemas e coordenação motora [Gee 2003]. Além disso, sua construção envolve competências multidisciplinares, como programação, *design* de interação, prototipagem iterativa e gerenciamento de projetos [Maloney et al. 2008]. Por esse motivo, o processo de desenvolvimento destes artefatos tem sido reconhecido como uma abordagem pedagógica eficaz, capaz de promover aprendizagem ativa e engajamento

[Politowski et al. 2021]. Esse cenário tem fomentado a indústria de jogos, que hoje em dia é mais rentável que os setores de música e cinema juntos [Portal News 2025].

Entretanto, para iniciantes, o desenvolvimento de um jogo é uma tarefa desafiadora. Além de dominar fundamentos de programação, é preciso assimilar, muitas vezes de maneira simultânea, conceitos como *game loops*, física, detecção de colisão, gerenciamento de estados e *level design* [Sobota e Pietriková 2023]. Soma-se a isso a dificuldade de selecionar a ferramenta de desenvolvimento mais adequada, diante da ampla gama de *game engines* disponíveis [Politowski et al. 2021]. Cada alternativa possui distintas curvas de aprendizado, conjuntos de recursos e experiências de uso, o que torna a escolha desafiadora para quem ainda está consolidando seus conhecimentos.

Esse cenário torna-se ainda mais crítico em estúdios independentes, onde desenvolvedores frequentemente acumulam múltiplas funções e dispõem de recursos escassos e prazos curtos [Freeman e McNeese 2019]. Dessa forma, a *game engine* escolhida influencia diretamente o tempo dedicado por cada desenvolvedor para concluir uma tarefa (custo em homem-hora), a sequência e organização das etapas de produção do jogo (*pipeline*) e, consequentemente, a capacidade do estúdio de entregar projetos dentro do prazo e do orçamento disponível [Engström 2020, Politowski et al. 2021]. Portanto, em alguns casos, a curva de aprendizado de uma *game engine* pode ser um fator decisivo para a continuidade de um projeto [Freeman e McNeese 2019].

Diante disso, considerando a dificuldade dos desenvolvedores novatos ao ingressar na indústria de jogos e a diversidade de *game engines* e bibliotecas, este trabalho apresenta como questão de pesquisa (QP): "Como as percepções de um desenvolvedor novato ao utilizar Unity, Godot e Pygame podem orientar a escolha de *game engines* por iniciantes na área de desenvolvimento de jogos?". Para respondê-la, conduziu-se um estudo de caso avaliando principalmente aspectos como tempo, complexidade de desenvolvimento, desafios ao utilizar os recursos nativos e impactos do conhecimento prévio em computação. As principais contribuições do trabalho são: apresentar evidências empíricas para orientar a escolha de *game engines* por desenvolvedores novatos; discutir as implicações dessa escolha em termos de custo-benefício para estúdios independentes; e ressaltar a importância da formação em computação para o desenvolvimento de jogos.

2. Fundamentos e trabalhos relacionados

O desenvolvimento de jogos digitais pressupõe a seleção de uma ferramenta fundamental, que pode ser uma *game engine* completa ou uma biblioteca especializada. Tais ferramentas podem reduzir parte da complexidade técnica envolvida na criação de jogos, disponibilizando ao desenvolvedor funcionalidades como renderização gráfica, gerenciamento de cenas, sistemas de física e processamento da entrada do usuário [Engström 2020]. A principal distinção entre essas ferramentas está no grau de abstração oferecido: enquanto *game engines* completas disponibilizam boa parte desses sistemas em componentes prontos, bibliotecas mais minimalistas exigem que o desenvolvedor combine esses elementos manualmente. Isso influencia tanto a curva de aprendizado quanto o nível de flexibilidade disponível [Politowski et al. 2021].

Entre as ferramentas mais utilizadas em contextos independentes, destacam-se a *game engine* Unity, o Godot e a biblioteca Pygame. A Unity é multiplataforma, baseada em C# e possui um editor visual robusto e ampla comunidade

[Christopoulou e Xinogalos 2017]. O Godot é *open source*, com linguagem semelhante ao Python, o GDScript, e suporte a C#, sendo valorizado pela leveza e ausência de custos [Sobota e Pietriková 2023]. Já o Pygame, voltado a jogos 2D, não possui editor visual nem sistemas nativos de física ou colisão, exigindo maior implementação manual [Engström 2020]. Enquanto Unity e Godot oferecem recursos que facilitam o desenvolvimento, o Pygame demanda mais esforço, mas favorece a compreensão dos fundamentos. Assim, a escolha da ferramenta impacta diretamente o processo de aprendizagem e o custo de produção.

Mercan e Onay Durdu [2017] avaliaram a usabilidade da *game engine* Unity sob a perspectiva de desenvolvedores. O estudo, realizado em laboratório com sete participantes (quatro iniciantes e três experientes), envolveu a execução de 26 tarefas relacionadas à criação de projetos, codificação, compilação, depuração e arquivamento, com registro de tempos, taxas de conclusão e aplicação de questionários. Os resultados mostraram que, embora a ferramenta seja utilizável, iniciantes enfrentaram dificuldades com mensagens de erro complexas, interface carregada e tarefas de animação e código. Ambos os grupos atribuíram avaliações moderadas, com menor pontuação para utilidade. Como contribuição, o estudo sugere melhorias de *design* voltadas ao suporte de iniciantes, como simplificação da interface e aprimoramento das mensagens de erro.

Dickson et al. [2017] compararam as *engines* Unity e Unreal quanto à sua viabilidade em uma disciplina de graduação. O curso foi ofertado quatro vezes (três com Unity e uma com Unreal), iniciando pelo aprendizado da ferramenta seguido do desenvolvimento de um jogo em equipe. A análise considerou fatores como curva de aprendizado, suporte, estabilidade, documentação, comunidade e custo. Os resultados indicaram que a Unity apresenta aprendizado mais fácil, maior estabilidade e melhor suporte, enquanto a Unreal oferece melhor qualidade visual, porém maior instabilidade e tempo de compilação. Ainda assim, a maioria dos estudantes preferiu a Unreal ao final do curso. O estudo contribui com um guia prático para a escolha de *game engines*.

Ezhil [2025] apresenta uma análise comparativa entre as *game engines* Unity, Unreal Engine, Godot e CryEngine, com foco em apoiar a escolha de ferramentas por desenvolvedores. O estudo avalia aspectos técnicos, modelos de licenciamento e adequação a diferentes contextos, utilizando uma abordagem exclusivamente revisional, baseada em documentação e literatura. Os resultados indicam que Unity se destaca pela acessibilidade, Unreal pela qualidade gráfica, Godot pela simplicidade e licença aberta, e CryEngine por aplicações visuais complexas, embora com menor comunidade. Como contribuição, o trabalho organiza critérios técnicos e financeiros em um único material de referência, sendo o único a incluir o Godot na comparação, ainda sem avaliação prática.

A Tabela 1 apresenta uma comparação dos trabalhos correlatos com esta pesquisa, dos quais pode-se observar semelhanças e diferenças. Por exemplo, Mercan e Onay Durdu [2017] limitaram-se à avaliação de uma única ferramenta sem envolver o desenvolvimento de um jogo. Dickson et al. [2017] compararam Unity e Unreal Engine em contexto educacional, mas sem contemplar Godot ou Pygame. Ezhil [2025], embora seja o único a incluir o Godot na comparação, não contempla o desenvolvimento prático ou foco em iniciantes. Diante disso, este estudo se diferencia ao: (i) comparar três *game engines*, incluindo o Pygame, que é pouco explorado em estudos dessa natureza; (ii) investigar percepções de tempo, complexidade e uso de conhecimentos de computação de

um desenvolvedor novato; e (iii) discutir as implicações dos resultados para o *pipeline* de produção em estúdios independentes.

Tabela 1. Comparativo entre este trabalho e os estudos relacionados.

Trabalho	Comparação de game engines	Unity	Godot	Pygame	Perfil
Este trabalho	✓	✓	✓	✓	Novato
Mercan e Onay Durdu [2017]	x	✓	x	x	Misto
Dickson et al. [2017]	✓	✓	x	x	Misto
Ezhil [2025]	✓	✓	✓	x	N/A

3. Materiais e métodos

Este trabalho analisa a percepção de um desenvolvedor novato em jogos, graduando em computação com experiência em programação, ao utilizar três ferramentas distintas para o desenvolvimento de mecânicas de um jogo de plataforma 2D: Unity, Godot e Pygame. Para atingir esse objetivo, é conduzido um estudo de caso, uma estratégia de pesquisa que investiga em profundidade um fenômeno dentro de seu contexto real, sendo especialmente adequado quando se busca compreender experiências subjetivas, processos e percepções de um indivíduo ou grupo em uma situação específica [Yin 2015].

3.1. Motivação e seleção de participantes

A motivação para este estudo surgiu a partir de um desenvolvedor novato na indústria de jogos que, embora tenha conhecimentos prévios de programação devido à sua formação superior (em andamento) em computação, apresentava dificuldades na escolha de uma ferramenta de desenvolvimento adequada ao seu perfil. Sem experiência prévia no domínio de jogos e diante de uma oferta ampla de *game engines* e bibliotecas disponíveis, o desenvolvedor não possuía critérios práticos suficientes para embasar essa decisão, situação comum entre desenvolvedores novatos [Politowski et al. 2021].

Nesse sentido, o participante deste estudo corresponde ao desenvolvedor mencionado: um graduando concluinte em Computação, com conhecimentos prévios em programação e desenvolvimento de *software*, adquiridos principalmente em disciplinas acadêmicas e em projetos vinculados a institutos de pesquisa, mas sem experiência prévia no desenvolvimento de jogos digitais. Esse perfil é relevante e comum na indústria independente, em que programadores passam a atuar no desenvolvimento de jogos [Freeman e McNeese 2019, Holfeld 2024].

3.2. Procedimentos

Os procedimentos realizados nesta pesquisa são apresentados em formato de etapas, conforme descritas a seguir.

1. Seleção das Game Engines: para determinar quais *game engines* seriam investigadas neste estudo, conduziu-se uma busca na literatura. Estudos baseados em dados das plataformas Steam e itch.io evidenciaram que a Unity mantém posição dominante entre os jogos publicados, enquanto o Godot apresentou crescimento expressivo a partir de 2020, consolidando-se como principal alternativa no segmento

independente, utilizado predominantemente para jogos 2D e por desenvolvedores autônomos [Holfeld 2024]. Portanto, ambas as ferramentas foram selecionadas, assim como o Pygame, que representa uma abordagem distinta das *game engines* completas: por exigir que o desenvolvedor implemente manualmente os componentes [Engström 2020], pode ser um diferencial para desenvolvedores com experiência prévia em programação.

2. Seleção do Jogo: esta etapa diz respeito à escolha do jogo que será desenvolvido nas três *game engines* para fins comparativos. O jogo selecionado é do tipo plataforma 2D de fase única, escolhido com base em dois critérios: adequação ao perfil novato em desenvolvimento de jogos e presença de mecânicas clássicas e simples, presentes em vários jogos de plataforma como Super Mario e Mega Man. Nesse sentido, o jogo não é extremamente simples, o que tenderia a não possibilitar identificar as diferenças arquiteturais entre as ferramentas, mas também não é complexo, ao ponto de inviabilizar a implementação dentro de um tempo razoável para um desenvolvedor sem experiência prévia. A inspiração do jogo deu-se a partir de um tutorial para o desenvolvimento de jogos com Unity, disponível no YouTube[1], que é composto por sete mecânicas centrais, conforme ilustradas na Figura 1 e descritas a seguir.

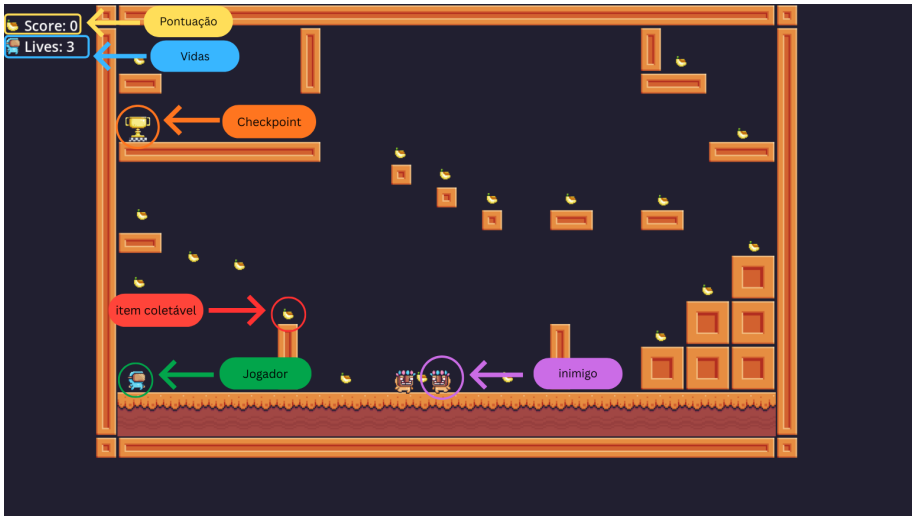

Figura 1. Mecânicas do jogo base para desenvolvimento.

(i) Movimentação lateral do personagem (em verde): controle de entrada do usuário, aplicação de velocidade e tratamento de colisão com o ambiente; **(ii) Sistema de vidas (em azul):** detecção de colisão entre o personagem e os inimigos, com limite de 3 colisões; **(iii) Sistema de coleta de itens (em amarelo):** detecção de colisão entre o personagem e objetos distribuídos pelo cenário, com contabilização dos itens; **(iv) Patrulha automática de inimigos (em roxo):** movimentação baseada em tempo, com inversão de direção; **(v) *Checkpoint* de conclusão de fase (em laranja):** marca o progresso do jogador ao final da fase; **(vi) Sistema de pontuação por estrelas:** o total de itens coletados define a quantidade de estrelas obtidas; e **(vii) Sistema de fim de jogo:** ao perder todas as vidas, o jogo exibe a tela de *game over* com opção de reinício.

3. Implementação: o participante iniciou o desenvolvimento dos jogos, conduzido de forma sequencial (Unity, Godot e Pygame). Em cada implementação,

[1]Disponível em: YouTube - Criando um jogo de plataforma 2D na Unity (texto clicável)

adotou um processo de aprendizagem autônoma, utilizando vídeos de tutoriais presentes no YouTube. Esse material foi complementado com consultas a fóruns de desenvolvimento de jogos e utilização de modelos de Inteligência Artificial Generativa, como o ChatGPT e Claude, para resolução de dúvidas pontuais. Ressalta-se que o participante não utilizou os modelos para geração de código, somente quando necessitava de suporte para correção de inconsistências ou dúvidas sobre o funcionamento das *game engines*. Além disso, nenhum curso pago ou orientação especializada externa foi utilizado, de modo a simular o contexto de aprendizagem independente, típico de um desenvolvedor novato no domínio [Politowski et al. 2021].

Durante o desenvolvimento, o participante utilizou as *game engines* Unity, Godot e Pygame nas respectivas versões: 6000.4.0f1, v4.6.1.stable.official e v2.6.1. Para a animação dos *sprites* e recursos para montar o *level design* do jogo foram utilizados *assets* gratuitos disponibilizados [Frog 2024]. Ao longo de cada implementação, o participante foi orientado a registrar em um diário de bordo suas percepções, dificuldades encontradas, decisões tomadas e a duração do tempo de desenvolvimento em cada *game engine*. Esses registros compõem um dos instrumentos de coleta de dados qualitativos deste estudo e são sintetizados na Subseção 3.3.

4. Aplicação de formulário: após as três implementações, aplicou-se um formulário de avaliação com o participante, com o objetivo de sistematizar e aprofundar as percepções construídas ao longo do processo de desenvolvimento. O formulário, descrito na Subseção 3.3, foi elaborado de forma autoral, dos quais três dos autores possuíam mais de sete anos de experiência com jogos educacionais.

3.3. Coleta e Análise de Dados

Os dados quantitativos do estudo referem-se aos registros de tempo de desenvolvimento por ferramenta, obtidos a partir do diário de bordo do participante ao longo das implementações, e às respostas a um formulário de avaliação composto por cinco questões em escala Likert de cinco pontos, variando de 1 (Discordo completamente) a 5 (Concordo completamente). As questões do formulário, disponível no link[2], abordaram: facilidade de aprendizagem e uso da ferramenta; suficiência de recursos, ferramentas e documentação disponíveis para o suporte durante implementação; adequação do tempo de desenvolvimento ao perfil novato do participante; necessidade de conhecimentos prévios em computação; e propensão a recomendar a ferramenta para outros desenvolvedores novatos. Já os dados qualitativos foram obtidos por meio de três perguntas abertas no mesmo formulário, em relação à percepção sobre o nível de dificuldade de cada ferramenta, os pontos positivos e negativos identificados e os conhecimentos de computação mobilizados durante o desenvolvimento. Todos os dados (quali e quanti) foram registrados em uma planilha para análise.

Como forma de analisar os dados quantitativos, é utilizado um gráfico de barras agrupadas, no qual cada grupo de barras representa uma das cinco afirmativas avaliadas, e cada barra corresponde a uma das três ferramentas: Unity, Godot e Pygame, permitindo a comparação visual direta entre as pontuações atribuídas pelo participante em cada aspecto. Para os dados de natureza qualitativa, provenientes das perguntas abertas do formulário, foi escolhida a análise descritiva, na qual as respostas

[2]https://drive.google.com/file/d/197ND0547EGO5z-NWJGvMXUuJr32DxTyZ/view?usp=sharing

do participante foram organizadas, sintetizadas e categorizadas em torno dos aspectos de interesse do estudo: nível de dificuldade percebido, pontos positivos e negativos, e conhecimentos de computação mobilizados. Essa combinação de representações visuais e análise descritiva permite articular os dados quantitativos e qualitativos de maneira complementar, oferecendo uma visão integrada da experiência do participante em cada ferramenta avaliada.

4. Resultados e discussões

Esta seção apresenta os resultados obtidos a partir dos dois instrumentos de coleta: o diário de bordo, mantido pelo participante ao longo das três implementações, e o formulário de avaliação respondido após o desenvolvimento. A Figura 2 contém um gráfico com as avaliações do participante para as questões em escala Likert do formulário, referentes a diferentes aspectos sobre as *game engines*, que são descritos a seguir.

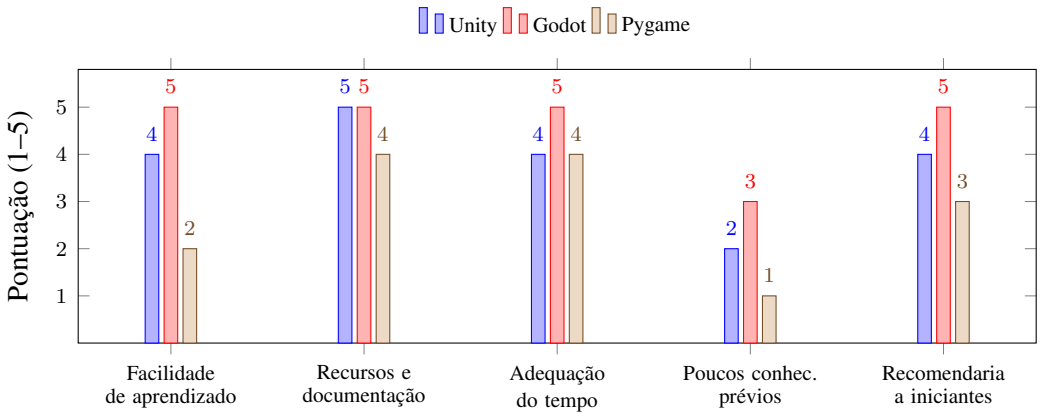

Figura 2. Respostas às afirmativas em escala Likert por ferramenta

Facilidade de aprendizado e uso: o Godot obteve a pontuação máxima (5) nesse critério, sendo descrito pelo participante como mais organizado e navegável, com interface traduzida para o português, editor de código integrado e sistema de nós com nomenclatura simples e direta. A Unity recebeu pontuação 4, com produtividade associada aos componentes nativos, mas com interface percebida como excessivamente densa, dificultando a localização de campos específicos e a interpretação de mensagens de erro. O Pygame recebeu a menor pontuação (2), cujo desenvolvedor pontuou a ausência de editor visual, que demandou múltiplos ciclos de execução e ajuste para posicionamento de objetos e construção da fase.

Recursos, ferramentas e documentação: Unity e Godot receberam pontuação máxima (5), indicando que ambas as *game engines* oferecem suporte suficiente para o desenvolvimento. O Pygame, apesar da menor estrutura nativa, recebeu pontuação 4, sugerindo que a documentação disponível foi considerada adequada, ainda que a ausência de sistemas prontos tenha elevado o esforço de implementação.

Adequação do tempo de desenvolvimento: o Godot recebeu pontuação 5 e registrou o menor tempo total de desenvolvimento, aproximadamente 13 horas. A Unity recebeu pontuação 4, com cerca de 19 horas registradas. O Pygame recebeu pontuação 4, mas totalizou aproximadamente 26 horas de desenvolvimento, sendo a implementação

mais longa entre as três ferramentas. Esses dados de tempo têm relevância direta para o contexto industrial, pois ao converter as horas registradas em custo de mão de obra, a diferença entre o Godot (13 h) e o Pygame (26 h) representa o dobro do esforço para um projeto equivalente, o que pode provocar um grande impacto em um estúdio independente com recursos limitados.

Necessidade de conhecimentos prévios em computação: esse foi o critério com as menores pontuações em todas as ferramentas, indicando que os conhecimentos prévios foram fundamentais para implementar as mecânicas do jogo. O Pygame foi tido como a *game engine* que mais demandou uso dos conhecimentos (1), seguida pela Unity (2) e Godot (3). O participante destacou que lógica de programação e orientação a objetos foram essenciais no desenvolvimento em Unity e Godot, enquanto no Pygame foram mobilizados de forma intensa e contínua também conhecimentos de estruturas de dados, desde o posicionamento de elementos simples até a implementação da lógica de colisões.

Recomendação para desenvolvedores novatos: o Godot recebeu pontuação máxima 5, sendo classificado pelo participante como adequado para iniciantes e explicitamente recomendado para outros desenvolvedores no mesmo perfil. A Unity recebeu pontuação 4, posicionando-se como alternativa viável. O Pygame recebeu pontuação 3, refletindo a avaliação de que sua curva de aprendizado mais acentuada o torna menos indicado para novatos sem um objetivo pedagógico específico.

A seguir, são discutidos os dados qualitativos obtidos por meio das perguntas abertas do formulário de avaliação, organizados pelos aspectos de análise.

Nível de dificuldade percebido: o participante classificou a Unity como de dificuldade moderada, o Godot como adequado para iniciantes e o Pygame como o mais difícil. Essa percepção é coerente com os demais dados coletados: o Pygame exigiu a implementação manual de elementos nativos nas demais ferramentas, como sistemas de física e detecção de colisão, resultando em volume consideravelmente maior de código e maior complexidade em praticamente todas as etapas do jogo.

Pontos positivos e negativos: em relação à Unity, os pontos positivos concentraram-se na disponibilidade de componentes nativos, como *Rigidbody* e *Collider*, e no bom suporte a *sprites* e animações. Como pontos negativos, foram relatadas dificuldades na manipulação de múltiplos *tilemaps*, na orientação dos *sprites* dos inimigos durante a patrulha e na ocorrência de um *bug* de colisores invisíveis que não foi resolvido ao longo do desenvolvimento. Quanto ao Godot, os pontos positivos abrangeram a interface mais amigável, a menor complexidade dos *scripts* e a analogia traçada pelo participante de que a facilidade de navegação e a interface intuitiva do Godot seria equivalente ao CLion, IDE da JetBrains, em contraposição à Unity que possui uma interface densa, comparada ao Code::Blocks. O único ponto negativo registrado foi a percepção de que o Godot oferece menos recursos nativos do que a Unity. Em relação ao Pygame, o principal ponto negativo foi a ausência de editor visual, enquanto os pontos positivos incluíram a ampla liberdade de personalização e o controle granular sobre todos os elementos do jogo, além de ter agregado um conhecimento mais robusto sobre o funcionamento de jogos digitais e o exercício de fundamentos de programação.

Conhecimentos de computação mobilizados: em todas as ferramentas, o participante mobilizou conhecimentos prévios de computação. Na Unity, lógica

de programação e orientação a objetos foram destacadas como essenciais para o desenvolvimento dos *scripts* e a implementação das mecânicas. No Godot, lógica de programação foi relevante para mecânicas e condições lógicas, como o sistema de coleta de itens e de premiação por estrelas. No Pygame, lógica de programação, orientação a objetos e estruturas de dados foram mobilizados de forma intensa e contínua ao longo de todo o desenvolvimento, desde tarefas simples até a implementação da lógica de colisões.

Em resposta à QP, os dados obtidos sugerem que o Godot é a ferramenta mais equilibrada para um desenvolvedor novato em jogos com formação em computação: menor tempo de desenvolvimento e maiores pontuações em facilidade de uso e adequação do tempo. A Unity posicionou-se como alternativa viável, com boa produtividade proporcionada por seus componentes nativos, mas com interface densa, curva de aprendizado mais exigente e ocorrência de problemas técnicos não resolvidos. O Pygame, embora o mais complexo, com cerca de 26 horas de desenvolvimento e as menores pontuações em facilidade de uso, destacou-se pela liberdade de personalização e pela contribuição ao aprimoramento de fundamentos de programação, sendo o mais adequado para exercitar os conteúdos aprendidos. Em todas as ferramentas, a mobilização de conhecimentos prévios de computação, especialmente lógica de programação e orientação a objetos, mostrou-se relevante no desenvolvimento.

Do ponto de vista da indústria independente, os resultados têm implicações práticas relevantes. O menor tempo de desenvolvimento no Godot, aproximadamente 32% inferior ao da Unity e 50% inferior ao do Pygame, representa uma redução direta no custo em homem-hora para um projeto de escopo equivalente [Holfeld 2024]. Além disso, a natureza *open source* e gratuita do Godot elimina custos de licenciamento presentes em outras ferramentas, tornando-o especialmente atrativo para estúdios independentes e desenvolvedores solo [Holfeld 2024, Unity Technologies 2023]. Esse aspecto ganhou relevância após a controvérsia de 2023 em que a Unity anunciou uma taxa por instalação (*runtime fee*) retroativa, evidenciando o risco financeiro de depender de *game engines* com modelos de licenciamento proprietário [Unity Technologies 2023]. Por outro lado, a Unity, apesar do maior tempo de adaptação inicial, oferece um ecossistema mais maduro com maior disponibilidade de *assets* e integrações de terceiros, o que pode ser vantajoso em projetos de maior escala ou com requisitos técnicos mais complexos [Engström 2020].

Os achados desta pesquisa convergem com outros estudos da literatura. A percepção de interface densa e curva de aprendizado exigente na Unity corrobora os resultados de Mercan e Onay Durdu [2017], que identificaram dificuldades semelhantes em desenvolvedores iniciantes. A avaliação moderadamente positiva da Unity é coerente com Dickson et al. [2017], mas a inclusão do Godot na comparação revelou uma alternativa ainda mais acessível para novatos, dado não contemplado pelos autores. As conclusões revisionais [Ezhil 2025] sobre a simplicidade e adequação do Godot a projetos 2D convergem com os dados deste estudo. Por fim, a mobilização intensa de conhecimentos prévios de computação nas três ferramentas evidencia que o perfil do desenvolvedor exerce papel determinante sobre a experiência de desenvolvimento, complementando a perspectiva dos demais autores.

Limitações da pesquisa incluem: (i) um único participante no estudo, o que impede a generalização dos achados para outros perfis de desenvolvedores; (ii) a ordem fixa das implementações (Unity, Godot e Pygame) pode representar uma ameaça à

validade interna do estudo, pois o aprendizado acumulado ao longo das etapas pode ter influenciado as implementações posteriores. Dessa forma, parte da agilidade observada no Godot pode estar associada à familiaridade já adquirida na etapa anterior, e não apenas às características da própria ferramenta; (iii) o escopo restrito a mecânicas de plataforma 2D limita as conclusões a esse tipo de projeto, não sendo possível afirmar que os resultados se repetiriam em jogos de outros gêneros ou maior complexidade; e (iv) o uso de ferramentas de apoio como ChatGPT e Claude, ainda que uniforme entre as três implementações, representa uma variável que pode ter atenuado dificuldades que seriam mais evidentes em um processo de aprendizagem estritamente autônomo.

5. Conclusões

Este trabalho foi orientado pela questão de pesquisa (QP): "Como as percepções de um desenvolvedor novato ao utilizar Unity, Godot e Pygame podem orientar a escolha de *game engines* por iniciantes na área de desenvolvimento de jogos?". Para respondê-la, conduziu-se um estudo de caso com quatro etapas: (i) realizou-se uma busca na literatura para selecionar as ferramentas mais relevantes para o perfil investigado; (ii) definiu-se o jogo a ser desenvolvido, com mecânicas representativas e escopo adequado ao nível do participante; (iii) o participante implementou o jogo nas três ferramentas de forma sequencial, registrando suas percepções em um diário de bordo; e (iv) aplicou-se um formulário de avaliação composto por afirmativas em escala Likert e perguntas abertas, cujos dados foram analisados de forma integrada com os registros do diário.

Em resposta à QP, o Godot mostrou-se a ferramenta mais adequada ao perfil investigado, combinando o menor tempo de desenvolvimento com as melhores avaliações em facilidade de uso e indicação para novatos. A Unity, embora produtiva, exigiu maior esforço de adaptação devido à densidade de sua interface e à curva de aprendizado mais elevada. O Pygame, o mais complexo e demorado entre os três, revelou-se menos indicado para o desenvolvimento ágil de jogos, mas o mais eficaz para consolidar fundamentos de programação, podendo ser uma alternativa em contextos pedagógicos. Em todas as ferramentas, a experiência prévia em computação mostrou-se um fator determinante, evidenciando que o perfil do desenvolvedor influencia a experiência de desenvolvimento tanto quanto a ferramenta escolhida.

Para a indústria independente, o Godot destacou-se como a opção mais custo-benefício para desenvolvedores novatos, combinando menor tempo de desenvolvimento, ausência de custos de licenciamento e menor curva de adaptação [Holfeld 2024, Politowski et al. 2021]. A Unity mantém-se como alternativa sólida para projetos de maior escala, enquanto o Pygame, por exigir implementação manual extensiva, é mais adequado a contextos com objetivo pedagógico explícito, como disciplinas voltadas a algoritmos e estruturas de dados [Engström 2020, Sobota e Pietriková 2023]

Como trabalhos futuros, pretende-se: (i) ampliar o escopo do estudo, envolvendo desenvolvedores com e sem experiência prévia em programação, visando verificar possíveis diferenças entre os perfis; (ii) incluir mecânicas com diferentes níveis de complexidade e contemplar outras *game engines*, como GameMaker e CryEngine; (iii) analisar o impacto do uso de ferramentas de IA Generativa no apoio ao desenvolvimento de jogos; e (iv) adotar um delineamento com ordem de implementação contrabalanceada para mitigar a limitação associada ao efeito de aprendizado.

6. Agradecimentos

Os autores expressam sua gratidão à Universidade do Estado do Amazonas (UEA) pelo apoio institucional e à PROPESP-UEA pelo suporte financeiro. Agradecem também aos seus colegas do ThinkTEd Lab pelas contribuições e discussões que enriqueceram este trabalho. Os autores reconhecem ainda o apoio do centro de pesquisa Nexus, que forneceu recursos materiais essenciais, incluindo espaço físico, computadores e infraestrutura relacionada.

Declaração sobre uso de Inteligência Artificial

Neste estudo, utilizou-se o ChatGPT, da OpenAI, como ferramenta de apoio à geração de códigos para gráficos e tabelas em formato LaTeX, com o objetivo de reduzir o tempo e o esforço na construção dessas representações. A ferramenta também foi utilizada para auxiliar na reformulação de frases e na tradução de trechos do texto. Os autores assumem integral responsabilidade pela revisão, validação e versão final do trabalho.

Referências

Christopoulou, E. e Xinogalos, S. (2017). Overview and comparative analysis of game engines for desktop and mobile devices. *International Journal of Serious Games*.

Dickson, P. E., Block, J. E., Echevarria, G. N., e Keenan, K. C. (2017). An experience-based comparison of unity and unreal for a stand-alone 3d game development course. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 70–75.

Engström, H. (2020). Game development research.

Ezhil, S. L. (2025). A comparative analysis of unity, unreal engine, godot, and cryengine: Technical features, licensing models, and suitability for modern game development. *International Neuro Rehabilitation Journal of Australia*.

Freeman, G. e McNeese, N. J. (2019). Exploring indie game development: Team practices and social experiences in a creativity-centric technology community. *Computer Supported Cooperative Work (CSCW)*, 28(3):723–748.

Frog, P. (2024). Pixel adventure. Acesso em: 25 abr. 2026.

Gee, J. P. (2003). What video games have to teach us about learning and literacy. *Computers in entertainment (CIE)*, 1(1):20–20.

Holfeld, J. (2024). On the relevance of the Godot Engine in the indie game development industry. *arXiv preprint arXiv:2401.01909*.

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., e Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. *SIGCSE Bull.*, 40(1):367–371.

Mercan, Ş. e Durdu, P. O. (2017). Evaluating the usability of unity game engine from developers' perspective. In *2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–5. IEEE.

Politowski, C., Petrillo, F., Montandon, J. E., Valente, M. T., e Guéhéneuc, Y.-G. (2021). Are game engines software frameworks? a three-perspective study. *Journal of Systems and Software*, 171:110846.

Portal News (2025). A economia dos jogos: como o setor de games já supera o cinema e a música. Acessado em: 28 de abril de 2026.

Sobota, B. e Pietriková, E. (2023). The role of game engines in game development and teaching. In *Computer Science for Game Development and Game Development for Computer Science*. IntechOpen.

Unity Technologies (2023). An open letter to our community. https://blog.unity.com/news/open-letter-to-our-community. Acesso em: jun. 2025.

Yin, R. K. (2015). *Estudo de caso: planejamento e métodos*. Bookman, 5 edition.