Avaliação de Desempenho de um Framework para Ajuste Dinâmico de Dificuldade em Jogos Single Player

Carlos Henrique R. Souza[1],
Luciana de O. Berretta[1], **Sérgio T. de Carvalho**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brazil

[1]{carloshenriquesouza, luciana, sergio}@inf.ufg.br

Abstract. *The development of Dynamic Difficulty Adjustment (DDA) mechanisms in digital games is an alternative to increasing player retention. There are many open questions, and there is still little discussion on the impact of using DDA mechanisms on game performance. In this context, this work presents an experiment carried out to verifying the possible impacts when mechanisms generated with DDA-MAPEKit, a framework developed for this purpose, are inserted into a game. Four metrics were considered: frames per second (FPS), frame time, and RAM and CPU consumption. The effects on FPS and frame time were noted (approximately* 16.1% *reduction in FPS), which can impact the player's experience from a certain number of simultaneous engines (n = 10). This reduction was considered acceptable at a value of less than 10 mechanisms.*

Resumo. *O desenvolvimento de mecanismos de Ajuste Dinâmico de Dificuldade (DDA) em jogos digitais é uma alternativa para aumentar a retenção do jogador. Muitas são as questões em aberto, e ainda é escassa a discussão acerca do impacto do uso de mecanismos de DDA no desempenho do jogo. Neste horizonte, este trabalho apresenta um experimento realizado para verificar possíveis impactos ao serem inseridos em um jogo mecanismos gerados com o DDA-MAPEKit, um framework desenvolvido para este propósito. Para isso, foram consideradas quatro métricas: quadros por segundo (FPS), frame time, além dos consumos de memória RAM e de CPU. Notaram-se efeitos no FPS e frame time (redução de aproximadamente* 16.1% *no FPS), que podem impactar na experiência do jogador a partir de um determinado número de mecanismos simultâneos (n = 10). Esta redução foi considerada aceitável em um valor inferior a 10 mecanismos.*

1. Introdução

Nos últimos anos, tem-se observado um aumento significativo na demanda por jogabilidade adaptativa em jogos digitais [Seyderhelm and Blackmore 2021]. Os jogadores buscam experiências imersivas que atinjam o equilíbrio entre o desafio e o seu nível de habilidade. Uma solução que emergiu para atender a essa demanda é o conceito de Ajuste Dinâmico de Dificuldade (*Dynamic Difficulty Adjustment* – DDA) [Seyderhelm and Blackmore 2021, Streicher and Smeddinck 2016]. Este conjunto de técnicas envolve o ajuste automático de parâmetros do jogo com base na *performance* do jogador, visando prevenir frustrações e desistências, mantendo, portanto, a retenção do jogador [Zohaib 2018].

Apesar dos avanços no DDA, ainda persistem questões de pesquisa em aberto. Primeiramente, observa-se uma crescente necessidade de se adotar abordagens modularizadas que

permitam a integração de múltiplas estratégias de DDA dentro do mesmo mecanismo ou entre diferentes mecânicas/dinâmicas de jogo [Seyderhelm and Blackmore 2021, Zohaib 2018]. Ademais, busca-se o desenvolvimento de ferramentas e modelos generalizáveis, que possam ser facilmente implementados em motores de jogos populares e que atendam a uma ampla gama de gêneros de jogos [Sepulveda et al. 2020, Mi and Gao 2022]. Além destas questões, nota-se que ainda é escassa a discussão acerca dos impactos do uso de mecanismos de DDA no desempenho do jogo. Esta é uma dimensão que não pode ser negligenciada, haja vista que, ao tentar gerar impactos positivos no jogador com o balanceamento da dificuldade, um mecanismo que provoque a queda no desempenho pode, ao contrário, gerar frustração e ocasionar o abandono do jogo [Wang et al. 2023, Claypool et al. 2006].

Neste contexto, este trabalho tem por objetivo contribuir para a investigação acerca destes possíveis impactos. Apresenta-se brevemente o DDA-MAPEKit [Souza et al. 2023b], um *framework* desenvolvido para facilitar a criação de mecanismos de DDA em jogos *single player*. Baseada no modelo MAPE-K [Weyns 2021], oriundo da área de sistemas autoadaptativos, a solução fundamenta-se neste ciclo principal de Monitorar-Analisar-Planejar-Executar, sobre uma Base de Conhecimento, e acopla diversas estratégias a esta "espinha dorsal". Ele baseia-se na construção de um mecanismo para cada grupo de mecânicas (ou *dinâmica* de jogo) que descreva a dificuldade do jogo. Nesse sentido, foi conduzido um experimento para verificar eventuais impactos no desempenho com a introdução dos mecanismos de DDA gerados com o *framework*, utilizando-se execuções simuladas em um pequeno jogo construído para este estudo. Nos testes, foram avaliadas métricas comumente usadas para a avaliação de desempenho de jogos [Li et al. 2020, Claypool and Claypool 2009, Dunlop 2003]: *frames* por segundo (FPS – que corresponde à quantidade de quadros renderizados no jogo em um segundo), *frame time* (inverso do FPS, corresponde ao tempo de renderização de um único quadro), consumo de memória RAM e de CPU. A análise destas métricas permitiu coletar evidências acerca dos efeitos no jogo, ao ser variada a quantidade de mecanismos utilizados simultaneamente.

Este artigo está organizado em outras seis seções: a Seção 2 discute a fundamentação teórica que embasou este estudo; a Seção 3, por sua vez, apresenta os trabalhos relacionados; a Seção 4 apresenta brevemente o DDA-MAPEKit, situando-o no contexto de soluções para o balanceamento dinâmico de dificuldade; a Seção 5 apresenta o procedimento adotado na realização do experimento; a Seção 6 discute os resultados obtidos; a Seção 7, por fim, apresenta as considerações finais e os horizontes da pesquisa.

2. Fundamentação Teórica

Nesta seção, são apresentados os conceitos que fundamentam este trabalho, a saber, questões sobre o Ajuste Dinâmico de Dificuldade (DDA) e ainda sobre as métricas de avaliação de desempenho em jogos digitais.

2.1. Ajuste Dinâmico de Dificuldade (DDA)

No domínio dos jogos digitais, há uma crescente demanda por jogabilidade adaptativa, que envolve o ajuste dinâmico dos parâmetros do jogo com base na experiência do jogador. O Ajuste Dinâmico de Dificuldade (DDA) é uma abordagem comumente usada para alcançar essa adaptabilidade. Assim, as técnicas de DDA visam equilibrar automaticamente o nível de dificuldade do jogo, reduzir a frustração e evitar que os jogadores abandonem o jogo [Seyderhelm and Blackmore 2021, Streicher and Smeddinck 2016]. Dessa forma, o conceito de DDA é um tópico central nas discussões sobre jogabilidade adaptativa [Seyderhelm and Blackmore 2021].

O principal objetivo dos ajustes é manter um equilíbrio entre o desafio do jogo e as habilidades do jogador. Quando esses dois aspectos estão alinhados, os jogadores entram no "canal de *flow*", caracterizado pelo envolvimento, retenção e imersão do jogador [Seyderhelm and Blackmore 2021]. No entanto, se o jogo se tornar muito fácil ou muito difícil, os jogadores podem experimentar frustração e lutar para manter seu interesse [Seyderhelm and Blackmore 2021, Streicher and Smeddinck 2016]. Para enfrentar esse desafio, os jogos com DDA precisam identificar quando os jogadores se desviaram do "canal de *flow*", usando um avaliador de desempenho, ou seja, um mecanismo que pode medir o desempenho do jogador. Em seguida, um mecanismo de ajuste é empregado para modular variáveis relacionadas à dificuldade e restaurar um nível apropriado de desafio [Streicher and Smeddinck 2016].

Nesse campo, muitos trabalhos recentes discutem vários problemas de pesquisa em aberto relacionados ao DDA, incluindo a integração de diferentes estratégias de DDA [Seyderhelm and Blackmore 2021, Zohaib 2018], o desenvolvimento de novos métodos e a otimização das abordagens existentes [Sepulveda et al. 2020]. Além disso, há uma necessidade de criar ferramentas que simplifiquem a implementação desses mecanismos [Mi and Gao 2022, Sepulveda et al. 2020] e ainda de projetar soluções versáteis aplicáveis a vários gêneros e motores de jogos populares. Outrossim, todas essas questões são objeto de investigação e pesquisa [Mi and Gao 2022, Sepulveda et al. 2020].

2.2. Avaliação de Desempenho em Jogos Digitais

Avaliar o desempenho em jogos digitais é uma necessidade crucial da indústria de desenvolvimento de jogos e também da pesquisa acadêmica. A crescente complexidade dos jogos, aliada às demandas dos jogadores por experiências cada vez mais imersivas e fluidas, torna essencial compreender como os jogos operam em termos de desempenho. Dessa forma, são métricas comumente utilizadas na literatura para avaliar o desempenho [Li et al. 2020, Claypool and Claypool 2009, Dunlop 2003][1]:

- **FPS (*frames per second*):** Representa o número de quadros (*frames*) renderizados por segundo e é diretamente relacionado à "suavidade" ou "fluidez" da experiência do jogador. Quanto maior o FPS, mais fluida a jogabilidade. Valores considerados ideais estão em torno dos 60 FPS, sendo 24 FPS o valor considerado mínimo.
- **Frame Time:** Trata-se da métrica inversa ao FPS, representando o tempo gasto para renderizar um único quadro. Embora seja menos comum entre os jogadores, é valioso na detecção de variações de desempenho. Um *frame time* constante indica uma experiência de jogo mais estável. Isso é especialmente importante para evitar "gargalos" (*stuttering*) que podem prejudicar a jogabilidade.
- **Consumo de memória RAM:** O consumo de memória RAM é outra métrica para avaliar o desempenho. Jogos que consomem uma quantidade excessiva de RAM podem levar a problemas de desempenho, como travamentos e lentidão, especialmente em sistemas com recursos limitados. É necessário avaliar o consumo de RAM para garantir que o jogo seja acessível a um público amplo e para otimizar o uso de recursos.
- **Consumo de CPU:** A CPU (processamento) é um componente-chave na execução de jogos, especialmente em títulos que dependem fortemente de física e simulações

[1] Além destes trabalhos, foram consultados:
https://unity.com/how-to/best-practices-for-profiling-game-performance;
https://unrealartoptimization.github.io/book/process/measuring-performance/.

complexas. Medir o consumo de CPU ajuda a identificar gargalos de processamento que podem afetar o desempenho do jogo.

Embora essas métricas sejam amplamente utilizadas na literatura e na indústria de jogos, elas apresentam limitações, que também são discutidas pelos estudos consultados. O FPS, por exemplo, pode variar significativamente dependendo da cena do jogo e da configuração do *hardware*, tornando-o, às vezes, uma métrica imprecisa para comparações absolutas. O *frame time*, por sua vez, pode ser influenciado por eventos ocasionais, como o carregamento de texturas, resultando em picos que não refletem o desempenho geral. O consumo de RAM e CPU também pode variar, a depender da complexidade da jogabilidade.

Apesar de suas limitações, essas métricas continuam sendo ferramentas valiosas, fornecendo uma visão geral do funcionamento do jogo. Neste estudo, essas métricas serão usadas como indicadores para identificar e evidenciar possíveis impactos nos mecanismos de Ajuste Dinâmico de Dificuldade (DDA) submetidos ao experimento.

3. Trabalhos Relacionados

Embora seja escassa a literatura que discute o desempenho de mecanismos de balanceamento da dificuldade (o enfoque está predominantemente atrelado a processos avaliativos que envolvam o jogador), alguns trabalhos apontam para essa necessidade e, portanto, fundamentaram esta pesquisa.

Os estudos de [Tagliaro 2022] e ainda de [Hunicke 2005], ao apresentarem as configurações de *setup* para a realização de experimentos envolvendo jogos com DDA, fazem uma descrição detalhada do processador, quantidade de memória RAM e ainda da frequência na qual as operações de DDA são realizadas no jogo. Ademais, de forma explícita, os autores mencionam que tais escolhas foram feitas em razão da prevenção de um cenário de baixo desempenho, o que poderia prejudicar os resultados do experimento e a experiência dos jogadores. Fica constatada, portanto, a preocupação por parte de alguns trabalhos, em relação ao desempenho.

Outras abordagens, porém, consideram o desempenho para fins de análise. O *framework* proposto por [Berseth et al. 2018], que faz o gerenciamento de objetos e personagens não-jogáveis, discute questões relativas ao tempo de execução do *framework* e possíveis impactos no jogo. Ainda que de maneira breve, os autores atrelam o desempenho à quantidade de polígonos da cena e à população de objetos a serem instanciados e gerenciados pelo *framework*. Por fim, ressaltam a importância de se adotar práticas de otimização para que estes números não impactem a jogabilidade. Nesta mesma direção caminha o trabalho de [de Lima et al. 2022], que discute o uso de mecanismos de modelagem de jogador e *machine learning* para ajustar jogos de terror. O *frame time* obtido de 3.1ms foi considerado satisfatório.

4. DDA-MAPEKit: Um Framework para DDA para Jogos Single Player

O *framework* que será submetido ao experimento é o DDA-MAPEKit, desenvolvido pelos autores para a aplicação em jogos *single player* [Souza et al. 2023b, Souza et al. 2023a]. Visando abordar as situações mencionadas anteriormente, busca-se apresentar um *framework* abrangente que não apenas incorpora a abordagem discutida, mas também possibilita uma implementação flexível. Ao introduzir o DDA-MAPEKit, a solução proposta incorpora os elementos do ciclo MAPE-K, conhecido por oferecer uma solução altamente modular e versátil para sistemas autoadaptativos, com base em quatro atividades (Monitorar, Analisar,

Planejar e Executar), sobre uma Base de Conhecimento [Weyns 2021]. Os ciclos MAPE-K foram integrados às especificidades dos mecanismos de DDA. O *framework* foi implementado para a Unity Engine[2].

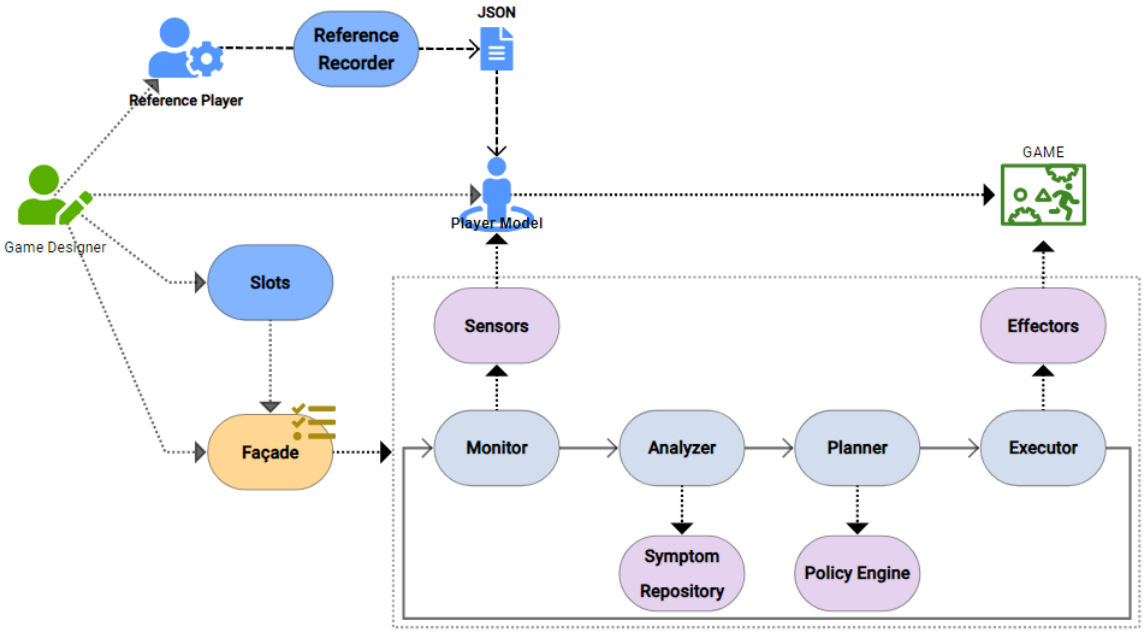

Figura 1. Diagrama que ilustra o funcionamento do DDA-MAPEKit.

Nesse sentido, a dinâmica de funcionamento do DDA-MAPEKit pode ser visualizada no esquema da Figura 1. As setas pontilhadas em cinza representam as atividades do Game Designer; a área delimitada pelo retângulo de linhas pontilhadas mostra o que é gerenciado pela fachada; as setas pontilhadas em preto representam a comunicação que acontece internamente entre os módulos; as setas abertas não-pontilhadas representam o *loop* principal do *framework*; por fim, as setas pretas pontilhadas representam inserções que devem acontecer previamente à execução do *framework*.

Em primeiro lugar, o Game Designer precisa delimitar um conjunto de atributos do jogador que será monitorado, adicionando-o em um Modelo de Jogador (Player Model – uma classe que descreve o estado atual do jogador). Estes atributos serão obtidos diretamente das variáveis de jogo. A seguir, para o cálculo da *performance* do jogador, realizado segundo a proposta de [Sepulveda et al. 2020] a partir da razão entre os valores dos atributos e valores de referência, será necessário obter estes "valores padrão". Assim, o Game Designer utiliza um Reference Player (que pode ser um ou mais jogadores reais, ou ainda um simulador) para obter estes dados, gravados por meio do Reference Recorder em um arquivo JSON.

Com estas definições feitas, o Game Designer pode configurar o framework, a partir da fachada (Façade), e implementar os Slots, ou seja, métodos que serão utilizados por alguns módulos do sistema, como aquele destinado a efetuar as modificações nas variáveis de jogo, chamado pelos Effectors.

Assim, o DDA-MAPEKit pode iniciar o seu ciclo. Primeiramente, o Monitor observa os valores do Player Model obtidos por meio de sensores. Assim, caso algum valor esteja fora da faixa estipulada no atributo do modelo, chama-se o Analyzer, que calcula a *performance*

[2]https://unity.com/

do jogador (basicamente a razão entre o valor atual do atributo e o valor de referência) e acessa o Symptom Repository, que coleciona sintomas. Cada sintoma é composto por uma descrição e um limiar (por exemplo, "baixo" como a descrição e [0.2,0.3] como o limiar). Esses limiares são utilizados para avaliar se o valor calculado está alinhado com um sintoma específico. Se for encontrada correspondência entre o desempenho e algum sintoma, são necessários ajustes nas variáveis de dificuldade da mecânica. Assim, um pedido de adaptação é criado para armazenar o sintoma identificado, e o Planejador é subsequentemente notificado.

Por conseguinte, o pedido de adaptação é recebido pelo Planner, que examina se há uma regra de ajuste associada ao sintoma identificado. Cada regra compreende um sintoma e sua ação correspondente. A Policy Engine, consultada pelo Planner, consiste em uma coleção de tais regras. Uma ação engloba um conjunto de valores para as variáveis que precisam ser modificadas para equilibrar a dificuldade das mecânicas específicas (por exemplo, pode haver uma regra que associa o sintoma "baixo" ao valor "10.5" para a variável "número de itens", diretamente relacionada à dificuldade do jogo). Ao encontrar regras aplicáveis, o Planejador elabora um Plano de Mudança que encapsula essas ações. O Plano de Mudança é então transmitido ao Executor, o qual é subsequentemente notificado. Após receber a notificação, o Executor organiza as ações e as encaminha aos Effectors. Os Efeitos utilizam um método de atualização (programado pelo Designer nos Slots) para implementar as mudanças nos valores das variáveis do jogo, ou seja, atribuem os novos valores. Com o sistema atualizado, o ciclo recomeça.

O *framework* proposto destina-se a ser utilizado separadamente para cada grupo de mecânicas de jogo[3] que contribui para descrever a dificuldade do jogo. Todos os mecanismos, no entanto, compartilham o Player Model. Essa compreensão visa modularizar o ajuste de dificuldade, permitindo uma abordagem que considere as especificidades de cada aspecto da dificuldade, bem como a combinação de abordagens, ou seja, a escolha e integração de diferentes estratégias para cada uma delas. Entretanto, compreende-se que existem outras possibilidades de uso da ferramenta desenvolvida.

5. Experimento

Para a realização deste experimento, foi elaborado um jogo, cuja tela pode ser observada na Figura 2. Ele apresenta um personagem (cápsula vermelha) que se movimenta pelo cenário tentando capturar moedas (objetos amarelos) ao longo do tempo de jogo (limitado a um minuto). Novas moedas são instanciadas quando o jogador termina de capturar todas as que estão disponíveis. Além disso, o personagem deverá fugir do inimigo (esfera verde), pois, ao colidir com ele, perde uma de suas vidas e, se perder todas as cinco disponibilizadas, o jogo é encerrado.

Nesse sentido, foram selecionadas três dinâmicas do jogo. Em primeiro lugar, o **sistema de pontuação** (*scoring*), que envolve a quantidade de moedas que serão instanciadas no cenário (no intervalo de 1 a 6). Ademais, a dinâmica de **perseguição** (*chasing*) também foi escolhida, que envolve a velocidade do movimento do inimigo (valores de 1 a 10). Por fim, a dinâmica de **penalização** (*penalization*), que envolve a quantidade de vidas que o inimigo captura do jogador a cada ataque (colisão), iniciando em 1 e podendo chegar a 2. Para cada uma destas, foi criado um mecanismo de DDA, utilizando-se o DDA-MAPEKit e o fluxo apresentado anteriormente.

A condução do experimento envolveu a realização de cinco simulações do jogo com

[3]Compreendidas também como "dinâmicas de jogo", segundo autores como [Vahlo et al. 2017].

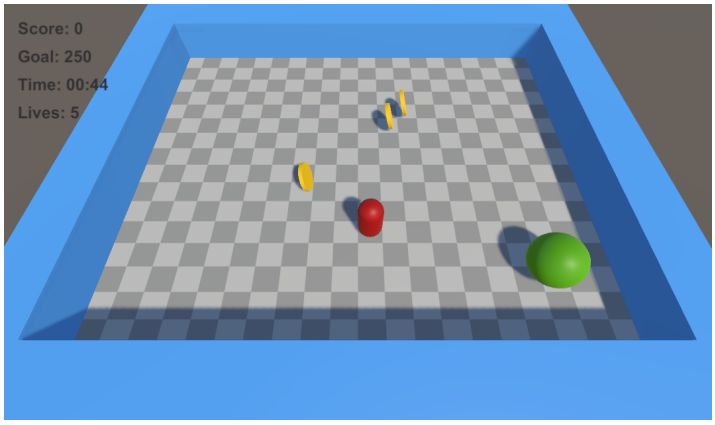

Figura 2. Jogo elaborado para a realização do experimento.

cada quantidade possível de mecanismos executados simultaneamente (iniciando com nenhum mecanismo – jogo sem DDA – até 3 mecanismos). Para controlar o personagem, foi utilizado um agente que conhecia os movimentos possíveis e as regras do jogo, tomando as decisões aleatoriamente. A cada execução, inferiu-se internamente os valores de FPS e *frame time* mínimos, máximos e médios. Além disso, externamente, foram verificados os consumos mínimos, máximos e médios de CPU e RAM, utilizando-se a ferramenta ProcessPerformance[4]. Todos os experimentos foram realizados no editor da Unity Engine. Foram usadas as seguintes configurações: Intel Core i7-2600 CPU 3.40 GHz, RAM 12GB, GPU NVIDIA GeForce GT 430.

6. Resultados e Discussão

Os resultados obtidos nas simulações estão dispostos na Tabela 1, e correspondem à média aritmética das medições realizadas nas cinco simulações para cada quantidade de mecanismos simultâneos (0, 1, 2 e 3 *loops*). Os valores foram dispostos com precisão de cinco casas decimais.

	0	**1**	**2**	**3**
Min FPS	3,00000	3,00000	3,00000	3,00000
Max FPS	440,21060	400,45490	400,37800	428,72820
AVG FPS	146,82500	140,85470	135,02430	123,28920
Min FT (s)	0,00227	0,00249	0,00249	0,00233
Max FT (s)	0,33333	0,33333	0,33333	0,33333
AVG FT (s)	0,00681	0,00710	0,00740	0,00811
RAM (MB)	2,44820	2,44060	2,44540	2,44200
CPU (%)	20,58600	20,68000	20,15600	20,79800

Tabela 1. Resultados das medições realizadas (por número de mecanismos).

Em relação ao FPS, foram notadas variações tanto nos valores máximos quanto nos valores médios (Figura 3). As variações de FPS máximo se mantiveram na faixa dos 400 quadros por segundo, apresentando uma queda de aproximadamente 9% do cenário sem DDA para o cenário com um mecanismo de DDA, quadro que foi mantido sem muitas variações quando adicionado mais um mecanismo. Por fim, notou-se uma subida de 7% quando foi acrescido o terceiro *loop*. Como não houve uma tendência de subida ou descida ao longo

[4]https://github.com/SoftwareImpacts/SIMPAC-2021-199

dos testes, é possível que as variações registradas correspondam a questões alheias à ação do *framework*. Os valores mínimos não apresentaram variação em nenhum momento.

Em relação aos valores médios, porém, registrou-se uma tendência de queda. Ao total, foram aproximadamente 16.1% de queda, ao serem comparados o primeiro com o último cenário. As variações ocorreram na ordem de 4% comparando-se o cenário sem a ação do mecanismo com a atuação de um *loop*, 4.1% ao ser adicionado o segundo *loop*, e ainda 8.6%, sendo habilitado o terceiro, considerando sempre valores aproximados. Esta última queda foi a maior dentre as variações observadas no FPS médio. Houve, portanto, um impacto no desempenho do jogo, causado pela ação dos mecanismos de DDA. Entretanto, ainda com este impacto, o valor de FPS foi mantido muito acima do valor considerado ideal pela literatura (em torno de 60 FPS), evidenciando ser possível a utilização do *framework* sem danos graves ao desempenho.

Dada a natureza da métrica de *frame time*, tem-se que ela apresentou tendências de mesma intensidade, naturalmente na direção oposta: uma relativa manutenção nos valores mínimos e máximos, além de um crescimento no *frame time* médio. Entretanto, também deve ser observada a constância entre os valores de *frame time* ao longo da execução do jogo. Observou-se que, nos primeiros segundos (correspondentes ao carregamento do jogo), o *frame time* apresentou flutuações significativas devido à maior carga de processamento. No entanto, após esse período inicial, houve uma relativa estabilização, indicando um *frame time* relativamente constante em todas as execuções, independentemente da presença dos mecanismos de DDA. Isso sugere que o uso do DDA-MAPEKit não afetou negativamente a estabilidade da experiência de jogo.

Em relação ao consumo de CPU e RAM, também não foram constatadas grandes alterações que indicassem tendência de impacto por parte dos mecanismos de DDA. As variações ocorridas não apresentaram tendências de subida ou queda significativas, e também podem ser atribuídas a razões alheias à ação dos mecanismos de balanceamento de dificuldade.

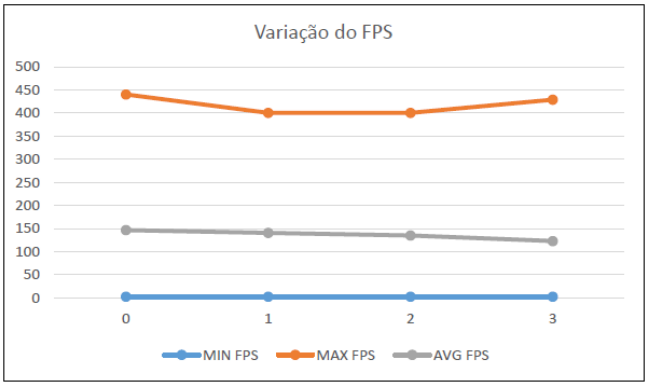

Figura 3. Variação do FPS.

Ainda que os impactos observados neste estudo não apontem para grandes problemas de desempenho, dada a tendência de queda do FPS médio, surge a possibilidade de, utilizando-se projeções estatísticas e os dados recolhidos no experimento, simular a manutenção das tendências observadas e projetar a curva de variação do FPS médio para valores maiores que 3 mecanismos simultâneos. O resultado dessa projeção pode ser observado no gráfico da Figura 4. Observou-se que, no caso dos dados deste experimento, os impactos atingem patamares que comprometem o FPS ideal (60 FPS) a partir de 10 *loops*

executados ao mesmo tempo. Esta informação aponta para o fato de que é necessário planejamento por parte de *designers* e desenvolvedores para não haver uma grande sobreposição de mecanismos.

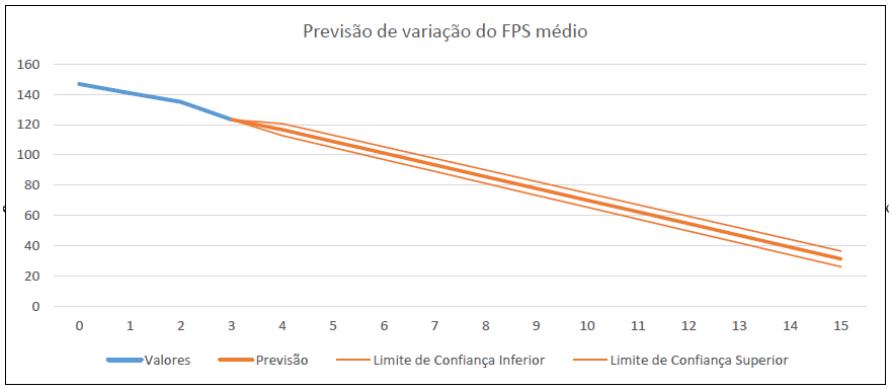

Figura 4. Projeção estatística para a variação do FPS.

Em resumo, os resultados são considerados satisfatórios. Foram registrados impactos no desempenho (dentro do que é tido por aceitável) relativos ao uso dos mecanismos de DDA construídos a partir do DDA-MAPEKit, principalmente nas métricas de FPS e *frame time* médios. Dessa maneira, estes impactos não representam impedimentos para o uso do *framework*, ainda que, dada uma determinada quantidade de mecanismos (que certamente deve variar a depender de cada jogo e de sua complexidade), as métricas analisadas possam assumir valores fora do que é considerado ideal. Para que se possa desfrutar das vantagens do DDA sem prejudicar o desempenho, é necessário considerar estes possíveis impactos e conceber os mecanismos em quantidade tal[5] que efeitos negativos não sejam sentidos pelo jogador.

7. Considerações Finais

Este estudo apresentou um experimento para investigar possíveis impactos no desempenho de jogos digitais ao serem inseridos mecanismos de ajuste dinâmico de dificuldade criados com o DDA-MAPEKit, um *framework* desenvolvido para este fim. Ao serem realizadas execuções simuladas de um pequeno jogo elaborado para este experimento, foram monitorados os valores de FPS (*frames per second*) e *frame time*, além dos consumos de CPU e memória RAM. Os resultados evidenciaram que, de fato, notam-se impactos no desempenho, principalmente quedas no FPS e aumento no *frame time*. Entretanto, todos os valores se mantiveram nas faixas consideradas aceitáveis.

Diversos horizontes despontam como continuidade desta pesquisa. O aprofundamento das análises e a inclusão de novas métricas para avaliar o desempenho dos jogos pode ampliar a compreensão acerca dos impactos discutidos neste trabalho. Ademais, experimentos com voluntários podem acrescentar informações acerca da percepção do jogador em relação aos efeitos no desempenho. Por fim, acredita-se que há muito a ser discutido acerca da relação entre o desempenho do jogo e a atuação dos mecanismos de DDA, com o intuito de desenvolver diretrizes e melhores práticas para a implementação eficaz desses mecanismos, garantindo uma experiência de jogo equilibrada e satisfatória para os jogadores, enquanto se mantém um desempenho técnico aceitável.

[5]Note-se que, nas projeções realizadas, nada foi dito a respeito de uma possível quantidade máxima de *loops* que podem ser construídos para um jogo, mas sim em relação a uma possível quantidade máxima de *loops* sendo executados ao mesmo tempo.

Referências

Berseth, G., Haworth, M. B., Kapadia, M., and Faloutsos, P. (2018). Characterizing and optimizing game level difficulty. In *Proceedings of the 7th International Conference on Motion in Games*, MIG '14, page 153–160, New York, NY, USA. ACM.

Claypool, M. and Claypool, K. (2009). Perspectives, frame rates and resolutions: It's all in the game. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, page 42–49, New York, NY, USA. ACM.

Claypool, M., Claypool, K., and Damaa, F. (2006). The effects of frame rate and resolution on users playing first person shooter games. In Chandra, S. and Griwodz, C., editors, *Multimedia Computing and Networking 2006*, volume 6071, page 607101. International Society for Optics and Photonics, SPIE.

de Lima, E. S., Silva, B. M., and Galam, G. T. (2022). Adaptive virtual reality horror games based on machine learning and player modeling. *Entertainment Computing*, 43:100515.

Dunlop, R. (2003). Fps versus frame time. Disponível em: https://www.mvps.org/directx/articles/fps_versus_frame_time.htm. Acesso em: 26 de set. de 2023.

Hunicke, R. (2005). The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACE '05, page 429–433, New York, NY, USA. ACM.

Li, Y.-C., Hsu, C.-H., Lin, Y.-C., and Hsu, C.-H. (2020). Performance measurements on a cloud vr gaming platform. In *Proceedings of QoEVMA'20*, QoEVMA'20, page 37–45, New York, NY, USA. ACM.

Mi, Q. and Gao, T. (2022). Improved belgian AI algorithm for dynamic management in action role-playing games. *Applied Sciences*, 12(22):11860.

Sepulveda, G. K., Besoain, F., and Barriga, N. A. (2020). Exploring dynamic difficulty adjustment in videogames. In *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*. IEEE.

Seyderhelm, A. J. A. and Blackmore, K. (2021). Systematic review of dynamic difficulty adaption for serious games: The importance of diverse approaches. *SSRN E-Journal*.

Souza, C., Oliveira, S., Berretta, L., and Carvalho, S. (2023a). The use of health data for dynamic difficulty adjustment in serious games (in portuguese). In *Proceedings of SBCAS 2023*, pages 479–484, Porto Alegre, RS, Brasil. SBC.

Souza, C. H. R., Oliveira, S. S. D., Berretta, L. O., and de Carvalho, S. T. (2023b). Ddamapekit: A framework for dynamic difficulty adjustment based on mape-k loop. In *Proceedings of SBGames 2023*, pages 01—-10, New York, NY, USA. ACM.

Streicher, A. and Smeddinck, J. D. (2016). Personalized and adaptive serious games. In *Entertainment Computing and Serious Games*, pages 332–377. Springer International Publishing.

Tagliaro, L. R. G. (2022). An implementation of adaptive difficulty systems for challenging video games. Undergraduate Thesis.

Vahlo, J., Kaakinen, J. K., Holm, S. K., and Koponen, A. (2017). Digital game dynamics preferences and player types. *JCMC*, 22(2):88–103.

Wang, J., Shi, R., Zheng, W., Xie, W., Kao, D., and Liang, H.-N. (2023). Effect of frame rate on user experience, performance, and simulator sickness in virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2478–2488.

Weyns, D. (2021). *An Introduction to Self-Adaptive Systems*. John Wiley & Sons Ltd.

Zohaib, M. (2018). Dynamic difficulty adjustment (DDA) in computer games: A review. *Advances in Human-Computer Interaction*, 2018:1–12.