UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

EDUARDO FOLLE MIOTTO

AVALIAÇÃO DE TÉCNICAS DE PLANEJAMENTO PARA NPCS: GOAP E UD GOAP, EM UM JOGO DE SIMULAÇÃO DE LOJA

CHAPECÓ
2025

EDUARDO FOLLE MIOTTO

AVALIAÇÃO DE TÉCNICAS DE PLANEJAMENTO PARA NPCS: GOAP E UD GOAP, EM UM JOGO DE SIMULAÇÃO DE LOJA

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal da Fronteira Sul (UFFS), como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Felipe Grando

CHAPECÓ
2025

Miotto, Eduardo Folle

AVALIAÇÃO DE TÉCNICAS DE PLANEJAMENTO PARA NPCS: GOAP E UD GOAP, EM UM JOGO DE SIMULAÇÃO DE LOJA / Eduardo Folle Miotto – 2025.

55 f.

Orientador: Felipe Grando

Trabalho de Conclusão de Curso (Graduacão) – Universidade Federal da Fronteira Sul, Curso de Ciência da Computação, Chapecó, SC, 2025.

1. Inteligência Artificial 2. Comportamento 3. Tomada de Decisão 4. NPCs 5. jogos eletrônicos 6. GOAP 7. UD GOAP I. Grando, Felipe, orient. II. Universidade Federal da Fronteira Sul. III. Título.

EDUARDO FOLLE MIOTTO

AVALIAÇÃO DE TÉCNICAS DE PLANEJAMENTO PARA NPCS: GOAP E UD GOAP, EM UM JOGO DE SIMULAÇÃO DE LOJA

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal da Fronteira Sul (UFFS), como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Felipe Grando

Este Trabalho de Conclusão de Curso foi avaliado e aprovado pela banca avaliadora em: 11/12/2025.

BANCA AVALIADORA

Felipe Grando - UFFS

Denio Duarte - UFFS

Geomar André schreiner - UFFS

DEDICATÓRIA

Dedico este trabalho a meus amigos, colegas, familiares, pais e professores que me ajudaram em minha jornada.

“ 人の夢は終わらなエ ”
(Marshall D. Teach, One Piece — Eiichiro Oda)

RESUMO

O uso de arquiteturas de planejamento para a inteligência artificial (IA) de personagens não jogáveis (NPCs) tem crescido no desenvolvimento de jogos eletrônicos, especialmente em contextos que requerem comportamento adaptativo e modular. Entre os modelos mais adotados, destaca-se o *Goal-Oriented Action Planning* (GOAP), que permite a tomada de decisões dinâmica a partir da formulação de uma sequência de ações baseada em objetivos. Entretanto, devido a restrições do design do GOAP, variações como o *Utility-Directed GOAP* (UD GOAP) foram propostas na literatura. Este trabalho desenvolveu um ambiente de jogo utilizado como uma base para testes que comparam GOAP e UD GOAP, avaliando desempenho computacional e a qualidade do comportamento produzida por cada abordagem. Foram implementados cenários controlados contendo múltiplos NPCs, sobre os quais foram coletadas métricas de tempo de execução, quadros por segundo e padrões comportamentais. Os resultados indicaram diferenças e similaridades relevantes entre as arquiteturas, evidenciando variações quanto a custo computacional e ao impacto no comportamento dos agentes. A análise obtida contribui para a compreensão das vantagens e limitações de cada técnica, oferecendo suporte tanto para estudos acadêmicos na área de IA aplicada a jogos quanto para aplicações práticas no desenvolvimento de sistemas interativos.

Palavras-Chave: Inteligência Artificial, Comportamento, Tomada de Decisão, NPCs, jogos eletrônicos, GOAP, UD GOAP.

ABSTRACT

The use of planning architectures for the artificial intelligence (AI) of non-player characters (NPCs) has increased in digital game development, especially in scenarios that require adaptive and modular behavior. Among the most adopted models, Goal-Oriented Action Planning (GOAP) stands out, allowing dynamic decision-making through the formulation of an objective-based sequence of actions. However, due to design limitations of GOAP, variations such as Utility-Directed GOAP (UD GOAP) have been proposed in the literature. This work developed a game environment used as a testing platform to compare GOAP and UD GOAP, assessing both computational performance and the quality of the behavior produced by each approach. Controlled scenarios containing multiple NPCs were implemented, from which execution time, frames per second, and behavioral patterns were collected. The results revealed relevant differences and similarities between the architectures, highlighting variations in computational cost and the impact on agent behavior. The analysis contributes to understanding the advantages and limitations of each technique, supporting both academic research in game-oriented AI and practical applications in the development of interactive systems.

Keywords: Artificial inteligence, Behavior, Decision-Making, NPCs, Games, GOAP, UD GOAP.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CPU – *Central Processing Unit* (Unidade Central de Processamento)

FPS – *Frames per Second* (quadros por segundo)

FSM – *Finite State Machine* (Máquina de estados finita)

GOAP – *Goal-Oriented Action Planning* (Planejamento de ações orientado a objetivo)

IA – Inteligência Artificial

MB – Megabyte

NPC – *Non-Playable Character* (Personagem não jogável)

PC – *Personal Computer* (Computador Pessoal)

RPG – *Role Playing Game* (Jogo de interpretação de papéis ou Jogo narrativo)

TV – Televisão

UD GOAP – *Utility-Directed Goal-Oriented Action Planning* (Planejamento de ações orientado a objetivo dirigido por utilidade)

SUMÁRIO

1. INTRODUÇÃO

A área de jogos eletrônicos utiliza inteligência artificial (IA) no desenvolvimento de *Non-Playable Characters (NPCs)*, ou personagens não controlados por um jogador, conceito cunhado em jogos de mesa tradicionais, os jogos de interpretação de personagem (Role Playing Game, RPG) (RIBEIRO, 2022). Quando se refere à inteligência artificial em jogos, existe uma diferença de foco entre simulação e racionalidade (BAILEY; KATCHABAW, 2008). O objetivo do uso de IA em NPCs, de acordo com Dill (2013), é passar a impressão de que há vida naquilo visto ou, como uma ferramenta que auxilia na criação da experiência proposta.

Para isso, foram criados diversos métodos ao longo dos anos, visando atender a uma demanda específica de cada jogo. Os métodos, segundo Dill (2013), variam em um espectro entre autoral e autônomo. Controle autoral implica que o comportamento de um NPC é definido pelo desenvolvedor, quanto mais autoral mais predefinido o modo como ele age; um personagem que somente anda de um lado ao outro da tela sem se importar com contexto ao seu redor é uma IA roteirizada, como visto na esquerda da Figura 1.1, ou seja, o comportamento dos NPCs será exato e unicamente o que o desenvolvedor determinou que eles façam, permitindo uma experiência mais detalhada, mas mais previsível. À direita da Figura 1.1 se encontram as IAs mais autônomas, que são projetadas para criar os próprios comportamentos a partir de determinados limites.

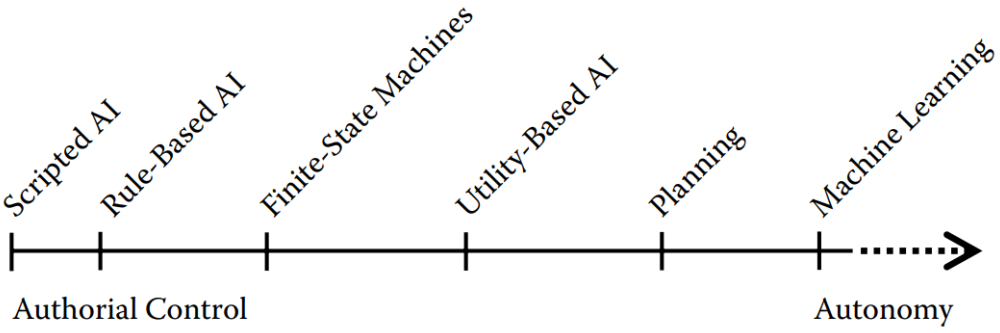

Figura 1.1: Comparação entre controle autoral e autonomia para as arquiteturas de IA mais aplicadas em jogos(DILL, 2013)

O *Planning* representado mais à direita da Figura 1.1 refere-se às IAs que planejam seu comportamento, o modo como agirá durante a execução, garantindo maior autonomia. Neste contexto, foi criado o *Goal-Oriented Action Planning (GOAP)* por Orkin (2003), uma arquitetura planejadora para produzir comportamentos menos repetitivos e previsíveis do que os vistos até então em muitos jogos. Formulando a

sequência de ações em tempo real, o desenvolvedor não precisa pensar em todos os cenários em que o NPC pode se encontrar e o que deve fazer neles, basta indicar para o sistema quais serão as ações disponíveis para o agente (NPC) e ele escolherá como agir (ORKIN, 2003).

Com o passar do tempo, surgiram variações do método GOAP, como o *Utility-Directed Goal-Oriented Action Planning (UD GOAP)*, que adiciona novos conceitos para aumentar a complexidade e o realismo dos planos gerados. Mas as vantagens e desvantagens desse método podem variar considerando sua aplicação em diferentes jogos com características específicas, considerando que o que funciona em um cenário não implica seu funcionamento em outro.

A definição de comportamentos para NPCs é essencial no desenvolvimento de jogos, e tais comportamentos estão sujeitos à arquitetura escolhida para desenvolvê-la. De acordo com Sloan, Namee e Kelleher (2011), o UD GOAP tem um resultado mais satisfatório, sem apresentar desempenho significativamente pior em sua execução. Apesar disso, ainda faz-se pertinente um novo estudo que avalie essas arquiteturas em um cenário distinto, visto que as particularidades do ambiente podem impactar diretamente os resultados.

Assim, durante a execução deste trabalho, teve como objetivo o desenvolvimento de um jogo eletrônico como base para a implementação do GOAP e UD GOAP, com alterações pontuais em diferentes partes da arquitetura, tais como: método de busca, escolha de objetivo e heurísticas de decisão para ações; implementá-los, analisar e comparar via resultados de testes. Buscando responder qual IA, GOAP ou UD GOAP, apresenta melhor desempenho computacional e tomada de decisões no cenário do jogo proposto: uma simulação de um mercado, onde os agentes têm o papel de clientes, que navegam simultaneamente pela loja em busca de produtos para comprar.

Neste contexto, foi considerado:

1. Como criar e o que torna um cenário de simulação adequado e que difere dos já analisados, para testar as arquiteturas de IA voltadas para o planejamento de ações de NPCs?

2. Como testar, avaliar e comparar o desempenho dos modelos de comportamento e planejamento gerados pelas variações do GOAP?

3. Qual é o desempenho ou custo computacional das arquiteturas utilizadas?

4. Como os resultados obtidos se comparam com os de estudos anteriores?

O uso de métodos de planejamento em tempo real para definir o comportamento de NPCs facilita o desenvolvimento de projetos de médio e grande porte devido

à sua modularidade, possibilitando maior flexibilidade e escalabilidade na implementação da IA, permitindo melhor uso dos recursos limitados (LONG, 2007).

Entretanto, para que essas arquiteturas sejam viáveis em ambientes interativos e com exigência de execução em tempo real, é necessário que apresentem desempenho compatível, garantindo a manutenção de quadros por segundo (FPS) satisfatórios, além de comportamentos coerentes com o design pretendido pelos desenvolvedores. Ou seja, não basta que a IA tome decisões funcionais, ela precisa fazê-lo de forma rápida, eficiente e alinhada com as expectativas de jogabilidade e imersão. De acordo com Huang et al. (2012), a IA para NPCs deve progredir com a melhora da parte gráfica dos jogos, cada vez mais realista, pois a dissonância entre os dois pode causar estranhamento ao jogador, quebrando a suspensão da descrença quando um personagem age de maneira não natural.

A técnica ou metodologia GOAP é uma das mais estabelecidas no campo em questão de IAs planejadoras utilizadas em jogos eletrônicos. Desde sua introdução no jogo F.E.A.R.[1], foi implementada em diversos títulos comercialmente relevantes, como Tomb Raider[2], Deus Ex: Human Revolution - Director's Cut[3] (JACOPIN, 2019), Middle Earth: Shadow of Mordor[4], Immortals Fenyx Rising[5] e a série Assassin's Creed desde Assassin's Creed Odyssey[6] (PRÉVOST et al., 2023).

Neste cenário, surgem variações do GOAP que buscam otimizar suas limitações, como o UD GOAP (Utility-Directed GOAP), que incorpora lógica baseada em sistemas de utilidade para priorização dinâmica de ações (SLOAN; NAMEE; KELLEHER, 2011). Segundo Sloan, Namee e Kelleher (2011), o UD GOAP apresenta resultados mais satisfatórios em termos de flexibilidade e comportamento emergente (que surgem da interação entre componentes do sistema, sem terem sido explicitamente programados ou antecipados), sem apresentar perdas significativas de desempenho. No entanto, os testes relatados por esses autores foram conduzidos em um contexto específico de jogo, o que limita a generalização dos resultados para outros ambientes com características distintas. Em destaque, a utilização de somente um NPC no ambiente de testes.

Dessa forma, faz-se pertinente o estudo comparativo entre o GOAP tradicional e sua variação UD GOAP em um ambiente de jogo diferente dos já utilizados em pesquisas anteriores, para verificar se os resultados obtidos pela literatura se mantêm em contextos diversos.

[1] https://store.steampowered.com/app/21090/FEAR/
[2] https://store.steampowered.com/app/203160/Tomb_Raider/
[3] https://store.steampowered.com/app/238010/Deus_Ex_Human_Revolution__Directors_Cut/
[4] https://store.steampowered.com/app/241930/Middleearth_Shadow_of_Mordor/
[5] https://store.steampowered.com/app/2221920/Immortals_Fenyx_Rising/
[6] https://store.steampowered.com/app/812140/Assassins_Creed_Odyssey/

Esta pesquisa se justifica, então, por contribuir academicamente, expandindo o conhecimento sobre técnicas de planejamento aplicadas à IA de jogos; e por contribuição no campo prático, ao fornecer uma análise fundamentada para desenvolvedores de jogos, auxiliando na escolha adequada da arquitetura de comportamento para NPCs.

Os resultados encontrados indicam o UD GOAP como uma evolução viável do método tradicional para cenários que requerem comportamentos mais realistas, assim como indicado por Sloan, Namee e Kelleher (2011). Concluindo que diferentes cenários possuem valor para uma análise dos métodos.

O trabalho está organizado da seguinte forma: o próximo capítulo apresenta o referencial teórico, no qual são abordados os jogos eletrônicos e seus gêneros, com ênfase no gênero de simulação, bem como as técnicas GOAP e UD GOAP, o algoritmo A* e os trabalhos relacionados. No capítulo 3 é descrita a metodologia adotada, detalhando os métodos e métricas utilizados, além da implementação do jogo base e dos modelos propostos, incluindo as especificações referentes à coleta de dados. O Capítulo 4 apresenta e discute os resultados obtidos. Por fim, o último capítulo traz as conclusões do trabalho e trabalhos futuros.

2. REFERENCIAL TEÓRICO

Este capítulo busca abordar as referências necessárias para o entendimento da pesquisa. Iniciando com uma breve explicação sobre jogos eletrônicos, o tamanho da indústria contemporânea, seus gêneros como o de simulação e o que o define. Após, será introduzido as técnicas implementadas para a pesquisa: GOAP, A*, o algoritmo utilizado como base estrutural para ambos os modelos e sua variação UD GOAP, finalizando com trabalhos relacionados.

2.1 Jogos Eletrônicos

Jogos eletrônicos surgiram junto à computação digital no meio do século XX e, atualmente, em questão de receita, é a maior indústria de mídia e entretenimento do mundo (GELONEZE; ARIELO, 2017). Em 2024 teve uma receita total de $187,7 bilhões de dólares (BUIJSMAN, 2024), maior do que os $30 bilhões da indústria cinematográfica (MITCHELL, 2025) e os $29,6 bilhões da indústria da música (International Federation of the Phonographic Industry (IFPI), 2025) somados.

Devido ao tamanho da indústria de jogos na atualidade, é natural o surgimento de grandes estúdios que, pela natureza dos jogos eletrônicos, têm equipes multidisciplinares de profissionais responsáveis pelo desenvolvimento de todas as etapas da criação do jogo, como a parte artística visual, musical, design e a parte computacional que, por sua vez, também possui caráter multidisciplinar na aplicação em jogos, como computação gráfica, algoritmos e estrutura de dados, engenharia de software e inteligência artificial (GELONEZE; ARIELO, 2017).

O restante do capítulo abordará gêneros de jogos, com destaque para jogos de simulação, explicará as técnicas de IA GOAP, UD GOAP, as métricas que serão utilizadas nos testes e, por fim, trabalhos relacionados.

2.1.1 Gêneros de Jogos Eletrônicos

Jogos eletrônicos possuem e se dividem em múltiplos gêneros. De acordo com Buijsman (2024), os gêneros mais populares em Computadores Pessoais e consoles são jogos de tiro, aventura, RPGs, *battle royales* e se diferenciando, como visto nas figuras 2.1 e 2.2, com jogos de estratégia entrando no *top 5* no PC, enquanto no

console jogos de esportes têm uma representatividade maior. Conforme visto nas figuras 2.1 e 2.2, que mostram gráficos de rosca, com a receita de jogos eletrônicos em 2024 agrupados por gênero, respectivamente em computadores pessoais com renda total de 43,2 bilhões de dólares americanos e consoles com 51,9 bilhões.

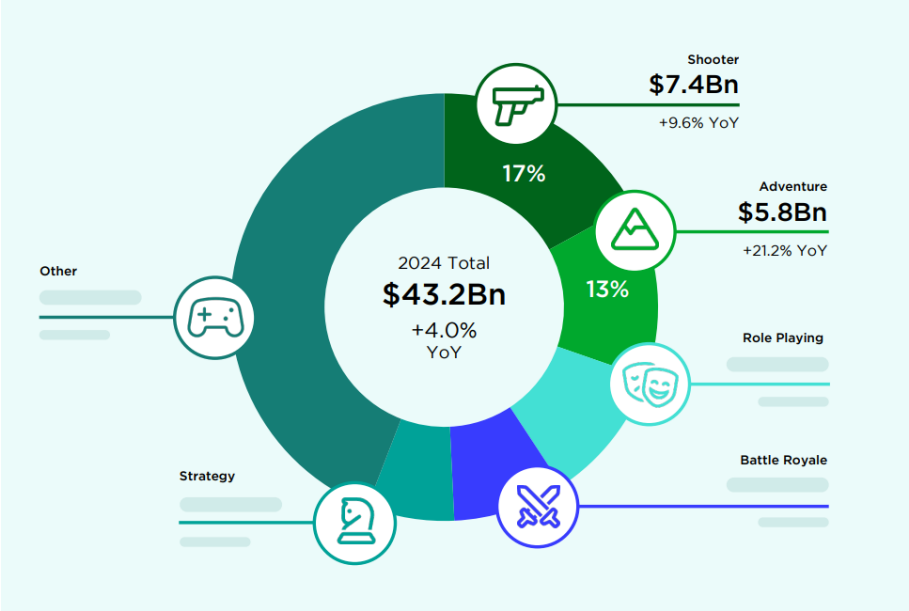

Figura 2.1: Gêneros mais populares no PC por receita (BUIJSMAN, 2024).

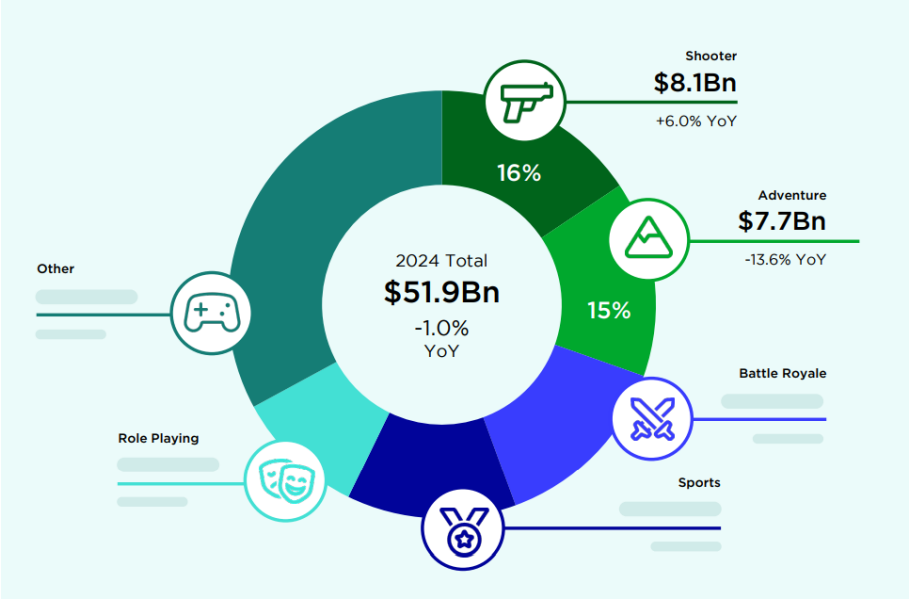

Figura 2.2: Gêneros mais populares em consoles por receita (BUIJSMAN, 2024).

2.1.2 Jogos de Simulação

Dentro das diversas categorias de jogos eletrônicos, temos os jogos de simulação, que têm como intenção replicar, com algum grau de fidelidade, certas atividades ou fenômenos da vida real (DESHPANDE; HUANG, 2011). Um subgênero de jogos de simulação são simulações de trabalhos de colarinho azul, onde o jogador toma o papel de um trabalhador em exercício da profissão, como PowerWash Simulator (FUTURLAB, 2022), no qual o jogador trabalha com um lavador a jato, lavando diferentes locais, veículos e objetos, representado na Figura 2.3, onde o jogador lava um automóvel.

Figura 2.3: PowerWash Simulator, jogador lavando um automóvel (FUTURLAB, 2022).

Vários jogos de simulação tiveram sucesso comercial, como Cities: Skylines (LTD., 2015) simulando o gerenciamento de uma cidade e Euro Truck Simulator 2 (SOFTWARE, 2012) simulando condução de caminhões em estradas, que, na plataforma de vendas de jogos para PC, Steam[1], estão respectivamente com mais de 200 mil e mais de 600 mil análises de usuários, números expressivos que convertem para mais de 12 milhões de cópias vendidas em ambos os casos, de acordo com os desenvolvedores (SKYLINES, 2022; SOFTWARE, 2022).

Mais especificamente para o escopo desta pesquisa, temos simulações de lojas, vistos nas Figuras 2.4, 2.5 e 2.6 que respectivamente mostram o jogador organizando a loja, comprando estoque e atende o caixa. Além de outras atividades relacionadas como a venda de produtos, execução de serviços e o crescimento do próprio empreendimento. Nos últimos anos se teve um grande crescimento de lança-

[1] https://store.steampowered.com/

mentos, como exemplos: Gas Station Simulator (2021), Supermarket Simulator (2024) (Figura 2.6), TCG Card Shop Simulator (2024) (figura 2.4 e 2.5), Supercar Collection Simulator[2] (2025) e Liquor Store Simulator[3] (2025).

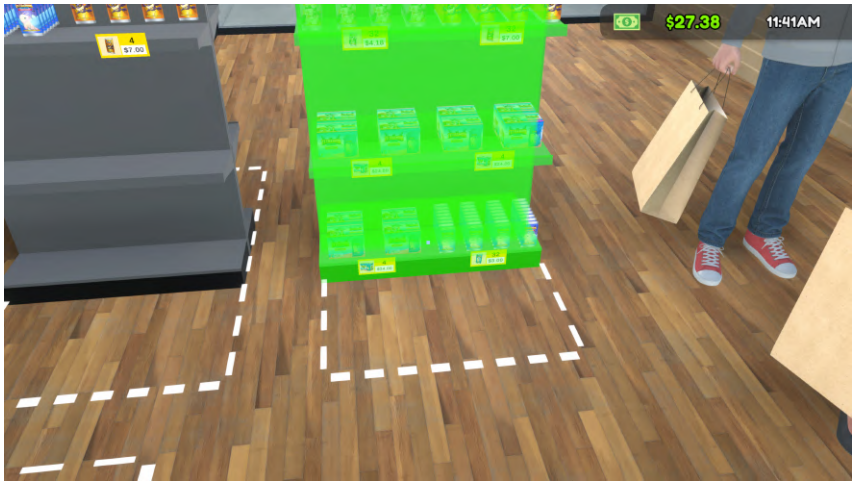

Figura 2.4: TCG Card Shop Simulator, construindo a loja (GAMES, 2024b).

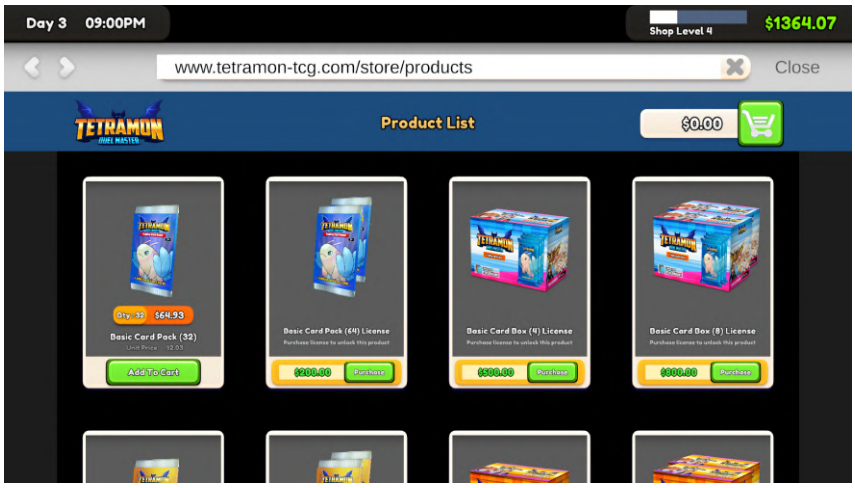

Figura 2.5: TCG Card Shop Simulator, comprando estoque (GAMES, 2024b).

[2]https://store.steampowered.com/app/3453600/Supercar_Collection_Simulator/
[3]https://store.steampowered.com/app/3124550/Liquor_Store_Simulator/

Figura 2.6: Supermarket Simulator, atendendo o caixa (GAMES, 2024a).

A maioria dos jogos ainda usa IAs com comportamento predefinido, como máquinas de estados finitas ou *Finite State Machines* (FSM) e árvores de comportamento. No entanto, o uso de técnicas como GOAP pode ser atrativo, tanto para o desenvolvimento quanto para o produto final (DU, 2025).

2.2 Técnicas

As técnicas de planejamento utilizadas na implementação desse trabalho são GOAP e UD GOAP, que têm caráter planejador, visando construir comportamentos em tempo real, para isso é utilizado o algoritmo A*. Estes conceitos serão abordados nesta seção.

2.2.1 GOAP

O *Goal-Oriented Action Planning* (GOAP), enquanto método original, proposto por Orkin (2003) tem como objetivo delegar o planejamento das sequências de ações que o agente deve tomar para o próprio agente, munindo-o com a tomada de decisões em tempo real para formar uma sequência de ações válidas que alcance um dado objetivo.

Ainda é necessário para o funcionamento do GOAP uma máquina de estados finita, mas a estrutura de transições e estados é separada modularmente, não precisando ser descrita explicitamente pelo desenvolvedor (ORKIN, 2003). Para isso é definido o conceito de objetivo, ação, plano e formulação do plano, além de utilizar uma representação do mundo. A Figura 2.7 mostra ao topo a representação do

mundo via um estado *State* definido como um conjunto de três variáveis booleanas: *isWeaponLoaded* indicando se a arma do NPC está carregada ou não, *inMeleeRange* se o agente está próximo do seu alvo e *isTargetDead* sinalizando se o alvo está vivo ou morto. Abaixo de *State* é declarado o estado atual do mundo, com as variáveis binárias com valores definidos. Isto consiste na máquina de estados finita implícita do GOAP, com três variáveis binárias se compõem oito estados diferentes possíveis.

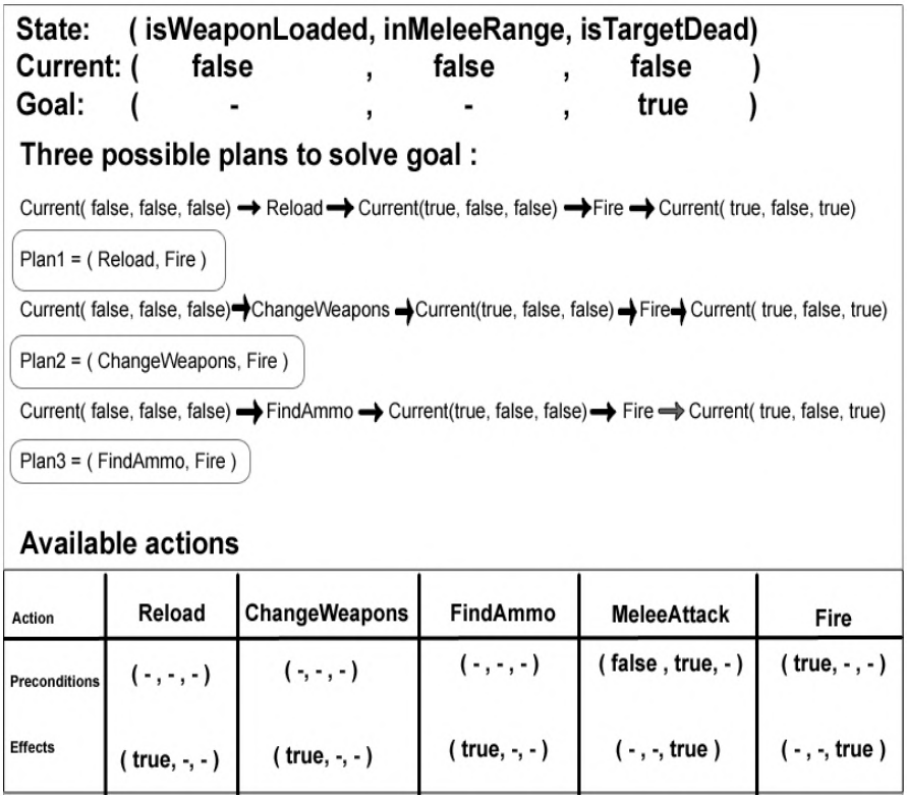

Figura 2.7: Exemplo de um planejamento, objetivo, ações e estados (LONG, 2007).

Objetivo

Um objetivo é um conjunto de condições que o agente quer satisfazer. Representadas por variáveis, que, ao atingirem o valor desejado, dão o objetivo como completo. Pode-se ter mais de um objetivo, mas para fazer o planejamento, escolhe-se um deles como o atual a ser considerado (ORKIN, 2003).

Na Figura 2.7 temos um objetivo representado por um estado do mundo a ser alcançado (*Goal*), ou seja, determinados valores para cada variável. Neste caso, somente uma variável importa para satisfazer o objetivo, então ele irá procurar ações que conseguem cumprir essa condição.

Ação

Uma ação representa um único passo indivisível do agente dentro do plano e que atua como transição de um estado para outro. A ação tem condições para ser executada assim como o objetivo tem o estado que quer chegar, e efeitos na forma do que é alterado na troca de estados caso a ação venha a ser posta a prática (ORKIN, 2003).

Na Figura 2.7, a parte inferior apresenta uma lista exemplo de ações para o agente executar, como a quarta ação *MeleeAttack* ou ataque corpo a corpo, que tem como condições para ser executada estar próximo ao alvo e não ter munição em sua arma, tendo como efeito a morte do alvo.

Plano

O plano é a sequência de ações escolhidas para ir do estado atual até o que satisfaz determinado objetivo (ORKIN, 2003). Nos planos exemplos da Figura 2.7, pode-se ver três caminhos diferentes válidos para atingir o objetivo. A figura 2.8 apresenta outro exemplo de um plano formulado, em que o estado inicial está ligado até o final via ações que mudam os parâmetros do mundo.

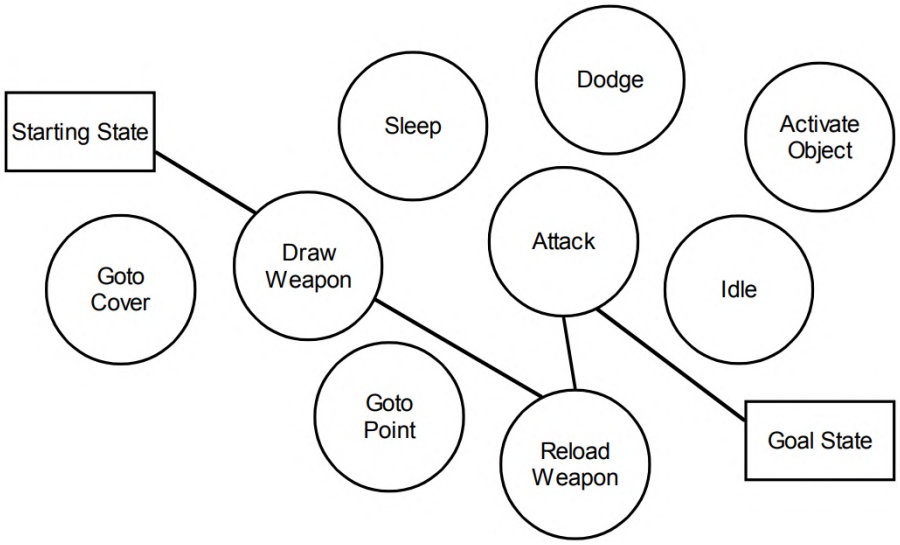

Figura 2.8: Exemplo de um plano (ORKIN, 2003).

O agente gera o plano em tempo real sob o objetivo atual, essa geração consiste em uma busca pelas ações disponíveis ao agente, começando pelo estado de aceitação em direção ao estado atual, para ver quais ações têm como efeito a realização de uma parte ou o todo do objetivo. Ao selecionar uma ação, verifica-se os pré-requisitos da mesma, e caso eles não estejam satisfeitos, continua a busca por

outra ação que os satisfaça. Quando não houver mais condições não satisfeitas para executar a última ação escolhida, significa que foi alcançado o estado atual, podendo o agente selecionar o plano e começar a execução. Depois de um plano ser escolhido e estiver em execução, ainda pode ser interrompido caso o objetivo troque ou o plano se torne inválido, começando uma nova busca (ORKIN, 2003).

Pode existir mais de um plano válido que levará o agente a completar seu objetivo, então, para guiar o algoritmo de busca, pode-se associar custo às ações, para a busca achar o plano com um menor custo (ORKIN, 2003).

Benefícios do GOAP

Quando o plano é gerado em tempo real, ele é feito considerando as condições atuais em que o agente se encontra, garantindo que o plano criado sempre seja válido, o que pode não ser o caso em um comportamento predeterminado e que depende do desenvolvedor antecipar todos os cenários que o NPC pode se encontrar (ORKIN, 2003).

Também é de benefício para o desenvolvimento a separação conceitual entre os estados e as ações, pois uma FSM escala exponencialmente quanto mais estados se adicionam, enquanto o GOAP só é necessário adicionar na lista a nova ação, seus pré-requisitos e efeitos (BRITTON, 2023).

Por fim, de acordo com Orkin (2003), o GOAP demonstra-se vantajoso na criação de NPCs variados, podendo selecionar diferentes subconjuntos de ações para cada um, permitindo que o agente se comporte da forma pretendida sem precisar construir um novo comportamento do zero e, por meio de pré-requisitos específicos, faz com que o agente use diferentes ações que alcançam o mesmo resultado dependendo do contexto.

Considerações da Implementação

Para o GOAP funcionar adequadamente, precisa-se considerar duas questões: o método de busca e a maneira de representar o mundo.

O método de busca utilizado será executado em tempo real, logo ele precisa ser rápido, para isso é utilizado o algoritmo A*, comumente utilizado em jogos eletrônicos para navegação de personagens no espaço. O algoritmo necessita do cálculo do custo de um nodo e de sua distância heurística até o objetivo. Os nodos representam estados do mundo e as bordas, as ações que movem entre os estados. O cálculo do custo é feito pela soma dos custos das ações necessárias para chegar até o estado

representado pelo nó, com a distância heurística sendo calculada como a soma das propriedades a serem satisfeitas no estado alvo (ORKIN, 2003).

Para representar o mundo de uma maneira que permite reconhecer quando chega-se ao estado alvo e aplicar pré-requisitos e efeitos das ações, pode-se usar uma lista de propriedades sobre o mundo, com atributos chaves, valores e identificadores dos sujeitos. Precisa-se representar somente o mínimo de propriedades relevantes ao objetivo atual, que crescem de acordo com as ações adicionadas para satisfazer tal objetivo, pois os pré-requisitos delas serão considerados para o objetivo também (ORKIN, 2003).

Além dos pré-requisitos normais, é definido o conceito de pré-condições contextuais, que precisam ser garantidas para executar uma ação, mas o planejador nunca tentará satisfazê-las (ORKIN, 2003).

2.2.2 Algoritmo A*

Para A busca regressiva do planejador de ações do GOAP, é recomendado por Orkin (2003) a busca em grafo A*, um algoritmo originalmente proposto por Hart, Nilsson e Raphael (1968), que se utiliza de uma heurística para achar a melhor solução final em tempo otimizado (JUNIOR, 2019).

A cada iteração o algoritmo seleciona o primeiro melhor caminho, que é definido como o menor $f(x)$ composto por:

$$f(x) = g(x) + b(x)$$

$g(x)$ é o custo do caminho percorrido até o nodo analisado, enquanto $b(x)$ é a heurística que pretende determinar o quão longe nodo analisado está do nodo objetivo, então a busca volta para nodos anteriores quando a heurística não melhorar o suficiente (JUNIOR, 2019).

O algoritmo A* permite identificar o caminho de menor custo sem a necessidade de explorar exaustivamente todos os nós do espaço de busca, amplamente aplicado em contextos de jogos eletrônicos para resolução de problemas de navegação em cenários virtuais. Um exemplo simples de heurística empregada nesse processo é a distância euclidiana entre o agente e o objetivo, assumindo deslocamento direto. Entretanto, quando o percurso ideal é bloqueado por obstáculos, o algoritmo ajusta progressivamente o custo acumulado ao considerar alternativas viáveis no entorno do

objeto que impede o avanço, recalculando a trajetória até identificar um caminho mais eficiente que contorne o obstáculo.

2.2.3 UD GOAP

O *Utility-Directed Goal-Oriented Action Planning* (UD GOAP) se baseia no GOAP, alterando a heurística de busca para se fundamentar em uma função de utilidade.

Na implementação do UD GOAP, a utilidade é a medida de quão desejável um estado é para o agente. Essa desejabilidade é representada por uma função utilidade que atribui números aos estados, tornando possível avaliar quão longe um estado está do estado preferido (valor máximo 1), e como ações irão impactar a utilidade. O agente selecionará suas ações então buscando maximizar essa utilidade, o que é vantajoso em cenários com objetivos conflitantes (como dirigir rápido e se manter seguro) e quando tem incerteza e pesos diferentes para os objetivos, com um valendo mais que o outro (SLOAN; NAMEE; KELLEHER, 2011).

Para saber como uma ação irá afetar a utilidade, são usados objetos inteligentes, definidos como objetos que guardam mais informações do que suas propriedades inerentes (como posição), por exemplo, informações úteis ao planejador que, caso a IA não estivesse presente no jogo, não precisariam existir (SLOAN; NAMEE; KELLEHER, 2011).

Então, o UD GOAP, durante o processo de planejamento, busca objetos inteligentes em que ações podem ser executadas. Essas ações têm notas associadas, ao contrário do GOAP, que tem custos, buscando não o plano com menor custo, mas o com maior nota. A nota é dada pela diferença entre a utilidade do próximo estado e o anterior, com a utilidade aumentando entre eles dando um valor positivo. Para o cálculo da utilidade é usado o conceito de motivações, motivação define o que o agente quer, representada por um ou mais parâmetros, que ele quer maximizar ou minimizar, tendo uma função própria de utilidade, então a utilidade de um estado é composto pela soma das utilidades das motivações do NPC multiplicadas pelos respectivos pesos. (SLOAN; NAMEE; KELLEHER, 2011).

A motivação também abstrai o conceito de objetivos do GOAP original, gerando ao invés de estados com variáveis que devem se igualar, uma tupla de objeto afetado e variável a considerar.

Na Figura 2.9 é representado um exemplo: o estado inicial com um valor de utilidade de 0,49, calculado pela média entre as utilidades de cada uma das duas mo-

tivações: se entreter e se energizar. Escolhendo se entreter para fazer a busca, gera o objetivo de aumentar o parâmetro entretenimento do objeto si próprio. Temos duas ações que aumentam o valor da variável entretenimento: pular e rir, onde pular diminui o valor da variável energia, o que causa um valor de utilidade do estado resultante menor, mesmo aumentando o dobro em questão de entretenimento, essa utilidade menor dá uma nota menor para a ação, fazendo com que a outra seja escolhida.

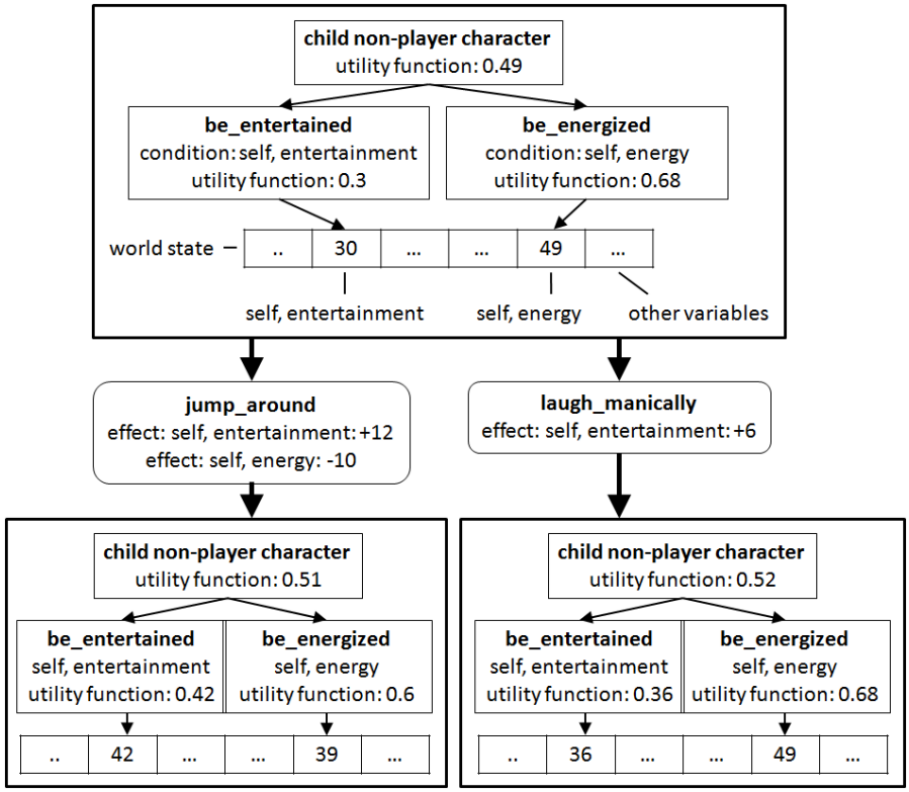

Figura 2.9: Exemplo de como a utilidade dos estados muda a partir de uma ação (SLOAN; NAMEE; KELLEHER, 2011).

2.3 Trabalhos Relacionados

Foram selecionados três trabalhos com contexto semelhante, usando como critérios na pesquisa a escolha de trabalhos que comparam métodos de IAs que definem comportamento e tomada de decisão para NPCs em jogos digitais.

2.3.1 Enhanced NPC Behaviour using Goal Oriented Action Planning

Long (2007) implementa, testa e compara duas técnicas de IA, uma utilizando GOAP e outra usando Máquina de Estados Finita (FSM), em um jogo 2D simplificado baseado modos de jogo de um jogo multijogador 3D de tiro onde duas equipes ou jogadores individuais competem em arenas.

O primeiro modo planejado é dominação, cada time deve ocupar (e impedir que o oponente ocupe) áreas específicas para ganhar pontos; após, foram adicionados outros modos como captura a bandeira, em que seu time deve roubar a bandeira da base adversária e levar até a própria; mata-mata, um modo sem times, onde o jogador que conseguir eliminar o maior número de jogadores dentro de um tempo limite vence; e último homem de pé, que funciona de forma similar ao mata-mata, mas o jogador ao morrer não retorna até o fim da partida, vencendo quem sobreviver até o final.

Os testes foram divididos em três partes, primeiro comparações de resultados nos modos de jogo, cada time composto somente de NPCs e utilizando somente uma das técnicas. As comparações para o mata-mata e último homem de pé foram uma partida de 60 minutos em dois mapas diferentes; para os modos em equipe: capture a bandeira e dominação, foram separados em 6 casos de combinação entre o GOAP e FSM com e sem táticas de esquadrão, cada caso sendo realizado 5 partidas de 20 minutos.

O segundo teste consistiu em testes de performance computacional e eficiência em gerenciamento de memória e processamento, sendo obtidos os dados com VTune em uma partida de mata-mata de 25 minutos com somente um tipo de IA por vez. O trabalho também analisou a facilidade de implementação, flexibilidade e a reusabilidade de cada técnica.

O trabalho constatou que a FSM só ganhou do GOAP no cenário específico de modo de jogo. Quanto à eficiência de recursos, o uso médio do processador para o GOAP foi 33.62% contra 35.44% da FSM, mas o GOAP teve um mínimo de 2.34% e máximo de 98.44%, enquanto a FSM teve 0.05% e 71.09% respectivamente, demonstrando ser mais estável e também utilizou, em média, 1.8MB a menos de memória. E por fim a média de renderização de quadros por segundo (FPS) foi de 439 para o GOAP e 464 para a FSM. Na implementação, notou-se um aumento de complexidade na FSM ao adicionar modos diferentes de jogo por ter necessidade de adicionar novos estados além de transições entre todos os já existentes. O GOAP se mostrou mais flexível, pois cada nova ação adicionada não requer que sejam realizados tratamentos

nas já existentes. Por outro lado, para implementar e depurar o código do FSM se provou mais prático, por ter uma arquitetura mais simples, ao contrário do GOAP que precisa fazer um gerenciador de objetivos, planejadores, busca A* para objetivos e para o mapa e contêiner para as ações.

2.3.2 Decision-Making Decisions: Artificial Intelligence In Computer Games

Britton (2023) examina três tipos de técnicas de IA usadas no comportamento de NPCs em jogos digitais: GOAP, Máquina de Estados Finita e Árvore de Comportamento. O objetivo principal da pesquisa foi o de analisar os pontos fortes e fracos de cada técnica, usando três jogos comerciais como exemplos de cenários ideais para cada uma: Hitman[4] (2016), The Elder Scrolls V: Skyrim[5] (2011) e DEFCON[6] (2006).

Hitman é um jogo 3D em terceira pessoa em que o jogador deve eliminar determinados alvos em um mapa populado de NPCs, que assumem diferentes papéis como guardas e civis, e reagem de acordo com a situação, mas também têm rotinas que seguem durante o passar do tempo.

Skyrim é um jogo 3D em terceira pessoa, RPG de ação e mundo aberto, onde o jogador explora o mapa e faz missões para evoluir e progredir na história do jogo. Neste contexto, o mundo tem uma grande quantidade de NPCs com funções simples, como dialogar, lutar e dormir à noite, seguindo também uma rotina.

DEFCON é um jogo de estratégia em tempo real, com partidas de até 6 jogadores ou IAs competindo num cenário inspirado na Guerra Fria e conflitos nucleares. As IAs no jogo tomam o mesmo papel dos jogadores de controlar um dos adversários e tomam decisões de acordo com as informações recebidas durante a partida.

Não foram feitos testes práticos nem implementações, focando em descrever as arquiteturas dos modelos em detalhes, produzindo nos resultados uma análise conceitual.

O texto conclui que jogos como Hitman, que possuem vários NPCs que lidam com decisões mais complexas de comportamento, baseadas em múltiplos fatores dinâmicos, se beneficiam do GOAP, onde a seleção dos objetivos e a execução deles está mais desconectada uma da outra, permitindo maior adaptabilidade a mudanças. Jogos de mundo aberto com grande quantidade de NPCs, como o jogo Skyrim, usam Máquinas de Estados Finitas que conseguem executar funções mais simples eficiente-

[4] https://store.steampowered.com/app/236870/HITMAN/
[5] https://store.steampowered.com/app/489830/The_Elder_Scrolls_V_Skyrim/
[6] https://store.steampowered.com/app/1520/DEFCON/

mente, sem um grande custo de processamento. Enquanto para jogos como Defcon, que precisam de um grande número de estados, a utilização da técnica de Árvores de Comportamento consegue gerenciar as transições entre eles sem escalar exponencialmente a quantidade de transições, a estrutura de árvore também permite o planejamento de comportamentos mais específicos em sub-árvores, tudo isso facilitando um grande número de possibilidades de respostas da IA.

2.3.3 Utility-Directed Goal-Oriented Action Planning: A Utility-Based Control System for Computer Game Agents

Neste artigo, Sloan, Namee e Kelleher (2011) propõem uma versão melhorada do Goal-*Oriented Action Planning* (GOAP) chamada *Utility-Directed Goal-Oriented Action Planning* (UD GOAP), com o objetivo de resolver pontos fracos do método original, utilizando objetos inteligentes e a *utilidade* de estados do mundo resultantes. Também implementa ambos os métodos para finalidade de comparar em três aspectos: tempo que o agente demora para completar seus objetivos, performance computacional e de memória e a complexidade dos planos que cada um consegue gerar.

Sloan, Namee e Kelleher (2011) determinam dois aspectos do GOAP como pontos fracos: (1) não conseguir considerar o resultado total de um plano, sempre escolhendo a alternativa com menor custo, mesmo que, proporcionalmente, a com maior custo dê um melhor resultado, usando no texto um exemplo de um agente com objetivo de recuperar pontos de vida, tendo como opções andar uma distância 5 e recuperar 20 pontos ou andar 15 e recuperar 100, com o GOAP sempre escolhendo a opção mais barata. E (2) o GOAP não consegue considerar múltiplos objetivos enquanto planeja, tendo dificuldade de, por exemplo, eliminar um inimigo enquanto também economiza recursos.

As comparações foram feitas num cenário baseado no jogo The Sims, onde o agente se encontra em um ambiente que representa uma casa, tendo motivações que representam: entretenimento, energia, fome, conforto, higiene, solidão e vontade de urinar. Cada motivação começa com um valor zero e o objetivo do agente é subir todas para 100, contando com 25 ações diferentes para planejar, possuindo pelo menos duas ações diferentes para uma mesma motivação, usando como regra geral que as ações que duram mais tempo contribuem mais proporcionalmente para o objetivo. A simulação não usou nenhuma aleatoriedade, sendo executada três vezes para cada modelo, ambos usando como seletor de objetivo a motivação com menor valor.

O GOAP demorou 2 minutos e 26 segundos para atingir um estado ideal, enquanto o UD GOAP demorou 1 minuto e 58 segundos, por conseguir escolher corretamente quando a ação mais demorada era também mais vantajosa. Para o uso de recursos computacionais, o GOAP usou 1.09% do processador e 0.54% da memória, e o UD GOAP utilizou 1.34% e 0.74% respectivamente, sendo mais pesado por causa de sua função utilidade. Na questão da complexidade, o GOAP se mostrou mais repetitivo em seus planos, alternando entre ações de baixo custo, já o UD GOAP primeiro escolhia as ações com maior incremento e, depois, selecionava as de baixo custo com baixo aumento de motivação para cobrir o aumento restante.

2.3.4 Similaridades

Os trabalhos se assemelham com a pesquisa proposta no quesito comparativo entre métodos de IA para NPCs, utilizando múltiplos agentes em um mesmo teste assim como Long (2007), tendo como diferenças os modelos selecionados (FSM e GOAP) e a natureza competitiva do jogo implementado.

O trabalho de Sloan, Namee e Kelleher (2011) possui maior semelhança, tendo os mesmos métodos sendo comparados, mas utilizando somente um agente por teste, e medindo uso do processador e memória para os recursos computacionais, sem utilizar a renderização de quadros por segundo como métrica. Também mede o comportamento do agente baseado no tempo de execução, enquanto a implementação proposta se baseia na razão entre satisfação e dinheiro gasto pelo cliente.

3. PROCEDIMENTOS METODOLÓGICOS

A pesquisa tem como princípio a comparação entre métodos de arquitetura para IAs de NPCs em jogos eletrônicos, sendo parte posterior à implementação dos métodos e do jogo. Para isso, foram definidos dois métodos a partir de pesquisa bibliográfica: GOAP e UD GOAP, métricas e ferramentas para a avaliação e desenvolvimento.

Neste contexto, a pesquisa tem abordagem de caráter quantitativa, referente aos testes de performance mensuráveis, como taxa de quadros por segundo e o uso de recursos computacionais como memória e processamento, junto com uma análise da qualidade dos comportamentos apresentados pelos agentes a partir de dados mensuráveis. Trata-se de uma pesquisa de natureza aplicada, buscando gerar conhecimento com aplicação prática no desenvolvimento de jogos e sistemas de IA. Quanto aos objetivos, classifica-se como uma pesquisa exploratória, pois busca aprofundar a compreensão sobre a arquitetura GOAP e suas variações em um novo ambiente experimental. Em relação aos procedimentos, consiste em uma pesquisa experimental, baseada no desenvolvimento de um ambiente controlado (jogo) para a realização de testes comparativos entre as arquiteturas estudadas.

Segue a descrição das principais etapas desenvolvidas na elaboração dessa pesquisa.

3.1 Determinação de Métodos e Métricas

Com a seleção do GOAP como método principal, fez-se um levantamento e análise de materiais relevantes sobre as arquiteturas GOAP e variações, que nesse contexto são UD GOAP, UD GOAP com *smart ambience* (ambiente inteligente) proposto posteriormente por Sloan (2015), GOAP com lógica *fuzzy* (ou lógica difusa) e outros métodos híbridos abordados por Uludağlı e Oğuz (2023).

Para fins de manutenção de escopo, foi selecionada a variação UD GOAP de Sloan, Namee e Kelleher (2011) para ser comparada ao GOAP tradicional, podendo usar os resultados da pesquisa anterior como referência de métricas e para análise de resultados.

As métricas de avaliação dos testes de caráter quantitativo no que tange ao desempenho incluem: porcentagem de tempo de uso do processador, quantidade de memória principal em megabytes (MB) alocada para o jogo, quadros renderizados por

segundo e tempo total de execução. Escolhidas com base nos trabalhos relacionados de Long (2007), de Sloan, Namee e Kelleher (2011) e de Sloan (2015).

Outras métricas que abordam o comportamento dos agentes apresentados pelos diferentes modelos foram selecionadas: quantidade de dinheiro gasto e nível de satisfação alcançado.

3.2 Especificação do Jogo e implementação dos Métodos

A base para testes é um jogo de simulação de loja, conforme apresentado na Seção 2.1.2, onde a intenção é o jogador gerenciar o funcionamento de um mercado e os NPCs popularem o mundo assumindo o papel dos clientes, navegando pela loja, escolhendo e comprando produtos.

Para o escopo deste trabalho, foi implementado apenas o necessário para a realização dos testes comparativos. Isso inclui as diferentes IAs utilizadas pelos NPCs e todos os elementos com os quais esses agentes interagem durante o jogo. Entre eles estão: o gerenciador de criação de agentes, funções globais para acesso ao estado do mundo, a disposição física da loja para a navegação dos NPCs, produtos com informações pertinentes à heurística de decisão do agente, caixa registradora para a efetuação de pagamentos. Além disso, foram adicionados objetos que permitem a troca inesperada de objetivos durante a execução de um plano, como uma televisão que, ao ser aproximada, aumenta sua prioridade.

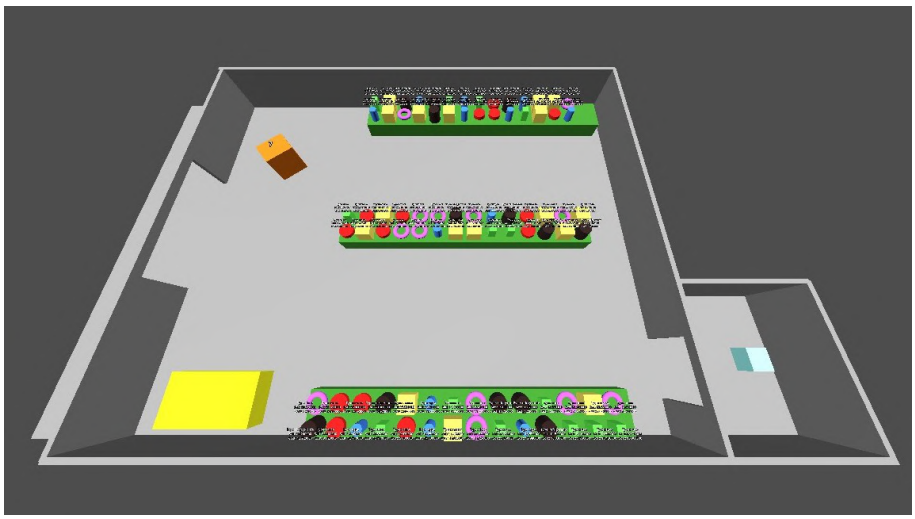

Figura 3.1: Vista aérea da loja utilizada durante os testes.

O jogo de simulação conta com uma câmera com visão aérea da loja para observação do comportamento apresentado pelos agentes (NPCs clientes). Representado na Figura 3.1, a loja foi organizada com:

- Uma porta de entrada, local onde os NPCs são instanciados e ao final, vão para simular a saída do estabelecimento, o tempo de execução do cliente é dado a partir desses dois momentos, e o teste ficará em execução até o último cliente sair da loja.
- Três prateleiras posicionadas em paralelo na cor verde, com produtos dispostos em fileiras duplas sobre elas. O agente deve andar ao redor das prateleiras para chegar onde deseja.
- Uma frente de caixa ou ponto de venda, onde os clientes que pegaram pelo menos um produto devem se direcionar para pagar antes de sair da loja. Representado na cor amarela no canto inferior esquerdo.
- Um banheiro aos fundos do mercado, indicado por uma caixa de cor azul no canto inferior direito. Os clientes que precisarem ir ao banheiro terão que ir até este local.
- Por último uma televisão na cor laranja, com um número disposto sobre ela, como visto na Figura 3.2. O número que varia de 1 à 8, trocando de valor a cada 1,5 segundos, indica quão alto está a vontade de assisti-la que somente ganha efeito caso o cliente esteja em um raio de 10 metros ao seu redor.

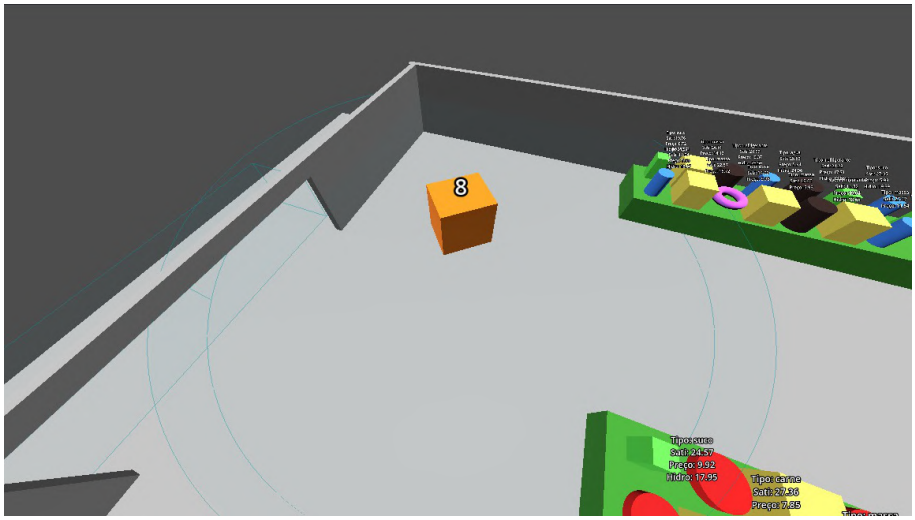

Figura 3.2: Televisão disposta na entrada da loja com sua área de efeito.

Os produtos dispostos nas prateleiras são a principal interação do cliente com a loja. A Figura 3.3 mostra os seis tipos da esquerda para a direita: massa,

doce, carne e três bebidas: refrigerante, suco e água. Possuem um preço entre 5,00 e 20,00, uma propriedade de satisfação com valor de 10,00 a 30,00 que é somada ao agente quando o item é pego, caso seja uma bebida, também possui a propriedade de hidratação (10,00 a 30,00), que é adicionada a vontade do NPC de ir ao banheiro.

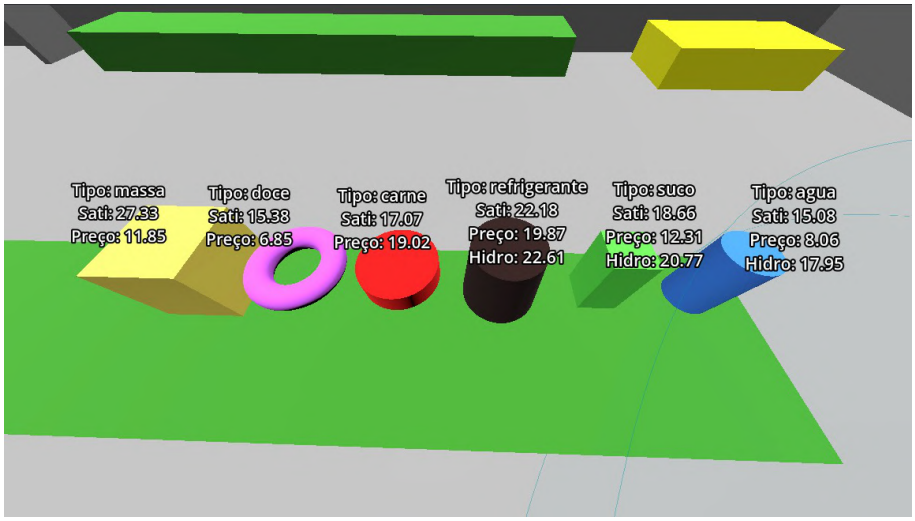

Figura 3.3: Produtos de cada tipo dispostos na prateleira.

Os NPCs são os objetos do jogo no qual os métodos GOAP e UD GOAP são implementados para controlar. Possuem:

- Barra de satisfação que começa em 0,0 e vai até um máximo de 100,0, sendo o principal foco do agente maximizar essa barra, comprando produtos que possuem um preço e um valor de satisfação a ser somado com o próprio.
- Barra de vontade de urinar, que quanto mais alta mais priorizada. Ao comprar bebidas este valor aumenta pelo montante indicado no produto. Este valor é inicializado entre 0,00 e 70,00, com objetivo de representar três possibilidades iniciais: não precisar do banheiro, precisar mas não é prioridade e ter como o foco inicial a ida ao banheiro.
- Dinheiro para comprar os produtos. O cliente não pode comprar um item mais caro do que consegue pagar. O agente possui entre 10,00 e 140,00, sem possibilidade de aumentar esse valor.
- Preferência por produtos de certos tipos, indo de um a três tipos em que o agente pode considerar a compra, cada preferência é um valor entre 0,2 e 0,8, que é usada como peso no custo da ação de pegar o produto. Quanto maior o valor, mais ele prioriza o tipo do produto durante o planejamento.

3.2.1 Implementação GOAP

A implementação do GOAP, que atua através do NPC, consistiu em classes para o agente, planejador e cada ação e objetivo desenvolvido. A classe do agente, instanciada para o NPC, gera os valores aleatórios, os fixos, instância a classes de objetivos e do planejador, cria um estado do mundo interno, contendo as informações necessárias para o próprio planejamento, mas não para outros agentes. Aqui a seleção e atualização de objetivos é feita, a chamada do planejador e o plano é executado no mundo.

O estado interno consiste nas variáveis booleanas: *está-satisfeito*, *acabou-de-comprar*, *pagou*, *precisa-usar-o-banheiro*, *usou-banheiro*, *assistindo*, *saiu*. Essas variáveis condizem com os efeitos e pré-condições das ações, estado ideal dos objetivos, assim como para ver quais são válidos.

As ações são classes que possuem funções: *pegar-efeitos*, *pegar-pré-condições*, *é-válida*, *pegar-custo* e *agir* foram criadas seis ações diferentes: ação pegar item, ir até local, assistir televisão, usar o banheiro, pagar e ir embora. A função *agir* é chamada pelo agente durante a execução do plano, que contém a lógica para executá-la, caso ela não consiga ser executada, o plano deve ser abortado.

Os objetivos são classes que possuem funções: *é-valida*, *prioridade* e *pegar-estado-objetivo*. A prioridade é usada pelo agente para definir qual estado objetivo irá mandar para o planejador. Os objetivos são: se satisfazer, se aliviar, assistir TV e ir embora.

O planejador é uma classe que ao ser instanciada, também instancia todas as ações do agente, recebe o estado objetivo enviado pelo agente e realiza uma busca A* regressiva, verificando os efeitos e pré-condições, filtra ações que não podem ser usadas com a função *é-válida* das mesmas. A heurística usada para cálculo do custo durante a busca é a quantidade de condições no estado-objetivo que não foram satisfeitas.

3.2.2 Implementação UD GOAP

Utilizando a implementação do GOAP realizada anteriormente, foi adaptada para o UD GOAP, dessa forma, possui classes para o agente, ações e planejador, trocando a classe de objetivos pela classe de motivações, foram definidos cinco motivações diferentes: *se-entreter*, *se-aliviar*, *ir-embora*, *se-satisfazer* e *economizar-dinheiro*.

Foram implementados os objetos inteligentes, que dizem ao agente como executar ações que os usam, como a ação ir até local, o item tem guardado a distância que o NPC deve ficar, então ir até uma TV te deixa mais longe dela do que ir até o banheiro, que você precisa estar mais perto para utilizar.

As motivações são classes instanciadas pelo agente, elas possuem uma função que retorna a tupla de objetivo, e a função utilidade que tem o peso a ser utilizado e a lógica para determinar o quão o estado está do ideal para a motivação específica, como explicado na Seção 2.2.3.

A motivação adicional de economizar dinheiro foi introduzida para possibilitar a demonstração da diferença entre os modelos, já que o UD GOAP pode utilizar informações sobre os objetos como preço e satisfação gerada. A motivação *se-satisfazer* utiliza a variável de satisfação do cliente, sendo considerada equivalente ao objetivo de mesmo nome. Para a outra variável de interesse, o preço, foi criada uma nova motivação que nunca tem prioridade na seleção da motivação principal utilizada no planejamento.

O planejador, como no GOAP, instancia as ações do agente e realiza uma busca regressiva A*, que, ao invés de equiparar condições com efeitos, verifica as ações válidas com maior nota para utilidade.

3.2.3 Ferramentas de Desenvolvimento

O motor de jogos Godot[1] na sua versão 4.5 foi escolhido para o desenvolvimento do jogo base, utilizando a linguagem própria do motor, GDScript. A implementação do GOAP para Godot foi inicialmente baseada no projeto de Gerevini (2023), que definiu a estrutura geral dos arquivos e classes, mas vários aspectos necessitaram de mudanças para se adequar ao modelo como definido pelo (ORKIN, 2003), como o planejador usar uma busca em largura ao invés do algoritmo A*.

3.3 Coleta de Dados

Após a finalização das etapas de implementação, foi conduzido um conjunto sistemático de testes com o objetivo de avaliar o desempenho e a eficácia das funcionalidades desenvolvidas. Esses testes foram realizados com base nas métricas previamente descritas na Seção 3.1.

[1] https://godotengine.org/

Para a coleta dos dados foi utilizado uma implementação dentro do próprio Godot via código, salvando os dados em arquivos de texto durante a execução. O dado quantitativo de quadros por segundo (frames per second, FPS) também foi obtido dentro do motor, com a função *get_monitor* da classe *Performance* passando como parâmetro a constante *TIME_FPS* da mesma classe (Godot Engine developers, 2025). A coleta do tempo de uso da CPU e de memória principal exclusiva do processo, métricas quantitativas, foram feitas com scripts PowerShell utilizando a classe *System.Diagnostics.PerformanceCounter* da plataforma .NET, com os argumentos *Process* para monitorar um processo específico, *% Processor Time* para retornar a porcentagem de tempo que o processo usou da CPU para executar instruções e *Working Set - Private* que indica páginas de memória recentemente referenciadas por um processo que não podem ser compartilhadas com outros processos.

A coleta do tempo de execução do caso, foi coletada via comando *Get-Date* do PowerShell, mas também pode ser inferida a partir dos dados, coletados a cada segundo.

O computador utilizado na execução dos testes e coleta de resultados foi um Notebook Acer Nitro AN515-44. A unidade de processamento central (CPU) utilizada foi um AMD Ryzen 7 4800H, configurado com 8 núcleos físicos e 16 threads. A placa de vídeo dedicada foi uma NVIDIA GeForce GTX 1650. A memória principal instalada totaliza 8 GB. O ambiente de execução do sistema operacional foi o Microsoft Windows 10 Pro (64-bit).

O conjunto de testes realizados consistiu em uma quantidade crescente de NPCs sendo criados, seguindo a ordem de: teste com 1, 2, 3, 4, 5, 10, 15, 20, 30 e 50 agentes, totalizando 10 quantidades diferentes. cada configuração de quantidade de NPCs foi executada 10 vezes para ambos modelos implementados, com a finalidade de calcular uma média e tirar peso de possíveis outliers. Também foi testado uma vez com 200 e outra com 1000 agentes para cada modelo, após identificar que as quantidades selecionadas ainda não chegavam no limite da máquina, totalizando 204 testes.

Para consistência entre os testes, foi feito um script PowerShell que executa os programas em laço esperando 10 segundos entre o fim de um caso e o começo do próximo, rodando um por vez, para evitar influências nos dados coletados.

O cenário ao ser rodado utilizou uma semente para gerar valores pseudoaleatórios, as sementes foram selecionadas com hash de uma string passada ao início do teste de formato: npc-1-experimento-5, o primeiro número identificando a quantidade de npcs presentes no experimento e o segundo número a instância do teste (1 a

10). O propósito desse formato foi garantir a variedade entre cada repetição enquanto consegue obter o mesmo cenário para ambos os modelos.

Durante os testes, também foi avaliada a qualidade dos comportamentos apresentados e foram obtidos dados que possam demonstrar tais comportamentos. A partir da interpretação desses dados, é possível extrair conclusões fundamentadas, contribuindo para responder às questões de pesquisa levantadas e para a reflexão sobre possíveis aprimoramentos futuros.

Cinco dos 102 testes feitos com o UD GOAP tiveram que ser refeitos com uma nova semente aleatória, que seguindo o padrão utilizou números acima de 10, os casos falhos sendo descartados, devido a um alinhamento das variáveis pseudoaleatórias que possibilitava o cliente assistir TV indefinidamente, ao invés de priorizar a saída.

4. RESULTADOS

A seguir estão os resultados registrados da porcentagem utilizada do processador, da porcentagem utilizada da memória principal e da renderização de quadros por segundo. Os gráficos indicados nas figuras mostram o GOAP na cor azul e o UD GOAP na cor vermelha, o tempo medido em segundos, o uso da CPU medido em porcentagem do tempo do processador utilizado e o uso da memória principal medido em megabytes.

Os testes realizados não têm um tempo predeterminado de execução. Assim, a variação da duração entre testes se evidencia nos gráficos de linha com um modelo terminando antes do outro, deixando um dos modelos sem dados nos períodos subsequentes de tempo. Isso também se demonstra no estreitamento do desvio padrão, pois a quantidade de casos para a média diminuiu com o passar do tempo, com alguns casos tendo uma única iteração que demorou mais que o resto, o que torna a média igual ao valor bruto e o desvio padrão zero.

Os dados utilizados para a análise da qualidade dos comportamentos dos modelos foram: quanto gastou, quando se satisfez e o tempo de execução. Usados para uma comparação entre os modelos e com os resultados dos trabalhos relacionados.

4.1 Uso da CPU

As Figuras 4.1 à 4.10 apresentam a média e o desvio padrão do uso de CPU dedicado ao programa ao longo do tempo, em segundos, durante a execução dos testes. Observam-se valores semelhantes entre as abordagens analisadas, ambas demandando percentuais próximos de processamento. Verifica-se um crescimento inicial do uso da CPU seguido de estabilização em torno de aproximadamente 6,5%, resultando em média geral entre os 102 casos de 5,97% para o GOAP e 6,19% para o UD GOAP.

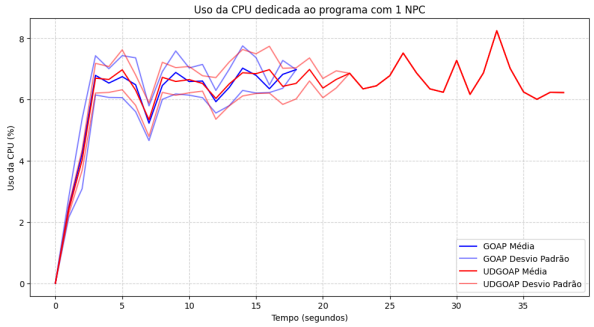

Figura 4.1: Uso da CPU dedicada ao programa com 1 NPC.

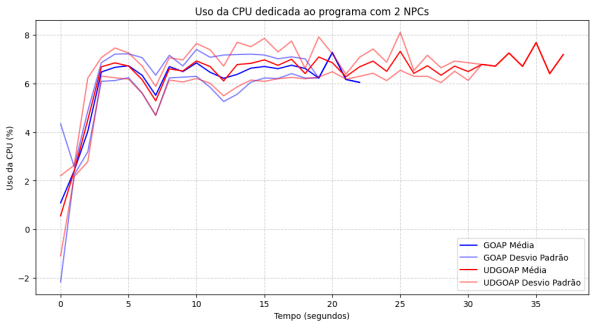

Figura 4.2: Uso da CPU dedicada ao programa com 2 NPCs.

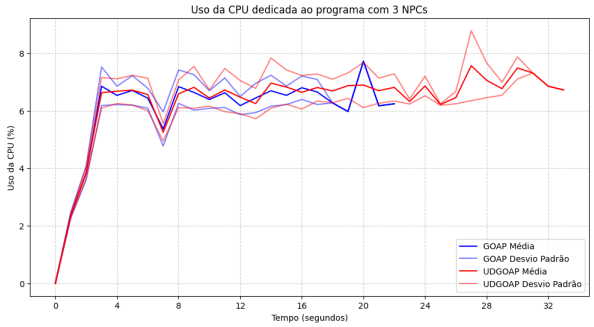

Figura 4.3: Uso da CPU dedicada ao programa com 3 NPCs.

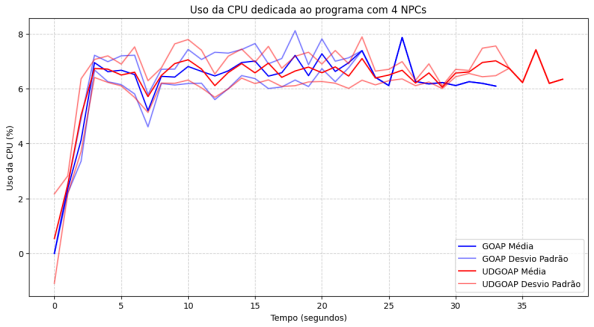

Figura 4.4: Uso da CPU dedicada ao programa com 4 NPCs.

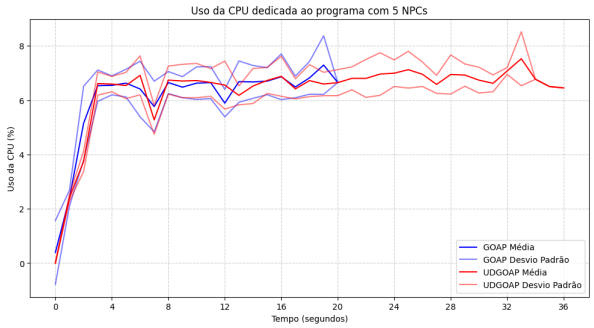

Figura 4.5: Uso da CPU dedicada ao programa com 5 NPCs.

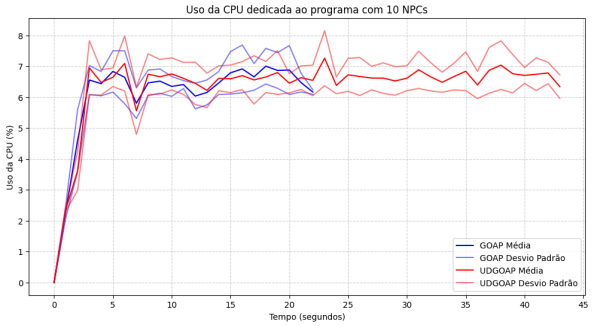

Figura 4.6: Uso da CPU dedicada ao programa com 10 NPCs.

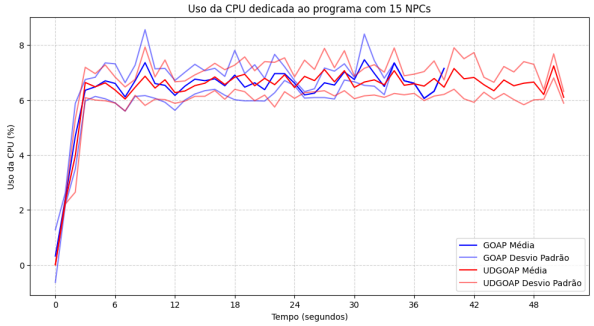

Figura 4.7: Uso da CPU dedicada ao programa com 15 NPCs.

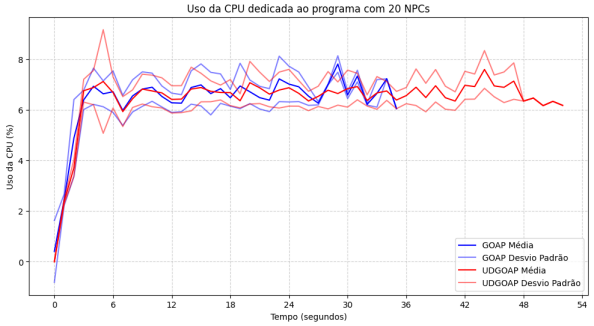

Figura 4.8: Uso da CPU dedicada ao programa com 20 NPCs.

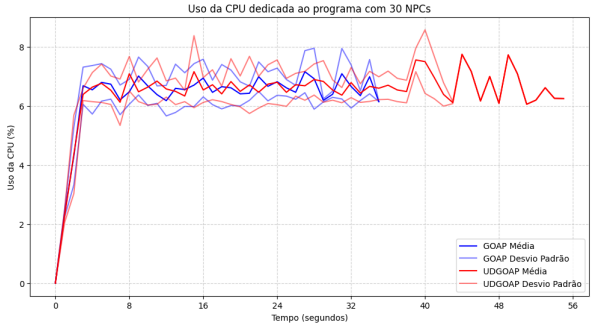

Figura 4.9: Uso da CPU dedicada ao programa com 30 NPCs.

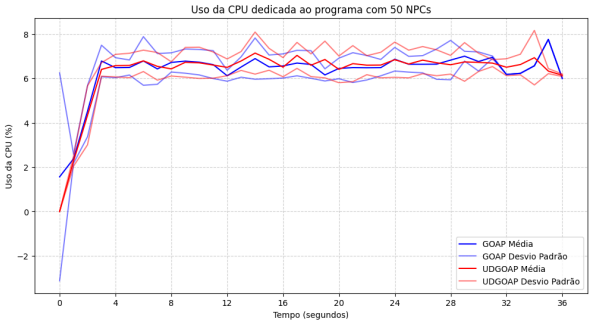

Figura 4.10: Uso da CPU dedicada ao programa com 50 NPCs.

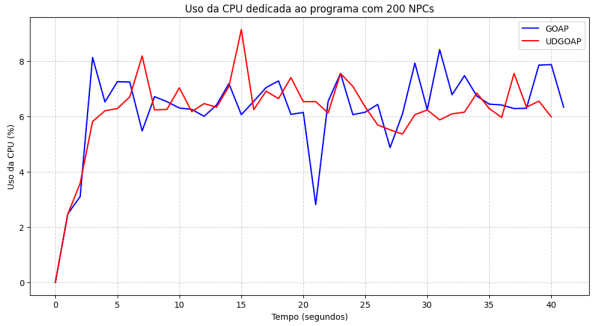

Figura 4.11: Uso da CPU dedicada ao programa com 200 NPCs.

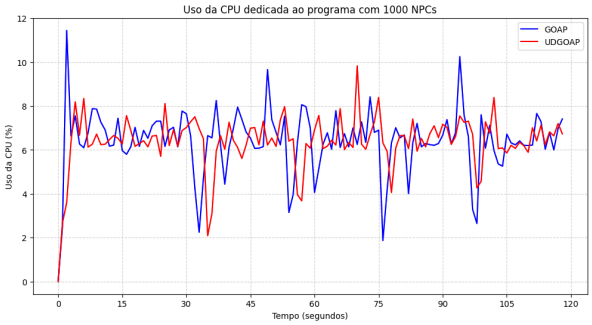

Figura 4.12: Uso da CPU dedicada ao programa com 1000 NPCs.

Além disso, destaca-se a baixa correlação entre o número de agentes e a utilização de CPU. Embora fosse esperado um aumento significativo de consumo com o incremento da quantidade de NPCs, verificou-se apenas um acréscimo de 0,73% ao comparar a execução com um e com mil agentes no GOAP, enquanto o UD GOAP apresentou aumento de 0,46% entre ambos. Esses resultados estão sintetizados na Figura 4.13, por meio de barras representativas da média de utilização da CPU durante toda a execução e agrupadas de acordo com o número de NPCs avaliados.

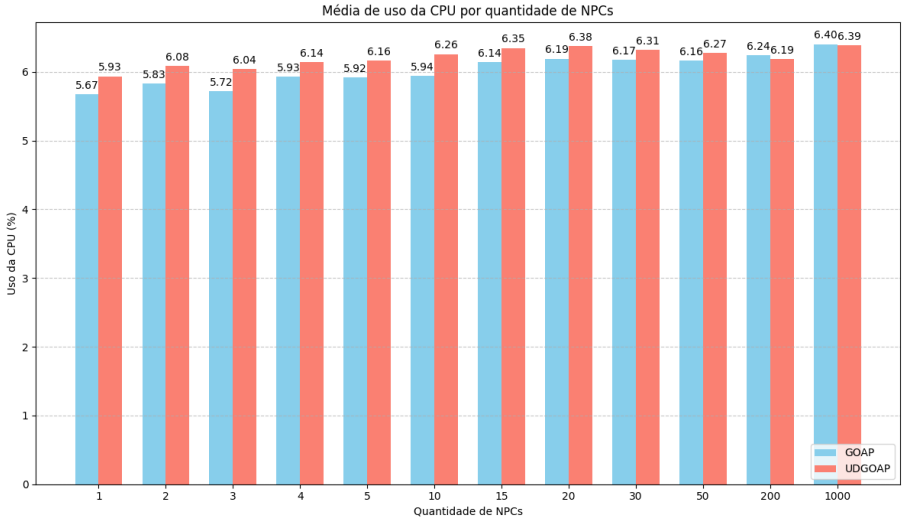

Figura 4.13: Média do uso da CPU com o aumento da quantidade de NPCs.

4.2 Uso da Memória Principal

As Figuras 4.14 até 4.23 apresentam o uso da memória principal, em megabytes (MB), registrado a cada segundo durante a execução dos testes. Observa-se um padrão relativamente uniforme ao longo do processo, caracterizado por um crescimento inicial seguido de estabilidade até o término da execução.

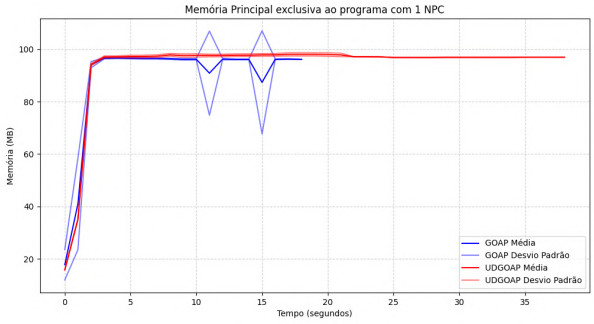

Figura 4.14: Uso da Memória Principal com 1 NPC.

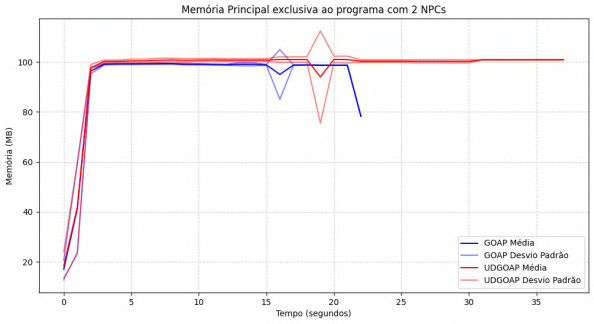

Figura 4.15: Uso da Memória Principal com 2 NPCs.

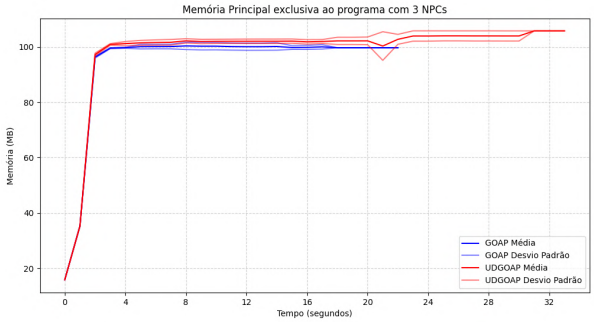

Figura 4.16: Uso da Memória Principal com 3 NPCs.

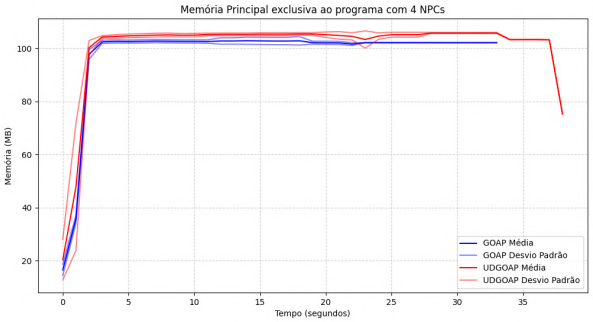

Figura 4.17: Uso da Memória Principal com 4 NPCs.

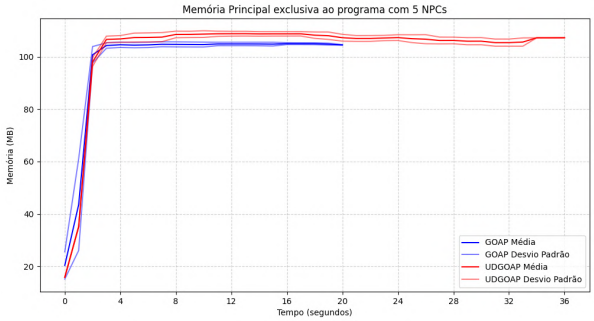

Figura 4.18: Uso da Memória Principal com 5 NPCs.

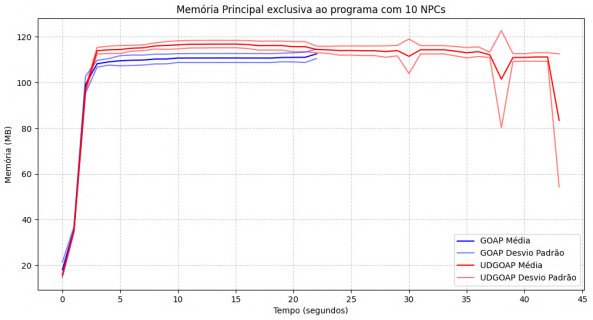

Figura 4.19: Uso da Memória Principal com 10 NPCs.

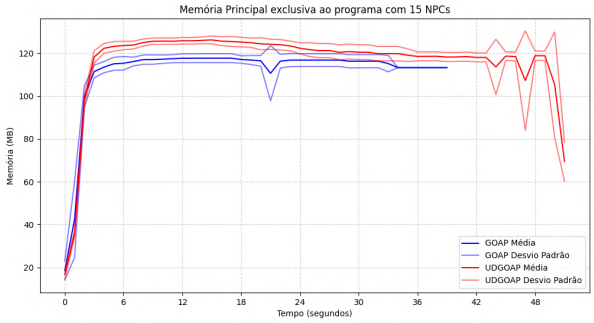

Figura 4.20: Uso da Memória Principal com 15 NPCs.

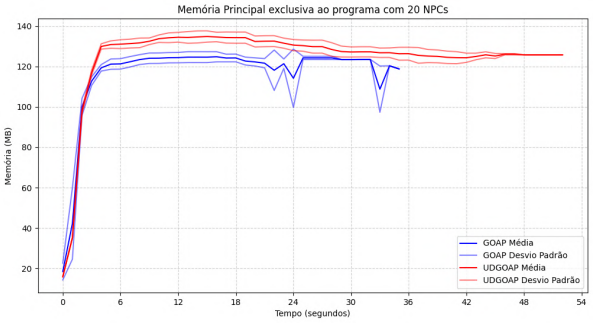

Figura 4.21: Uso da Memória Principal com 20 NPCs.

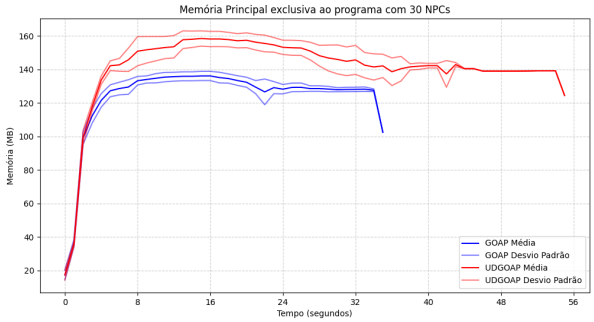

Figura 4.22: Uso da Memória Principal com 30 NPCs.

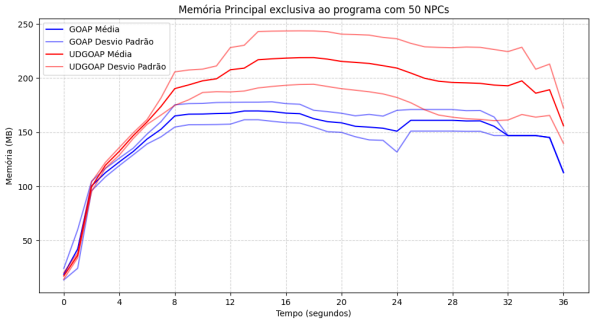

Figura 4.23: Uso da Memória Principal com 50 NPCs.

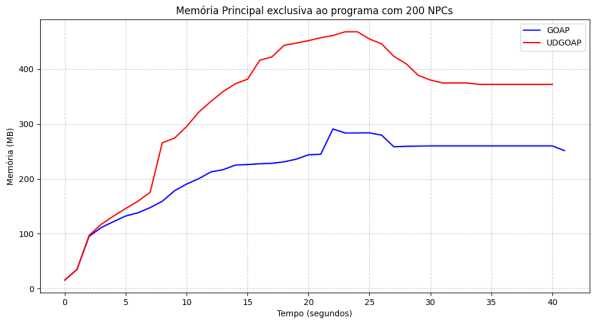

Figura 4.24: Uso da Memória Principal com 200 NPCs.

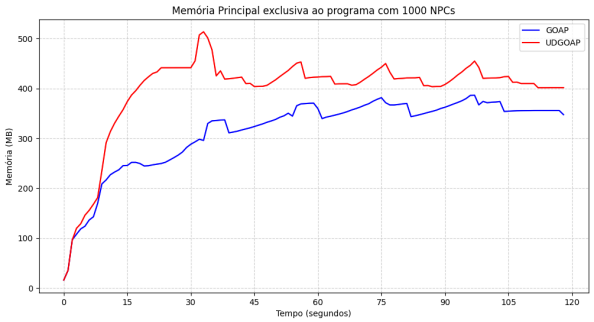

Figura 4.25: Uso da Memória Principal com 1000 NPCs.

Em contraste com o uso da CPU, a quantidade de agentes influenciou diretamente o consumo de memória principal, conforme evidencia a Figura 4.26. Em ambas as arquiteturas avaliadas, verificou-se aumento proporcional da utilização de memória em função do número de NPCs. O modelo UD GOAP apresentou média geral de 120,30 MB, superior ao valor observado para o GOAP, de 108,61 MB, além de indicar maior crescimento conforme o número de agentes aumentava em cada teste.

Nas instâncias com 200 e 1000 agentes, ilustradas nas Figuras 4.24 e 4.25, respectivamente, é possível observar mais claramente a variação temporal do consumo de memória em uma execução individual. Particularmente na Figura 4.25, identifica-se um padrão cíclico após o período inicial de crescimento, aspecto que será discutido posteriormente na análise detalhada do comportamento observado.

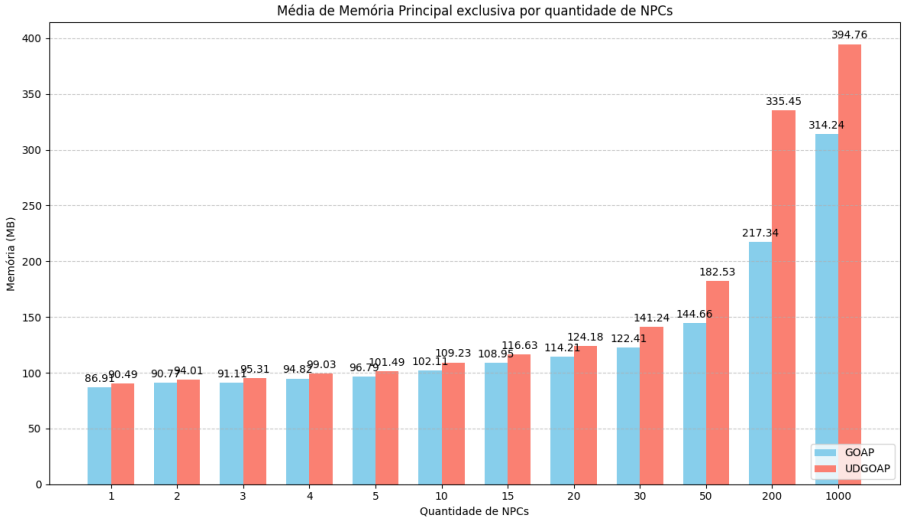

Figura 4.26: Média do uso da memória principal com o aumento da quantidade de NPCs.

4.3 Renderização de Quadros por segundo

As Figuras 4.27 até 4.36 apresentam a taxa de renderização de quadros ao longo do tempo, em segundos. Observa-se, para ambos os modelos, uma variação inicial seguida de um aumento gradativo até o final da execução dos testes. De modo geral, o modelo GOAP apresentou um aumento de quadros por segundo mais cedo quando comparado ao UD GOAP, evidenciado à medida que o número de agentes aumentou, e a diferença entre os modelos tornou-se mais pronunciada na região intermediária da execução, sendo reduzida nos instantes finais, quando ocorre diminuição no número de NPCs ativos no modelo UD GOAP.

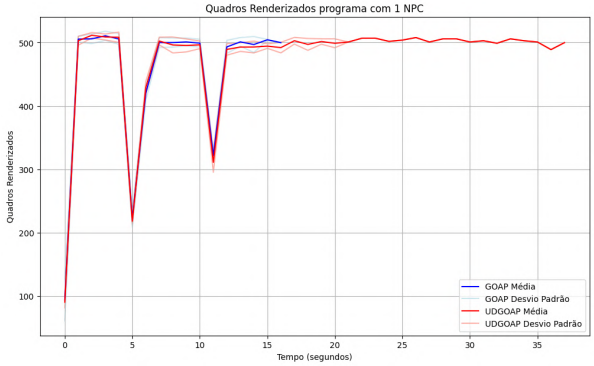

Figura 4.27: Renderização de Quadros por segundo com 1 NPC.

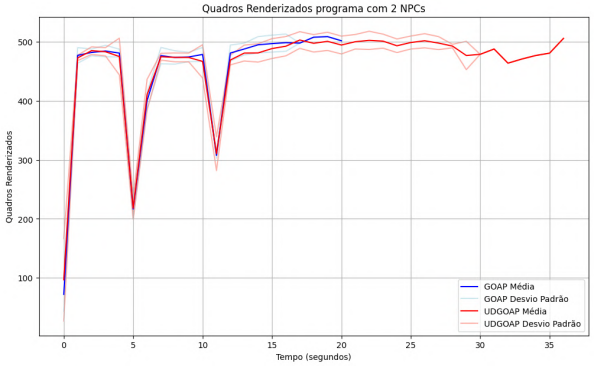

Figura 4.28: Renderização de Quadros por segundo com 2 NPCs.

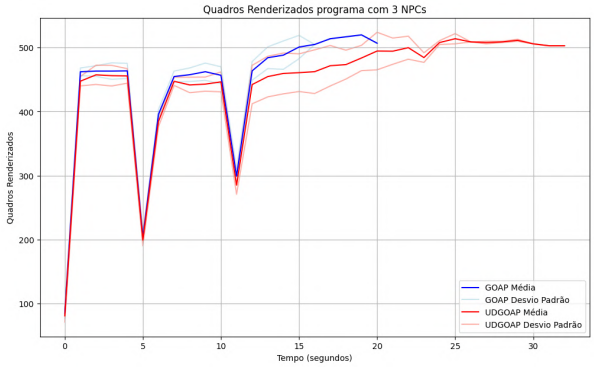

Figura 4.29: Renderização de Quadros por segundo com 3 NPCs.

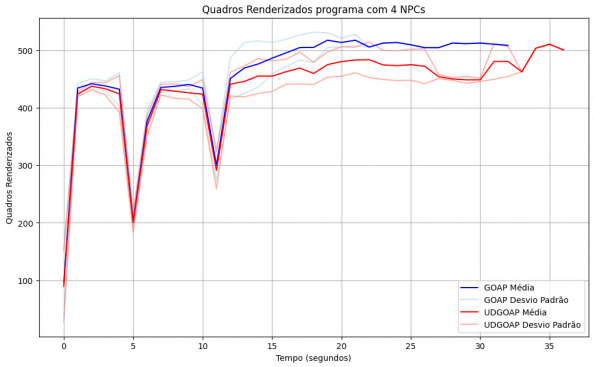

Figura 4.30: Renderização de Quadros por segundo com 4 NPCs.

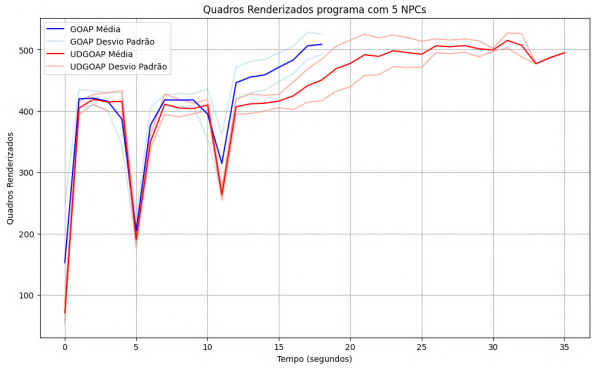

Figura 4.31: Renderização de Quadros por segundo com 5 NPCs.

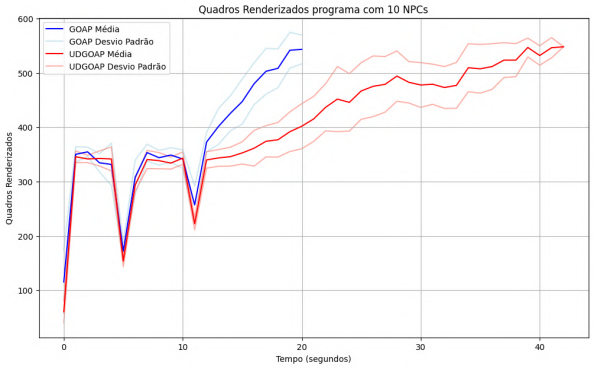

Figura 4.32: Renderização de Quadros por segundo com 10 NPCs.

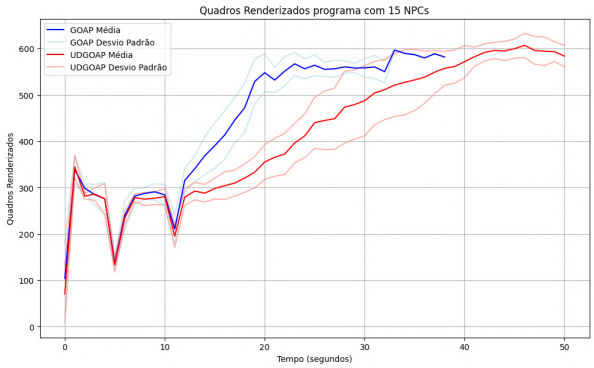

Figura 4.33: Renderização de Quadros por segundo com 15 NPCs.

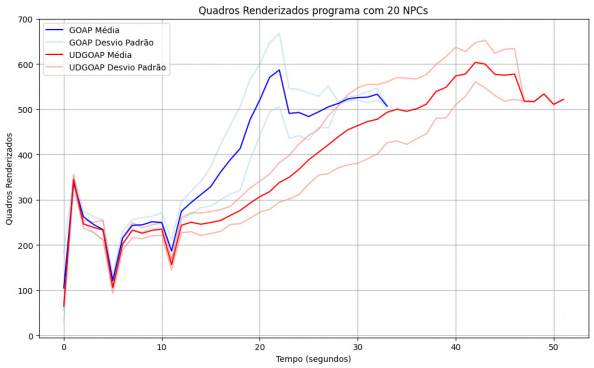

Figura 4.34: Renderização de Quadros por segundo com 20 NPCs.

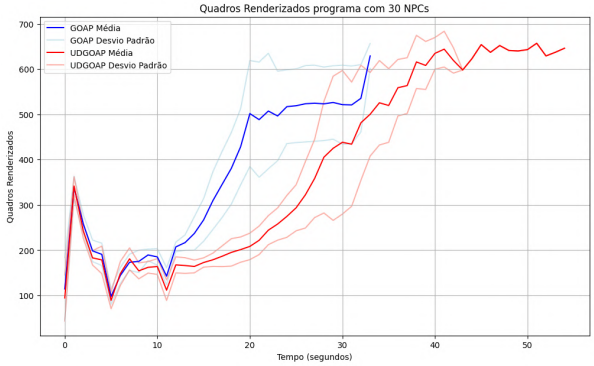

Figura 4.35: Renderização de Quadros por segundo com 30 NPCs.

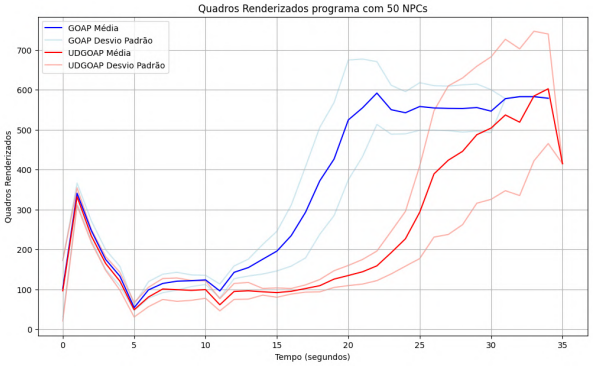

Figura 4.36: Renderização de Quadros por segundo com 50 NPCs.

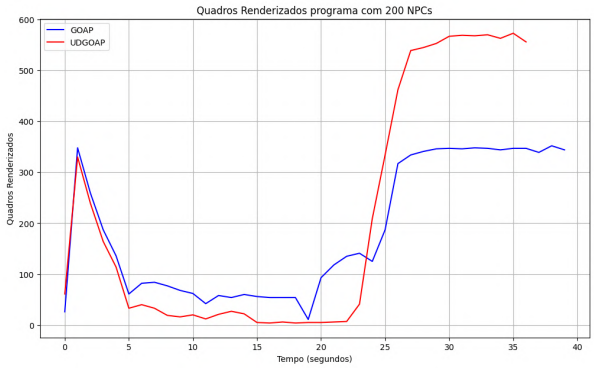

Figura 4.37: Renderização de Quadros por segundo com 200 NPCs.

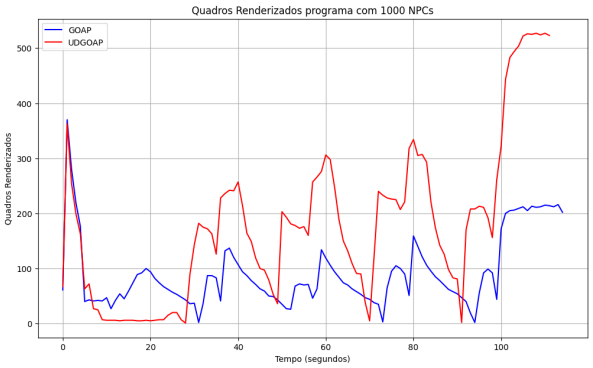

Figura 4.38: Renderização de Quadros por segundo com 1000 NPCs.

A Figura 4.39 evidencia que o incremento no número de NPCs exerce influência inversamente proporcional sobre a taxa de quadros por segundo, ou seja, quanto maior a quantidade de agentes, menor a taxa de renderização observada. No entanto, nota-se uma elevação desse valor ao final da execução, associada à saída gradual dos NPCs do ambiente, o que reduz a carga de processamento.

As médias gerais obtidas foram 364,74 quadros por segundo para o modelo GOAP e 372,87 para o modelo UD GOAP. Essa diferença pode ser atribuída à duração distinta das execuções, uma vez que o UD GOAP mantém-se por mais tempo na fase final, caracterizada por maior crescimento da taxa de quadros, elevando assim sua média geral. Esse comportamento explica também o fato de o UD GOAP apresentar redução menos significativa na taxa de quadros por segundo conforme o número de NPCs aumenta, como ilustrado na Figura 4.39.

Por fim, os resultados mostrados nas Figuras 4.37 e 4.38 reforçam a tendência observada previamente. Nessas instâncias, o UD GOAP supera o GOAP após determinado ponto da execução e apresenta novamente o padrão cíclico identificado anteriormente, conforme mencionado na Seção 4.2.

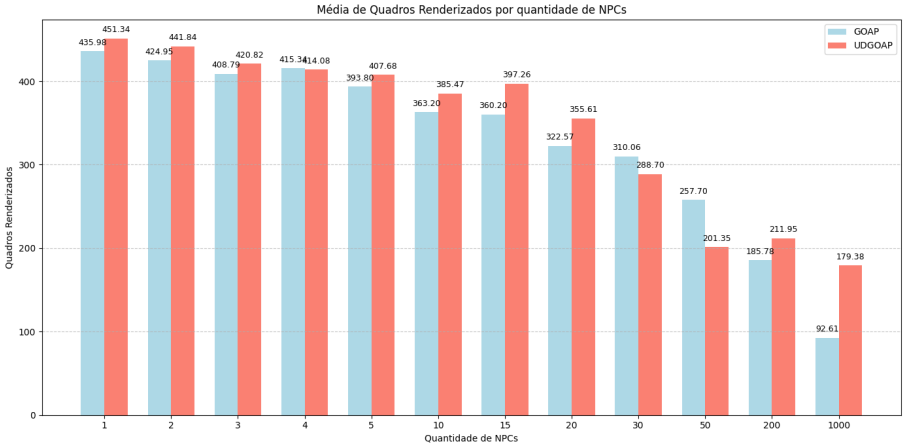

Figura 4.39: Média de renderização de quadros com o aumento da quantidade de NPCs.

4.4 Comportamentos

O padrão cíclico identificado nas Figuras 4.25 e 4.38 caracteriza um fenômeno diretamente associado ao comportamento emergente dos NPCs diante das condições impostas pelo ambiente de jogo. Esse comportamento corrobora a observação discutida por Long (2007), segundo a qual a execução simultânea de planejamento por múltiplos agentes resulta em queda de desempenho global. Nos cenários com elevada quantidade de NPCs, após o período inicial de execução, todos os produtos disponíveis na loja são adquiridos. Dessa forma, os agentes criados pelo gerenciador passam a possuir apenas duas opções viáveis: utilizar o banheiro, caso o estado interno permita, ou assistir televisão. Quando o valor associado à televisão diminui, conforme descrito na Seção 3.2, todos os agentes previamente parados passam a replanejar simultaneamente. Esse evento coletivo resulta em picos periódicos no gráfico de 4.25 e quedas em 4.38, evidenciando oscilações aproximadas de 21 segundos, tempo necessário para a variável associada à televisão retornar ao seu valor inicial.

A natureza do ambiente implementado implica que testes com quantidades muito elevadas de agentes se tornam pouco expressivos para fins de análise comparativa comportamental. Após a aquisição do último produto disponível, o sistema comporta-se como um fluxo contínuo de entrada e saída de NPCs, produzindo uma variação de desempenho somente devido à televisão que funciona como um gargalo, explicado no parágrafo anterior.

Esse fenômeno também repercute na qualidade dos planos formados. Em cenários com ampla densidade de agentes, a escolha de produtos torna-se limitada; selecionar um produto durante o planejamento não implica necessariamente que o

agente conseguirá obtê-lo, reduzindo a correspondência entre a decisão lógica e o resultado efetivo. As Figuras 4.40 e 4.41 ilustram essa situação. Ambas apresentam a razão entre a satisfação final do cliente e o gasto realizado, agrupadas conforme a quantidade de agentes por teste, sendo que a segunda exclui os casos em que não houve aquisição de produtos. Observa-se que, com o aumento da quantidade de NPCs, a diferença entre os resultados diminui, refletindo a redução da qualidade e precisão da ação planejada.

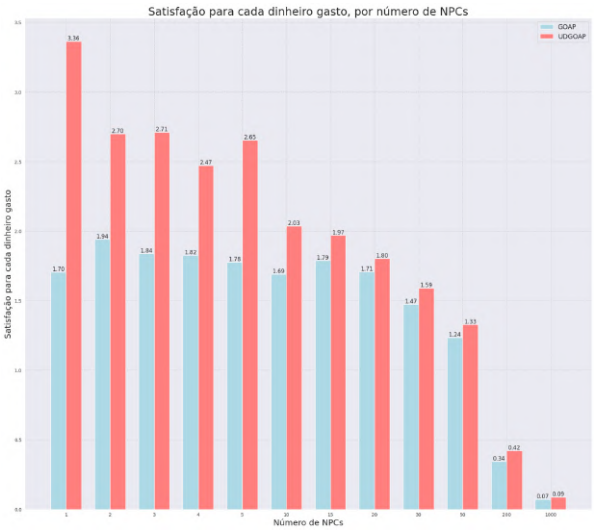

Figura 4.40: Média de satisfação por dinheiro gasto com o aumento da quantidade de NPCs.

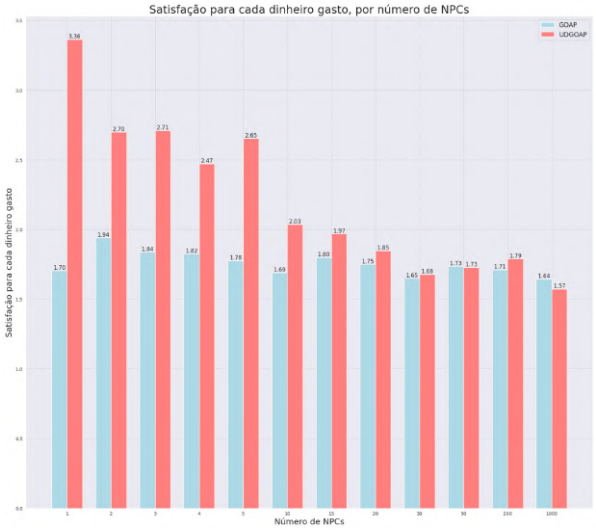

Figura 4.41: Média de satisfação por dinheiro gasto com o aumento da quantidade de NPCs, excluindo quem não comprou.

Adicionalmente, as Figuras 4.40 e 4.41 evidenciam o impacto das diferenças na heurística de seleção de ações entre os modelos, bem como o efeito da implementação de objetos inteligentes. Como o UD GOAP possui acesso a informações adicionais sobre os produtos, informações estas não consideradas pelo GOAP tradicional, esse modelo consegue ponderar o valor utilitário das ações com maior precisão, resultando em decisões mais coerentes em termos de custo e benefício.

Esse resultado também evidencia a influência do ambiente sobre as métricas utilizadas na avaliação. No estudo de Sloan, Namee e Kelleher (2011), o UD GOAP apresentou desempenho superior em termos de qualidade por concluir as tarefas mais rapidamente, decorrente da seleção de ações com maior relação custo-benefício. Entretanto, no cenário proposto neste trabalho, essa mesma característica resultou em maior tempo de execução, uma vez que a escolha de ações mais vantajosas frequentemente implicava maior deslocamento dentro da loja, aumentando o tempo necessário para sua conclusão. Além disso, enquanto no trabalho relacionado a maximização da satisfação era sempre possível, no ambiente aqui desenvolvido essa variável é li-

mitada pelo recurso monetário disponível, o que altera significativamente a dinâmica de decisão e o impacto das escolhas realizadas pelo agente.

5. CONSIDERAÇÕES FINAIS

Neste trabalho de conclusão de curso, foram avaliadas duas arquiteturas para a produção de inteligências artificiais em jogos, GOAP e UD GOAP. Arquiteturas definidas por Orkin (2003) e Sloan, Namee e Kelleher (2011), ambas relacionadas no quesito de planejamento de ações como maneira de determinar o comportamento de NPCs. Neste intuito, foram testadas quanto ao desempenho computacional e à qualidade dos resultados obtidos pelas execuções agentes em um mesmo cenário.

Os resultados computacionais demonstraram desempenho similar no uso do processador, não se tornando um aspecto decisor entre os modelos. Para uso de memória, UD GOAP apresentou desempenho inferior, utilizando em média 10,69 MB a mais que o GOAP, mas dependendo da quantidade de NPCs que se pretende utilizar, não é um impeditivo na escolha do modelo. Nos testes de renderização de quadros por segundo, o UD GOAP teve uma média 2% maior que o GOAP.

Os resultados de qualidade dos comportamentos apresentados, indicam o UD GOAP como melhor, por usar um contexto mais aprimorado para o julgamento das ações, consegue alcançar um maior realismo na escolha de ações. Essas informações indicam que o UD GOAP se destacou como uma melhoria em relação ao modelo original proposto.

5.1 Trabalhos futuros

Uma opção para trabalho futuro seria a implementação de novos ambientes para testes, visto que o implementado neste trabalho demonstrou certos resultados diferentes sobre os modelos. Outra possibilidade seria a expansão do escopo dos ambientes já testados, para ter mais ações, diferentes tipos de NPCs ou interação direta entre os agentes.

Pode-se também utilizar outras variações do GOAP, ou do UD GOAP, como GOAP com lógica difusa, ou Ambiência inteligente. Além de utilizar uma análise qualitativa dos resultados, com o jogo sendo testado por voluntários para determinar o realismo dos agentes.

Para obtenção mais clara de resultados de performance, tem a possibilidade de implementar e realizar testes mais robustos somente com as buscas A*, considerando o custo contra a utilidade, sem o contexto do jogo ao redor que pode afetar a distribuição de dados.

REFERÊNCIAS BIBLIOGRÁFICAS

BAILEY, C.; KATCHABAW, M. An emergent framework for realistic psychosocial behaviour in non player characters. In: *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*. ACM, 2008. p. 17–24. Disponível em: <https://doi.org/10.1145/1496984.1496988>.

BRITTON, G. Decision-making decisions: Artificial intelligence in computer games. *Available at SSRN 4513757*, 2023.

BUIJSMAN, M. *Newzoo's Global Games Market Report 2024*. Newzoo, 2024. Disponível em: <https://newzoo.com/resources/trend-reports/newzoos-global-games-market-report-2024-free-version>.

DESHPANDE, A. A.; HUANG, S. H. Simulation games in engineering education: A state-of-the-art review. *Computer applications in engineering education*, Wiley Online Library, v. 19, n. 3, p. 399–410, 2011.

DILL, K. What is game ai? *Game AI pro*, AK Peters/CRC Press, p. 3–10, 2013.

DU, H. The progress and trend of intelligent npcs in games. *Applied and Computational Engineering*, v. 133, p. 158–164, 2025.

FUTURLAB. *PowerWash Simulator*. 2022. Disponível em: <https://store.steampowered.com/app/1290000/PowerWash_Simulator/>.

GAMES, N. *Supermarket Simulator*. 2024. Disponível em: <https://store.steampowered.com/app/2670630/Supermarket_Simulator/>.

GAMES, O. *TCG Card Shop Simulator*. 2024. Disponível em: <https://store.steampowered.com/app/3070070/TCG_Card_Shop_Simulator/>.

GELONEZE, F. R.; ARIELO, F. S. Um breve análise sobre a indústria de jogos eletrônicos e os indie games. *Revista Multiplicidade*, v. 8, n. 8, 2017.

GEREVINI, V. *godot-goap[software]*. [S.l.]: Github, 2023.

Godot Engine developers. *Performance class*. 2025. Godot Engine Documentation. Accessed: December 5, 2025. Disponível em: <https://docs.godotengine.org/en/stable/classes/class_performance.html>.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, v. 4, n. 2, p. 100–107, 1968.

HUANG, X. et al. Goal-oriented action planning in partially observable stochastic domains. In: IEEE. *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*. [S.l.], 2012. v. 3, p. 1381–1385.

International Federation of the Phonographic Industry (IFPI). *Global Music Report 2025: State of the Industry*. 2025. Disponível em: <https://ifpi-website-cms.s3.eu-west-2.amazonaws.com/GMR_2025_State_of_the_Industry_Final_83665b84be.pdf>.

JACOPIN, É. Optimizing practical planning for game ai. In: *Game AI Pro 360: Guide to Architecture*. [S.l.]: CRC Press, 2019. p. 269–280.

JUNIOR, J. de J. F. S. *Otimização do tempo de execução em algoritmos de roteamento global: Explorando métricas e paralelismo*. Dissertação (Mestrado) — Universidade Federal do Rio Grande – FURG, Programa de Pós-Graduação em Computação, Rio Grande, RS, Brasil, 2019. Dissertação de Mestrado.

LONG, E. *Enhanced NPC Behaviour using Goal Oriented Action Planning*. Dissertação (Mestrado) — University of Abertay Dundee, September 2007. Disponível em: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=472acfbf6b9a1a3faaec7bf61e7f018b471272c1>.

LTD., C. O. *Cities: Skylines*. 2015. Disponível em: <https://store.steampowered.com/app/255710/Cities_Skylines/>.

MITCHELL, R. *Gower Street Revises 2025 Global Box Office Estimate Ahead of CinemaCon*. 2025. Disponível em: <https://gower.st/articles/gower-street-revises-2025-global-box-office-estimate-cinemacon/>.

ORKIN, J. Applying goal-oriented action planning to games. *AI Game Programming Wisdom 2*, Charles River Media, Inc., 2003.

PRÉVOST, G. et al. Topological planning with post-unique and unary actions. In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2023. p. 5429–5436.

RIBEIRO, G. L. F. Técnicas de inteligência artificial na criação de personagens não jogáveis: uma revisão de literatura. In: *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)*. Natal, RN: Sociedade Brasileira de Computação (SBC), 2022. p. 208–217. Disponível em: <https://sol.sbc.org.br/index.php/sbgames_estendido/article/view/23650>.

SKYLINES, C. *Cities: Skylines | Celebrating 12M Copies Sold*. 2022. YouTube. Acesso em: 7 jul. 2025. Disponível em: <https://youtu.be/e4c09dYBnGQ>.

SLOAN, C. *Drive-based utility-maximizing computer game non-player characters*. Tese (Doutorado) — Technological University Dublin, 2015.

SLOAN, C.; NAMEE, B. M.; KELLEHER, J. D. Utility-directed goal-oriented action planning: A utility-based control system for computer game agents. *Artificial Intelligence and Cognitive Science*, 2011.

SOFTWARE, S. *Euro Truck Simulator 2*. 2012. Disponível em: <https://store.steampowered.com/app/227300/Euro_Truck_Simulator_2/>.

SOFTWARE, S. *Euro Truck Simulator 2's 10th Anniversary*. 2022. Acesso em: 7 jul. 2025. Disponível em: <https://blog.scssoft.com/2022/10/euro-truck-simulator-2s-10th-anniversary.html>.

ULUDAĞLI, M. Ç.; OĞUZ, K. Non-player character decision-making in computer games. *Artificial Intelligence Review*, v. 56, n. 12, p. 14159–14191, Dec 2023. ISSN 1573-7462. Disponível em: <https://doi.org/10.1007/s10462-023-10491-7>.