UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ERIC TROFINO

INTELIGÊNCIA ARTIFICIAL VERSUS APRENDIZADO DE MÁQUINA: ANÁLISE EM UM JOGO DESENVOLVIDO EM UNITY

Bauru, São Paulo, Brasil
2024

ERIC TROFINO

INTELIGÊNCIA ARTIFICIAL VERSUS APRENDIZADO DE MÁQUINA: ANÁLISE EM UM JOGO DESENVOLVIDO EM UNITY

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus de Bauru.
Orientadora: Profa. Dra. Juliana da Costa Feitosa

Bauru, São Paulo, Brasil
2024

T843i

Trofino, Eric
INTELIGÊNCIA ARTIFICIAL VERSUS APRENDIZADO DE MÁQUINA: ANÁLISE EM UM JOGO DESENVOLVIDO EM UNITY / Eric Trofino. -- Bauru, 2024
37 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru
Orientadora: Juliana da Costa Feitosa

1. Aprendizado do computador. 2. Inteligência artificial. 3. Computer games Design. 4. Jogos eletronicos Industria. 5. Jogos eletronicos. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Eric Trofino

Inteligência Artificial versus Aprendizado de Máquina: Análise em um Jogo Desenvolvido em Unity

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus de Bauru.

Banca Examinadora

Profa. Dra. Juliana da Costa Feitosa
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Profa. Dra. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Prof. Dr. Kelton Augusto Pontara da Costa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 14 de Novembro de 2024.

Dedico este trabalho à minha mãe,
a qual lutou muito para que eu pudesse chegar aqui.

Agradecimentos

Gostaria de expressar imensa gratidão a todos que contribuíram para a realização deste trabalho.

Agradeço primeiramente à minha família, especialmente minha mãe, pelo apoio constante e pelos valores ensinados que tornaram quem eu sou. Vocês são uma fonte de motivação e guiamento, sempre acreditando em mim e me incentivando a seguir em frente, principalmente nos momentos que mais precisei.

À Juliana Pelegrini, pelo carinho, suporte e compreensão, especialmente nos dias em que a dedicação ao trabalho nos afastaram temporariamente. Sua calma, compaixão e ternura foram essenciais nos últimos 5 anos, e sou profundamente grato por tê-la caminhando ao meu lado nesta jornada.

Aos meus amigos, que me acompanham a anos e até décadas, compartilhando momentos de alegria, desenvolvimento e companheirismo. A presença de vocês foi fundamental para que eu continuasse perseverando em meus projetos e a troca de ideias não só fortalece nossa amizade, mas também gera frutos para o progresso, como este trabalho.

Aos professores, que se dedicaram a me orientar ao longo do curso e fizeram um ótimo trabalho. Vocês foram responsáveis por ampliar meus conhecimentos e rede de conexões, me oferecendo oportunidades e conselhos. Em especial, agradeço à minha orientadora, Dra. Juliana da Costa Feitosa, por sua orientação diligente e de prontidão, pelo embasamento da pesquisa, também pelo constante apoio. Sua orientação foi essencial para que eu norteasse o projeto e alcançasse os objetivos propostos.

Por fim, agradeço a todos que, de alguma forma, contribuíram para que este trabalho fosse realizado. Cada gesto de incentivo para que esta ideia se tornasse realidade foi lembrado com carinho.

Resumo

Este trabalho compara técnicas de Inteligência Artificial (IA) e Aprendizado de Máquina (ML, do inglês *Machine Learning*) aplicadas ao desenvolvimento de personagens não-jogáveis (NPCs, do inglês *Non-Playable Characters*) em jogos eletrônicos, focando no uso do aprendizado por reforço para treinar um agente em um ambiente simulado criado no Unity. Foi implementado um jogo em que o jogador controla um caçador tentando capturar um cervo, sendo este último controlado por um agente com IA tradicional ou treinado por ML. O estudo utiliza métricas de desempenho, como taxa de fuga, tempo de captura e distância média entre os dois, para avaliar a eficácia dos dois agentes. Os resultados mostraram que o agente treinado por ML apresentou melhor desempenho, fugindo por mais tempo do caçador e ocasionalmente aplicando estratégias de fuga inesperadas. A pesquisa contribui para o avanço no desenvolvimento de jogos ao explorar como o ML pode gerar comportamentos mais dinâmicos e imersivos para NPCs, possibilitando novas abordagens para desafios e interações em jogos eletrônicos.

Palavras-chave: *Machine Learning*, Inteligência Artificial, Desenvolvimento de Jogos, Jogos Digitais

Abstract

This work compares Artificial Intelligence (AI) and Machine Learning (ML) techniques applied to the development of Non-Playable Characters (NPCs) in video games, focusing on the use of reinforcement learning to train an agent in a simulated environment created in Unity. A game was implemented where the player controls a hunter trying to capture a deer, with the latter controlled by either a traditional AI or an ML-trained agent. The study uses performance metrics, such as escape rate, capture time, and average distance between the two, to evaluate the effectiveness of both agents. The results showed that the ML-trained agent performed better, evading the hunter for longer periods and occasionally employing unexpected escape strategies. This research contributes to game development advancements by exploring how ML can produce more dynamic and immersive behaviors for NPCs, enabling new approaches to challenges and interactions in video games.

Keywords: Machine Learning, Artificial Inteligence, Game Development, Digital Games

Lista de figuras

Lista de quadros

Lista de abreviaturas e siglas

ML	*Machine Learning*
IA	*Inteligência Artificial*
NPC	*Non Playable Character*
RTS	*Real Time Strategy*

Sumário

1 Introdução

Nos últimos anos, a Inteligência Artificial (IA) tem testemunhado avanços impressionantes, alinhados lado a lado com técnicas de Aprendizado de Máquina (ML, do inglês *Machine Learning*) (HALEEM, 2023). Segundo Damaceno e Vasconcelos (2018):

> A IA resume-se à capacidade das máquinas de aprender e tomar decisões com base em dados previamente programados, utilizando algoritmos complexos. A mesma pode ser subdividida em diferentes áreas, como o conceito de ML. Este é um processo contínuo no qual as máquinas aprendem com os dados de entrada fornecidos, utilizando algoritmos estruturados para elaborar saídas que resolvam problemas específicos. Essa abordagem permite que as máquinas se adaptem e evoluam à medida que recebem novos dados e situações em tempo real.

Em jogos, é comum a aplicação de IA em personagens não-jogáveis (do inglês: *Non-Playable Characters* ou NPCs), os quais simulam o comportamento e a racionalidade de jogadores reais (SILVA; RIBEIRO, 2021). A constante interação com os NPCs na maioria dos jogos modernos torna-os um ponto focal ideal para a aplicação de algoritmos de ML, já que desempenham papéis diversos, desde aliados até antagonistas (TUPE et al., 2016). Além disso, sua inteligência influencia diretamente a imersão do jogador no mundo virtual (TUPE et al., 2016).

Este trabalho apresenta uma investigação sobre a utilização de técnicas de ML para o treinamento de IA dentro de um jogo, explorando como esses métodos podem ser aplicados para melhorar a experiência do jogador e a jogabilidade. Foi utilizada a abordagem de ML chamada de aprendizado por reforço, para treinar um NPC em um ambiente simulado de um jogo desenvolvido para este propósito. Ao analisar e comparar essas técnica com outras técnicas comuns na indústria de jogos, pretende-se identificar as vantagens e limitações de cada uma em termos de desempenho, eficiência e adaptabilidade aos diferentes contextos de jogo.

Além disso, este estudo visa contribuir para o avanço do conhecimento no campo da IA aplicada a jogos eletrônicos, explorando e servindo de exemplo sobre como os desenvolvedores de jogos podem integrar efetivamente técnicas de ML para criar experiências mais dinâmicas e imersivas.

Por fim, este trabalho apresenta uma análise crítica dos resultados obtidos, destacando as contribuições significativas para a área de *Game Design* (em português, design de jogos) e *Game Development* (em português, desenvolvimento de jogos), pavimentando

possíveis caminhos para pesquisas futuras e para o desenvolvimento de jogos, em que a IA não apenas executa a mesma programação, mas também desafia e surpreende os jogadores em cada interação.

No Capítulo 2, observa-se a fundamentação teórica que conduziu o desenvolvimento deste trabalho. No Capítulo 3, são apresentadas as ferramentas utilizadas durante o desenvolvimento, detalhando as tecnologias e recursos que sustentam a implementação. No Capítulo 4, o foco é o processo de desenvolvimento do jogo, abordando as decisões técnicas e criativas que nortearam a construção do projeto. O Capítulo 5 discute a jogabilidade do jogo, explicando as mecânicas e a experiência do jogador. O Capítulo 6 traz a análise dos resultados obtidos, com uma avaliação das medidas de desempenho e sua comparação com as expectativas iniciais. Finalmente, o Capítulo 7 apresenta as conclusões gerais, destacando os principais achados, limitações e possíveis direções para trabalhos futuros.

2 Revisão Bibliográfica

A IA e o ML revolucionaram o design de jogos eletrônicos, passando de comportamentos básicos para sistemas dinâmicos e adaptáveis. Este capítulo revisa o progresso dessas tecnologias, abordando suas aplicações mais comuns, desde o controle de NPCs até a personalização da experiência do jogador. O conteúdo destaca como essas inovações ampliam a complexidade e a imersão nos jogos modernos.

2.1 Introdução à Inteligência Artificial em Jogos

A IA tem desempenhado um papel essencial no desenvolvimento de jogos eletrônicos desde suas origens. Inicialmente, a IA era utilizada para criar comportamentos simples de inimigos em jogos clássicos, tais como *Space Invaders* (1978) e *Pac-Man* (1980), nos quais os inimigos seguiam padrões previsíveis (GNANASEKARAN; FABA; AN, 2017). No entanto, com o avanço da tecnologia e o aumento das expectativas dos jogadores, as IAs começaram a evoluir para proporcionar interações mais dinâmicas e imprevisíveis, enriquecendo a experiência de jogo. Atualmente, a IA em jogos não só controla inimigos, mas também gera cenários, cria narrativas e personaliza o jogo para o comportamento do jogador (BUONGIORNO et al., 2024).

Entre os algoritmos mais utilizados historicamente em jogos, um dos exemplos mais conhecidos é o A* (A estrela), amplamente empregado em jogos para determinar o melhor caminho de um ponto a outro no ambiente (LIU, 2023). Além disso, os algoritmos baseados em árvores de decisão e máquinas de estados finitos são muito populares para controlar o comportamento dos NPCs (BLOEMBERGEN et al., 2015).

Segundo Jagdale (2021), Máquinas de Estados Finitos são uma abordagem clássica e eficaz para modelar o comportamento de NPCs em jogos, especialmente em cenários com estados e transições bem definidos. Cada estado representa uma ação ou condição, e as transições entre eles são acionadas por regras predefinidas, como iniciar um ataque ou fugir dependendo da proximidade do jogador ou da saúde do personagem. Embora sejam fáceis de implementar, estas máquinas podem se tornar complexas à medida que o número de estados e transições cresce, dificultando a manutenção. Apesar de não serem ideais para jogos dinâmicos ou ambientes amplos, as Máquinas de Estados Finitos ainda são amplamente utilizadas devido à sua simplicidade e eficiência em comportamentos previsíveis e estruturados.

Por outro lado, Árvores de Decisão oferecem uma abordagem mais flexível e escalável para lidar com comportamentos complexos de NPCs. Essa técnica organiza ações

e condições em uma estrutura hierárquica de nós, o que facilita alterações e expansões sem comprometer a lógica existente. Diferentemente das Máquinas de Estados Finitos, Árvores de Decisão permitem que os NPCs priorizem objetivos ou ajustem suas respostas com base no contexto do jogo, tornando-as ideais para ambientes mais dinâmicos. Além disso, ferramentas visuais para criação e edição tornam essa abordagem acessível a desenvolvedores menos experientes em programação. Por sua flexibilidade e modularidade, Árvores de Decisão são amplamente empregadas em cenários que exigem maior clareza e escalabilidade (SEKHAVAT, 2017).

2.2 Introdução do Machine Learning em Jogos

Com o avanço do ML, novos horizontes se abriram no design de jogos. Enquanto a IA tradicional dependia de regras e scripts predefinidos, o ML introduziu a capacidade de aprendizado a partir de dados e comportamentos passados, permitindo que os NPCs e sistemas de jogos adaptem suas ações de maneira autônoma e inteligente (AIOLLI; PALAZZI, 2008). O uso de ML em jogos começou a ganhar espaço nos anos 2000, com estudos que mostravam como agentes treinados por meio de aprendizado por reforço poderiam superar jogadores humanos em jogos de estratégia, como o Go (MARCOLINO; MATSUBARA, 2011) (SILVER; SUTTON; MÜLLER, 2008).

Um dos maiores marcos dessa revolução foi o desenvolvimento do AlphaGo, da DeepMind, que em 2016 derrotou o campeão mundial de Go, demonstrando o potencial do ML em jogos complexos que exigem planejamento e adaptação em tempo real (SILVER; SUTTON; MÜLLER, 2008). Em torno desse momento, o ML passou a ser integrado em diversos tipos de jogos, como jogos de estratégia em tempo real (do inglês: *Real Time Strategy* ou RTS), onde agentes controlados por ML podem ajustar suas estratégias de acordo com o estilo do jogador e o progresso no jogo (ONTANON et al., 2015).

2.3 Formas de Machine Learning Aplicadas a Jogos

O ML é uma área ampla, com diferentes subtipos, sendo três os principais aplicados em jogos:

- **Aprendizado Supervisionado**: Neste tipo, o algoritmo aprende a partir de um conjunto de dados rotulados, ou seja, cada entrada é associada a uma saída correta (BISHOP, 2006). Em jogos, essa técnica pode ser usada para treinar modelos baseados em exemplos de jogadas ideais ou padrões de comportamento que o jogador deve seguir;

- **Aprendizado Não Supervisionado**: Aqui, o algoritmo explora os dados sem rótulos explícitos, buscando padrões e estruturas por conta própria (BISHOP, 2006). Em jogos, pode ser aplicado para descobrir novos comportamentos de NPCs ou padrões em como os jogadores interagem com o ambiente; e
- **Aprendizado por Reforço**: Esse é talvez o mais revolucionário em jogos, onde os agentes aprendem a partir da interação com o ambiente, recebendo recompensas ou penalidades com base em suas ações (BISHOP, 2006). A DeepMind aplicou essa abordagem para treinar o AlphaGo e outros sistemas avançados, e jogos como StarCraft têm explorado fortemente essa técnica (USUNIER et al., 2017). Recentemente, essa abordagem tem sido combinada com redes neurais profundas, formando o *Deep Reinforcement Learning*, que permite o aprendizado em ambientes mais complexos e de alta dimensionalidade, como nos jogos tridimensionais e em tempo real (BLOEMBERGEN et al., 2015).

Hoje em dia, o ML está cada vez mais presente em diferentes camadas do design de jogos (YANNAKAKIS; TOGELIUS, 2018). Algoritmos de ML são usados para criar comportamentos dinâmicos de NPCs (USUNIER et al., 2017), ajustar a dificuldade de acordo com o nível de habilidade do jogador (MORTAZAVI; MORADI; VAHABIE, 2024), e até mesmo gerar conteúdo proceduralmente, como mapas e histórias (ARAÚJO; ARANHA; MADEIRA, 2018).

2.4 Considerações Finais

A evolução de ambos IA e ML em jogos eletrônicos não só expandiu as possibilidades do design de jogos, mas também trouxe desafios e oportunidades para a indústria. O uso dessas tecnologias tem permitido que jogos sejam mais dinâmicos, adaptáveis e personalizados, elevando a experiência do jogador a novos patamares. Nos próximos anos, espera-se que o ML continue a desempenhar um papel fundamental na criação de jogos ainda mais inteligentes e envolventes.

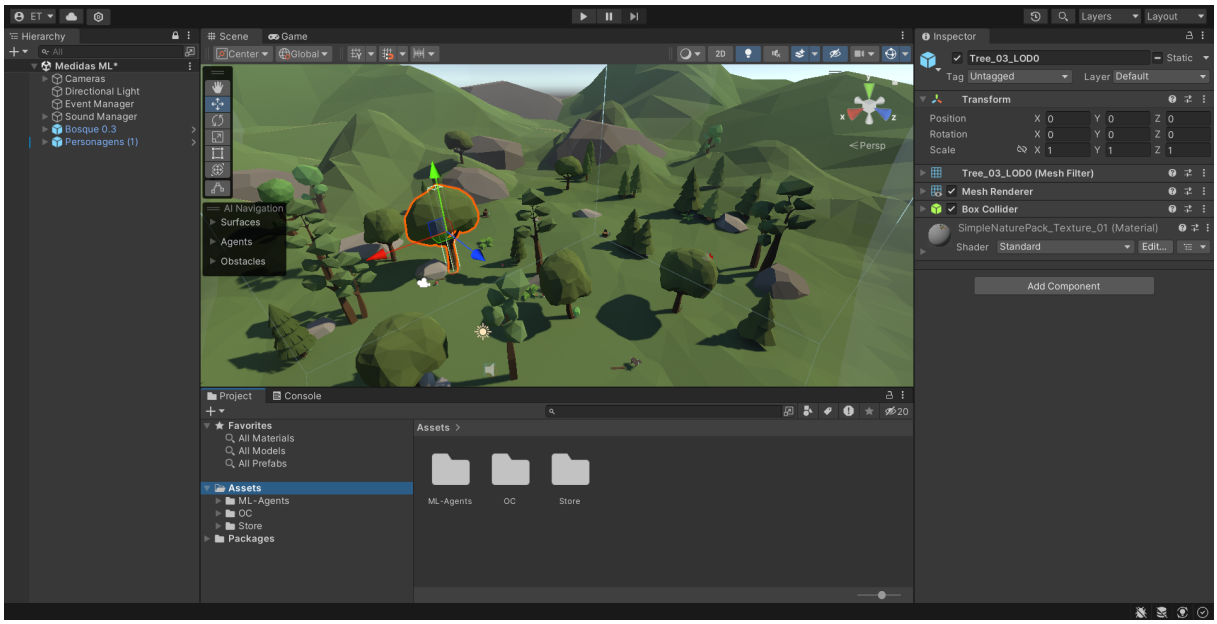
3 Ferramentas Utilizadas

Este capítulo apresenta as ferramentas empregadas no projeto, explicando um pouco sobre suas funcionalidades e citando suas qualidades.

3.1 Introdução ao Unity

O Unity[1] é uma das plataformas de desenvolvimento de jogos mais populares, amplamente utilizada por desenvolvedores devido à sua flexibilidade e capacidades multiplataforma. Com suporte a diversas linguagens, como C#, e uma vasta biblioteca de ferramentas, o Unity permite a criação de ambientes 2D e 3D complexos (FOXMAN, 2019). Devido à sua versatilidade, o Unity foi escolhido como a principal ferramenta para o desenvolvimento deste projeto, combinando facilidade de uso e poderosas funcionalidades de IA (JULIANI et al., 2020). Sua interface pode ser vista na Figura 1.

Figura 1 – Interface do Unity



Fonte: Elaborada pelo autor.

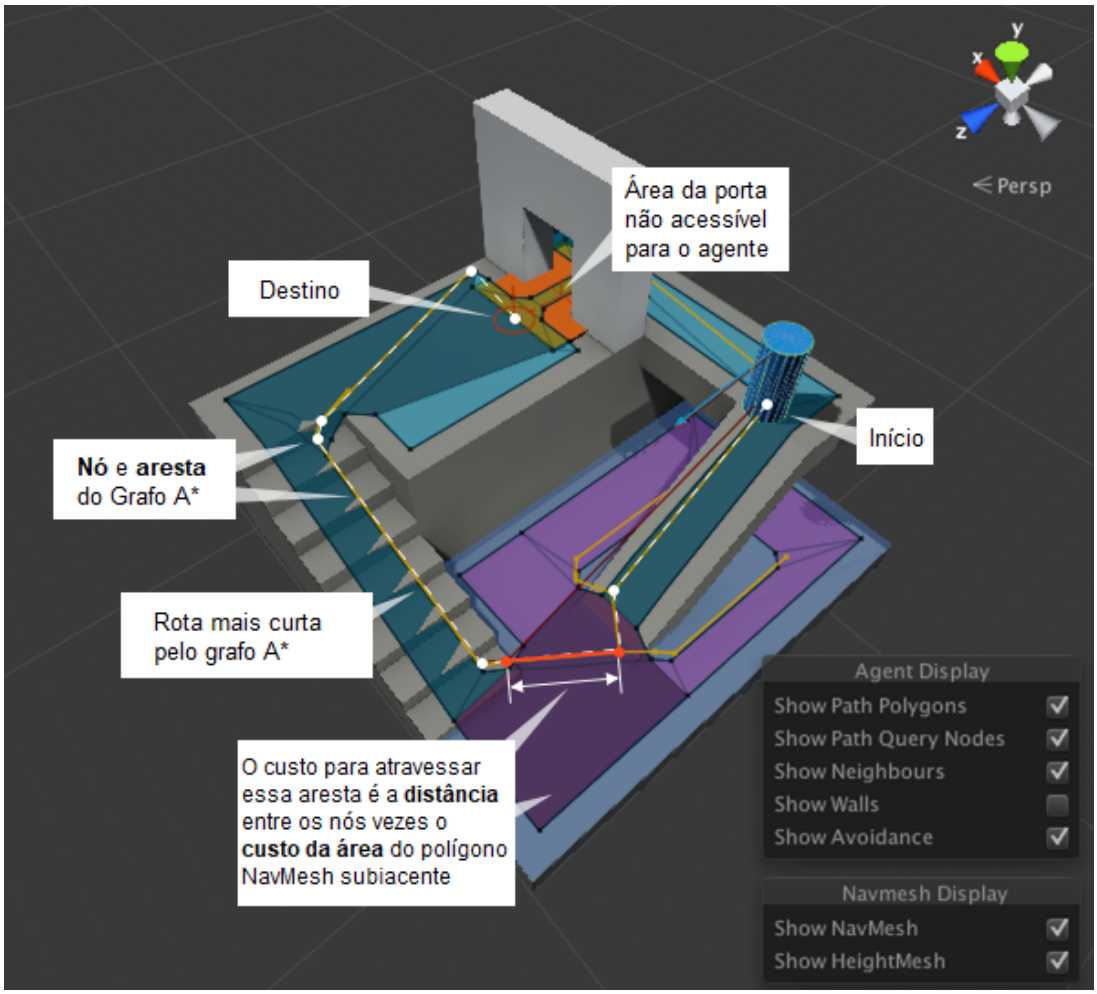
Entre suas ferramentas avançadas estão o AI Navigation e o ML-Agents Toolkit, que permitem a criação de comportamentos de IA e o treinamento de agentes inteligentes através de algoritmos de ML.

[1] https://unity.com/pt, Acesso em: 24 out. de 2024

3.2 AI Navigation

O AI Navigation[2], também conhecido como NavMesh (Navegação de Malha), é uma das ferramentas mais robustas do Unity para o controle de movimento de agentes no ambiente de jogo. Ele permite a criação automática de superfícies de navegação (NavMeshes) que delimitam onde os personagens podem se mover. Ótimo para a criação de NPCs e personagens controlados por IA que precisam seguir rotas complexas ou evitar obstáculos no cenário. O método utilizado pelo AI Navigation é o algoritmo de busca A* (A estrela), que calcula o caminho mais curto entre dois pontos com base nos nós do NavMesh, levando em conta a topologia do terreno e evitando colisões (LIU, 2023). O algoritmo A* utilizado pela ferramenta pode ser visualizado na Figura 2.

Figura 2 – Demonstração do NavMesh Agent



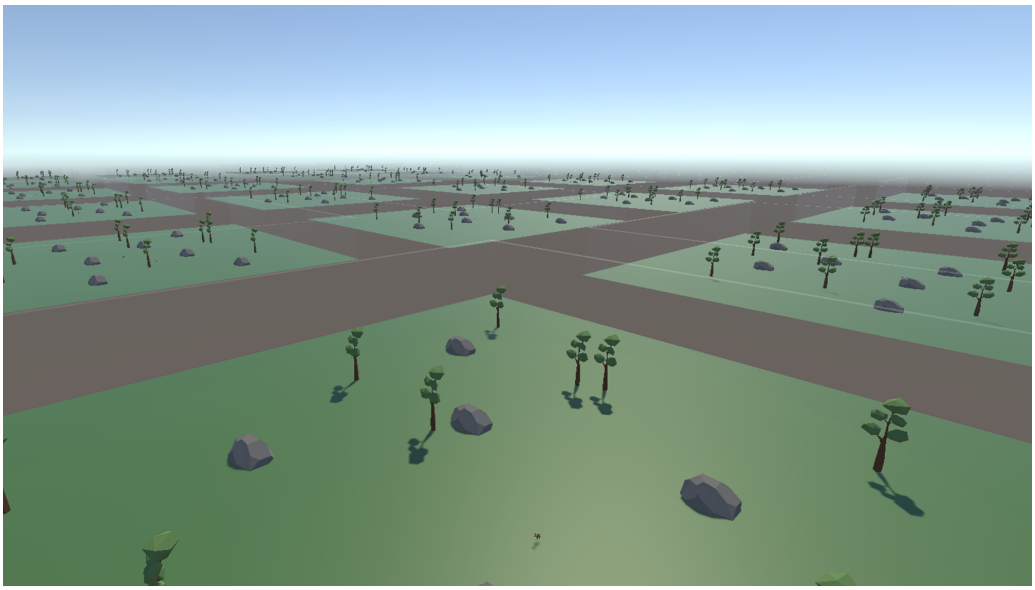
Fonte: Documentação AI Navigation[2]

[2] https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/index.html, Acesso em: 18 out. de 2024

3.3 ML-Agents Toolkit

O ML-Agents Toolkit[3] é uma extensão poderosa do Unity que permite a integração de algoritmos de ML, incluindo aprendizado por reforço e aprendizado supervisionado, para o treinamento de agentes inteligentes. Essa ferramenta possibilita a criação de cenários onde os agentes aprendem com base em suas interações com o ambiente, melhorando progressivamente suas ações através de recompensas e punições. Um ambiente simplificado de treinamento pode ser conferido na Figura 3.

Figura 3 – Treinamento com ML Agents



Fonte: Elaborada pelo autor.

3.4 Considerações Finais

A integração das ferramentas Unity, ML-Agents e AI Navigation demonstrou-se altamente eficaz para a criação de ambientes simulados e o treinamento de agentes inteligentes. Juntas, elas oferecem uma combinação poderosa para o desenvolvimento de movimentos realistas e comportamentos complexos em simulações. Trabalhos anteriores, como o de Juhola (2019), já utilizaram essas ferramentas com sucesso para abordar desafios como navegação autônoma e aprendizado por reforço, destacando sua versatilidade em aplicações de IA.

O uso do Unity como ferramenta principal deste trabalho permitiu não só o desenvolvimento ágil do ambiente de jogo, mas também a implementação eficiente de IA e ML. O AI Navigation garantiu a movimentação precisa dos personagens, enquanto o ML-Agents facilitou o treinamento de comportamentos complexos para os agentes controlados

[3] https://github.com/Unity-Technologies/ml-agents/releases/tag/release_10, Acesso em: 18 out. de 2024

por ML, resultando em uma comparação sólida entre IA tradicional e técnicas modernas de ML.

4 Metodologia

O desenvolvimento do jogo foi orientado por decisões de design voltadas para otimizar o desempenho dos agentes e a funcionalidade dos sistemas de IA e ML. A estética, os controles e a construção do cenário foram planejados para simplificar a implementação e criar um ambiente de testes eficiente, permitindo a avaliação do comportamento dos agentes e o treinamento do agente de ML.

4.1 Estética Visual e Simplificação

A fim de priorizar a simplicidade visual, a estética escolhida para o jogo utiliza *assets low poly*, exemplificado na Figura 4, comumente empregados em jogos independentes. Essa abordagem acelerou o desenvolvimento, focando nas interações dentro do ambiente. Azoubel et al. (2016) destaca que a simplicidade no design visual é essencial para evitar distrações que possam comprometer a jogabilidade.

Figura 4 – Comparação entre Low Poly e Realismo

Fonte: 3Dstudio, 2022

4.2 Esquema de Controles

O esquema de controles adotado foi o padrão W, A, S, D, amplamente utilizado em jogos de terceira pessoa para a movimentação. Teclas livres foram reservadas para futuras mecânicas, garantindo que o sistema fosse expansível e adaptável a modificações.

4.3 Construção do Cenário

O cenário do jogo, um bosque com obstáculos, reflete a necessidade de um espaço desafiador, mas mantendo uma estrutura clara para o treinamento do agente de ML. A decisão de evitar variações grandes de altitude baseou-se na alta probabilidade de erros e na complexidade desnecessária que isso poderia trazer. Essa abordagem simplifica a movimentação dos agentes, permitindo foco nas interações.

4.4 Mecânica de Jogo e Sistema de Pontuação

A mecânica de jogo se assemelha a um "pega-pega", cujo jogador deve controlar o caçador para que ele encoste em um cervo, que está fugindo, para capturá-lo. Como o cervo acelera e tem uma velocidade máxima maior que a do caçador, o jogador deve usar suas habilidades estratégicas para encurralar o cervo e capturá-lo.

O sistema de pontuação baseia-se em um cronômetro que inicia junto com o jogo. A pontuação é determinada pelo tempo restante no momento que o caçador captura o cervo, incentivando eficiência no movimento e na tomada de decisões. Essa estrutura permite uma avaliação clara do desempenho dos agentes.

4.5 Desenvolvimento das IAs

Devido ao objetivo principal do jogo ser capturar um NPC, o NPC deve conseguir fugir, portanto, é necessário que sua movimentação seja eficiente, não colidindo com obstáculos e andando numa direção que aumente essa distância. Entretanto, esta movimentação para fuga requer não só que o agente encontre um caminho ótimo para um ponto, mas também determine para qual ponto ele deve ir, trazendo um grau de dificuldade mais alto para a determinação do movimento do NPC. Na movimentação do caçador durante a etapa de treino e medição e de ambos os agentes, as destinações definidas são alcançadas pelo AI Navigation.

4.5.1 IA Tradicional

A programação do agente de IA Tradicional, também referido no trabalho por "agente IA", segue a arquitetura de Árvore de Decisão, contendo estrutura semelhante a um fluxograma, na qual cada nó representa um teste em um atributo, e cada folha representa uma decisão final (MÜLLER; BITTENCOURT, 2021). Esta arquitetura foi escolhida devido a sua flexibilidade na tomada de decisões e popularidade na indústria.

A programação de escolha do destino usando árvore de decisão normalmente é feita criando um vetor oposto a direção do objeto no qual se quer fugir. Porém, esse ponto de destino pode estar fora do mapa, obstruído por obstáculos, ou simplesmente ser inacessível. Devido a esta limitação, o ponto futuro deve ser testado antes do comando de movimento ser executado e, caso não esteja disponível, outro ponto deve ser testado até encontrar um disponível. O código do algoritmo pode ser visto na Figura 5.

Em suma, este agente foi programado manualmente e não aprendeu com as experiências ao longo do desenvolvimento. Ele escolhe, em tempo real, uma rota não obstruída com a menor rotação possível de seu corpo para percorrê-la.

4.5.2 Machine Learning

A programação do agente com ML, também referido no trabalho por "agente ML", foi feita utilizando recompensas e punições, ou seja, indicando quais ações são benéficas ou não para seu objetivo. O ML-Agents foi utilizado para treinar o comportamento e o método de treinamento empregado foi o aprendizado por reforço profundo.

A indicação de quais destinos são benéficos para seu objetivo foi ensinada recompensando o agente por ficar longe, ou aumentar a distância entre o cervo e o caçador. A punição ocorre por se aproximar do caçador ou obstáculos e também por ser capturado. Parte do código pode ser visto na Figura 6.

Em suma, este agente foi treinado com aprendizado profundo por reforço, melhorando após cada etapa de treino. Ele, além de tentar aumentar a sua distância do caçador, também escolhe rotas que fazem o caçador ter de rotacionar muito para capturar o cervo e, ocasionalmente, se aproxima do caçador para ter uma rota mais favorável futuramente.

Segundo a Figura 7, na qual representa a última etapa do treinamento do agente, é observado que o treinamento foi um sucesso. A recompensa acumulada representa a soma das recompensas para cada partida. Esse valor deve aumentar num treinamento bem sucedido. A perda na função de valor demonstra o quão bem o modelo pode prever os resultados. Portanto, este gráfico indica o erro na previsão de valor para cada partida. A perda deve decrescer ao longo de um treinamento bem sucedido.

Figura 5 – Algoritmo de Fuga

```
//calculo do movimento
dirAway = transform.position - playerTransform.position;
newPos = transform.position + dirAway;

//testa se o caminho escolhido está livre
if(Physics.Raycast(transform.position, newPos, out RaycastHit hit, wallSafeDistance))
{
    //se há um obstáculo no caminho, a direção está bloqueada
    if (hit.transform.CompareTag("Wall")) dirBlocked = true;

    else
    {
        dirBlocked = false;
        agent.SetDestination(newPos);
        return; //se o caminho não está obstruido por um obstáculo, não é necessario calcular curva
    }
}

else
{
    dirBlocked = false;
    agent.SetDestination(newPos);
    return; //se o caminho está livre, não é necessario calcular curva
}

//calculo da curva
//escolhe aleatoriamente se a próxima curva iniciara o calculo para direita ou esquerda
if (dirBlocked && count == 1) turnLeft = System.Convert.ToBoolean(Random.Range(0,2));

//calcula a curva
if (!turnLeft) possiblePos = Quaternion.Euler(0,minTurn*count,0)*newPos;
if (turnLeft) possiblePos = Quaternion.Euler(0,-minTurn*count,0)*newPos;

//testa se o possível caminho esta obstruido
if(Physics.Raycast(transform.position, possiblePos, out RaycastHit hit2, wallSafeDistance))
{
    if (!hit2.transform.CompareTag("Wall")) //se não estiver obstruido por um obstaculo, está limpo
    {
        CurvaFinalizada();
    }

    else
        count += 1; //aumenta a contagem e no proximo frame tentara uma curva mais fechada
}

else CurvaFinalizada();

agent.SetDestination(newPos); //se não estiver obstruido, está limpo
```

Fonte: Elaborada pelo autor.

4.6 Considerações Finais

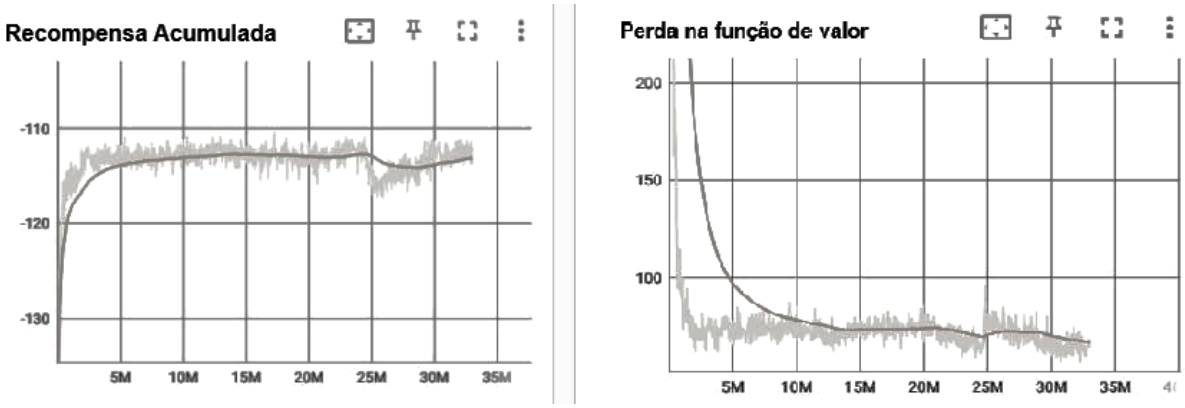
O desenvolvimento do jogo seguiu uma abordagem minimalista, priorizando clareza nas interações e consistência no ambiente de treinamento. As decisões de design garantiram foco no comportamento dos agentes, minimizando interferências externas.

Figura 6 – Parte do código de ML

```
public override void OnActionReceived(ActionBuffers actions)
{
    //if (Vector3.Distance(playerTransform.localPosition, transform.localPosition) < hearingRange)
    float horizontalMovement = actions.ContinuousActions[0];
    float verticalMovement = actions.ContinuousActions[1];
    moveDirection = new UnityEngine.Vector3(horizontalMovement, 0, verticalMovement);
    agent.SetDestination(transform.position+moveDirection*3);

    // Atribua recompensas com base na distância
    if (distanceToPlayer > previousDistanceToPlayer)
    {
        SetReward(1f); // Recompensa por se afastar
    }
    else if (distanceToPlayer <= previousDistanceToPlayer)
    {
        SetReward(-1f); // Penalidade por se aproximar
    }

    if (distanceToPlayer > 5f)
    {
        SetReward(0.1f);
    }

    previousDistanceToPlayer = distanceToPlayer; // Atualize a distância anterior
}
```

Fonte: Elaborada pelo autor.

Figura 7 – Progresso do treinamento



Fonte: Elaborado pelo autor.

5 Jogabilidade

A jogabilidade deste jogo combina elementos de estratégia e ação, criando uma experiência focada em testes de comportamento. O jogador controla o caçador, enquanto o cervo é controlado por um dos dois modelos de agente: um utilizando IA simples na forma de árvores de decisão e outro treinado por ML. O principal objetivo do jogo é capturar o cervo, que tenta fugir quando o caçador se aproxima, como se o animal ouvisse o som dos passos.

5.1 Objetivos do Jogo

O objetivo direto é capturar o cervo no menor tempo possível e, para isso, o jogador deve utilizar suas habilidades estratégicas. O objetivo indireto é deixar o jogador perceber a diferença na dificuldade entre capturar o cervo com o agente IA e o cervo com o agente ML. A dificuldade de captura dos dois modelos é observada pelo desempenho deles em fugir do caçador, permitindo avaliar a eficácia das abordagens.

5.2 Mecânicas de Jogo

O jogador controla o caçador com comandos de teclado e mouse. O caçador pode mover-se livremente pelo ambiente, como visto na Figura 8, utilizando as teclas mais comuns: W, A, S e D. A detecção do cervo ocorre por meio do controle da câmera sobre a cabeça do caçador, numa perspectiva de terceira pessoa, pelo movimento do mouse. O cervo toma decisões baseadas em se afastar do caçador o mais rápido possível.

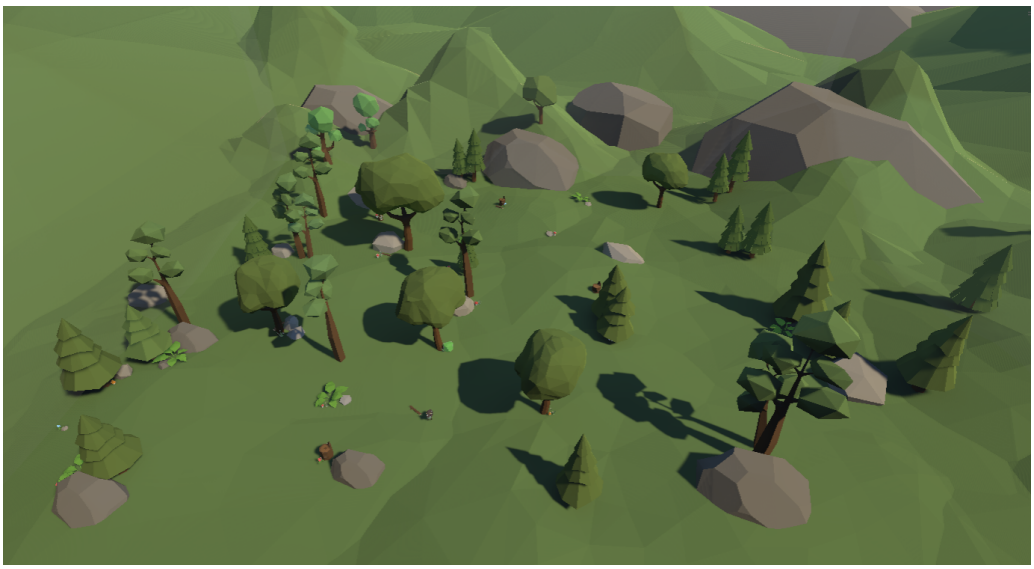
Figura 8 – Caçador se movendo e cervo ao fundo

Fonte: Elaborada pelo autor.

5.3 Ambiente de Jogo

O jogo se passa em um pequeno bosque que inclui árvores, pedras e decorações, como cogumelos, gravetos e um plano de fundo montanhoso, além de pequenas variações no relevo do solo. O ambiente possui áreas com folhagem e outros objetos que obstruem a visão, criando esconderijos onde o cervo pode, ao acaso, se esconder. Além disso, existem passagens estreitas onde o cervo consegue se mover com mais facilidade do que o caçador, proporcionando oportunidades de fuga. O cenário pode ser observado na Figura 9.

Figura 9 – Vista de pássaro do cenário



Fonte: Elaborada pelo autor.

5.4 *Feedback*

O *feedback* fornecido ao jogador é quanto tempo demorou para o caçador capturar o cervo. Essa métrica permite que o jogador avalie seu desempenho e refine suas estratégias em tentativas futuras. O tempo pode ser visto na tela final do jogo, representada na Figura 10, acessada imediatamente após o tempo acabar ou o cervo ser capturado.

Figura 10 – Tela Final

Fonte: Elaborada pelo autor.

5.5 Considerações Finais

A jogabilidade do jogo é simples e elementar, servindo apenas ao propósito de realizar testes. A combinação de diferentes abordagens de IA em um ambiente controlado proporciona uma base para a análise e avaliação de estratégias de captura e fuga.

6 Resultados

Foram realizadas duas simulações distintas: a primeira avaliando o desempenho do agente IA e a segunda, do agente ML. Cada simulação consistiu em dezenas de milhares de partidas, executadas até o evento de captura do cervo ou até sua fuga ao final do tempo de cinco minutos.

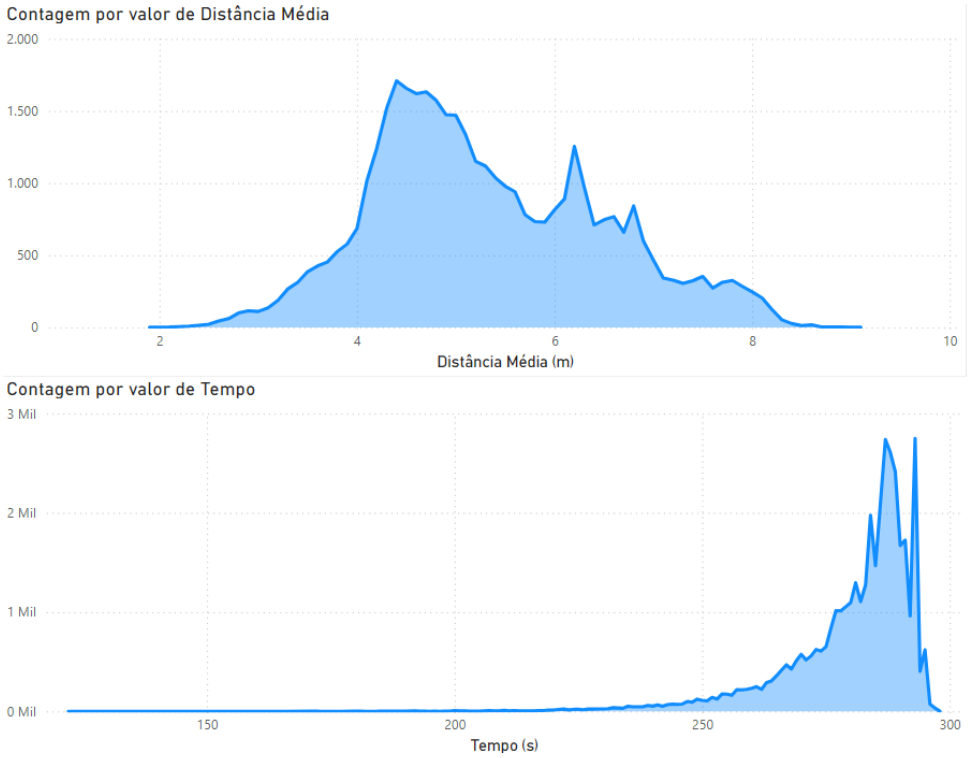
As medições foram conduzidas com base em três principais métricas de desempenho: a porcentagem de partidas em que o cervo conseguiu escapar por simulação (eficácia), o tempo restante nas partidas em que ele foi capturado (eficiência) e a distância média mantida entre o cervo e o caçador em cada partida.

O tempo restante varia de 300 segundos a zero segundos, como o cronômetro do jogo é iniciado com cinco minutos (300s) e decresce ao longo da partida, um valor menor indica que o cervo fugiu por mais tempo. Ou seja, um desempenho melhor e um valor de zero segundos indica que o cervo completou a fuga durante cinco minutos e venceu a partida.

A distância média varia de dez metros até próximo de zero metros. O agente é ativado ao perceber o caçador numa distância igual ou menor a dez metros e então começa a medir a distância entre eles a cada *frame*, ao fim da partida é calculado a média entre essas distâncias. Uma distância média maior ao fim da partida geralmente indica um desempenho melhor.

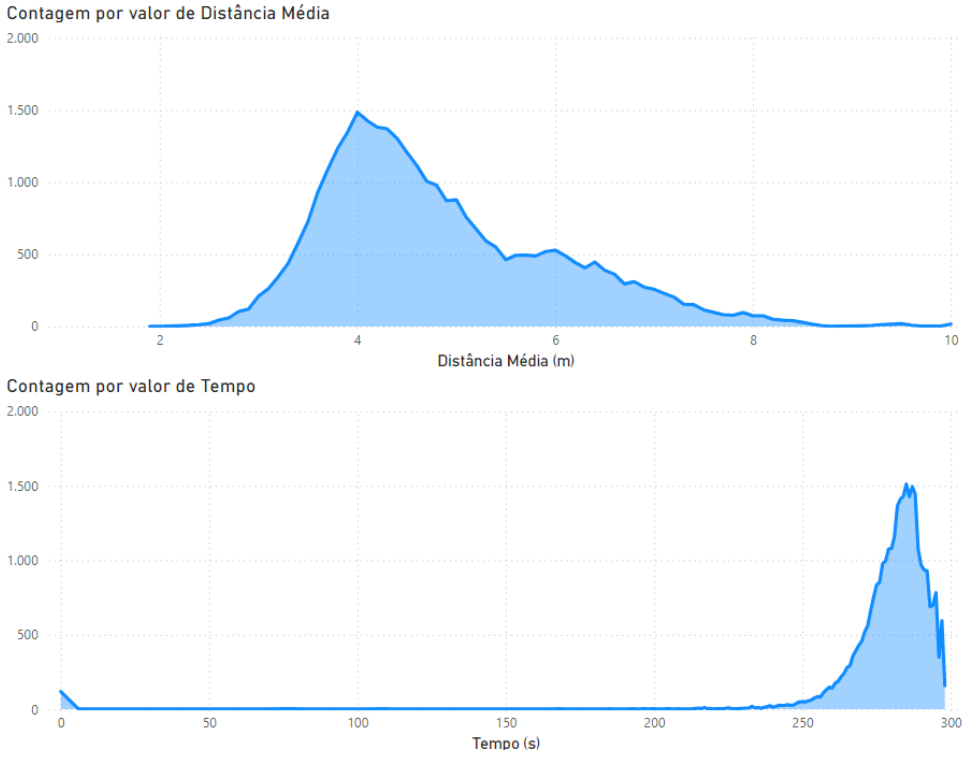
As Figuras 11 e 12 apresentam gráficos da contagem de vezes que um valor de distância média ou tempo foi registrado. A contagem está no eixo das ordenadas e cada valor está no eixo das abscissas. A Figura 11 apresenta os dados da simulação do agente IA, enquanto a Figura 12 apresenta os dados da simulação do agente ML

Figura 11 – Gráficos das medidas do agente IA



Fonte: Elaborada pelo autor.

Figura 12 – Gráficos das medidas do agente ML



Fonte: Elaborada pelo autor.

6.1 Comparação entre os agentes

A seguir, serão comparados os resultados entre os dois agentes pela Tabela 1. Destacando para cada medida, qual agente foi superior. Os dados foram arredondados para duas casas decimais.

Quadro 1 – Estatísticas de Desempenho dos Agentes

	Agente IA	Agente ML
Número de partidas	40.450	31.457
Número de fugas	0	119
Porcentagem de fugas	0%	0,38%
Estatísticas do Tempo Restante		
Pior desempenho	298,00s	298,00s
Melhor desempenho	122,00s	0,00s
Média	279,99s	279,25s
Mediana	284,00s	283,00s
Moda	293,00s	285,00s
Variação	181,87s	495,11s
Desvio padrão	13,49s	22,25s
Estatísticas da Distância Média		
Menor registrada	1,90m	1,90m
Maior registrada	9,10m	10,00m
Média	5,32m	4,83m
Mediana	5,10m	4,60m
Moda	4,40m	4,00m
Variação	1,37m	1,39m
Desvio padrão	1,17m	1,18m

Fonte: Elaborada pelo autor.

- **Eficácia**: Considerando a alta dificuldade do cervo em vencer uma partida, a IA treinada por ML demonstrou uma taxa de vitória ligeiramente superior. Conseguindo escapar do caçador em uma porcentagem maior de partidas do que a IA tradicional. Esse resultado indica que a IA por ML desenvolveu uma capacidade mais avançada para atingir o objetivo final do cervo, que é sobreviver por cinco minutos sem ser capturado.

- **Eficiência**: Nos casos em que o cervo treinado por ML foi capturado, o tempo restante para o término da partida foi, em média, menor ao obtido pela IA tradicional. Esse comportamento reflete uma maior eficiência do agente ML em retardar a captura, mesmo nas ocasiões em que não conseguiu sobreviver até o final dos cinco minutos. Porém a imensa variação observada neste modelo, indica uma instabilidade em seu comportamento, variando de um desempenho muito ruim a um perfeito.

- **Distância Média**: O agente ML manteve, em média, uma distância menor do caçador ao longo das partidas em comparação com a IA tradicional. Isso pode ser devido a um comportamento do agente ML, em que ele se aproxima do caçador propositalmente para ultrapassá-lo e forçá-lo a virar 180º para continuar a perseguição. O agente ML também manteve a distância máxima possível ao longo de toda a partida em 0,05% da simulação, ou seja, 17 jogos.

6.2 Descobertas

Um resultado interessante foi que, em 119 partidas, o agente ML conseguiu sobreviver por cinco minutos completos, indicando uma capacidade real de cumprir com sucesso o objetivo de fuga. O resultado esperado era que nenhum dos modelos conseguisse fugir por 5 minutos, devido a alta dificuldade desta tarefa designada para um agente com programação simples e outro com um período de treinamento relativamente curto. Essa performance revela que, embora o cervo não consiga escapar em todas as partidas, há potencial para que ele atinja o objetivo consistentemente com mais tempo de treinamento. Como o cervo possui velocidade superior à do caçador, sua captura nas partidas se deve unicamente a decisões não ótimas tomadas pela IA, o que poderia ser refinado com um período de treinamento mais longo, fazendo com que a fuga completa fosse o padrão e não a exceção.

A métrica de distância média é relevante, pois mostra que a IA treinada por ML não só executa bem uma técnica esperada, mas também aprende a realizar novas técnicas. Por exemplo, a técnica da ultrapassagem, que tira vantagem de sua maior velocidade de rotação em comparação com o caçador, reduzindo a distância entre eles para no futuro conseguir aumentá-la, conseguindo assim, tornar sua fuga mais duradoura. Por fim, os resultados deste estudo apontam uma superioridade do agente treinado por ML em comparação com o agente baseado em árvore de decisão para controlar o cervo.

7 Conclusão

Este trabalho propôs comparar a utilização de técnicas de ML para o treinamento de IA dentro de um jogo com métodos mais difundidos, focando na aplicação de aprendizado por reforço para desenvolver um NPC em um ambiente simulado. A pesquisa teve como objetivo progredir a experiência do jogador e a jogabilidade no setor de jogos digitais, comparando o desempenho de técnicas de ML com abordagens tradicionais da indústria.

Os resultados obtidos foram surpreendentes, uma vez que, em 119 partidas, o agente treinado por ML conseguiu sobreviver por cinco minutos completos, superando as expectativas iniciais de que ambos os modelos dificilmente conseguiriam alcançar esse objetivo. Essa performance destaca não apenas a eficácia do aprendizado por reforço, mas também o potencial que o cervo possui para escapar consistentemente, dado período de treinamento mais longo. A captura do cervo em diversas partidas pode ser atribuída a decisões não ótimas da IA, que poderiam ser aprimoradas através de treinamento adicional.

Em termos de contribuições, este estudo fornece realizações significativas sobre como as técnicas de ML podem ser integradas ao desenvolvimento de jogos. A pesquisa demonstra que a IA não precisa ser restrita a comportamentos pré-programados, mas pode evoluir e se adaptar, criando experiências de jogo mais dinâmicas e envolventes. Essa abordagem pode inspirar desenvolvedores a explorar novos caminhos na criação de NPCs que desafiem e surpreendam os jogadores a cada interação.

Embora o estudo tenha gerado resultados positivos, algumas limitações foram identificadas. Se houvesse mais tempo e recursos, uma gravação das simulações permitiria análises mais detalhadas das partidas que apresentaram comportamentos interessantes. Além disso, a inclusão de mecânicas adicionais no jogo proporcionaria experiências mais intrigantes, quando fosse observado o agente aprendendo a utilizar a mecânica eficientemente ou inovadoramente. Ademais, seria interessante realizar uma análise qualitativa dos resultados gerados pela IA e pela ML, permitindo uma compreensão mais aprofundada das estratégias e decisões adotadas pelos agentes durante as simulações. Sequencialmente, seria ótimo transformar o ambiente de jogo para uma experiência interativa semelhante a produtos comercializados, pois, além de proporcionar um ambiente de medições, tornaria mais divertido e envolvente jogá-lo, atraindo um público mais amplo. Diante disso, recomenda-se que futuras pesquisas se concentrem em superar as limitações mencionadas, explorando melhorias no treinamento de agentes de ML e incluindo mais mecânicas de jogo.

As descobertas deste estudo destacam a capacidade surpreendente dos agentes de ML, mas também ressaltam o desafio envolvido em sua programação. Esses agentes,

embora promissores, demandam um esforço significativo para configuração e ajuste, além de haver a possibilidade de explorarem rapidamente vulnerabilidades tanto na simulação quanto no próprio jogo.

Em suma, este trabalho não apenas contribui para o entendimento das aplicações de ML em jogos eletrônicos, mas também abre caminho para novas pesquisas e inovações que poderão moldar o futuro do desenvolvimento de jogos.

Referências

AIOLLI, F.; PALAZZI, C. E. Enhancing artificial intelligence in games by learning the opponent's playing style. In: *International Federation for Information Processing Digital Library*. Padova: [s.n.], 2008. v. 279. Acesso em: 5 abr. 2024. Disponível em: <https://www.researchgate.net/publication/29825872_Enhancing_Artificial_Intelligence_in_Games_by_Learning_the_Opponent's_Playing_Style>.

ARAÚJO, W.; ARANHA, E.; MADEIRA, C. Geração procedural de conteúdo aplicada a jogos digitais educacionais. In: _____. [S.l.: s.n.], 2018. p. 1–20. ISBN 9788576694625.

AZOUBEL, P. B.; PINA, L. V.; DEMAISON, A. L.; CAMPOS, L. F. d. A. Game design e hci: A importância de estudos e pesquisa no processo de desenvolvimento de jogos digitais. In: *Anais do 12º Congresso Brasileiro de Pesquisa e Desenvolvimento em Design*. São Paulo: Blucher, 2016. (Blucher Design Proceedings, 2), p. 4410–4418. ISSN 2318-6968.

BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738.

BLOEMBERGEN, D.; TUYLS, K.; HENNES, D.; KAISERS, M. Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research*, v. 53, p. 659–697, 08 2015.

BUONGIORNO, S.; KLINKERT, L. J.; CHAWLA, T.; ZHUANG, Z.; CLARK, C. *PANGeA: Procedural Artificial Narrative using Generative AI for Turn-Based Video Games*. 2024. Acesso em: 18 out. de 2024. Disponível em: <https://arxiv.org/abs/2404.19721>.

DAMACENO, S. S.; VASCONCELOS, R. O. Inteligência artificial: Uma breve abordagem sobre seu conceito real e o conhecimento popular. *Caderno de Graduação - Ciências Exatas e Tecnológicas - UNIT - SERGIPE*, v. 5, n. 1, p. 11, 2018. Acesso em: 5 abr. 2024. Disponível em: <https://periodicos.grupotiradentes.com/cadernoexatas/article/view/5729>.

FOXMAN, M. United we stand: Platforms, tools and innovation with the unity game engine. *Social Media + Society*, v. 5, n. 4, p. 2056305119880177, 2019. Acesso em: 18 out. de 2024. Disponível em: <https://doi.org/10.1177/2056305119880177>.

GNANASEKARAN, A.; FABA, J. F.; AN, J. *Reinforcement Learning in Pacman*. 2017. CS229: Machine Learning, Stanford University.

HALEEM, M. S. *Advances in Artificial Intelligence, Machine Learning and Deep Learning Applications*. Coventry: Eletronics, 2023. Acesso em: 5 abr. 2024. Disponível em: <https://www.mdpi.com/2079-9292/12/18/3780>.

JAGDALE, D. Finite state machine in game development. *International Journal of Advanced Research in Science Communication and Technology*, Naksh Solutions, p. 384–390, oct 2021. Acesso em: 21 nov. de 2024. Disponível em: <https://doi.org/10.48175/IJARSCT-2062>.

JUHOLA, O.-P. Creating self-learning ai using unity machine learning. 2019. Acesso em: 21 nov. de 2024. Disponível em: <https://www.theseus.fi/bitstream/handle/10024/261509/Juhola_Olli-Pekka.pdf?sequence=2>.

JULIANI, A.; BERGES, V.-P.; TENG, E.; COHEN, A.; HARPER, J.; ELION, C.; GOY, C.; GAO, Y.; HENRY, H.; MATTAR, M.; LANGE, D. *Unity: A General Platform for Intelligent Agents*. 2020. Acesso em: 01 nov. de 2024. Disponível em: <https://arxiv.org/abs/1809.02627>.

LIU, D. Research of the path finding algorithm a* in video games. *Highlights in Science, Engineering and Technology*, v. 39, p. 763–768, 04 2023.

MARCOLINO, L. S.; MATSUBARA, H. Multi-agent monte carlo go. In: *Multi-agent Monte Carlo Go.* [S.l.: s.n.], 2011. v. 1, p. 21–28.

MORTAZAVI, F.; MORADI, H.; VAHABIE, A. Dynamic difficulty adjustment approaches in video games: a systematic literature review. *Multimedia Tools and Applications*, v. 83, p. 83227–83274, 03 2024.

MÜLLER, A. A.; BITTENCOURT, J. R. Ud3 - unity decision tree integrator: Uma extensão de Árvores de decisão para unity. In: *2021 IEEE Latin America Conference on Computing, Communications and Electronics (LACCE)*. [S.l.]: IEEE, 2021. p. 65–70.

ONTANON, S.; SYNNAEVE, G.; URIARTE, A.; RICHOUX, F.; CHURCHILL, D.; PREUSS, M. Rts ai problems and techniques. In: _____. [S.l.: s.n.], 2015. ISBN 978-3-319-08234-9.

SEKHAVAT, Y. A. Behavior trees for computer games. *International Journal on Artificial Intelligence Tools*, v. 26, 01 2017. Acesso em: 21 nov. de 2024. Disponível em: <https://doi.org/10.1142/S0218213017300010>.

SILVA, G. A. d.; RIBEIRO, M. W. d. S. Desenvolvimento de npcs com comportamentos engajados. In: *Trilha de Indústria – Artigos Completos – Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGAMES)*. Porto Alegre: Sociedade Brasileira de Computação, 2021. p. 755–762. Acesso em: 5 abr. 2024. Disponível em: <https://sol.sbc.org.br/index.php/sbgames_estendido/article/view/21511/21335>.

SILVER, D.; SUTTON, R. S.; MÜLLER, M. Sample-based learning and search with permanent and transient memories. In: *Annual International Conference on Machine Learning*. [S.l.: s.n.], 2008.

TIIGIMÄGI, S. *What is Low Poly and High Poly Modeling?* 2022. Acesso em: 17 out. 2024. Disponível em: <https://3dstudio.co/low-and-high-poly-modeling/>.

TUPE, K.; SINGH, P.; PARMAR, S.; PANDEY, T. Ai & npc in games. *International Journal of Computer Engineering in Research Trends*, India, v. 3, n. 3, p. 129–133, 2016. Acesso em: 5 abr. 2024. Disponível em: <https://ijcert.org/ems/ijcert_papers/V3I307.pdf>.

USUNIER, N.; SYNNAEVE, G.; LIN, Z.; AL. et. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2017.

YANNAKAKIS, G.; TOGELIUS, J. *Artificial Intelligence and Games*. Springer International Publishing, 2018. Acesso em: 18 out. de 2024. ISBN 9783319635194. Disponível em: <https://books.google.com.br/books?id=HK1MDwAAQBAJ>.