Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Avaliação e Análise de Desempenho de Distribuições Linux para a Execução de Jogos Eletrônicos

Francis Lucas de Sá

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Alexandre Magno de Sousa

Maio, 2025
João Monlevade–MG

Francis Lucas de Sá

Avaliação e Análise de Desempenho de Distribuições Linux para a Execução de Jogos Eletrônicos

Orientador: Alexandre Magno de Sousa

Monografia apresentada ao curso de Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Maio de 2025

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

S111a Sá, Francis Lucas de.
Avaliação e análise de desempenho de distribuições Linux para a execução de jogos eletrônicos. [manuscrito] / Francis Lucas de Sá. - 2025.
66 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Alexandre Magno de Sousa.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Engenharia de Computação .

1. Análise de variância. 2. Jogos - Desempenho - Análise. 3. Jogos eletrônicos. 4. Linux (Sistema operacional de computador). 5. Medição. 6. Projetos de experimentos fatoriais. 7. Sistemas de coleta automática de dados. I. Sousa, Alexandre Magno de. II. Universidade Federal de Ouro Preto. III. Título.

CDU 519.2:004

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Francis Lucas de Sá

Avaliação e Análise de Desempenho de Distribuições Linux para a Execução de Jogos Eletrônicos

Monografia apresentada ao Curso de Engenharia de Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharelado em Engenharia de Computação

Aprovada em 16 de Abril de 2025.

Membros da banca

Doutor - Professor Alexandre Magno de Sousa - Orientador - Universidade Federal de Ouro Preto
Doutor - Professor Eduardo da Silva Ribeiro - Universidade Federal de Ouro Preto
Doutora - Professora Gilda Aparecida de Assis - Universidade Federal de Ouro Preto

Professor Alexandre Magno de Sousa, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 07/05/2025

Documento assinado eletronicamente por **Alexandre Magno de Sousa**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 07/05/2025, às 16:32, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0906226** e o código CRC **01AD46B5**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Este trabalho é dedicado à minha mãe (in memoriam), que tornou tudo possível.

Agradecimentos

A realização deste trabalho não seria possível sem a ajuda e apoio de diversas pessoas, às quais expresso meu profundo agradecimento:

Ao meu orientador, que dedicou seu tempo e paciência para tornar este trabalho possível.

À comunidade de software livre e de código aberto, cuja paixão pela tecnologia, espírito colaborativo e compromisso com a inovação transformam ideias em soluções concretas e fazem do mundo um lugar melhor e mais interessante.

Em especial à minha mãe, que sempre forneceu apoio e tornou tudo possível. E também a todos que me ajudaram após a sua partida.

"If I have seen further, it is by standing on the shoulders of giants."

— Isaac Newton (1642 – 1727),
in: Letter to Robert Hooke.

Resumo

O desempenho de jogos eletrônicos para computador é uma preocupação central quando os usuários decidem escolher desde os recursos para configuração do computador até as configurações de cada jogo. Ainda, em se tratando de sistemas operacionais Linux, existem diferentes distribuições a serem consideradas. Apesar dessas distribuições executarem com o mesmo kernel, elas apresentam propostas distintas, então, isso pode gerar uma preocupação em relação ao desempenho. Diante disso, este trabalho visa analisar o desempenho de diferentes distribuições Linux para identificar os seus impactos na execução de jogos. Para isso, é utilizado um projeto de experimentos fatorial completo com dois fatores e replicações. O primeiro fator conta com diferentes distribuições Linux, tais como Arch, Bazzite, CachyOS, Debian, Fedora e Ubuntu. Já, para o segundo fator, foram selecionados os seguintes jogos Assassin's Creed Origins, Far Cry 6, Killer Instinct (2013), Metro Exodus Enhanced Edition e War Thunder. A ferramenta MangoHud foi utilizada para coleta das métricas e dos dados da execução dos experimentos dos jogos. Como métrica de avaliação de desempenho foi definida a métrica de frames por segundo, além desta, outras métricas de desempenho também foram coletadas, tais como utilização, frequência e temperatura da GPU e CPU, e frametime. Por meio do projeto fatorial completo foi possível identificar que a variação explicada das distribuições Linux foi de apenas 0,34%, no entanto, de forma surpreendente, os jogos representam 98,16% da variação explicada. Portanto, as distribuições Linux apresentam desempenhos semelhantes entre si e a carga executada apresenta maior impacto no desempenho do sistema do que as distribuições Linux. Ainda assim, foi possível observar que o sistema Ubuntu obteve o maior desempenho, em cerca de 1,12% acima da média de frames por segundo, enquanto o sistema Fedora apresentou o menor desempenho, com cerca de 2,58% abaixo da média. Ressalta-se que estes resultados se limitam a uma única configuração de hardware para plataformas desktop. No entanto, este trabalho contribui com a aplicação do projeto fatorial completo para avaliar o desempenho da execução de jogos, além de mostrar que as distribuições Linux apresentam pouca influência no desempenho.

Palavras-chaves: Análise de Desempenho. Medição e Coleta de Dados. Distribuições Linux. Jogos. Projeto de Experimentos. Projeto Fatorial Completo. ANOVA.

Abstract

The performance of computer games is a key concern when users choose from resources to configure their computer to the settings for each game. When it comes to Linux operating systems, there are several distributions to consider. Although these distributions run with the same kernel, they have different proposals, so this can be a concern regarding performance. With this in mind, this work aims to analyze the performance of different Linux distributions to determine their impact on game execution. For this purpose, a full factorial experimental design with two factors and replications is used. The first factor includes various Linux distributions, such as Arch, Bazzite, CachyOS, Debian, Fedora and Ubuntu. For the second factor, the following games were selected: Assassin's Creed Origins, Far Cry 6, Killer Instinct (2013), Metro Exodus Enhanced Edition and War Thunder. The tool MangoHud was used to collect metrics and data from the execution of the game experiments. Frames per second was defined as a performance evaluation metric, and other performance metrics were also collected, such as GPU and CPU utilization, frequency, and temperature, and frametime. Through the full factorial design, it was found that the explained variation of the Linux distributions is only 0.34%; however, surprisingly, the games are responsible for 98.16% of the explained variation. Therefore, the Linux distributions have similar performance and the executed load has a greater impact on system performance than the Linux distributions. Still, it was possible to observe that the Ubuntu system obtained the highest performance, at around 1.12% above the frames per second average, while the Fedora system presented the lowest performance, at around 2.58% below the average. It should be noted that these results are limited to a single hardware configuration for desktop platforms. However, this work contributes to the application of the full factorial design to evaluate the execution performance of games and also shows that Linux distributions have little impact on performance.

Key-words: Performance Analysis. Measurement and Data Collection. Linux Distributions. Games. Design of Experiments. Full Factorial Design. ANOVA.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CPU *Central Processing Unit*

GPU *Graphics Processing Unit*

API *Application Programming Interface*

FPS *Frames Per Second*

ANOVA *Analysis of Variance*

Sumário

1 Introdução

A avaliação de desempenho é crucial para a gestão eficiente da infraestrutura computacional, otimizando recursos através da compreensão de sua utilização e viabilizando a alocação eficaz, incluindo redistribuição, virtualização e balanceamento de carga, especialmente em sistemas computacionais onde vários processos são executados para realizar a comunicação entre componentes físicos e programas em tempo real (JAIN, 1991).

Na indústria de jogos, essa avaliação influencia diretamente a escolha do usuário, impactando a decisão de jogar, desistir ou investir em novos componentes de computador ou máquinas (consoles, celulares ou computadores), onde os próprios usuários avaliam as peças disponíveis, considerando especificações técnicas e métricas de desempenho como quadros por segundo e "1% low" para tomar decisões (GAMEBENCH, 2019).

Os celulares e consoles de videogame possuem *hardware* fixo, sem a possibilidade de alterar componentes internos, onde os jogos multiplataforma podem ter otimizações específicas e os exclusivos temporários precisam ser portados para computador, podendo impactar no desempenho por problemas de incompatibilidade entre as plataformas. Enquanto os computadores são modulares, podendo trocar as peças de acordo com a disponibilidade financeira, além da flexibilidade de *softwares* para customizações diversas (KINGSTON, 2023).

A possibilidade de executar jogos nativos de Windows em sistemas Linux tem crescido consideravelmente nos últimos anos, impulsionada por ferramentas de compatibilidade como o Wine[1] e *softwares* como a Steam, Lutris[2] e Bottles[3], oferecendo uma usabilidade simplificada. Além disso, o lançamento do Steam Deck[4] pela Valve[5], um console portátil que utiliza o SteamOS[6], uma distribuição Linux, como sistema padrão, contribui positivamente para essa tendência (NOGUEIRA, 2024).

Para *desktops*[7] em geral, a plataforma mais conhecida é a Steam[8], que conta com uma média diária de aproximadamente 30 milhões de jogadores, onde aproximadamente 96,10% dos jogadores utilizam Windows, 1,58% utiliza OSX e 2.33% utiliza Linux, de acordo com a pesquisa de *hardware* e *software* mais recente, sendo que esses números variam, pois a pesquisa é realizada por mensalmente (VALVE, 2025).

[1] <https://www.winehq.org/>
[2] <https://lutris.net/>
[3] <https://usebottles.com/>
[4] <https://www.steamdeck.com/en/>
[5] <https://www.valvesoftware.com/en/>
[6] <https://store.steampowered.com/steamos>
[7] Desktop: computadores de mesa.
[8] <https://store.steampowered.com/>

1.1 Motivação e Justificativa

Cada sistema operacional lida com seus mecanismos internos de funcionamento, como as chamadas de sistema e *Application Programming Interface* (API)s para comunicações entre processos e com componentes de *hardware*, e devido a essa diferença de funcionamento entre sistemas, como Windows e Linux, surge o problema de compatibilidade de aplicações, que algumas vezes dependem de APIs específicas de um sistema, tornando difícil haver uma compatibilidade nativa entre sistemas, caso a aplicação não seja projetada para ser portável (TANENBAUM; BOS, 2014).

A compatibilidade, é um grande problema a ser resolvido, impossível de ser remediado, sem que haja alguma solução intermediária para criar essa ponte, para que os programas de um sistema funcionem no outro com a menor quantidade de problemas apresentados ou impactos perceptíveis de desempenho. E, para resolver esse problema, surgiu uma alternativa com a proposta de traduzir as chamadas de sistema de um sistema operacional para o outro se forma rápida, por tentativa e erro, e com o tempo, as soluções como o Wine foram se aprimorando, pra traduzir chamadas de sistema do Windows para sistemas como o BSD e Linux (DAVIES et al., 2024).

A dependência dessas camadas de compatibilidade e as nuances do suporte de *hardware* e *software* no Linux geram preocupações significativas em relação ao desempenho relativo dos jogos em comparação com o ambiente nativo do Windows, para o qual foram originalmente projetados. E a maioria dos jogos não oferece uma versão nativa para sistemas Linux, sendo necessário utilizar a camada de compatibilidade para acessar jogos clássicos e lançamentos recentes, e também, é comum que jogos competitivos bloqueiem os usuários de Linux com a alegação de que não é possível usar um sistema anti-trapaças eficiente (BONFIM, 2024).

Então, para as pessoas que optam por usar um sistema livre, resta usar uma camada de compatibilidade para a maior parte dos jogos disponíveis no mercado, e como um esforço coletivo da comunidade, sites como o ProtonDB[9], um projeto de Buck DeFore e diversos usuários da internet para classificar a compatibilidade de jogos, e funciona como uma lista de jogos onde os usuários relatam se houve problemas para executar algum jogo em seu sistema Linux, e a classificação vai de “Platina”, para jogos que rodam sem problemas, até “Quebrado” para jogos com sistemas Anti-Trapaças que bloqueiam o acesso ou algum problema desconhecido de desempenho ou por não ser possível rodar por incompatibilidade completa (PROTONDB, 2025).

Além disso, existem diversas alternativas de sistemas Linux, as denominadas distribuições, que possuem suas particularidades de funcionamento, sistemas de segurança e tecnologias alternativas implementadas sobre o *Kernel* Linux. Existem distribuições que

[9] <https://www.protondb.com/>

visam o conforto máximo do usuário, seja para tarefas diárias ou mesmo para jogos, e outras possuem a filosofia de "construa você mesmo", nas quais os usuário é o responsável por escolher os pacotes iniciais, com um controle maior sobre o seu sistema (CASTRO, 2016).

Essas preocupações podem representar um dos principais obstáculos para uma adoção mais ampla do Linux em computadores *desktop*s, limitando a liberdade de escolha dos usuários, principalmente os que possuem um computador relativamente antigo, e restringindo as possibilidades de desenvolvimento e aprimoramento de produtos, pois, além dos usuários precisarem se adaptar a um sistema novo, ainda se encontram divididos entre diversas opções disponíveis, e podem se questionar se uma opção de sistema é melhor que a outra.

1.2 Definição do Problema

O problema de pesquisa é a perda ou ganho de desempenho entre distribuições Linux distintas, ao executar jogos. Visto que existem diversas distribuições com filosofias de desenvolvimento e versionamento próprias, é importante elucidar a dúvida de se vale a pena um usuário abrir mão de algumas funcionalidades em busca de um desempenho maior em outra distribuição ou se as variações de distribuições são efetivamente pequenas para o desempenho.

Diante desse contexto, para analisar um conjunto de distribuições Linux que pode ser extrapolado para outras, o trabalho visa usar ferramentas estatísticas e medição empírica do desempenho de jogos em algumas distribuições Linux diferentes para encontrar alguma evidência de que se há ou não uma perda de desempenho entre as diferentes distribuições Linux.

1.3 Objetivos Geral e Específicos

O projeto tem como objetivo geral analisar o desempenho de sistemas operacionais, algumas distribuições Linux de propostas e filosofias distintas, para executar jogos eletrônicos. Para que o objetivo geral seja alcançado, os seguintes objetivos específicos são definidos:

1. Execução, medição de desempenho e coleta de diferentes tipos de jogos em dois tipos de sistemas operacionais;
2. Comparação de desempenho utilizando métricas de *hardware* e de *software* para cada sistema operacional;

3. Investigação de como cada sistema operacional lida com os seus recursos quando exposto à tarefas intensivas e como isso resulta em mais ou menos performance para cada sistema.

1.4 Resultados e Contribuições

Os resultados demonstram que, com a análise do projeto fatorial completo com dois fatores, através da *Analysis of Variance* (ANOVA) e intervalos de confiança, observou-se que os jogos são responsáveis pelo maior impacto no desempenho, com uma contribuição de 98,16%, enquanto os sistemas operacionais possuem uma contribuição de 0,34%, ou seja, os sistemas operacionais possuem uma diferença pequena de desempenho entre si, com apenas alguns casos demonstrando uma vantagem pequena de ganho ou perda de desempenho para jogos específicos. Então, é possível dizer que a escolha de uma distribuição tem pouco impacto no desempenho de jogos quando comparadas entre si. Os resultados mostraram que, em termos percentuais, mesmo que a variação explicada vinda dos sistemas operacionais seja pequena, os sistemas operacionais Ubuntu 22.10 e Fedora 41 obtiveram, respectivamente as porcentagens de diferença relativa à média mais acentuadas, mesmo que pequenas, com respectivamente, o Ubuntu apresentando aproximadamente uma desempenho de 1,12% acima da média e o Fedora, um desempenho de 2,58% abaixo da média, ou seja, no pior caso, há uma perda de poucos *Frames Per Second* (FPS) de forma geral.

As contribuições envolvem a constatação de que as distribuições Linux possuem pouca diferença entre si, em relação ao desempenho de jogos. E a partir da aplicação da metodologia do projeto de experimentos, foi possível observar o comportamento de cada sistema através dos intervalos de confiança, pois, apesar de pouca diferença, não houveram intervalos não significativos. Além disso, o trabalho demonstrou que é possível aplicar o projeto de experimentos também para a análise de desempenho de jogos, para obter um resultado mais robusto ao tentar observar o comportamento de um sistema para esse propósito.

1.5 Estrutura da Monografia

Os conceitos teóricos importantes para a análise de desempenho são apresentadas no próximo capítulo, tais como o funcionamento de jogos e dos sistemas operacionais, as ferramentas de compatibilidade, as ferramentas de monitoramento e as métricas e trabalhos relacionados. Em seguida, o Capítulo 3 apresenta as definições da metodologia de desenvolvimento, referente ao *hardware* e *software*. Então, no Capítulo 4 apresentará um resumo dos dados coletados e então irá executar o projeto fatorial completo. Por fim, o Capítulo 5 traça as conclusões, limitações e trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os conceitos fundamentais para compreender a proposta do trabalho e as tecnologias envolvidas na análise de desempenho e a importância dessa análise para o desempenho de sistemas operacionais para jogos. A Seção 2.1 introduz conceitualmente o que pode ser definido como um jogo eletrônico e sua interação com o *hardware*. Já a Seção 2.2 descreve o conceito fundamental que é responsável pela comunicação entre o jogo, o sistema operacional e a placa de vídeo. Por sua vez, a Seção 2.3 se refere aos conceitos importantes referentes aos sistemas operacionais e distribuições diferentes, e sua relação aos jogos. A Seção 2.4 se refere à necessidade da camada de compatibilidade para executar jogos nativos de Windows em sistemas Linux. Na Seção 2.5 são apresentadas as ferramentas de compatibilidade e as métricas de desempenho coletadas. Em seguida, a Seção 2.6 introduz os conceitos importantes para as análise dos projetos fatoriais. Por fim, a Seção 2.7 se refere aos trabalhos relacionados, que ajudaram a criar uma base para o estudo.

2.1 Jogos Eletrônicos

No contexto do entretenimento digital, de acordo com Bartle (2003), os videogames consistem em experiências interativas que ocorrem dentro de mundos em ambiente virtual, onde um ou mais personagens são controlados por jogadores humanos, que interagem dentro de um espaço comum, e são regidos por regras específicas ao seu universo, que definem a dinâmica de interação e desafios com o objetivo de alcançar metas. E esses mundos incluem elementos cosméticos (como ambientação e estética temática), estilos de jogabilidade, como ação, aventura, single-player, multiplayer e jogos de mundo aberto, com diversas mecânicas de interação, incluindo visão em primeira pessoa ou terceira pessoa, e também envolve diferentes perfis de jogadores.

Para viabilizar essa experiência interativa, utiliza-se o motor de jogo (*game engine*). Trata-se de uma estrutura de *software* complexa responsável por gerenciar subsistemas essenciais como renderização gráfica, física, áudio, entrada de usuário, rede, *scripting*, e lógica de jogo. Um motor de jogo atua tanto na execução do jogo quanto no suporte ao desenvolvimento técnico e criativo. Existem dois entendimentos comuns sobre motores de jogo, sendo que um desses entendimentos é o motor de jogo como parte integrada de um jogo específico, em que todos os sistemas e subsistemas necessários para o funcionamento e operação por parte do usuário são desenvolvidos sob medida e acoplados diretamente ao código-fonte. O segundo entendimento é o motor de jogo como uma plataforma reuti-

lizável, muitas vezes acompanhada por um editor visual (por exemplo, Unreal Engine[1], ou Unity[2]), nessa concepção, o motor oferece uma infraestrutura genérica que pode ser usada em diversos projetos, com suporte para personalização e extensão via programação e edição gráfica, onde os desenvolvedores podem implementar e ajustar o *loop* de *gameplay*, inserir objetos, criar interações físicas, animar personagens, definir texturas, entre outras tarefas com facilidade, podendo também, exportar o trabalho para outros desenvolvedores (GREGORY, 2018; MESSAOUDI et al., 2017).

2.2 APIs Gráficas

De modo geral, os jogos eletrônicos podem ser considerados como *softwares* graficamente intensivos, isto é, dependem intensamente de uma *Graphics Processing Unit* (GPU), sendo este um componente crucial para um motor de jogo, para que os gráficos do jogo e interface gráfica sejam renderizados em tempo real da forma desejada pelos desenvolvedores, através da *pipeline* gráfica. O motor de renderização atua como ponte entre a lógica do jogo, *Central Processing Unit* (CPU) e a GPU, convertendo informações do mundo virtual em comandos para a *pipeline* gráfica produzir um *frame*. Além disso, a *pipeline* gráfica possui etapas fixas e programáveis, através de *shaders*, entendidos como pequenos programas, que podem ser orquestrados através de uma API gráfica, para enviar as instruções para a GPU realizar uma série de processamentos e cálculos (LENGYEL, 2019).

Essa *pipeline*, por sua vez, é o conjunto de etapas com outras várias sub-etapas, necessárias para preparar uma renderização em tempo real para a tela de um jogador, e este processo todo envolve um fluxo de dados contínuo, do carregamento de objetos de jogo e efeitos visuais, para a transformação em uma imagem bidimensional com base no ponto de vista do jogador usada para recortar uma parte da cena, objetos e informações de iluminação e reflexões acuradas, relativo às suas respectivas localizações em uma cena, sendo um processo fundamental para esse tipo de processamento de imagens para os jogos eletrônicos (AKENINE-MöLLER et al., 2018).

Parte do trabalho prévio à GPU ocorre na CPU, que prepara e executa um processamento inicial de geometria e recorte da cena, e então, invoca chamadas de desenho via *game engine* ou outra aplicação gráfica. Um jogo inicia o processo de gerar um frame ao chamar as funções de uma API gráfica, então o processador leva as instruções necessárias até a GPU, que executa uma série de etapas até renderizar um frame. É possível dizer que a CPU orquestra o que deve ser processado com poucos núcleos bem sincronizados, enquanto a GPU executa esse processamento com milhares de núcleos focados em processamento altamente paralelo para maior quantidade de dados processados. (HWU; KIRK; HAJJ, 2022; VARCHOLIK, 2014).

[1] <https://www.unrealengine.com/pt-BR>
[2] <https://unity.com/pt>

Portanto, existem diversas APIs gráficas, que promovem a comunicação entre a GPU e CPU, porém, se destacam o DirectX[3] desenvolvido pela Microsoft, que é o conjunto de APIs proprietárias da mesma, que operam principalmente nos sistemas Windows e Xbox e é amplamente utilizada para desenvolver jogos, e o Direct3D é uma API contida nesse conjunto, dedicada a renderização 3D (VARCHOLIK, 2014). O OpenGL[4] e Vulkan[5] são APIs gráficas multiplataforma desenvolvidas pela Khronos Group, onde uma grande diferença entre ambos, é que Vulkan é uma API mais nova e permite ter mais controle sobre o *hardware* de forma mais explícita, o que permite a criação de aplicações com mais desempenho (THE KHRONOS GROUP, 2025).

2.3 Sistemas Operacionais

O sistema operacional age como uma camada de *software* fundamental, pois permite que um jogo seja executado, providenciando os recursos de *hardware* de um computador, como a CPU, GPU e dispositions de entrada e saída, gerenciando vários programas ao mesmo tempo, alocando memória, e permitindo o acesso e carregamento dos arquivos de um jogo para a memória principal. Apesar de jogos serem sistemas contidos, eles ainda dependem de um sistema operacional para que haja a comunicação plena com o *hardware*, por causa dos *drivers* e certas APIs, que dependem de chamadas de sistema específicas (GREGORY, 2018).

O Linux moderno pode ser entendido como a junção do *kernel* Linux, criado por Linus Torvalds como um projeto pessoal, e a junção com o projeto GNU, iniciado por Richard Stallman em 1983, que hoje, o sistema resultante é distribuido software livre sob a licença GPL (GNU *General Public License*)[6] versão 2.0, essa licença garante que qualquer pessoa possa usar, estudar, modificar e redistribuir o software, desde que os termos da respectiva licença sejam respeitados (FREE SOFTWARE FOUNDATION, INC, 2025; LINUX KERNEL ORGANIZATION, 2024). Isso permite a existência de variações chamadas distribuições, que visam suprir alguma proposta. Onde cada distribuição opera com base no *kernel* Linux, com ou sem pequenas customizações visando algum objetivo específico como mais desempenho em tarefas de servidor, tarefas diárias ou tarefas em tempo real (CASTRO, 2016).

O kernel Linux adota um modelo monolítico, no qual todo o código central do sistema incluindo gerenciamento de memória, escalonamento de processos e *drivers* de dispositivo é executado em espaço privilegiado, e é desenvolvido ativamente, e a cada ano, novas funcionalidades são incorporadas ao código principal. Ele coordena diretamente a

[3] <https://learn.microsoft.com/en-us/windows/win32/directx>
[4] <https://www.khronos.org/opengl/>
[5] <https://www.vulkan.org/>
[6] <https://www.gnu.org/licenses/old-licenses/lgpl-2.0.html>

interação do *software* com o *hardware* e expõe interfaces de alto nível aos programas por meio das chamadas de sistema, um conjunto de funções padronizadas, ou API que atua como ponte entre aplicativos e recursos do kernel, e além disso, é possível interagir com essas chamadas de sistema através de certos comandos de terminal (TANENBAUM; BOS, 2014).

Para complementar suas capacidades sem reinicializar o núcleo inteiro, o Linux suporta módulos de *kernel* carregáveis, que podem ser incorporados *kernel* de forma relativamente fácil, e são geralmente carregados ao iniciar o computador, no carregamento do kernel, ou durante a execução do sistema. Esses componentes adicionais permitem a incorporação dinâmica de novos *drivers* e funcionalidades como suporte a equipamentos de *hardware* adicionais garantindo compatibilidade e flexibilidade ao sistema operacional em execução, como por exemplo, caso um componente de *hardware* não funcione corretamente, pode ser possível encontrar um módulo de kernel desenvolvido por um outro usuário e carregá-lo no sistema (LOVE, 2010).

Apesar de todos os sistemas baseados em Linux operarem sobre um mesmo *kernel*, o ecossistema Linux pode ser visto como fragmentado pelas opções diversificadas de tecnologias oferecidas pelas distribuições, que podem implementar tecnologias de ponta para o funcionamento do sistema ou realizar customizações ou alterações estéticas e de experiência de usuário, para atrair mais pessoas e oferecer uma melhor qualidade de serviço para uma determinada finalidade. Com propostas que variam entre ser uma distribuição amigável ao usuário, direcionada a servidores, testes, entre outras. Além das distribuições completas, é importante mencionar que também existe a opção de escolha de ambientes gráficos de *desktop* onde cada um pode ser escolhido pela sua estética, usabilidade ou utilização de recursos de *hardware* (CASTRO, 2016).

2.4 Ferramentas de Compatibilidade

Para promover uma solução para a compatibilidade entre sistemas Windows e Linux, e outros, o Wine [7], surgiu como uma solução de código aberto, que usa o conceito de camada de compatibilidade. O seu funcionamento consiste em traduzir as chamadas do sistema Windows para chamadas equivalentes do sistema operacional alvo em tempo real, sem utilizar virtualização, e isso permite que os jogos e *software*s nativos de Windows podem rodar diretamente no Linux. Porém esta ferramenta ainda não é uma solução que permite a compatibilidade universal, podendo ter instabilidades com certos programas e até não funcionar para outros, para isso, está em desenvolvimento ativo (DAVIES et al., 2024; JULLIARD, 2024).

O Wine possui uma plataforma que informa sobre a compatibilidade de milhares

[7] <https://www.winehq.org/>

aplicativos em geral, a *Wine Application Database* [8]. Para jogos em especial, uma forma que os usuários de Linux podem verificar o estado de funcionamento de jogos é a plataforma ProtonDB, que funciona por *crowd-sourcing* e é ativamente atualizada, com informações sobre a compatibilidade atual e a configuração dos sistemas dos usuários que relatarem o estado do desempenho (PROTONDB, 2025; WINEHQ, 2025).

Para jogos em especial, foi desenvolvido o Proton [9], uma camada de compatibilidade de código aberto desenvolvida pela Valve, tem se mostrado um marco no cenário de jogos para Linux. O Proton foi baseado no Wine, através de um *fork* do projeto, e lançado em 2018 junto ao Steam Play[10], o Proton permite que jogos de Windows que utilizam o DirectX, sejam executados no Linux com diversas otimizações extra sobre o Wine, que visam melhorar o desempenho de jogos, e possui uma integração fácil e rápida com o Launcher da Steam. O Proton converte chamadas Direct3D 8, 9, 10 e 11 para Vulkan ou OpenGL, dependendo do suporte do sistema a essas tecnologias, através do DXVK, e a conversão do DirectX 12 para o Vulkan através do VKD3D (REBOHLE et al., 2025; VALVE, 2018).

2.5 Ferramentas de Monitoramento e Métricas de desempenho

Existem, como elucidado por Gregg (2021), existem diversas ferramentas de observabilidade e monitoramento de sistemas disponíveis, e para propósitos diferentes. Cada ferramenta de observabilidade possui uma proposta diferente, como observar o consumo geral de recursos de um sistema com métricas de desempenho padrão como utilização de disco, memória, CPU e redes, e também existem ferramentas para monitorar processos do sistema, ajudando a entender o comportamento de processos individuais, entre outros tipos.

Ao analisar um sistema ao executar jogos eletrônicos, o mais comum são ferramentas que monitoram as métricas de desempenho do sistema inteiro, coletando diversas informações de sensores do *hardware*, e as métricas observadas serão principalmente as relacionadas a CPU, GPU e memória, como a utilização de CPU, utilziação de GPU, utilização de memória, quadros por segundo e *frametime*. A utilização de CPU é dada pelo tempo em que a CPU ativamente executou alguma tarefa, sendo medida em porcentagem e sua temperatura é o resultado da dissipação de potência decorrente do uso e de características físicas do processador e da solução térmica, como o dissipador de calor (GAMEBENCH, 2019; JULIAN, 2017).

Para a placa de vídeo (GPU), a utilização também pode ser vista em termos de porcentagem, e também possui um contador próprio de memória dedicada (VRAM) em

[8] <https://appdb.winehq.org/>
[9] <https://github.com/ValveSoftware/Proton>.
[10] <https://steamcommunity.com/games/221410/announcements/detail/1696055855739350561>.

gigabytes, que funciona de forma parecida com a memória RAM, é dedicada a armazenar temporariamente dados de imagens e texturas de objetos e informações de luz de jogos, sendo que quanto mais, melhor, porém não é o único determinante de desempenho de uma placa de vídeo, pois esse desempenho depende de outros componentes como a contagem de núcleos e velocidade de cada um deles (HWU; KIRK; HAJJ, 2022).

A métrica de quadros por segundo (FPS) é uma das métricas mais importantes ao analisar o desempenho de um jogo. De acordo com NVIDIA Corporation (2022) o seu cálculo é geralmente dado pela coleta de dados a partir do *frametime*, que se da renderização da *pipeline* gráfica, entre a detecção da instrução "present" e "display", e então é feito o cálculo de frames renderizados em um segundo. A medida de "1% low" dos FPS representa as quedas bruscas de frame, que pode ser calculado pelo 1° percentil dos piores frames registrados, que são responsáveis por um usuário notar uma interrupção brusca na sua tela, caso ocorra abaixo de uma certa quantidade de frames, como por exemplo, uma queda brusca abaixo de 60 quadros por segundo.

Para o monitoramento de jogos Linux, o MangoHUD[11] é uma solução para monitoramento de desempenho. O MangoHUD exibe informações de desempenho em tempo real na tela, através de uma sobreposição gráfica, que mostra diversas métricas ao usuário e também permite a gravação dessas métricas em arquivo ".csv" (FLIGHTLESSMANGO, 2025). Assim como algumas alternativas para Windows, como o PresentMon[12] da Intel, o FrameView[13] da NVIDIA e a combinação do MSI Afterburner [14] + Rivatuner Statistics Server[15] , captura métricas relacionadas ao *hardware* e ao *software*, como o FPS e *frametime* (DAMNJANOVIC, 2024; NVIDIA CORPORATION, 2022).

2.6 Análise de Desempenho

A análise de desempenho de sistemas envolve a aplicação de diversas técnicas e ferramentas que funcionam em conjunto para entender o comportamento de sistemas computacionais, com a finalidades diferentes, como realizar predições de capacidade, otimizar um sistema para lidar melhor com carga, entre outros. E, antes de iniciar a análise e avaliação de desempenho de um sistema, deve-se definir primeiro os objetivos do estudo, listar as variáveis que podem afetar o desempenho, discriminar resultados desejados e indesejados e, então, escolher a técnica de avaliação adequada (MENASCE; ALMEIDA; DOWDY, 2004).

De acordo com Jain (1991), as principais técnicas de avaliação de desempenho são:

[11] <https://github.com/flightlessmango/MangoHud>
[12] <https://game.intel.com/us/intel-presentmon/>
[13] <https://www.nvidia.com/en-us/geforce/technologies/frameview/>
[14] <https://www.msi.com/Landing/afterburner/graphics-cards>
[15] <https://www.guru3d.com/page/rivatuner-rtss-homepage/>

(*i*) a medição real; (*ii*) a modelagem analítica; e (*iii*) a simulação. Para que essa análise seja realizada, existe a medição dos sistemas através de ferramentas de monitoramento de sistemas, envolvendo o monitoramento de redes, CPU ou GPU, dependendo do tipo de sistema. Além disso, podem ser empregadas modelagem analítica e simulação, as quais utilizam leis operacionais (e.g., Lei de Little, Lei da Utilização, Lei da Demanda, Lei do Fluxo Forçado), Teoria de Filas, Processos Birth-Death e Modelos de Markov como modelos descritivos e preditivos de como um sistema reage, a partir dos dados disponíveis.

Uma das formas de testar um sistema é através do *benchmarking*, que podem ser considerados como testes controlados, e de acordo com Gregg (2021), entre as categorias de *benchmark* destacam-se os testes focados em operações pontuais, os *benchmarks* de nível de aplicação, para verificar como uma aplicação reage ao ser submetida a uma carga sintética e os testes de estresse (*stress tests*), que submetem o sistema a condições extremas. E, então, com base nos resultados desses testes, utilizando ferramentas de monitoramento com um teste de estresse em ambiente controlado podem identificar problemas sem que o sistema vá para produção ou evitando criar um modelo de predição futura do sistema e, posteriormente, aplicar uma técnica de avaliação estatística para análises dos resultados de desempenho.

Dentre as técnicas de avaliação estatística existem os projetos fatoriais, neles, os fatores referem-se às variáveis de um sistema, que podem ter diversas alternativas, como, por exemplo, processadores diferentes, onde os níveis de um fator são as variações ou alternativas de um fator. As replicações se referem a quantas vezes um experimento foi executado e as interações se referem ao caso onde dois ou mais fatores possuem níveis que dependem um do outro. A análise dos efeitos dos fatores é feita por meio de ferramentas estatísticas como a ANOVA, testes estatísticos por meio de intervalos de confiança dos fatores e inspeções visuais dos quantis teóricos e empíricos dos resíduos. E, além disso, pode ser que haja a necessidade de aplicar transformação dos dados antes de colocá-los para a análise e para verificar essa necessidade são feitos alguns testes visuais, para verificar a normalidade dos dados (MONTGOMERY, 2012).

A ANOVA, é uma ferramenta estatística importante para o projeto fatorial, pois tem a finalidade de analisar o quanto os grupos dos fatores e seus níveis são significativos para a análise e indicar uma direção para a realização da análise ao rejeitar a hipótese nula, que é quando não há diferença estatisticamente significativa entre os grupos de fatores e todos os fatores estão dentro da média. Então, em seguida, a análise dos intervalos de confiança fornece uma visão sobre o quanto cada fator, seus níveis e interações impactaram no desempenho geral do sistema que será testado e essas duas ferramentas em conjunto fornecem uma visão com uma grande sensibilidade para visualizar os impactos dos fatores, mesmo que pequenos (TURNER; THAYER, 2001).

A inspeção visual ocorre através de gráficos dos quantis e histogramas dos. O

gráfico de Quantil-Quantil Normal compara um quantil teórico e a distribuição dos dados ajustados a uma curva normal e, caso a maioria dos dados se aproximem da linha teórica, pode indicar que os dados seguem uma distribuição aproximadamente normal. Para os resíduos, um gráfico desejado possui uma dispersão de dados sem tendências visíveis e sem padrões definidos, onde os casos em que existem padrões claros ou "clusters" podem indicar um número insuficiente de amostras ou que o modelo não foi bem especificado (JAIN, 1991).

2.7 Trabalhos Relacionados

Os estudo para entender o desempenho relativo a sistemas operacionais é amplamente explorado através da análise de como diversos sistemas de configurações de *hardware* e *software* desempenham, abrangendo diversas áreas, como aplicações de servidores e código, enquanto a análise que envolve jogos eletrônicos geralmente envolve a tarefa de analisar configurações de *hardware* diferentes para executar jogos e auxiliar os consumidores a escolherem a melhor combinação de *hardware*, tentando balancear desempenho e preço, ou apenas a experimentação de peças exóticas.

Esta seção irá abordar alguns trabalhos relacionados, considerados relevantes para esse tipo de pesquisa. Cada pesquisa geralmente utiliza de técnicas visuais e de análise de dados para realizar conclusões, com base em métricas como a média das medições, enquanto a abordagem deste trabalho é uma tentativa de utilizar o projeto fatorial aplicado à análise de desempenho de jogos.

O estudo de Messaoudi et al. (2017) explora a game engine Unity3D para amostras de jogos em diversos tipos de sistemas diferentes, como Windows, Linux e Android em amostras de jogos em Unity 3D classificados entre gêneros diferentes. Entre as descobertas do estudo, encontrou que o tempo necessário para gerar um frame depende da qualidade da resolução do jogo, do tamanho da tela e do tipo ou gênero do jogo, além de que o motor de renderização é responsável por até 70% da utilização de CPU, o que neste contexto pode indicar que a GPU pode estar limitando o desempenho, pois grande parte do tempo, a CPU ficou em espera por resposta da GPU, além do que o consumo de GPU e CPU mostraram uma forte correlação entre si. Esse estudo serve como base para estudar os efeitos dos sistemas operacionais no desempenho e as métricas principais para realizar essa análise e o que esperar ao submeter sistemas diferentes a cargas como jogos diferentes, e como o *hardware* pode afetar a resposta de módulos de game engine, porém nem todo sistema operacional para as máquinas pode ser testado em todas as configurações de *hardware* compatíveis.

O estudo de Sulaiman e Raffi (2021) tem o objetivo de analisar como aplicações comuns do dia-a-dia se comportam em dois sistemas operacionais diferentes, o Windows 10

e Linux Mint. Para isso, utilizaram algumas ferramentas de monitoramento como o NZXT CAM, Gnome System Monitor, Gnome Terminal, VBScript, entre outros, e escolheram os programas Steam, Discord, LibreOffice, Google Chrome e Mozilla Firefox. Com isso, conseguiram identificar que o Linux Mint quando parado consumiu poucos recursos de CPU e RAM, com respectivamente 1,8% e 24%, e o Windows 10, consumiu aproximadamente o dobro de CPU e RAM, com respectivamente 5% e 41%, além do Linux Mint conseguir abrir os programas um pouco mais rápido e também encontrou que o desempenho de rede entre os dois sistemas não é significativo. Esse estudo pode ser usado como uma forma de entender o que pode se esperar da estabilidade básica de cada sistema e se há alguma perda de desempenho já em aplicações básicas, que apesar de se basearem em elementos gráficos, não são graficamente intensivas como jogos, entretanto, o estudo realiza apenas uma replicação para cada sistema, e também, as ferramentas de monitoramento de recursos usadas não permitem uma coleta temporal, se baseando apenas na observação de um momento do sistema.

O estudo de Kopel e Bożek (2023) tem o objetivo de avaliar a efetividade das ferramentas de compatibilidade e se há perda de desempenho entre Windows (Windows 10) e Linux (Pop OS 22.04) quando o Proton é utilizado, e além disso, relata possíveis frustrações que podem surgir quando um usuário comum tente rodar algum jogo e outros problemas técnicos inerentes da solução de compatibilidade e dos *drivers*[16] na época da publicação. O estudo encontrou uma diferença de desempenho, com perdas consideráveis através do Proton, em comparaçao ao Windows, além de quedas bruscas de desempenho muito piores, através do 0.1% low, fora um uso maior de VRAM em Linux, e também relatou a impossibilidade de executar alguns jogos. Esse estudo serve como base para compreender se há uma perda de desempenho decorrente da camada de compatibilidade, executar jogos com APIs gráficas nativas de Windows em Linux, através da camada de compatibilidade. O estudo realiza as execuções em diversos *presets* gráficos, porém realiza uma replicação válida para cada um desses *presets*, e também usa um método simples de comparação, o que limita a estimativa de erros experimentais ou eventuais erros atípicos de uma replicação.

O estudo de Martinovic, Balen e Cukic (2012) tem o objetivo de realizar a análise de desempenho dos sistemas Windows 7, Windows Vista e Windows XP através do *benchmark*, utilizando sete ferramentas especializadas em *benchmark*s de *hardware*, como o PCMark05, 3DMark06, SPECviewperf, ScienceMark, Everest eSuperPI, e o D-ITG. Realizou 3 experimentos: o primeiro em um sistema com *hardware* considerado High-End, o segundo em um sistema com *hardware* considerado Low-End e um terceiro experimento focado em redes, com dois computadores idênticos. O primeiro experimento mostrou que o Windows 7 possui um pouco mais de desempenho, em comparação aos outros sistemas,

[16] *Driver*: programa que fornece uma comunicação entre o dispositivo e a abstração de *software*.

e o Windows XP consumiu menos memória quando parado. No segundo experimento, o Windows XP obteve o melhor desempenho e em geral, com um melhor gerenciamento de CPU. O terceiro experimento, o Windows 7 e Vista obtiveram o melhor desempenho em TCP e o Windows XP obteve um melhor desempenho em UDP. Este estudo tem a importância de demonstrar que mesmo entre a mesma família de sistemas operacionais, podem existir alguma variação entre os resultados de desempenho, e além disso, os resultados podem variar de acordo com a capacidade do *hardware*, porém o estudo foca nas versões de 32-bits dos sistemas.

2.8 Considerações Finais

O capítulo apresentou os conceitos básicos para compreender a importância e o fundamento em realizar uma análise de desempenho de jogos diferentes em sistemas operacionais ou distribuições diferentes.

Inicialmente, foi apresentado o conceito de jogos e game engines, que são programas complexos que fazem um uso intensivo do *hardware*, e principalmente de GPUs, para apresentar uma imagem de suas ações dentro de um jogo para um usuário em tempo real, e como os jogos podem ser diferentes entre si.

Em seguida, foi apresentado o conceito dos sistemas operacionais Linux e as APIs gráficas e as camadas de compatibilidade para jogos. Os sistemas operacionais Linux, através das suas distribuições, podem implementar várias tecnologias diferentes, e essas tecnologias podem atrair um usuário por essa tecnologia, seja pela promessa de desempenho ou porque atende às suas necessidades. Porém, o fator comum entre todo o ecossistema é a incompatibilidade inerente em relação às chamadas de sistema da API do Windows, e isso se estende às APIs gráficas como o DirectX, que é uma tecnologia amplamente utilizada em jogos, por fornecer uma interface para programar e abstrair elementos gráficos utilizados em jogos. Para traduzir essa API para os sistemas Linux executarem os jogos, é usada uma camada de compatibilidade, que traduz o DirectX para Vulkan ou OpenGL.

Então, a partir dos conceitos envolvendo a análise de desempenho, as ferramentas de monitoramento e as métricas de desempenho para jogos, é possível formular um modelo para analisar o desempenho de um sistema. E, quando aplicado a jogos em conjunto às ferramentas de monitoramento de sistema e com as métricas coletadas, podem auxiliar na avaliação de como um sistema está utilizando o seu *hardware*, ou pode ajudar a diagnosticar o desempenho na etapa de desenvolvimento de um jogo.

Por fim, os trabalhos relacionados formam os fundamentos para realização de um estudo envolvendo jogos eletrônicos em diferentes sistemas operacionais, mais especificamente, no estudo da avaliação e análise de desempenho de jogos eletrônicos em distribuições Linux.

3 Ferramentas e Métodos

O capítulo apresenta as ferramentas e tecnologias usadas para realizar os experimentos, além de detalhes sobre cada fator que influencia o desempenho do ambiente de execução dos jogos. A Seção 3.1 introduz as tecnologias de *hardware* e *software* utilizadas no experimento, em que a Subseção 3.1.1 descreve o *hardware* em detalhes, por sua vez, a Subseção 3.1.2 descreve a ferramenta de monitoramento e a Subseção 3.1.3 descreve as ferramentas para a análise das métricas. A Seção 3.2 apresenta a definição dos sistemas operacionais selecionados. Na Seção 3.3 são definidos os jogos selecionados. Em seguida, a Seção 3.4 define as etapas de instalação e coleta de dados para cada jogo em cada sistema operacional. Na Seção 3.5 são descritos os resultados desejados e indesejados ao executar jogos por parte do que um usuário busca durante a execução de jogos. A Seção 3.6 define os parâmetros que serão utilizados no projeto fatorial completo. Por fim, a Seção 3.7 descreve todas as etapas do processamento de dados.

3.1 Ferramentas e Tecnologias Utilizadas

As ferramentas e tecnologias referem-se às configurações de *hardware*, o *software* utilizado para coletar as métricas necessárias para a realização do estudo e as ferramentas que serão utilizadas para realizar o processamento, transformação e análise das métricas coletadas.

3.1.1 Configuração de *hardware*

As configurações de *hardware* utilizadas para executar os experimentos são descritos a seguir:

- Processador: AMD Ryzen 7 5700X3D;
- Placa de Vídeo: NVIDIA RTX 4060;
- Placa Mãe: MSI MPG B550 Gaming Plus;
- Memória RAM: 32GB DDR4 CL22;
- Armazenamento: WD Black SN770 1TB.

A placa de vídeo é uma RTX 4060, que, de acordo com o TechPowerup (2023), foi lançada pela NVIDIA em 2023, possui 8GB de memória de vídeo dedicada, com uma largura de banda de 128 bits, do tipo GDDR6, operando a 17000 MHz de taxa de transferência

máxima. Além disso, possui 3072 núcleos CUDA, com clock gráfico entre 210 MHz e 3105 MHz, além de outras unidades dedicadas a *raytracing* e cálculos computacionais de *machine learning*. E também, possui suporte nativo ao DirectX 12, Vulkan e OpenGL mais recentes. Funciona no slot PCIe 4.0 x16, porém é eletricamente PCIe 4.0 x8.

O processador é um Ryzen 7 5700X3D, lançado pela AMD em 2024 para a plataforma AM4, utilizando a arquitetura Zen 3. Possui 8 núcleos e 16 threads, com um clock máximo de 4.1 GHz (efetivamente 4.05 GHz). Possui a tecnologia 3D-Vcache, que aumenta a quantidade do cache L3 para 96MB, em comparação ao Ryzen 7 5700X com 32MB de cache L3 e 4.6 GHz de clock máximo. A vantagem de possuir mais cache L3 tem relação a quantidade de acessos á memória RAM que algumas aplicações realizam, e isso, para certos jogos, pode melhorar a estabilidade e o desempenho geral. Para o experimento, o processador será usado em *Stock*, sem modificações manuais no *Performance Boost Overdrive* (PBO) e não necessitou da instalação de drivers adicionais (CPU-WORLD, 2024; CPU-WORLD, 2022; ROACH; CONNATSER, 2023).

A placa mãe é uma MSI MPG B550 Gaming Plus, com o *Chipset* B550, lançada pela Micro-Star International (MSI), que possui suporte ao PCIe 4.0 (16 gigatransfers por segundo ou GT/s) no slot principal para placa de vídeo e SSD NVME, e outros slots auxiliares com o PCIe 3.0 (com 8 GT/s). Possui um VRM (*Voltage Regulator Module* ou Módulo Regulador de Tensão) de 13 fases e dissipadores térmicos capazes de resfriar esses componentes de forma efetiva, de forma geral, é capaz de extrair o potencial do processador sem causar problemas térmicos na placa ou desacelerar o processador (MICRO-STAR INTERNATIONAL, 2023).

O SSD NVME gen 4 é um WD Black SN770, lençado pela empresa Western Digital, que possui leitura de 5150MB/s e gravação de 4900MB/s. Apesar da alta velocidade, o seu impacto no desempenho de jogos se limita ao carregamento inicial (DOWNING; WEBSTER, 2022).

3.1.2 Ferramenta para Monitoramento

A ferramenta de monitoramento utilizada foi o MangoHud, que salvará logs contendo todas as métricas descritas pela Tabela 4, podendo ser configurado para outras métricas conforme a documentação oficial[1] por meio das variáveis do arquivo de configuração.

3.1.3 Ferramentas para Análise

Para a análise das métricas coletadas e a realização do projeto fatorial, as ferramentas utilizadas neste trabalho são:

[1] <https://github.com/flightlessmango/MangoHud?tab=readme-ov-file#environment-variables>.

- Python[2] como linguagem de programação (PYTHON SOFTWARE FOUNDATION, 2024b);
- pip[3] como gerenciador de pacotes Python (PYTHON SOFTWARE FOUNDATION, 2024a);
- Numpy[4] como a biblioteca para cálculos gerais (NUMPY TEAM, 2025);
- Pandas[5] para gerenciar tabelas e dataframes (THE PANDAS DEVELOPMENT TEAM, 2025);
- statsmodels[6] para o cálculo de ANOVA (PERKTOLD et al., 2024);
- scipy[7] para o cálculo da tabela z ou t e a transformação Box-Cox (SCIPY CONTRIBUTORS, 2025);
- Matplotlib[8] para a criação de gráficos (MATPLOTLIB DEVELOPMENT TEAM, 2025);
- Seaborn[9] também para a criação de gráficos (WASKOM, 2024).

3.2 Definição de Fatores: Seleção de Sistemas Operacionais

Os sistemas operacionais selecionados foram: Arch, Bazzite, CachyOS, Debian, Fedora e Ubuntu. Para todos, o ambiente de *desktop* utilizado foi o GNOME, e na sua maioria, o protocolo de composição de janelas utilizado foi o Wayland, com exceção do Debian, e o sistema de arquivos EXT4, exceto o Bazzite, com BTRFS. Referente à versão do Proton, o recomendado é sempre utilizar uma versão mais recente, porém, para evitar uma possível interferência de versão entre os intervalos de coleta, a versão selecionada do Proton foi a Proton 9.0-4, através do Steam Play.

A Tabela 1 descreve um resumo dos sistemas operacionais selecionados, suas versões, a versão dos *drivers* de vídeo disponíveis e a versão disponível de cada *kernel*.

[2] <https://www.python.org/>, versão: 3.13.3.
[3] <https://pypi.org/project/pip/>, versão: 25.0.1.
[4] <https://numpy.org/>, versão: 2.1.3.
[5] <https://pandas.pydata.org/>, versão: 2.2.3.
[6] <https://www.statsmodels.org/stable/index.html>, versão: 0.14.4.
[7] <https://scipy.org/>, versão: 1.15.2.
[8] <https://matplotlib.org/>, versão: 3.9.3.
[9] <https://seaborn.pydata.org/>, versão: 0.13.2.

Tabela 1 – Versões dos Sistemas Operacionais.

Nome	Arch	Bazzite	CachyOS	Debian	Fedora	Ubuntu
Versão	Rolling	41	Rolling	12.10	41	24.10
Driver	570.133.07	570.133.07	570.133.07	535.216.01	570.133.07	560.35.03
GNOME	48	47	48	43.9	47	47
Kernel	6.13.8-arch1-1	6.13.9-103.bazzite	6.14.1-2-cachyos	6.1.0-32-amd64	6.13.9-200.fc41	6.11.0-21-generic

Fonte: Linux, Informações coletadas dos detalhes de sistema, 2025.

O Arch Linux[10] é uma distribuição independente, que utiliza o pacman como instalador de pacotes e o método de disponibilização de atualizações de pacotes é o "Rolling", significando que as atualizações são disponibilizadas rapidamente para que os usuários tenham versões atualizadas de cada pacote no sistema e não possui uma versão fixa definida, e por padrão, não possui *softwares* prontos para uso. O seu padrão *kernel* possui poucas modificações, sendo próximo do *kernel* "*vanilla*", e também tem suporte a versões alternativas do *kernel* como a variação Zen, porém apenas a versão padrão será usada no estudo (ARCH LINUX WIKI CONTRIBUTORS, 2025a; ARCH LINUX WIKI CONTRIBUTORS, 2025b).

O Bazzite[11] é um sistema operacional baseado no Fedora Silverblue, uma distribuição do Fedora que utiliza o conceito de *desktops* imutáveis através do rpm-ostree, onde as aplicações por padrão funcionam em um sandbox e a cada instalação de *software* fora do flatpak, é necessário reiniciar o sistema. O seu *kernel* possui extensões de compatibilidade para dispositivos de mão (Steam Deck, HOG Ally, etc.) e inclui o agendador de CPU customizado BORE, para tentar aprimorar o desempenho do sistema (BAZZITE, 2023; BAZZITE, 2025).

O CachyOS[12] é um sistema baseado no Arch Linux, com foco em jogos, e disponibiliza vários programas para facilitar a usabilidade e também a modificações de *kernel*, que podem ser mais efetivas para processadores das arquiteturas x86-64-v3, x86-64-v4 e Zen4+. O seu *kernel* possui algumas modificações focadas em jogos, utilizando o agendador de CPU BORE, construído com otimizações de compilação, para tentar extrair mais desempenho, além de possuir módulos de *kernel* para GPUs NVIDIA inclusos (CACHYOS, 2025b; CACHYOS, 2025a).

O Debian[13] é um sistema independente, que utiliza o Advanced Packaging Tool (APT) para instalar seus pacotes, com a extensão ".deb" pelo terminal ou alguma loja de aplicativos inclusa no ambiente gráfico (por exemplo, GNOME Software) e foca na estabilidade de pacotes, com o modelo de lançamento periódico com versões estáveis de

[10] <https://archlinux.org/download/>
[11] <https://bazzite.gg/>
[12] <https://cachyos.org/>
[13] <https://www.debian.org/distrib/>

longa duração, porém possui outras versões como a Experimental, que possui pacotes em teste e a "Sid" que possui pacotes na última versão possível, que podem ser instáveis, e essa versão em específico não é recomendada para o uso cotidiano. A instalação padrão possui diversos pacotes que visam facilitar a experiência do usuário e seu *kernel* possui poucas modificações, sendo perto do *kernel* Vanilla, com algumas remoções devido a questõed de termos de licença (DEBIAN PROJECT, 2025; DEBIAN KERNEL HANDBOOK PROJECT, 2023).

O Fedora Workstation[14] é uma distribuição independente que utiliza o DNF como gerenciador de pacotes, e instala pacotes ".rpm" e também inclui a opção de instalar flatpaks por padrão, através de uma loja de aplicativos proporcionada por algum ambiente gráfico. Por padrão, a versão Workstation inclui diversos pacotes instalados para proporcionar ao usuário uma usabilidade do cotidiano, e possui um ambiente de *desktop* com poucas modificações estéticas, na versão Fedora Server é possível escolher entre diversos ambientes de *desktop* e pacotes para incluir na instalação. O seu *kernel* possui algumas modificações, mas se aproxima da versão Vanilla (FEDORA PROJECT, 2025b; FEDORA PROJECT, 2025a).

O Ubuntu *desktop*[15] é uma distribuição mantida pela empresa Canonical, e é baseada no Debian e também usa o APT como gerenciador de pacotes, e pode instalar os pacotes ".deb" em conjunto aos Snaps. Possui um foco na usabilidade e desempenho, com dois tipos de lançamento de versão, sendo o LTS uma versão fixa de lançamento periódico, com suporte comum de até 5 anos de e 10 anos de suporte estendido, e uma versão com suporte curto de 9 meses, mas com *softwares* mais recentes e o seu *kernel* possui poucas modificações e tem o suporte ao Live Patching em versões dedicadas (CANONICAL LTD., 2025b; CANONICAL LTD., 2025a).

3.3 Definição de Fatores: Seleção de Jogos

A técnica de avaliação será o *benchmarking*, que consiste em uma carga sintética e previsível, que nos casos dos jogos, geralmente consiste em uma cena pré programada para ser um teste de estresse durante um determinado tempo, com o objetivo de avaliar a capacidade do *hardware* do jogador, para que esse jogador escolha as melhores opções gráficas para executar o jogo no seu *hardware*, e também, para orientar a decisão de compra de um componente de *hardware* com base no *benchmark* de outros jogadores.

Inicialmente, os jogos selecionados para teste são os que possuem uma ferramenta interna de *benchmark* para fornecer uma base para que os experimentos sejam realizados de forma padronizada e replicável, uma abordagem parecida com a utilizada no estudo de

[14] <https://fedoraproject.org/>
[15] <https://ubuntu.com/download>

Kopel e Bożek (2023), pois cada ferramenta de *benchmark* de cada jogo possui um tempo certo de execução, e mesmo que não represente de forma completamente precisa um cenário completo de jogo, onde há a interação de usuários, cenas cheias de objetos, entre outros, ainda serve como um ponto de referência de como cada jogo se comporta em cada sistema operacional, salvo os casos de *bugs* ou outros problemas aleatórios que podem aparecer durante um jogo. Os jogos foram selecionados usando o critério de que haja ferramentas internas de *benchmark*, e que estejam devidamente adquiridos na biblioteca da Steam, e a seleção buscou diversificar os gêneros dos jogos, para que os experimentos possam abranger uma amostra diversificada de jogos. Os jogos selecionados foram: "Assassin's Creed Origins", "Far Cry 6", "Killer Instinct" "Metro Exodus Enhanced" e "War Thunder".

O jogo Assassin's Creed Origins[16] foi desenvolvido pela Ubisoft Montreal e publicado pela Ubisoft, lançado em 2017 é um jogo de ação e aventura em terceira pessoa de mundo aberto. A sua cena de *benchmark* possui aproximadamente 120 segundos e é composta por um passeio pelo cenário do jogo com diversos elementos gráficos como partículas e texturas de alta resolução, além de objetos estáticos, como construções e objetos móveis, como personagens não jogáveis (UBISOFT, 2017a).

O jogo FarCry 6[17] foi desenvolvido pela Ubisoft Toronto e publicado pela Ubisoft, lançado em 2021 é um jogo first-person shooter de ação e aventura em mundo aberto. A cena de *benchmark* tem a duração de aproximadamente 60 segundos, e é composta de elementos gráficos diversificados, com a composição de objetos estáticos, partículas e personagens não jogáveis que se movem durante a execução (UBISOFT, 2021).

O jogo Killer Instinct[18] é desenvolvido pela Iron Galaxy Studios, publicado pela XBOX Game Studios, lançado em – é um jogo de luta e a sua taxa de quadros é travada em 60 FPS. O *benchmark* dura aproximadamente 50 segundos, e é composto pela metade de uma partida pré programada em um cenário com diversos objetos e dois personagens que emitem diversas partículas móveis, que servem para representar a situação onde o jogo pode apresentar a maior dificuldade para executar no *hardware* (XBOX GAME STUDIOS, IRON GALAXY STUDIOS, 2013).

O jogo Metro Exodus[19] Enhanced Edition foi desenvolvido pela 4A Games e publicado pela empresa Deep Silver, lançado em 2021 é um jogo de ação first-person shooter. A sua versão Enhanced possui o aprimoramento de visuais e Ray Tracing para placas de vídeo modernas. A sequência de *benchmark* possui aproximadamente 105 segundos e apresenta um passeio por um cenário do jogo com diversos efeitos de partículas, texturas e reflexões, com uma quantidade moderada de elementos móveis. Para o experimento, foi utilizado a ferramenta de *benchmark* do próprio jogo, com todas as opções no máximo

[16] <https://store.steampowered.com/app/582160/Assassins_Creed_Origins/>
[17] <https://store.steampowered.com/app/2369390/Far_Cry_6/>
[18] <https://store.steampowered.com/app/577940/Killer_Instinct/>
[19] <https://store.steampowered.com/app/412020/Metro_Exodus/>

exceto a "Shading Quality", ajustada no *Low* porque as outras opções de qualidade resultavam em um erro desconhecido (DEEP SILVER, 4A GAMES, 2021a).

O jogo War Thunder[20] é desenvolvido pela Gaijin Entertainment e publicado pela empresa Gaijin Network Ltd, lançado em 2013 e atualizado constantemente, é um jogo de combate veicular com elementos históricos e física complexa. O jogo possui diversas cenas de *benchmark*, a escolhida para o experimento foi a "Tank Battle (CPU)" com uma duração de aproximadamente 60 segundos, que é composta por um cenário em grande escala onde vários veículos se movem e emitem partículas e possui diversos elementos estáticos e móveis sendo renderizados na tela. Apesar do jogo conseguir operar em DirectX e Vulkan, no teste, apenas a versão Vulkan foi executada (GAIJIN ENTERTAINMENT, 2025).

Tabela 2 – Informações Gerais dos Jogos Selecionados.

Jogo	Assassin's Creed Origins	Far Cry 6	Killer Instinct (2013)	Metro Exodus Enhanced Edition	War Thunder
Gênero	Ação e RPG	first-person shooter	Luta	first-person shooter e sobrevivência	Combate Veicular e Simulação de Voo
Ano	2017	2021	2013	2021	2013
Desenvolvedores	Ubisoft Montreal	Ubisoft Toronto	Iron Galaxy Studios	4A Games	Gaijin Entertainment
Tempo de *benchmark*	120 segundos	60 segundos	50 segundos	105 segundos	60 segundos

Fonte: Informações coletadas das páginas oficiais dos jogos: Assassin's Creed Origins (UBISOFT, 2017a), Far Cry 6 (UBISOFT, 2021), Killer Instinct (XBOX GAME STUDIOS, IRON GALAXY STUDIOS, 2013), Metro Exodus Enhanced Edition (DEEP SILVER, 4A GAMES, 2021a), War Thunder (GAIJIN ENTERTAINMENT, 2025).

[20] <https://store.steampowered.com/app/236390/War_Thunder/>

A Tabela 3 compila os requisitos de sistema de cada jogo com base nas especificações fornecidas pelos respectivos desenvolvedores, através das páginas oficiais de cada um.

Tabela 3 – Requisitos de *hardware* Recomendados para 1080P.

Jogo	Assassin's Creed Origins	Far Cry 6	Killer Instinct (2013)	Metro Exodus Enhanced Edition	War Thunder
CPU	Intel Core i7-3770 ou AMD FX-8350	AMD Ryzen 5 3600X ou Intel i7-9700	Intel Core i5-750 ou AMD Phenom II X4 965	CPU Octa-Core	Intel Core i5 ou equivalentee
GPU	NVIDIA GTX 760 ou AMD R9 280X	AMD RX 5700 XT ou NVIDIA RTX 2070 Super	NVIDIA GeForce GTX 670 ou AMD Radeon HD 7950	NVIDIA RTX 2070/3060 ou AMD RX 6700 XT	NVIDIA GeForce GTX 1060 ou Radeon RX 570
RAM	8 GB	16 GB	8 GB	16 GB	16 GB
Disco	42 GB	60 GB	10 GB	80 GB	95 GB
API Gráfica	DirectX 11	DirectX 12	DirectX 11	DirectX 12, Vulkan	DirectX 11, DirectX 12, Vulkan

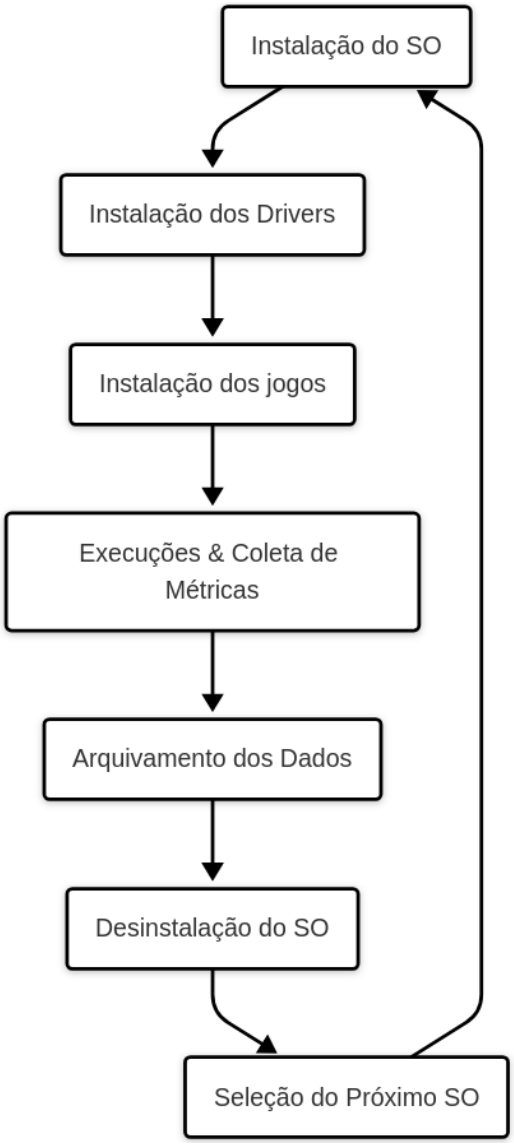
Fonte: Informações coletadas das páginas oficiais dos jogos: Assassin's Creed Origins (UBISOFT, 2017b), Far Cry 6 (UBISOFT, 2022), Killer Instinct (STEAM, XBOX GAME STUDIOS, IRON GALAXY STUDIOS, 2013), Metro Exodus Enhanced Edition (DEEP SILVER, 4A GAMES, 2021b), War Thunder (STEAM, GAIJIN ENTERTAINMENT, 2025).

Para a coleta de dados, cada jogo será executado várias vezes na opção gráfica que utiliza de maneira mais intensa os recursos computacionais em cada sistema operacional e, posteriormente, os dados serão analisados.

3.4 Planejamento dos experimentos

O planejamento dos experimentos seguirá a metodologia de instalar um sistema operacional, instalar os *drivers* de vídeo, instalar os jogos, executar cada jogo e suas replicações em cada sistema e arquivar as execuções completas em uma pasta para a análise futura, desinstalar um sistema e selecionar o próximo, conforme o fluxograma da Figura 1

Figura 1 – Fluxograma de Instalação dos sistemas operacionais e jogos.



Fonte: Elaborado pelo autor.

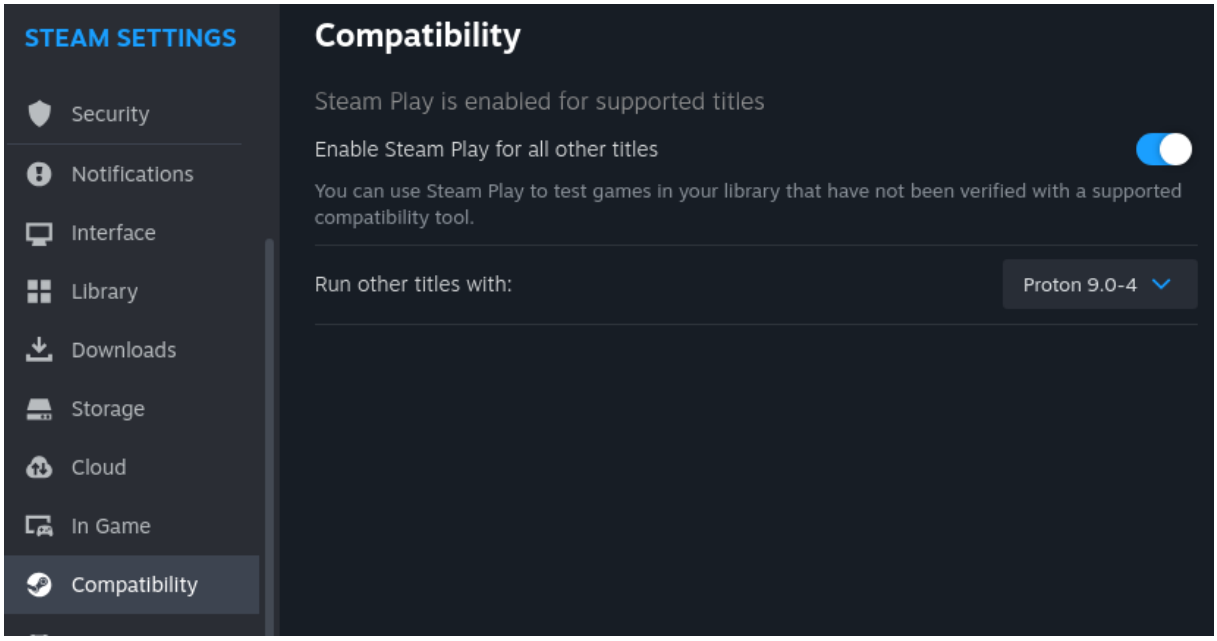
A instalação de cada sistema operacional é um processo relativamente simples, apenas variando o seu procedimento de acordo com o instalador de cada distribuição, que pode ou não ter mais ou menos flexibilidade, como as opções de pacotes para serem instalados e outros complementos. O sistema Arch Linux em especial, possui duas formas principais de instalação, sendo uma através de um processo manual, onde o usuário deve realizar várias tarefas como o particionamento manual do disco e outra simplificada, através do script de instalação archinstall, que já é incluso nas imagens de instalação, e o método de instalação usado foi por esse script, para reduzir qualquer chance de erro de instalação. Alguns sistemas oferecem diversas opções durante a instalação, e outros oferecem opções customizadas ao realizar o download da imagem de disco (.iso).

Em seguida, ao ter uma instalação bem sucedida so sistema, prossegue-se para a

etapa de instalação dos *drivers* de vídeo. Esta etapa varia muito de um sistema para o outro, com alguns sistemas como o Fedora e Debian exigindo que um repositório extra fosse habilitado, e outros como o Bazzite e CachyOS incluindo os *drivers* automaticamente. Entretanto, os *drivers* de vídeo para a placa de vídeo utilizada neste estudo funcionam de forma parecida, carregando um módulo de *kernel* que permite a compatibilidade do dispositivo. Para placas de vídeo AMD, o procedimento é considerado mais simples, entretanto, para placas de vídeo NVIDIA, atualmente, os *drivers* proprietários não são sempre incluídos por padrão.

Após a instalação bem sucedida dos *drivers*, o próximo passo é instalar a plataforma Steam e os jogos. Isso é feito de forma simples, instalando o pacote específico de cada distribuição, sendo que apenas uma das distribuições, o Bazzite, usou a versão em flatpak. Após realizar o login, foi necessário habilitar o Steam Play para adicionar o suporte ao Proton aos jogos nativos de Windows, o que é feito apenas clicando em um botão. Esse procedimento pode ser feito clicando em "*Steam > Settings > Compatibility > Enable Steam Play for all other titles*", então, o Proton 9.0-4 foi selecionado por padrão para todos os jogos. Por fim, os jogos foram instalados normalmente.

Figura 2 – Opção para habilitar o Steam Play para todos os jogos.



Fonte: Steam, captura de tela da interface do usuário, 2025.

Com os jogos instalados, o programa MangoHud também foi instalado e configurado para salvar os arquivos em disco e também foi vinculado aos jogos através das opções extra de launcher para cada um, através do "mangohud %command%", para que seja possível realizar a captura das métricas de desempenho, e em seguida, os jogos foram executados. A primeira rodada de *benchmark* não foi registrada, pois esta serve como um aquecimento,

ou seja, os jogos foram executado onze vezes, sendo que a primeira não foi computada e as outras dez foram registradas, salvo os casos onde ocorreu algum erro de início de gravação, que demandou o descarte da execução de *benchmark* ou reinício da rodada.

Ao realizar toda a coleta para todos os jogos, as respectivas pastas de saída foram rotuladas e arquivadas em uma pasta, para permitir que o próximo sistema seja instalado e todos os procedimentos sejam repetidos.

3.5 Serviços e Resultados do Sistema

Os serviços envolvidos na análise serão principalmente os jogos, com o efeito colateral de outros serviços que estão em execução nos sistemas operacionais de propósito geral em segundo plano, que, sendo impossível isolar apenas o processo do jogo, os outros processos do sistema podem interferir com uma porcentagem baixa de uso de algum componente, como por exemplo, os processos em segundo plano utilizarão um pouco de tempo de CPU e memória, que serão capturados com as ferramentas de captura, junto às leituras durante a execução do jogo.

Se tratando dos resultados desejados, eles giram em torno das altas taxas de quadros, que permanecem acima de aproximadamente 60 quadros por segundo, e são essenciais para uma experiência de usuário de qualidade. As altas taxas de quadros garantem que os usuários tenham a melhor experiência e jogo, com as imagens renderizadas de forma nítida e clara, e uma alta taxa constante é especialmente importante na classe de jogos competitivos, onde uma queda brusca pode ser a diferença entre vitória e derrota, sendo essas quedas bruscas representadas pelo "1% low", que registra os piores 1% dos *frames*, e a partir disso, é possível inferir que quando há muitos registros de queda brusca abaixo de 60 FPS, pode-se dizer que a experiência foi negativamente afetada (CLAYPOOL; CLAYPOOL, 2007).

Outro resultado desejado é a utilização ótima dos recursos de *hardware*. Como os jogos são aplicações graficamente intensivas, o ideal é a utilização máxima da GPU, enquanto a CPU se mantém em um nível relativamente mais baixo. Caso ocorra o contrário, com CPU em alta utilização e GPU em utilização baixa constante, pode indicar algum problema com o *hardware*, com um gargalo de GPU, onde o processador não consegue utilizar a GPU em todo o seu potencial, ou algum *bug* do jogo (MESSAOUDI et al., 2017).

Em relação aos resultados indesejados, estão as quedas de quadro ou stutters, representados por quedas bruscas de quadros registrados e muita instabilidade durante o tempo de execução, que levem a uma queda constante na taxa de quadros para abaixo dos 60 quadros por segundo, isso leva a uma experiência frustrante, com uma experiência visual instável e irregular. Esse resultado podem ser causados por gargalos no processamento, limitação na memória de vídeo ou RAM do sistema ou bugs do jogo. É possível constatar

esse fenômeno através de alta instabilidade no *frametime*, e um "1% low" muito baixo, que representa que as piores taxas de quadros registradas são ruins (NVIDIA CORPORATION, 2022).

Os travamentos e *crashes* são outro resultado indesejado, que podem ser interrupções indesejadas na execução de um jogo, enquanto um *crash* acontece quando um jogo encerra a execução e exibe ou não uma mensagem de erro. Esses travamentos e *crashes* podem acontecer por uma série de fatores, como conflito de *softwares*, *drivers* desatualizados, *hardware* insuficiente, ou problemas na camada de compatibilidade e erros desconhecidos (SHAH, 2024).

A coleta das métricas aconteceu pelo processo de inicializar o cenário da ferramenta de *benchmark* de cada jogo e, em seguida, pressionando as teclas para iniciar o registro, de acordo com o atalho de cada uma das ferramentas. Após a coleta, cada *dataframe* será limpo e normalizado, removendo os cabeçalhos, para que a análise e cálculos possam prosseguir sem problemas.

A Tabela 4 apresenta as métricas coletadas durante cada rodada de experimentos, suas descrições e unidades de medida.

Tabela 4 – Descrição das Métricas.

Métrica	Medida	Descrição
fps	segundo	Quadros por Segundo
frametime	nanossegundo	Tempo gasto até um frame ser renderizado e exibido na tela
cpu_load	Porcentagem	Porcentagem geral do uso do processador
GPU_load	Porcentagem	Porcentagem geral do uso da placa de vídeo
cpu_temp	Graus Celsius – ºC	Temperatura do processador
GPU_temp	Graus Celsius – ºC	Temperatura da placa de vídeo
GPU_core_clock	Mega Hertz	Frequência do processador gráfico
GPU_mem_clock	Mega Hertz	Frequência da memória de vídeo da placa gráfica
GPU_vram_used	Mega Byte	Quantidade preenchida da memória RAM da placa gráfica
ram_used	Mega Byte	Quantidade preenchida da memória RAM do processador

Fonte: Adaptado de flightlessmango (2025).

3.6 Projeto Fatorial Completo com Dois Fatores

Um projeto fatorial completo, com dois fatores, seguindo a metodologia descrita por Jain (1991), pode ser usado para comparar diversas configurações de sistemas diferentes

utilizando cargas diferentes. Para a realização dessa metodologia, será neceessário realizar os cálculos e gráficos, que serão desenvolvidos utilizando ferramentas como python como linguagem de programação, utilizando as bibliotecas estatísticas scipy para os cálculos e o seaborn para criar os gráficos. Esses gráficos serão o Box Plot e Cumulative Distribution Function (CDF) para observar a distribuição inicial da coleta, e em seguida, os gráficos Quantil-Quantil normal (QQ-Plot) para os resíduos, com o objetivo de verificar a normalidade dos dados, e o gráfico de dispersão e histograma para os resíduos e o gráfico de barras da magnitude absoluta dos efeitos dos fatores.

Para este projeto, os fatores serão definidos como sendo um dos fatores os Sistemas Operacionais e o outro fator serão os jogos. Para cada interação do fator A com o fator B, ou seja, para cada jogo em cada sistema, haverá 10 replicações. A métrica principal será o FPS. De acordo com Montgomery (2012), esse projeto fatorial é definido pelo modelo da Equação 3.1

$$Y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + e_{ijk}, \tag{3.1}$$

onde o Y_{ijk} representa a resposta de cada fator em uma dada replicação, μ é a resposta média, α_i e β_j representam respectivamente o primeiro e o seguindo fator, que no caso deste estudo, serão os sistemas operacionais e jogos e i e j representam os seus níveis, $(\alpha\beta)_{ij}$ representa a interação entre dois fatores i e j, e e_{ijk} representa o erro experimental.

Após carregar os dados para o algoritmo, através das médias de cada replicação, por sistema operacional e por jogo, essas médias foram transformadas através da função de Box-Cox.

3.7 Pré-Processamento de Dados

Para que os cálculos sejam realizados e para lidar com a quantidade de dados, como exemplificado no exemplo da Figura 3, onde cada linha corresponde à leitura de 300 milissegundos, com aproximadamente 3 registros de métricas por segundo, foi tomado a decisão de prosseguir o projeto de experimentos utilizando a média de FPS de cada replicação. Para isso, no algoritmo, cada repetição será representada pela média de FPS do seu arquivo correspondente.

A coluna “cpu_power”, que representa a potência do processador (CPU) específico, apresentou um erro de leitura em todas as replicações, por isso, foi registrada como zero [21].

[21] Uma possível solução pode ser instalar o módulo de *kernel* zenpower (<https://github.com/ocerman/zenpower>) para CPUs AMD Zen.

Figura 3 – Arquivo de exemplo de saida do log do MangoHud.

	fps	frametime	cpu_load	cpu_power	gpu_load	cpu_temp	gpu_temp	gpu_core_clock
0	133.330	7.50017	37.3087	0	96	64	63	2835
1	120.974	8.26622	37.3087	0	96	64	63	2835
2	125.800	7.94913	37.3087	0	96	64	63	2835
3	120.080	8.32777	36.0153	0	97	64	63	2835
4	133.641	7.48274	36.0153	0	97	64	63	2835
...	...	...	...	...	...	...	...	...
394	106.642	9.37716	46.1002	0	96	59	63	2835
395	103.603	9.65223	46.1002	0	96	59	63	2835
396	124.450	8.03538	46.1002	0	96	59	63	2835
397	112.794	8.86572	46.1002	0	96	59	63	2835
398	111.258	8.98808	49.5019	0	96	60	63	2835

Fonte: MangoHud, arquivo de registro, 2025.

Em seguida, a média dos dados será transformado em uma tabela onde existem as colunas correspondentes aos jogos, sistemas operacionais e a média de FPS para uma determinada replicação. Essa transformação é importante para facilitar a execução dos cálculos dos algoritmos necessários para o projeto fatorial.

Figura 4 – Arquivo de exemplo dos dados pré-processados.

	OS	Game	Replication	FPS
0	Arch	ACOrigins	1	122.087600
1	Arch	ACOrigins	2	121.735677
2	Arch	ACOrigins	3	122.722136
3	Arch	ACOrigins	4	121.797149
4	Arch	ACOrigins	5	122.316452
...	...	...	...	...
295	Ubuntu	Warthunder	6	88.634453
296	Ubuntu	Warthunder	7	88.132579
297	Ubuntu	Warthunder	8	87.507292
298	Ubuntu	Warthunder	9	87.681191
299	Ubuntu	Warthunder	10	87.896445

Fonte: Elaborado pelo autor.

Após os processamentos iniciais, para que os dados sejam normalizados, eles foram transformados pela função de Box-Cox, que, para $\lambda \neq 0$ é dada pela Equação 3.2

$$y^{(\lambda)} = \frac{y^{\lambda} - 1}{\lambda} \, (\lambda \neq 0) \, . \tag{3.2}$$

Na equação, o y corresponde aos valores das amostras e $y^{(\lambda)}$ é o resultado da transformação, e o expoente λ caso não seja zero, precisa ser estimado para um número ótimo que garanta que os dados se adéquem à normalidade. É responsável por ajudar a estabilizar a variância em dados que não seguem uma distribuição normal, possibilitando um modelo simples para a análise da variância (BOX; COX, 1964).

Em seguida, será feito o cálculo da ANOVA, que irá fornecer as informações sobre a variância dos fatores e suas interações. Considerando os sistemas operacionais como fator A, com 6 níveis e os jogos como fator B com 5 níveis e 10 repetições, e, de acordo com Jain (1991), o grau de liberdade será dado por $\alpha \times \beta \times (r-1) = 6 \times 5 \times (10-1) = 270$, então, usa-se a distribuição normal para calcular os intervalos de confiança, que serão calculados em seguida, seguindo a Equação 3.3

$$X_i \pm z_{0.95} \times var, \tag{3.3}$$

onde X_i representa o fator ou interação entre fatores, o intervalo de confiança de 90% representado por $z_{0.95} \approx 1.6448$, e a variância de cada fator ou suas interações.

3.8 Considerações Finais

A metodologia tem a importância de definir todas as ferramentas necessárias para executar o projeto fatorial, como os fatores e seus detalhes, os jogos, o equipamento e os procedimentos de tratamento dos dados coletados.

Primeiramente, definição dos fatores é importante para poder estabelecer a quantidade de sistemas e jogos analisados, além dos detalhes de cada sistema e jogo, que serão executados durante a análise, além dos detalhes do *hardware*, que será o responsável por executar os jogos, em conjunto aos sistemas operacionais e também pode ter influência nos resultados, por causa das versões dos *drivers* de vídeo, por exemplo, pois estes *drivers* são um ponto crucial para estabelecer a comunicação entre o *hardware*, sistema operacional e jogo.

Além disso, o estabelecer o formato dos dados que serão tratados e as ferramentas utilizadas para executar o projeto fatorial, gráficos e tabelas, será possível conduzir um experimento que pode ser facilmente repetido ou futuramente expandido para, por exemplo, mais jogos e mais sistemas operacionais além dos mencionados.

4 Resultados Experimentais

O capítulo de resultados apresentará os resultados da coleta e processamento dos dados, além dos resultados do projeto fatorial completo para os dois fatores, necessários para realizar uma conclusão sobre o desempenho de cada sistema operacional, quando executaram cada jogo. A Seção 4.1 apresenta os resumos das coletas dos dados. Por sua vez, a Seção 4.2 apresenta a etapa de preparação dos dados para realizar os cálculos do projeto fatorial completo e a transformação dos dados utilizando a função Box-Cox. Na Seção 4.3 são apresentados os resultados dos cálculos de ANOVA necessários para analisar a variância. A Seção 4.4 foca na análise dos efeitos dos fatores individuais e das interações entre os fatores, ou seja, nos efeitos de cada jogo e sistema. Por fim, a Seção 4.6 apresenta um resumo geral do que foi encontrado.

4.1 Sumarização Estatística da Coleta de Dados

A Tabela 5 apresenta um resumo geral das 10 replicações de cada jogo dos sistemas no total, para a métrica de quadros por segundo (FPS), apresentando o "1% low" dos piores frames, a média e o 95º percentil. Essa tabela tem a função de apresentar uma visão geral do que foi coletado.

Tabela 5 – Resumo das médias gerais de 10 repetições.

	ACOrigins			FarCry6			KillerInstinct		
	avg	1%low	95th	avg	1%low	95th	avg	1%low	95th
Arch	122.07	96.16	145.82	98.68	81.83	111.66	60.05	57.91	61.25
Bazzite	122.37	96.62	146.21	99.38	80.46	113.53	60.18	57.67	61.39
CachyOS	121.19	99.56	143.64	99.26	83.49	112.35	60.06	56.90	61.20
Debian	115.61	90.69	135.27	98.41	79.28	111.20	60.29	58.69	60.69
Fedora	120.06	93.40	144.18	97.79	80.98	110.66	60.11	56.27	61.35
Ubuntu	120.18	96.43	143.70	98.33	81.82	112.62	60.08	56.06	61.69

	MetroExodus			Warthunder		
	avg	1%low	95th	avg	1%low	95th
Arch	74.40	48.05	101.49	87.77	42.07	106.56
Bazzite	73.56	47.79	100.42	87.32	57.38	106.77
CachyOS	75.07	48.60	104.14	88.10	52.61	106.96
Debian	74.62	48.61	103.79	92.35	54.33	106.82
Fedora	73.62	47.13	101.36	76.59	40.58	88.98
Ubuntu	79.27	48.50	116.49	87.86	58.33	106.95

Fonte: Elaborado pelo autor.

A Tabela 6 apresenta a média das utilizações de GPU e GPU em porcentagem de

todos os jogos em todos os sistemas operacionais. Tem a função de apresentar como cada jogo e cada sistema utilizou os recursos de *hardware* durante as execuções dos jogos, e de forma geral, é possível observar que as utilizações ficaram dentro do esperado, com uma alta utilização de GPU enquanto a utilização de GPU ficou menor. A única exceção é um jogo, que apresenta uma utilização de GPU baixa pois é travado em 60 FPS por padrão, ou seja, isso limitou a quantidade de frames processados, reduzindo a utilização de GPU, enquanto todos os outros não tem limitação de taxa de quadros.

Tabela 6 – Utilização de processador e placa de vídeo, em porcentagem.

	ACOrigins		FarCry6		KillerInstinct		MetroExodus		Warthunder	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
Arch	47.85	96.40	36.13	98.41	9.88	37.56	29.56	99.81	9.83	93.58
Bazzite	48.48	96.41	37.29	97.67	9.90	37.65	30.73	99.75	10.33	93.55
CachyOS	45.46	96.58	35.67	98.36	9.80	37.56	29.23	99.89	9.97	94.06
Debian	47.21	96.45	35.27	98.55	8.36	37.98	28.56	99.22	10.93	96.35
Fedora	47.79	96.48	36.92	98.61	10.04	37.71	30.54	99.79	9.05	96.14
Ubuntu	48.45	96.53	35.08	98.16	10.17	37.57	31.88	99.51	10.53	93.86

Fonte: Elaborado pelo autor.

A Tabela 7 será utilizada para resumir visualmente o nome dos jogos e dos sistemas operacionais, para facilitar as visualizações dos resultados do projeto fatorial e suas respectivas tabelas.

Tabela 7 – Abreviação para os Jogos e Sistemas.

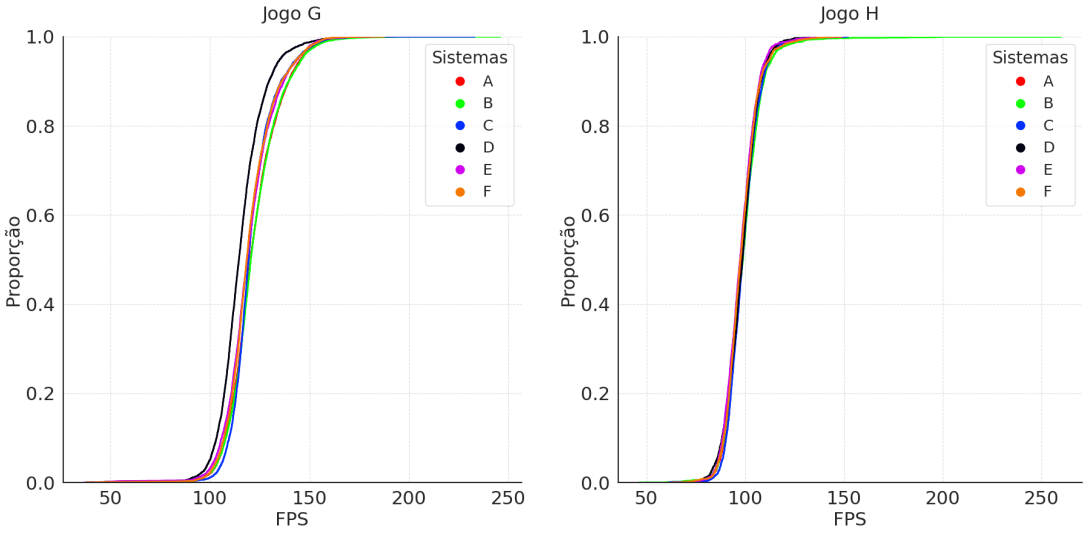
Código	Sistema Operacional	Código	Jogo
A	Arch	G	ACOrigins
B	Bazzite	H	FarCry6
C	CachyOS	I	KillerInstinct
D	Debian	J	MetroExodus
E	Fedora	K	Warthunder
F	Ubuntu		

Fonte: Elaborado pelo autor.

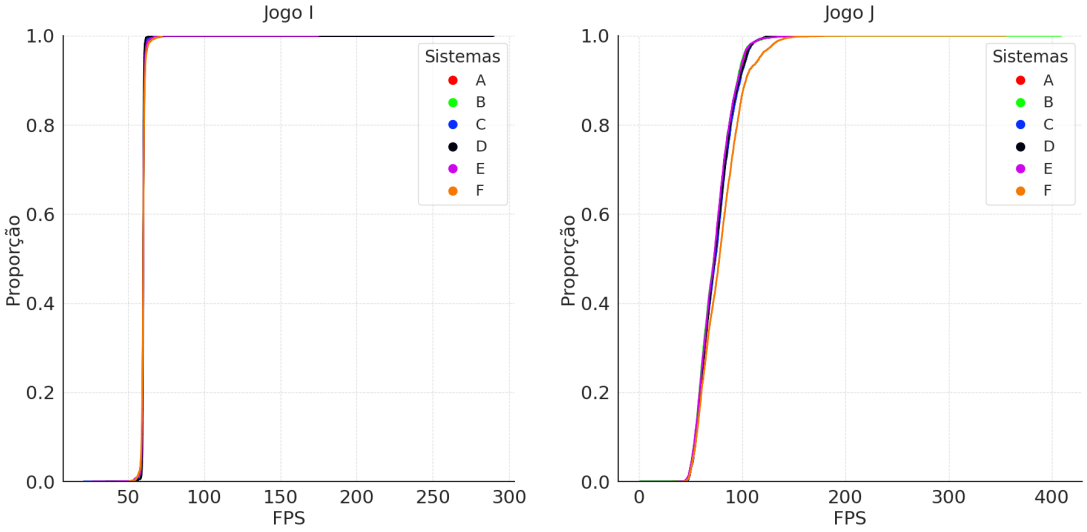
A Figura 5 apresenta os gráficos da função de distribuição acumulada de cada jogo, a Cumulative Distribution Function (CDF). É possível observar que cada jogo apresenta uma distribuição diferente para cada sistema operacional, com alguns casos com pouca diferença, representando o jogo com a limitação de quadros por segundo, e alguns jogos representando uma distribuição mais disperda na escala, como é o caso do Jogo K.

Figura 5 – CDF plot para cada jogo.

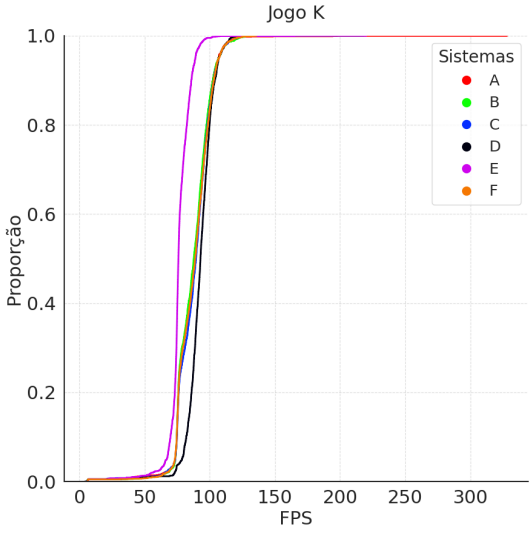
(a)



(b)



(c)



Fonte: Elaborado pelo autor.

A Figura 6 é o resultado do Box Plot para todos os dados de todas as replicações de cada jogo em cada sistema operacional. Para o primeiro jogo, é possível observar que em todos os sistemas houve uma pequena quantidade de *outliers* abaixo da média, e ainda uma grande quantidade de *outliers* abaixo de 60 FPS, podem ser interpretados como os *stutters*, que se apresentaram de forma consistente em todos os sistemas.

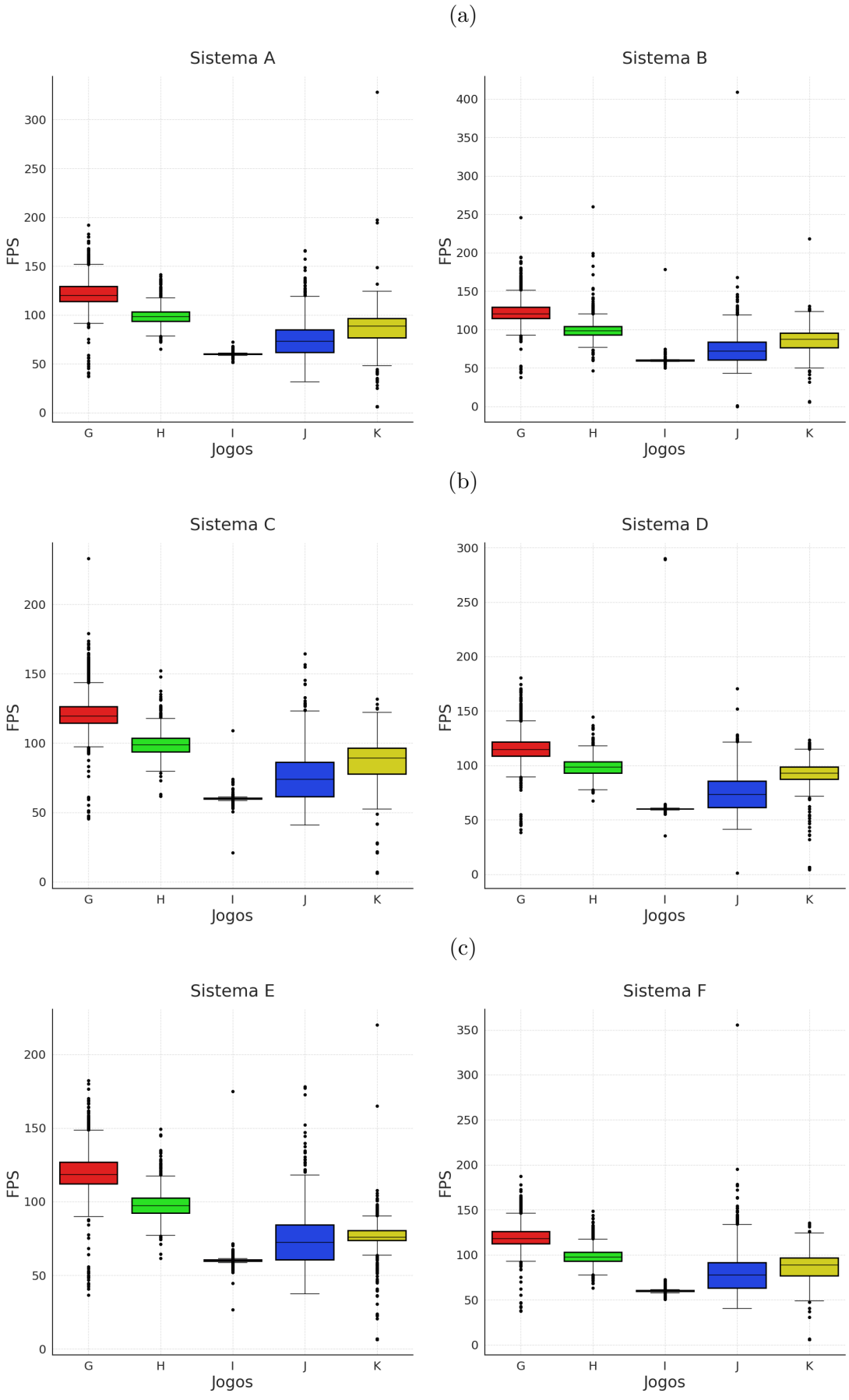
Para o segundo jogo, a maioria dos sistemas apresentou mais *outliers* acima da média, com poucos pontos, sendo considerados como *stutters*, o que representa uma boa estabilidade ao longo do *benchmark*, ainda que, como no caso do primeiro jogo, todos os sistemas responderam de forma consistente.

Para o terceiro jogo, que é limitado em 60 FPS, alguns sistemas como o Bazzite, Cachy e o Fedora apresentaram *outliers*, com pontos incomuns muito acima de 60 FPS e muito abaixo de 60 FPS, o que pode representar alguma anomalia indeterminada durante a coleta ou durante a execução do jogo, e como esses pontos são uma exceção e possuem poucos registros, possivelmente não tiveram um grande impacto na média calculada para o experimento.

O quarto jogo apresentou uma grande quantidade de *outliers* acima da média em todos os sistemas, com grandes picos de FPS e poucos registros de queda brusca abaixo de 60 FPS, também de forma não tão consistente para todos os sistemas operacionais, onde alguns sistemas apresentam velas mais espessas, como o E e C.

O quinto jogo apresentou uma grande variação entre os sistemas operacionais, o que confirma o que foi observado pela CDF, o jogo apresentou uma certa instabilidade com alguns pontos abaixo da média em todos os sistemas operacionais, com mais pontos em sistemas como o Fedora, Debian e Cachy, e menos em sistemas como o Arch, Bazzite e Ubuntu.

Figura 6 – Box Plots para cada sistema operacional.



Fonte: Elaborado pelo autor.

4.2 Preparação dos Dados

A Tabela 8 apresenta a média dos dados das execuções de cada replicação de cada jogo em cada sistema operacional sem a transformação pela função de Box-Cox. Os cálculos necessários para o projeto fatorial completo se baseiam na média do total de amostras para cada replicação.

Tabela 8 – Resposta Média de cada Repetição para cada Jogo.

Sistema	Replicação / Jogo	1	2	3	4	5	6	7	8	9	10
A	G	122.09	121.74	122.72	121.80	122.32	122.55	121.97	122.01	121.38	122.14
	H	98.66	98.84	98.30	98.98	98.70	96.97	98.89	99.39	98.36	99.69
	I	60.07	60.14	60.00	60.13	60.08	60.06	59.99	59.95	60.13	59.98
	J	74.74	74.98	73.60	74.57	73.97	73.54	74.66	75.08	73.89	74.91
	K	87.80	88.29	86.53	87.95	87.56	87.66	87.11	87.71	89.15	87.91
B	G	123.45	122.16	121.32	121.32	121.66	121.97	124.13	121.85	123.35	122.50
	H	99.12	100.20	99.52	99.27	98.76	98.41	99.53	99.40	100.59	98.97
	I	60.15	60.17	60.74	60.18	59.96	60.17	60.00	60.21	60.05	60.14
	J	73.67	73.10	73.57	74.00	73.67	73.40	72.71	73.65	73.30	74.56
	K	86.64	87.68	86.11	87.85	87.55	86.68	88.60	87.15	87.67	87.23
C	G	120.65	120.45	120.72	120.78	120.29	122.38	122.06	121.56	121.32	121.73
	H	98.79	99.53	98.54	98.85	99.12	100.12	99.50	99.66	99.45	99.07
	I	60.02	59.75	60.04	60.13	60.24	60.01	60.02	60.14	60.03	60.20
	J	74.44	74.63	74.80	75.43	76.28	74.52	75.51	76.55	74.18	74.40
	K	86.77	88.21	87.95	89.38	88.80	87.89	86.86	88.05	88.60	88.53
D	G	116.05	117.41	117.01	116.28	116.60	113.74	115.54	115.17	114.51	113.77
	H	97.96	98.43	98.22	98.75	98.63	99.05	97.93	98.20	98.90	98.07
	I	59.85	60.07	60.00	60.05	61.36	59.99	60.09	61.39	60.08	60.07
	J	74.01	73.97	74.75	74.59	75.45	74.75	75.79	75.04	74.12	73.70
	K	91.51	93.15	92.66	92.96	91.97	92.13	92.23	92.19	92.52	92.21
E	G	120.38	119.15	120.72	120.33	120.51	120.75	118.76	120.53	118.87	120.59
	H	97.94	97.25	97.10	97.44	98.31	98.18	96.88	98.61	97.84	98.38
	I	60.79	59.82	60.14	59.94	60.18	60.06	60.08	60.01	59.95	60.13
	J	73.15	73.00	72.72	74.05	73.95	73.80	74.73	74.27	73.32	73.26
	K	75.29	77.19	76.36	76.22	76.60	76.27	77.66	77.20	76.52	76.58
F	G	120.33	119.83	120.69	120.22	119.47	120.59	120.80	120.66	119.84	119.38
	H	98.93	98.57	98.30	97.93	97.97	98.23	98.64	98.67	98.02	98.02
	I	59.93	60.10	60.25	60.06	60.06	60.05	59.92	60.25	60.11	60.04
	J	83.43	83.91	82.61	82.91	83.58	83.88	74.04	72.56	72.71	73.10
	K	87.32	88.65	87.21	88.09	87.52	88.63	88.13	87.51	87.68	87.90

Fonte: Elaborado pelo autor.

A Tabela 9 apresenta o resumo dos efeitos de cada sistema operacional e de cada jogo, já com a transformação dos dados utilizando a função Box-Cox, com um $\lambda = 0,2121$ ótimo, para a Equação 3.2.

Tabela 9 – Sumarização dos Efeitos.

	G	H	I	J	K	Row Sum	Row Mean	Row Effect
A	8.3481	7.7717	6.5234	7.0456	7.4652	37.1540	7.4308	0.0108
B	8.3549	7.7904	6.5283	7.0176	7.4519	37.1431	7.4286	0.0086
C	8.3281	7.7874	6.5236	7.0682	7.4750	37.1824	7.4365	0.0165
D	8.1982	7.7647	6.5328	7.0530	7.5975	37.1462	7.4292	0.0092
E	8.3021	7.7479	6.5256	7.0196	7.1183	36.7135	7.3427	-0.0773
F	8.3050	7.7624	6.5244	7.2010	7.4681	37.2607	7.4521	0.0322
Column Sum	49.8364	46.6245	39.1581	42.4049	44.5761	222.5999		
Column Mean	8.3061	7.7707	6.5264	7.0675	7.4293		7.4200	
Column Effect	0.8861	0.3507	-0.8936	-0.3525	0.0093			

Fonte: Elaborado pelo autor.

A Tabela 10 representa os efeitos das interações, calculada pela subtração das médias das coluna e linhas, e somando a média geral da Tabela 9. Por exemplo: $E_{ag} = 8,3481 - 8,3061 - 7,4308 + 7,42 = 0,0312$. Em destaque nessa tabela estão os maiores e menores valores em termos de desempenho, nas cores azul e vermelho, respectivamente. Com valores positivos indicam que os efeitos das interações são maiores do que a média geral e valores negativos indicando efeitos das interações menores do que a média geral.

Desse modo, para cada Sistema Operacional (linha) com relação à execução do seu respectivo Jogo (coluna) pode-se observar que os maiores valores positivos (destacados em azul) indicam que as respectivas interações obtiveram o maior número de quadros por segundo, enquanto que os menores valores negativos (destacados em vermelho) obtiveram o pior desempenho em termos de quadros por segundo. Posteriormente, na Seção 4.4 serão apresentados os intervalos de confiança para os efeitos das interações, assim será possível identificar de fato qual desempenho da relação Sistema Operacional versus Jogo obtém o melhor desempenho estatisticamente significativo.

Tabela 10 – Efeitos das Interações.

	G	H	I	J	K
A	0.0312	−0.0098	−0.0137	−0.0327	0.0250
B	0.0402	0.0111	−0.0067	−0.0585	0.0139
C	0.0056	0.0002	−0.0192	−0.0158	0.0292
D	−0.1171	−0.0153	−0.0028	−0.0237	0.1590
E	0.0733	0.0544	0.0766	0.0294	−0.2338
F	−0.0332	−0.0405	−0.0341	0.1013	0.0066

Fonte: Elaborado pelo autor.

4.3 Análise de Variância (ANOVA)

A Tabela 11 apresenta os resultados do cálculo da ANOVA para cada grupo de fatores, a análise da variância. Ela apresenta a somaa dos quadrados, a porcentagem de variação, os graus de liberdade que são referentes às quantidades de fatores e interações,

a média dos quadrados, o F-computed que analisa o quanto a média entre os grupos varia, o F-table que é um número derivado da distribuição F e o p-value que representa a probabilidade de obter um número igual ao observado em F-computed, assumindo uma hipótese nula verdadeira.

Tabela 11 – Resultados do cálculo de ANOVA.

	Sum of Squares	Percentage of Variation	Degree of Freedom	Mean Square	F-Computed	F-Table	p-value
Operating System	0.38	0.34	5.00	0.08	58.61	2.25	3.69e-41
Game	109.87	98.16	4.00	27.47	21300.08	2.41	0.00
Interaction	1.34	1.19	20.00	0.07	51.80	1.61	2.72e-80
Residual	0.35	0.31	270.00	0.001			

Fonte: Elaborado pelo autor.

Conforme mostra a Tabela 11, para os sistemas operacionais, como o F-computed é muito maior que o valor de F-Table e o valor do p-value é efetivamente zero, rejeita-se a hipótese nula de que todos os níveis de Operating System (OS), Sistema Operacional, têm a mesma média. Embora a OS contribua com uma pequena fração da variação, essa contribuição é altamente consistente e estatisticamente significativa.

Os jogos dominam a variação com mais de 98% (destacado em vermelho na Tabela 11), com a maior dos quadrados, apresentando um valor de F-Computed maior do que F-Table, e o valor de p-value em declínio confirmam que nem todos os jogos produzem a mesma resposta, ou seja, a escolha do jogo é o maior impulsionador das diferenças de desempenho.

A interação, que representa 1,19% da variação explicada, indica que o efeito do sistema operacional difere de forma estatisticamente significativa por jogo, com certas combinações de sistema operacional e jogo apresentando desempenho melhor ou pior do que o esperado pela soma de seus efeitos individuais.

Por fim, os resíduos representam apenas 0,31% da variação explicada, o que é bom para o modelo uma vez que os erros não representam uma fração significativa da variação explicada e os jogos representam a grande maioria da variação explicada o que, por sua vez, reforça seu maior impacto no desempenho do sistema.

4.4 Analise Comparativa dos Efeitos e suas Interações

A Tabela 12 representa os intervalos de confiança dos efeitos de cada fator individual, os jogos e sistemas operacionais. Onde caso haja interceptação com o zero dentro dos intervalos, o fator pode ser potencialmente não significativo. Enquanto caso o efeito seja

positivo, indica que ele está acima da média, enquanto os efeitos negativos estão abaixo da média.

Tabela 12 – Intervalos de confiança dos Efeitos.

Parameter	Mean Effect	Std Dev Effect	90% Confidence Interval
μ	7.420	0.002	(7.417, 7.423)
Operating System			
A	0.011	0.005	(0.003, 0.018)
B	0.009	0.005	(0.001, 0.016)
C	0.016	0.005	(0.009, 0.024)
D	0.009	0.005	(0.002, 0.017)
E	−0.077	0.005	(−0.085, −0.070)
F	0.032	0.005	(0.024, 0.040)
Game			
G	0.886	0.004	(0.879, 0.893)
H	0.351	0.004	(0.344, 0.358)
I	−0.894	0.004	(−0.900, −0.887)
J	−0.353	0.004	(−0.359, −0.346)
K	0.009	0.004	(0.002, 0.016)

Fonte: Elaborado pelo autor.

Os intervalos de confiança para os efeitos dos sistemas operacionais, de A a F, demonstram que os sistemas A, B, C, D e F possuem efeitos positivos. Em contrapartida, o sistema E apresenta efeito negativo e um intervalo de confiança negativo. Para os intervalos de confiança dos efeitos dos jogos, a maioria, exceto o jogo K, exibiu efeitos maiores, tanto positivamente quanto negativamente, em relação aos sistemas operacionais. No geral, os efeitos dos jogos são uma ou duas ordens de grandeza maiores do que os efeitos dos sistemas operacionais. Como cada jogo é uma carga diferente, é esperado que haja uma grande variação da magnitude dos seus efeitos. Em suma, todos os intervalos de confiança dos efeitos são estatisticamente significativos e, por isso, validam os coeficientes do modelo do projeto fatorial completo com dois fatores.

Os efeitos podem ser interpretados conforme descrito a seguir. Em média os fatores resultaram em um FPS com transformação Box-Cox aplicada de 7,42, o que aplicando-se a transformação Box-Cox inversa resulta em uma média geral de **88,15** FPS, desse modo, a Tabela 13 apresenta a diferença percentual de cada fator, dada pela fórmula:

$$\Delta FPS = \frac{efeito - media}{media} \times 100 \tag{4.1}$$

Tabela 13 – Diferença percentual dos Efeitos para cada fator.

Fator	Diferença Percentual (%)	Fator	Diferença Percentual (%)
A	0.497	G	36.434
B	0.460	H	11.889
C	0.662	I	-31.788
D	0.116	J	-14.680
E	-2.858	K	-1.697
F	1.123		

Fonte: Elaborado pelo autor.

Para os fatores entre A e F, que representam os sistemas operacionais, foi possível observar que o sistema Fedora (E), em geral, registrou menos FPS do que a média, com aproximadamente 2,85% abaixo da média, enquanto o sistema Ubuntu (F) apresentou de aproximadamente 1,12% acima da média.

De forma geral, é possível concluir que a escolha do sistema operacional teve menos influência no desempenho geral do que os jogos em si, ou seja, a escolha do sistema operacional se mostrou pouco relevante para esses casos.

A Tabela 14 apresenta o intervalo de confiança das interações dos fatores. Os intervalos que incluem o zero são marcadas com um a, e representam uma leitura de quadros por segundo que ficaram dentro da média, não sendo considerados estatisticamente significativos.

Tabela 14 – Intervalos de confiança de 90% das interações dos fatores.

	G	H	I	J	K
A	(0.016, 0.046)	$(-0.025, 0.005)^a$	$(-0.029, 0.002)^a$	$(-0.048, -0.017)$	(0.010, 0.040)
B	(0.025, 0.055)	$(-0.004, 0.026)^a$	$(-0.022, 0.009)^a$	$(-0.074, -0.043)$	$(-0.001, 0.029)^a$
C	$(-0.010, 0.021)^a$	$(-0.015, 0.015)^a$	$(-0.034, -0.004)$	$(-0.031, -0.001)$	(0.014, 0.044)
D	$(-0.132, -0.102)$	$(-0.031, -0.000)$	$(-0.018, 0.012)^a$	$(-0.039, -0.008)$	(0.144, 0.174)
E	(0.058, 0.089)	(0.039, 0.070)	(0.061, 0.092)	(0.014, 0.045)	$(-0.249, -0.219)$
F	$(-0.048, -0.018)$	$(-0.056, -0.025)$	$(-0.049, -0.019)$	(0.086, 0.117)	$(-0.009, 0.022)^a$

^{a} indica que o intervalo de confiança não é significativo.

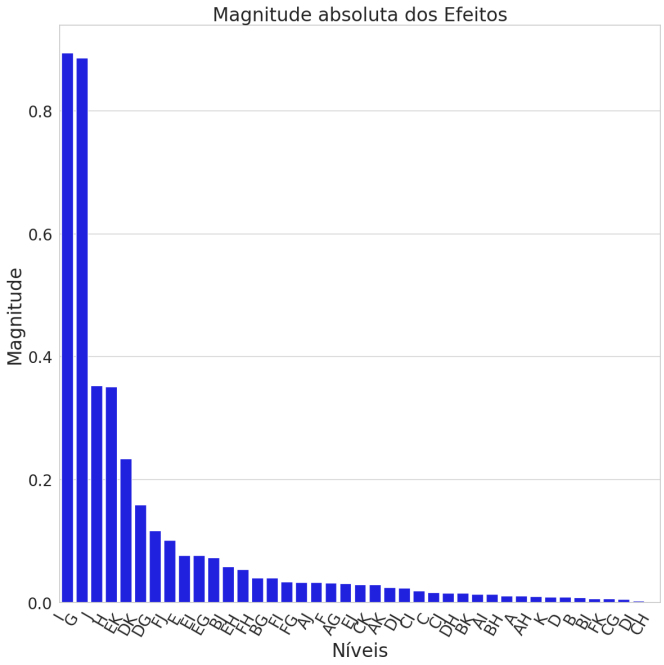
Fonte: Elaborado pelo autor.

As colunas G e I da tabela são correspondentes aos jogos que demonstraram os intervalos de confiança com as maiores magnitudes. Entretanto, nas interações, esses jogos obtiveram um intervalo de confiança relativamente pequeno, porém o jogo G obteve um desempenho um pouco menor do que a média no sistema D e um desempenho maior em E (destacado em vermelho na Tabela 14). Enquanto o jogo I obteve melhor desempenho no sistema E (também destacado em vermelho na Tabela 14), muito acima dos outros sistemas para esse jogo.

Referente à interação entre sistemas e jogos, apenas algumas interações, do sistema e jogos marcados com "a" na Tabela 14, se mostraram potencialmente insignificantes, pois essas interações ficaram dentro da média, e no cálculo dos intervalos de confiança, interceptaram o zero. Enquanto isso, sobre os intervalos em geral, se demonstraram em geral pequenos, tanto positivamente quanto negativamente, o que pode demonstrar que os efeitos das interações foram pequenos.

A Figura 7 apresenta a magnitude absoluta dos efeitos dos fatores e suas interações, para demonstrar como cada um dos fatores estudados e seus níveis tiveram contribuição no experimento.

Figura 7 – Magnitude absoluta dos efeitos de cada fator, interações e seus níveis.



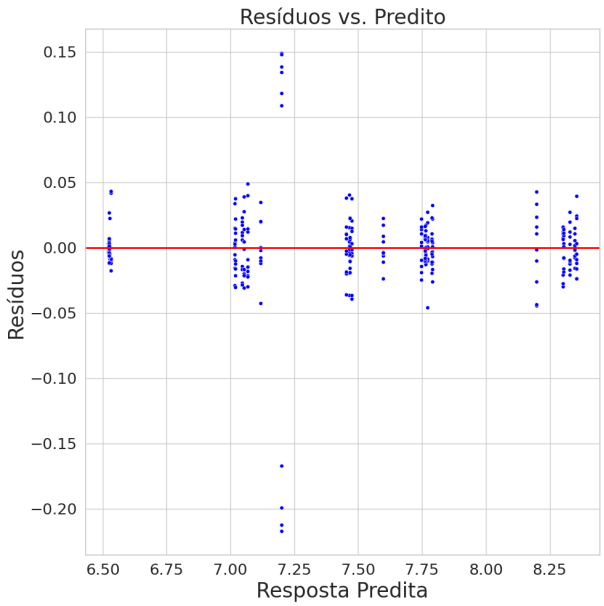
Fonte: Elaborado pelo autor.

É possível observar que a maior parte dos efeitos vieram dos jogos I e G, e ainda que os sistemas, representados entre as letras A e F, e sua interações tiveram uma contribuição muito pequena em geral. Isso ajuda a confirmar que a maior parte da variação do experimento tem natureza nos jogos selecionados.

4.5 Inspeção Diagnóstica para Validação do Modelo

Para inspeção diagnóstica do modelo do projeto fatorial completo, a Figura 8 representa o gráfico dos resíduos calculados. Um resultado desejado é que os pontos não apresentem um padrão ou agrupamento, mas que fiquem dispersos ao redor da linha.

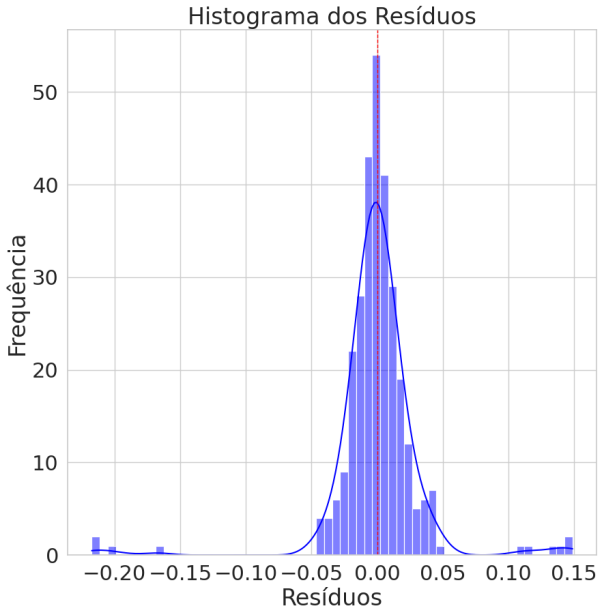
Figura 8 – Gráfico de resíduos.



Fonte: Elaborado pelo autor.

De acordo com a Figura 8, os resíduos são até duas ordens de grandeza menor do que as respostas preditas, exceto para os *outliers* apresentados na parte superior e inferior que são uma ordem de grandeza menor. Se ignorados os *outliers*, os resíduos são distribuídos de forma aproximadamente normal. Para confirmar essa observação a Figura 9 apresenta o histograma da distribuição de resíduos. Conforme é exibido nessa figura, o histograma dos resíduos apresenta formato de sino com alguns *outliers* na parte esquerda e direita do formato de sino, com um só pico e centrado próximo de zero, o que é um sinal positivo em relação ao modelo dos experimentos. Então, esse gráfico confirma que os resíduos são distribuídos de forma aproximadamente normal e valida a premissa de independência dos resíduos e de serem distribuídos de forma idêntica (o que implica na mesma variância para todos os valores das predições, excluindo os *outliers*).

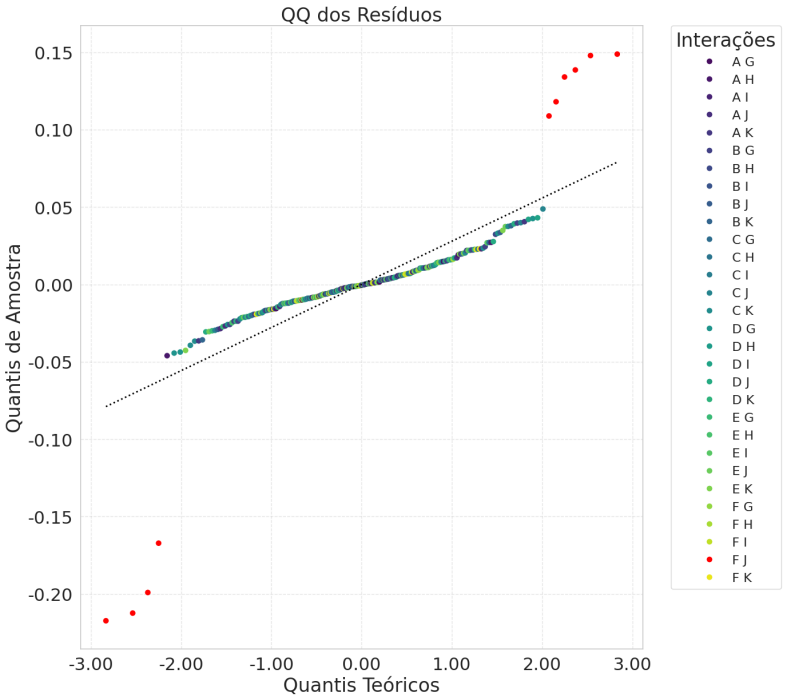
Figura 9 – Histograma dos resíduos.



Fonte: Elaborado pelo autor.

A Figura 10 representa o gráfico dos quantis teóricos, também chamado de Quantil-Quantil Normal (QQ-Plot), representados por uma linha inclinada e os calculados para os efeitos, representados pelos pontos. O resultado desejado é que os pontos fiquem ajustados à linha.

Figura 10 – Gráfico dos quantis.



Fonte: Elaborado pelo autor.

A parte central do gráfico da Figura 10, onde se concentram a maior parte dos pontos, indica uma distribuição de resíduos que aproximadamente segue a distribuição normal. Além disso, nessa figura estão destacados na cor vermelha os *outliers*, os quais se concentram nas extremidades esquerda e direita, conforme mostra a legenda, esses *outliers* são referentes aos fatores F e J. Na cauda direita, onde os pontos começaram a ultrapassar positivamente a linha, indica que os resíduos são muito maiores do que é esperado para uma distribuição normal. A cauda esquerda indica o oposto, que os resíduos são muito menores do que o esperado para uma distribuição normal.

Essas anomalias das caudas podem ter relação com os efeitos produzidos por um jogo testado em um sistema operacional específico e são considerados *outliers*, estes são os mesmos das Figuras 8 e 9. Inspeções manuais mostraram que esses *outliers* pertencem ao Sistema Operacional Ubuntu (F) executando o jogo Metro Exodus Enhanced Edition (J). Ao observar a Tabela 8 percebe-se que a linha F (Ubuntu) da coluna "Sistema" e linha J (Metro Exodus Enhanced Edition) da coluna "Jogo" apresentam maior variação em termos de amostra do que as demais amostras coletadas para o mesmo jogo nos demais sistemas operacionais.

4.6 Considerações Finais

Com base nos resultados, foi possível observar o impacto do desempenho de cada sistema diferente para cada jogo. Com base nisso, é possível observar que de forma geral, os sistemas na sua maioria possuem intervalos significativos, porém os jogos são os maiores responsáveis pela variação de desempenho no estudo.

A exibição inicial dos dados, através do gráfico CDF e Box Plot demonstraram que cada sistema obteve um resultado diferente, mesmo que ligeiramente, e especialmente no Box Plot, foi possível observar uma certa variação nos registros extremos deFPS em cada sistema operacional, que podem indicar as questões de possíveis erros de leitura ou pequenas instabilidades ou *stutters* ocasionais. Para avaliar a questão da estabilidade de cada sistema, que vai além da análise da média, seria necessário um outro experimento, que focaria especialmente em estabilidade, e para isso, seria importante testar uma quantidade maior de jogos.

Em relação à ANOVA, os resultados mostram que a maior parte da variação explicada é referente ao fator jogo, com uma 98,16% da variação explicada e, em ambos os fatores e nas interações, o F-computed foi muito maior do que o F-Table, o que indica que os grupos são significativos, apesar dos sistemas operacionais terem uma contribuição pequena, o que possivelmente ocasiona um impacto irrelevante no desempenho do sistema na execução do jogo.

O gráfico dos quantis apresentou uma distribuição próxima da normal, com alguns

outliers nas extremidades do gráfico de dispersão dos resíduos, do QQ-Plot e do histograma dos resíduos. Esses *outliers* são causados pela interação do Sistema Operacional Ubuntu com a execução do Jogo Metro Exodus Enhanced Edition conforme foi exibido no gráfico da Figura 10, no entanto, isso não inválida a hipótese de que os resíduos são independentes e identicamente distribuídos.

5 Conclusões e Trabalhos Futuros

A coleta de métricas dos sistemas ocorreu de forma bem sucedida, para os sistemas e jogos e suas replicações. E, em seguida, foi tomado a média de cada replicação da série temporal de cada jogo, e esta média foi usada para representar a replicação, usando apenas a métrica de FPS, como mostra a Tabela 8, e finalmente, os resultados foram utilizados para realizar o projeto fatorial completo com replicações, que conseguiu capturar as nuances de desempenho, mesmo que a média dos dados possa ocultar detalhes de estabilidade durante a execução.

A metodologia utilizada, de realizar a abordagem da análise de desempenho de jogos utilizando o projeto fatorial completo foi uma boa forma para identificar pontos de desempenho que seriam dificilmente detectados com uma análise visual simples. Além disso, a abordagem com replicações é uma forma de ajudar a reduzir a possibilidade de ter discrepâncias mais agudas por erros intermitentes, como poderia acontecer em uma repetição única, na qual um fator poderia sofrer uma queda intermitente de desempenho por causas externas do sistema operacional ou um *bug*, que levariam a uma percepção muito distorcida de uma leitura de dados.

A partir dos resultados da exibição inicial de dados, já foi possível notar visualmente uma diferença nos registros coletados entre cada sistema operacional, porém não demonstram uma resposta definitiva sobre o desempenho real. Com a análise do projeto fatorial, através da ANOVA e intervalos de confiança, foi possível observar que os jogos são responsáveis pelo maior impacto no desempenho, com uma contribuição de 98,16%, enquanto os sistemas operacionais possuem uma contribuição de 0,34%, ou seja, os sistemas operacionais possuem uma diferença pequena de desempenho entre si, com apenas alguns casos demonstrando uma vantagem quase negligenciável de ganho ou perda de desempenho para jogos específicos. Então, é possível dizer que a escolha de de uma distribuição Linux tem pouco impacto se um jogo irá executar com bom desempenho ou não, desde que haja *drivers* de vídeo atualizados. Por fim, em termos percentuais, indicados pela Tabela 13, foi possível observar que o sistema Ubuntu obteve o maior desempenho, em cerca de 1,12% acima da média de FPS, enquanto o sistema Fedora apresentou o menor desempenho, com cerca de 2,58% abaixo da média.

5.1 Limitações do Trabalho

A coleta de métricas se dá pelo resultado da interação específica do *hardware* com os programas, ou seja, ainda que o objetivo do trabalho seja observar o comportamento de sistemas operacionais, pode existir alguma influência desconhecida provocada pelos *drivers*

de cada *hardware* em cada sistema operacional, algo que pode ser observado na Figura 6, que apresenta certas leituras de dados anormais, que podem ter naturezas diversas, além da sinergia da configuração da máquina.

E também, para este estudo, apenas a média do FPS foi utilizada, porém, além do FPS, existem outras métricas que foram coletadas ao mesmo tempo e não foram analisadas, sendo essas outras métricas as listadas na Tabela 4, como o *frametime*, utilização de CPU e GPU que foram brevemente mencionados na Tabela 6, e temperatura. E, além isso, por causa de um erro de leitura, que pode ser observado na Figura 3 onde a coluna "cpu_power" está com registros zerados, não foi possível ler a potência do processador utilizado no estudo. Portanto, é necessário uma pesquisa mais aprofundada com relação à coleta dessa métrica ou ainda dedicar um trabalho para estudar a potência gasta pelos componentes usando um equipamento dedicado.

A seleção dos jogos foi limitada pela disponibilidade na biblioteca pessoal, e também pela disponibilidade de uma ferramenta interna de *benchmark*. Então, os resultados representam apenas uma fração dos jogos e engines disponíveis, principalmente faltando jogos competitivos disponíveis nas duas plataformas, além do que as ferramentas de *benchmark* podem nem sempre representar o caso mais custoso para cada jogo. Para um resultado melhor, seria necessário aumentar a quantidade de jogos executados, e escolher remover ou não os jogos que possuem uma limitação de quadros por segundo padrão.

Pode ser que hajam pequenos erros de coleta, pois a ferramenta de coleta pode ter um pequeno impacto no desempenho e também o início da gravação depende do pressionamento de teclas no tempo certo para iniciar a gravação de dados, porém, na sua maioria, a gravação aconteceu no tempo preciso, e quando houveram falhas visíveis, a replicação em questão foi descartada e regravada.

Além disso, é importante notar que podem haver erros aleatórios de *hardware* ou *software* que podem impactar no desempenho e estão fora do controle humano, mesmo que os experimentos do estudo foram meticulosamente controlados. Ou seja, pode ser que em uma nova tentativa de replicar o estudo, os resultados sejam melhores ou piores, mesmo com a mesma configuração de *hardware*, *drivers* e jogos.

5.2 Principais Contribuições

Ao seguir a metodologia do projeto de experimentos, foi possível verificar a influência das distribuições no desempenho de jogos, e com essa comparação de desempenho, é possível observar o quanto algumas distribuições atualmente interferem no desempenho de um jogo, que mesmo em um cenário controlado de *benchmark*, ainda pode ser extrapolado para um cenário real, demonstrando que as distribuições possuem pouca influência efetiva sobre o quão bem um jogo desempenha, ou seja, é possível dizer que a distribuição Linux

que um usuário escolha fará pouca diferença para o desempenho em jogos.

E também, ao aplicar essa metodologia, a mesma se provou eficiente também para detectar as variações e impactos em cada sistema operacional no desempenho para a área de jogos eletrônicos, mostrando que essa abordagem pode potencialmente ser usada em trabalhos futuros para estudar o desempenho de sistemas para jogos em geral, tanto sistemas operacionais quanto *hardware* em larga escala.

5.3 Trabalhos Futuros

Para uma análise mais completa, seria necessário aumentar a quantidade de sistemas operacionais para teste, incluindo os sistemas Windows e BSD e também testar outras configurações de kernel Linux customizados com algumas otimizações que potencialmente podem aumentar o desempenho em jogos, e também testar o sistema Debian utilizando os *drivers* mais atualizados. Além disso, aumentar a quantidade de jogos envolvidos no teste, pois como demonstrado no gráfico dos resíduos, o ajuste da curva não mostra um padrão ideal para este tipo de modelo, e também, expandir a análise realizando um outro projeto fatorial para as outras métricas da Tabela 4, que não foram utilizadas neste estudo, e também uma análise de FPS envolvendo o "1% low", para ajudar a detectar as nuances de estabilidade de cada configuração.

Além disso, existe a possibilidade de testar a abordagem do projeto fatorial para outros fatores, ajustando o experimento para um objetivo mais específico, como, por exemplo, estudar o desempenho de *hardware*s diferentes em jogos diferentes, para que seja possível criar uma base de dados com o ganho ou perda de desempenho com os *hardware*s disponíveis no mercado, para desempenho de jogos, pois como foi mostrado por Martinovic, Balen e Cukic (2012), podem existir diferenças de desempenho entre sistemas operacionais e entre configurações de *hardware* diferentes, podendo um *hardware* mais antigo mudar os resultados dos jogos estudados ao executá-los em um *hardware* diferente ou mais antigo. Para este experimento, juntamente ao FPS, seria importante analisar as utilizações de outros componentes, também para ajudar a identificar se alguma sub-utilização está causando uma perda de desempenho.

Para além da análise de sistemas operacionais e *hardware*, há a possibilidade de estudar a utilização da metodologia do projeto de experimentos para a otimização de jogos na etapa de desenvolvimento, pois esta ferramenta se mostrou muito eficiente para detectar a variação de leituras em um conjunto de dados com uma boa sensibilidade, ou seja, na etapa de desenvolvimento, ao analisar dados como a latência, FPS, utilização de *hardware*, em conjunto outras métricas e ferramentas disponíveis no desenvolvimento, é possível apontar um norte para aprimorar o desempenho de um jogo.

Referências

AKENINE-MöLLER, T. et al. *Real-time rendering*. 4. ed. Boca Raton, FL, USA: CRC Press, 2018. Citado na página 19.

ARCH LINUX WIKI CONTRIBUTORS. *Arch Linux*. 2025. ArchWiki. Disponível em: <https://wiki.archlinux.org/title/Arch_Linux>. Acesso em: 15 abr. 2025. Citado na página 31.

ARCH LINUX WIKI CONTRIBUTORS. *Kernel*. 2025. ArchWiki. Disponível em: <https://wiki.archlinux.org/title/Kernel>. Acesso em: 15 abr 2025. Citado na página 31.

BARTLE, R. *Designing Virtual Worlds*. Upper Saddle River, NJ, USA: New Riders, 2003. Citado na página 18.

BAZZITE. *Bazzite Kernel*. 2025. Github. Disponível em: <https://github.com/bazzite-org/kernel-bazzite>. Acesso em: 15 Abr. 2025. Citado na página 31.

BAZZITE. *FAQ*. 2025. Bazzite Documentation. Disponível em: <https://docs.bazzite.gg/General/FAQ/>. Acesso em: 15 Abr. 2025. Citado na página 31.

BONFIM, A. *Precisamos falar sobre anti-cheat no Linux*. 2024. Diolinux. Disponível em: <https://diolinux.com.br/editorial/falar-sobre-anti-cheat-no-linux.html>. Acesso em: 20 jan 2025. Citado na página 15.

BOX, G. E.; COX, D. R. An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, Oxford University Press, v. 26, n. 2, p. 211–243, 1964. Citado na página 42.

CACHYOS. *CachyOS Kernel*. 2025. <https://wiki.cachyos.org/features/kernel/>. Acesso em: 15 Abr. 2025. Citado na página 31.

CACHYOS. *Why CachyOS?* 2025. <https://wiki.cachyos.org/cachyos_basic/why_cachyos/>. Acesso em: 15 Abr. 2025. Citado na página 31.

CANONICAL. *The Ubuntu lifecycle and release cadence*. 2025. <https://ubuntu.com/about/release-cycle/>. Acesso em: 15 Abr. 2025. Citado na página 32.

CANONICAL. *Ubuntu packages*. 2025. <https://ubuntu.com/about/packages/>. Acesso em: 15 Abr. 2025. Citado na página 32.

CASTRO, J. D. *Introducing Linux distros*. 1. ed. New York, NY, USA: APRESS, 2016. Citado 3 vezes nas páginas 16, 20 e 21.

CLAYPOOL, K. T.; CLAYPOOL, M. On frame rate and player performance in first person shooter games. *Multimedia systems*, Springer, v. 13, n. 1, p. 3–17, 2007. Citado na página 38.

CPU-WORLD. *AMD Ryzen 7 5700X specifications*. 2022. <https://www.cpu-world.com/CPUs/Zen/AMD-Ryzen%207%205700X.html>. Acesso em: 21 Abr.2025. Citado na página 29.

CPU-WORLD. *AMD Ryzen 7 5700X3D specifications*. 2024. <https://www.cpu-world.com/CPUs/Zen/AMD-Ryzen%207%205700X3D.html>. Acesso em: 21 Abr.2025. Citado na página 29.

DAVIES, H. et al. *Wine User's Guide*. 2024. Disponível em: <https://gitlab.winehq.org/wine/wine/-/wikis/Wine-User's-Guide>. Acesso em: 15 Abr. 2025. Citado 2 vezes nas páginas 15 e 21.

DEEP SILVER, 4A GAMES. *Metro Exodus Enhanced Edition*. 2021. Jogo eletrônico. Disponível em: <https://www.metrothegame.com/en-us/>. Acesso em: 19 Mar. 2025. Citado na página 34.

DEEP SILVER, 4A GAMES. *PC SPECS*. 2021. <https://www.metrothegame.com/pcspecs/>. Acesso em: 20 Abr.2025. Citado na página 35.

DOWNING, S.; WEBSTER, S. *WD Black SN770 SSD Review: A Wolf in Sheep's Clothing*. 2022. <https://www.tomshardware.com/reviews/wd-black-sn770-ssd-review>. Acesso em: 21 Abr.2025. Citado na página 29.

FEDORA DOCUMENTATION TEAM. Getting Started. *In*: FEDORA DOCUMENTATION TEAM. *Fedora User Docs* . 2024. Disponível em: <https://docs.fedoraproject.org/en-US/fedora/latest/getting-started/>. Acesso em: 15 Abr. 2025. Citado na página 32.

FLIGHTLESSMANGO. *MangoHud. Github*. 2025. Disponível em: <https://github.com/flightlessmango/MangoHud>. Acesso em: 15 Jan. 2025. Citado 2 vezes nas páginas 23 e 39.

FREE SOFTWARE FOUNDATION. *What is GNU?* 2025. <https://www.gnu.org/>. Acesso em: 23 Jan. 2025. Citado na página 20.

GAIJIN ENTERTAINMENT. *War Thunder*. 2025. Jogo eletrônico. Disponível em: <https://warthunder.com/en/>. Acesso em: 19 Mar. 2025. Citado na página 34.

GAMEBENCH. *Game Performance Metrics That Matter: Guide To Interpretation And Action*. 2019. Disponível em: <https://blog.gamebench.net/game-performance-metrics-that-matter>. Acesso em: 19 mar. 2024. Citado 2 vezes nas páginas 14 e 22.

GREGG, B. *Systems Performance: Enterprise and the Cloud*. 2. ed. Boston, MA: Addison-Wesley Professional, 2020. Citado 2 vezes nas páginas 22 e 24.

GREGORY, J. *Game engine architecture*. 3. ed. Boca Raton, FL, USA: CRC Press, 2018. Citado 2 vezes nas páginas 19 e 20.

GURU3D. *RivaTuner (RTSS) homepage*. 2025. Guru3D. Disponível em: <https://www.guru3d.com/page/rivatuner-rtss-homepage/>. Acesso em: 12 Jan. 2025. Citado na página 23.

HERTZOG, R.; MAS, R. *The Debian Administrator's Handbook*. Sorbiers, FR: Freexian SARL, 2020. Citado na página 32.

HWU, W.; KIRK, D.; HAJJ, I. E. *Programming massively parallel processors*. 4. ed. Cambridge, MA, USA: Morgan Kaufmann, 2023. Citado 2 vezes nas páginas 19 e 23.

INTEL. *Intel® PresentMon*. 2025. <https://game.intel.com/us/intel-presentmon/>. Acesso em: 12 Jan. 2025. Citado na página 23.

JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York, NY, USA: John Wiley & Sons, 1991. Citado 5 vezes nas páginas 14, 23, 25, 39 e 42.

JULIAN, M. *Practical Monitoring: Effective Strategies for the Real World*. Sebastopol, CA, USA: O'Reilly Media, 2017. Citado na página 22.

JULLIARD, A. *Wine History*. 2024. Disponível em: <https://gitlab.winehq.org/wine/wine/-/wikis/Wine-History>. Acesso em: 15 Abr. 2025. Citado na página 21.

KHRONOS GROUP. What is Vulkan?. *In*: KHRONOS GROUP. *Vulkan Guide*. 2025. Disponível em: <https://docs.vulkan.org/guide/latest/what_is_vulkan.html>. Acesso em: 14 jan 2025. Citado na página 20.

KINGSTON. *PC or console? What's your preference?* 2023. <https://www.kingston.com/en/blog/gaming/pc-vs-console>. Acesso em: 21 Jan. 2025. Citado na página 14.

KOPEL, M.; BOżEK, M. "Is Proton Good Enough?" - A Performance Comparison Between Gaming on Windows and Linux. In: NGUYEN, N. T. et al. (Ed.). *Computational Collective Intelligence*. Cham: Springer Nature Switzerland, 2023. p. 634–646. Citado 2 vezes nas páginas 26 e 32.

LENGYEL, E. *Foundations of Game Engine Development*: Volume 2: Rendering. Lincoln, CA, USA: Terathon Software LLC, 2019. Citado na página 19.

LINUX KERNEL ORGANIZATION. *Frequently asked questions*. 2024. <https://www.kernel.org/category/faq.html/>. Acesso em: 23 Jan. 2025. Citado na página 20.

LOVE, R. *Linux Kernel Development*. 3. ed. Boston, MA: Addison-Wesley Professional, 2010. Citado na página 21.

MARTINOVIC, G.; BALEN, J.; CUKIC, B. Performance evaluation of recent windows operating systems. *JUCS - Journal of Universal Computer Science*, Journal of Universal Computer Science, v. 18, n. 2, p. 218–263, 2012. Citado 2 vezes nas páginas 26 e 60.

MATPLOTLIB DEVELOPMENT TEAM. *Matplotlib*. 2025. <https://matplotlib.org/>. Versão 3.9.3. Acesso em: 17 Abr. 2025. Citado na página 30.

MENASCE, D. A.; ALMEIDA, V.; DOWDY, L. *Performance by Design: Computer Capacity Planning*. Upper Saddle River, NJ: Prentice Hall, 2004. v. 1. Citado na página 23.

MESSAOUDI, F. et al. Performance analysis of game engines on mobile and fixed devices. *ACM Transactions on Multimedia Computing, Communications, and Applications*, Association for Computing Machinery, New York, NY, USA, v. 13, p. 1–28, 09 2017. Citado 3 vezes nas páginas 19, 25 e 38.

MICRO-STAR INTERNATIONAL. *MPG B550 GAMING PLUS*. 2023. <https://br.msi.com/Motherboard/MPG-B550-GAMING-PLUS/>. Acesso em: 21 Abr.2025. Citado na página 29.

MICRO-STAR INTERNATIONAL. *MSI AFTERBURNER*. 2025. <https://www.msi.com/Landing/afterburner/graphics-cards>. Acesso em: 15 Jan. 2025. Citado na página 23.

MICROSOFT. *Windows game development guide*. 2024. Disponível em: <https://learn.microsoft.com/en-us/windows/uwp/gaming/e2e>. Acesso em: 12 dez 2024. Citado na página 20.

MONTGOMERY, D. C. *Design and Analysis of Experiments*. 8. ed. Hoboken, NJ, USA: John Wiley & Sons, 2012. Citado 2 vezes nas páginas 24 e 40.

NOGUEIRA, J. G. *Número alto! Steam Deck já consegue rodar mais de 14 mil jogos*. 2024. Disponível em: <https://meups.com.br/tecnologia/steam-deck-rodar-mais-de-14-mil-jogos/>. Acesso em: 24 mar. 2024. Citado na página 14.

NUMPY TEAM. *NumPy*. 2025. <https://numpy.org/>. Versão 2.1.3. Acesso em: 17 Abr. 2025. Citado na página 30.

NVIDIA CORPORATION. *FrameView Integrated Frame Benchmarking & Power Tool*. 2022. <https://images.nvidia.com/content/geforce/technologies/frameview/frameview-1-4-user-guide-web-version.pdf>. Acesso em: 10 Fev. 2025. Citado 2 vezes nas páginas 23 e 39.

PERKTOLD, J. et al. *Statsmodels*. 2024. <https://www.statsmodels.org/stable/index.html>. Versão 0.14.4. Acesso em: 17 Abr. 2025. Citado na página 30.

PROTONDB. *ProtonDB*. 2025. <https://www.protondb.com/>. Acesso em: 15 Jan. 2025. Citado 2 vezes nas páginas 15 e 22.

PYTHON SOFTWARE FOUNDATION. *pip*. 2024. <https://pypi.org/project/pip/>. Versão 25.0.1. Acesso em: 17 Abr. 2025. Citado na página 30.

PYTHON SOFTWARE FOUNDATION. *Python*. 2024. <https://seaborn.pydata.org/>. Versão 3.13.3. Acesso em: 17 Abr. 2025. Citado na página 30.

REBOHLE, P. et al. *DXVK. Github*. 2025. Disponível em: <https://github.com/doitsujin/dxvk>. Acesso em: 20 mar 2025. Citado na página 22.

ROACH, J.; CONNATSER, M. *What is AMD 3D V-Cache? Extra gaming performance unlocked*. 2023. <https://www.digitaltrends.com/computing/what-is-amd-3d-v-cache/>. Acesso em: 21 Abr.2025. Citado na página 29.

SCIPY CONTRIBUTORS. *SciPy*. 2025. <https://scipy.org/>. Versão 1.15.2. Acesso em: 17 Abr. 2025. Citado na página 30.

SHAH, P. *10 Reasons Why Your Games Keep Crashing (And How to Fix Issues). Make Use Of (MUO)*. 2024. Disponível em: <https://www.makeuseof.com/tag/why-do-my-games-keep-crashing/>. Acesso em: 22 jan 2025. Citado na página 39.

STEAM, GAIJIN ENTERTAINMENT. *War Thunder*. 2025. <https://store.steampowered.com/app/236390/War_Thunder/>. Acesso em: 20 Abr.2025. Citado na página 35.

STEAM, XBOX GAME STUDIOS, IRON GALAXY STUDIOS. *Killer Instinct*. 2013. <https://store.steampowered.com/app/577940/Killer_Instinct/>. Acesso em: 20 Abr.2025. Citado na página 35.

SULAIMAN, N. S.; RAFFI, A. S. H. A. Comparison of operating system performance between windows 10 and linux mint. *International Journal of Synergy in Engineering and Technology*, v. 2, n. 1, p. 92–102, 2021. Citado na página 25.

TANENBAUM, A. S.; BOS, H. *Modern Operating Systems*. 4. ed. Upper Saddle River, NJ, USA: Prentice Hall, 2014. Citado 2 vezes nas páginas 15 e 21.

TECHPOWERUP. *NVIDIA GeForce RTX 4060*. 2023. <https://www.techpowerup.com/gpu-specs/geforce-rtx-4060.c4107>. Acesso em: 21 Abr.2025. Citado na página 28.

THE PANDAS DEVELOPMENT TEAM. *pandas*. 2025. <https://pandas.pydata.org/>. Versão 2.2.3. Acesso em: 17 Abr. 2025. Citado na página 30.

TURNER, J. R.; THAYER, J. F. *Introduction to Analysis of Variance: Design, Analysis & Interpretation*. Thousand Oaks, CA, USA: SAGE Publications, 2001. Citado na página 24.

UBISOFT. *Assassin's Creed Origins*. 2017. Jogo eletrônico. Disponível em: <https://www.ubisoft.com/pt-br/game/assassins-creed/origins>. Acesso em: 19 Mar. 2025. Citado 2 vezes nas páginas 33 e 34.

UBISOFT. *System requirements for Assassin's Creed Origins*. 2017. <https://www.ubisoft.com/en-us/help/assassins-creed-origins/connectivity-and-performance/article/system-requirements-for-assassins-creed-origins/000064383>. Acesso em: 20 Abr.2025. Citado na página 35.

UBISOFT. *Far Cry 6*. 2021. Jogo eletrônico. Disponível em: <https://www.ubisoft.com/pt-br/game/far-cry/far-cry-6>. Acesso em: 19 Mar. 2025. Citado 2 vezes nas páginas 33 e 34.

UBISOFT. *System requirements for Far Cry 6*. 2022. <https://www.ubisoft.com/en-us/help/far-cry-6/article/system-requirements-for-far-cry-6/000098943>. Acesso em: 20 Abr.2025. Citado na página 35.

VALVE. *Steam :: Steam for Linux :: Introducing a new version of Steam Play*. 2018. <https://steamcommunity.com/games/221410/announcements/detail/1696055855739350561>. Acesso em: 12 Dez. 2024. Citado na página 22.

VALVE. *Steam Hardware & Software Survey*. 2025. Disponível em: <https://store.steampowered.com/hwsurvey/>. Acesso em: 19 Abr. 2025. Citado na página 14.

VARCHOLIK, P. *Real-time 3D rendering with DirectX and HLSL: A Practical Guide to Graphics Programming*. Boston, MA: Addison-Wesley Professional, 2014. Citado na página 19.

WASHINGTON, T.; ARBULU, L. F.; M, M. Fedora Linux Kernel Overview. Versão F40. *In*: FEDORA DOCUMENTATION TEAM. *Fedora Quick Docs*. 2024. Disponível em: <https://docs.fedoraproject.org/en-US/quick-docs/kernel-overview/>. Acesso em: 15 Abr. 2025. Citado na página 32.

WASKOM, M. L. *Seaborn*. 2024. <https://seaborn.pydata.org/>. Versão 0.13.2. Acesso em: 17 Abr. 2025. Citado na página 30.

WINEHQ. *Welcome*. 2025. Wine Application Database. Disponível em: <https://appdb.winehq.org/>. Acesso em: 15 Jan. 2025. Citado na página 22.

XBOX GAME STUDIOS, IRON GALAXY STUDIOS. *Killer Instinct*. 2013. Jogo eletrônico. Disponível em: <https://www.xbox.com/pt-BR/games/store/killer-instinct/9NBLGGH1Z149>. Acesso em: 19 Mar. 2025. Citado 2 vezes nas páginas 33 e 34.