UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ - UNIOESTE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Levantamento de aplicações das técnicas de Inteligência Artificial para o controle de NPCs em jogos eletrônicos

Trabalho de Conclusão de Curso

Lucas Frank Hollmann

Cascavel–PR

2021

UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ - UNIOESTE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Lucas Frank Hollmann

Levantamento de aplicações das técnicas de Inteligência Artificial para o controle de NPCs em jogos eletrônicos

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel.

Orientador(a): Prof. Dr. André Luiz Brun

Cascavel–PR

2021

LUCAS FRANK HOLLMANN

LEVANTAMENTO DE APLICAÇÕES DAS TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL PARA O CONTROLE DE NPCS EM JOGOS ELETRÔNICOS

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. André Luiz Brun (Orientador)
Colegiado de Ciência da Computação, UNIOESTE

Prof. Dr. Adair Santa Catarina
Colegiado de Ciência da Computação, UNIOESTE

Prof. Msc. Edmar André Bellorini
Colegiado de Ciência da Computação, UNIOESTE

Cascavel, 25 de Julho de 2022

Agradecimentos

Agradeço aos meus pais, à minha irmã, ao professor Dr. André Luiz Brun.

Resumo

Uma das partes mais importantes do desenvolvimento de jogos eletrônicos é o comportamento dos NPCs do *videogame*, sendo necessário criar personagens com credibilidade, que estejam de acordo com a proposta do jogo e que garantam a melhor experiência possível para o jogador. Com o objetivo de entender melhor os métodos utilizados no desenvolvimento desses personagens, neste trabalho são levantadas as técnicas de IA aplicadas em alguns jogos comerciais, analisando os objetivos buscados pelas implementações e os resultados obtidos, além de comparar as diferentes técnicas utilizadas entre si. Os jogos analisados são *DOOM 2016*, que utiliza Máquinas de Estados Finitas Hierárquicas, *F.E.A.R.*, que adota Planejamento de Ações Orientado a Objetivos e Máquinas de Estados, *Alien: Isolation*, que emprega Árvores de Comportamento, *Horizon Zero Dawn*, que faz uso de Redes de Tarefas Hierárquicas, e a franquia *Forza*, que tem como foco as Redes Neurais Artificiais. A partir desse levantamento foi possível constatar que as MEFs são de fácil implementação, porém apresentam dificuldades em lidar com casos complexos, pela falta de modularidade. Poréms esses problemas são mitigados pela sua abordagem hierárquica, as MEFHs. O GOAP tem uma maior complexidade, mas possui suas ações desacopladas dos objetivos, aumentando a modularidade e a capacidade de fazer adaptações no sistema sem ter que alterar a maior parte dele. As BTs utilizam estruturas de árvore para a tomada de decisões. Isso permite um bom entendimento do sistema e uma visualização gráfica compreensível, além de serem bastante modulares, permitindo alterar galhos das árvores sem afetar os outros. As HTNs são baseadas no GOAP e agem de forma muito parecida. Entretanto, o desenvolvedor consegue ter mais controle das sequências de ações ao criar ações compostas com estruturas hierárquicas, contendo outras ações compostas ou ações simples. A RNA é a técnica que mais difere das demais, permitindo criar redes que, ao receberem as variáveis de entrada, podem calcular as ações do NPC, sendo, o processo interno da rede, treinado através de bases de dados, e não definido por interferência direta do programador.

Palavras-chave: *Videogames*, Máquinas de Estados, Planejamento de Ações Orientado a Objetivo, Árvores de Comportamento, Redes de Tarefas Hierárquicas, Redes Neurais Artificiais.

Lista de figuras

Lista de abreviaturas e siglas

NPC	*Non Playable-Character*, ou Personagem Não Jogável, em português
IA	Inteligência Artificial
AM	Aprendizado de Máquina
BT	*Behaviour Tree*, ou Árvore de Comportamento, em português
RPG	Role Playing Game
RBC	Raciocínio Baseado em Casos
RNA	Redes Neurais Artificiais
SBR	Sistema Baseado em Regras
MEF	Máquina de Estados Finita
MEFH	Máquina de Estados Finita Hierárquica
GOAP	*Goal-Oriented Action Planning*, ou Planejamento de Ações Orientado a Objetivo, em português
STRIPS	*STanford Research Institute Problem Solver*, ou Solucionador de Problemas do Instituto de Pesquisa de Stanford, em português
HTN	*Hierarchical Task Networks*, ou Rede de Tarefas Hierárquica, em português

Sumário

1 Introdução

Jogos sempre foram uma atividade presente na sociedade, fazendo parte da cultura e da economia e, com o advento da tecnologia, tornaram-se muito comuns e difundidos no formato digital, ganhando o nome de videogames ou jogos eletrônicos. Com um mercado crescente e consumidores ansiosos por avanços, destaca-se a importância do desenvolvimento dessa área.

O principal objetivo dos jogos eletrônicos é o entretenimento, tendo se tornado um dos principais meios de lazer na sociedade moderna. Contudo, os videogames já existem há muito tempo. Por exemplo, um dos primeiros videogames, *Tennis for Two*, foi criado em 1958, por William A. Higinbotham, com visuais básicos e sem som, o jogo tentava simular uma partida de tênis (KENT, 2001).

Desde então, devido às melhorias de *hardware* e *software*, os jogos evoluíram consideravelmente. Existem muitos tipos de jogos eletrônicos, que diferem na temática, estilo gráfico, jogabilidade, entre outros fatores. Existem videogames de muitas temáticas diferentes, como jogos de aventura, batalha e esportes, por exemplo. O estilo gráfico de um jogo pode variar entre duas e três dimensões apresentando diversos estilos visuais, inclusive, aqueles que se aproximam da realidade.

O mercado de videogames atual é comparável com o de cinema, crescendo 11% ao ano, além de haverem previsões para o ano de 2023 onde o rendimento desse mercado ultrapassará a marca dos 200 bilhões de dólares (PURCHIO, 2021).

A Inteligência Artificial (IA) é uma área que pesquisa técnicas para desenvolver dispositivos capazes de simular o raciocínio humano (RUSSELL; NORVIG, 2010). Devido à premissa de imitar o comportamento humano, a IA é frequentemente relacionada ao desenvolvimento de jogos pois um importante elemento dos jogos eletrônicos são os NPCs (*Non-Player Character*), que interagem com o jogador e o ambiente em que se encontram. Tais personagens precisam ser controlados e, dentro da área de desenvolvimento de jogos, é chamado de IA qualquer algoritmo utilizado no controle dos NPCs.

Os NPCs podem aparecer nos jogos de formas variadas, como oponentes, aliados ou como um personagem com qualquer propósito que tenha sido definido para ele. Porém, a IA pode ser aplicada, também, de outras formas, como uma estratégia para resolver outros problemas específicos.

1.1 Objetivo

O objetivo principal deste trabalho é realizar um levantamento das técnicas de Inteligência Artificial utilizadas para o controle dos NPCs em videogames desenvolvidos ao longo da história e compreender como essas técnicas funcionam, de que forma foram aplicadas aos casos levantados e os resultados obtidos através do desenvolvimento com o uso da IA. Também objetiva-se apresentar uma visão geral comparando abordagens com aplicações de diferentes técnicas de IA em diferentes jogos, observando as características da implementação de cada método, com o objetivo de entender as vantagens ou desvantagens obtidas ao utilizar os métodos de Inteligência Artificial.

Como contribuição deste trabalho, objetiva-se construir um texto que venha a auxiliar desenvolvedores e pesquisadores de IA em videogames, exibindo uma variedade de abordagens possíveis para resolver o problema de criar comportamentos convencíveis e que melhorem a experiência do jogador.

1.2 Estrutura do Texto

O primeiro capítulo (1) deste trabalho é uma introdução aos temas a serem abordados, definindo, também os objetivos buscados e estrutura do texto. No capítulo 2, são apresentados os conceitos e a fundamentação sobre jogos eletrônicos e inteligência artificial, explicando as principais técnicas de controle de agentes inteligentes em videogames. O capítulo 3 contém a metodologia utilizada para desenvolver o trabalho. O capítulo 4 apresenta os resultados do levantamento, exibindo técnicas de IA utilizadas em jogos comerciais, a forma como cada técnica foi implementada em cada caso e os resultados obtidos ao empregar cada uma delas. O capítulo 5 contem a conclusão do trabalho, com as considerações finais.

2

Fundamentação e Conceitos

Neste capítulo serão apresentados os conceitos utilizados ao longo do trabalho, definindo o que são jogos e jogos eletrônicos, apresentando um resumo da história dos jogos e do desenvolvimento de videogames, além de apresentar o conceito de NPCs. Também são abordados os conceitos de inteligência artificial e IA aplicada aos jogos eletrônicos, apresentando as principais técnicas utilizadas.

2.1 Jogos Eletrônicos

Rogers (2013) define que um jogo é uma atividade que requer no mínimo um jogador, tem regras e uma condição de vitória. Um jogo eletrônico, também chamado de *videogame*, portanto, é um jogo disponibilizado em uma tela de vídeo. Já Battaiola (2000) define um jogo eletrônico como um sistema interativo que permite ao usuário experimentar uma situação de conflito, onde, quase sempre, existe, como plano de fundo, um enredo, que define do que se trata o jogo, as regras às quais o jogador deve seguir e os objetivos que ele deve se esforçar para atingir.

Apesar dos jogos eletrônicos representarem uma parte integral da cultura popular contemporânea, a história dos videogames foi construída a partir de influências e acontecimentos que já eram escritos desde muito antes da existência do primeiro jogo eletrônico (KENT, 2001).

2.1.1 Histórico dos Jogos

Os jogos estão presentes na sociedade há milhares de anos, com relevância social, econômica e cultural, não limitando-se apenas aos eletrônicos, que só apareceram nas últimas décadas. Jogos de cartas, de tabuleiro e até mesmo esportes são exemplos de atividades não eletrônicas que ajudaram a construir a cultura de diversas civilizações e contribuíram para a evolução dos jogos ao longo da história.

No século XX a indústria de jogos se desenvolveu bastante. Em 1931, foi criado, por David Gottlieb, o *Baffle Ball*, uma das primeiras versões dos *pinballs*, sem o uso de eletricidade, utilizando apenas a gravidade e dispositivos mecânicos, sendo considerado um grande sucesso e influenciando a criação de várias outras versões desses jogos. Ao longo das décadas seguintes, foram criados vários outros jogos que simulavam esportes, corridas e outras atividades (KENT, 2001).

Na década de 1950, foram desenvolvidos programas que ficaram conhecidos como os primeiros jogos eletrônicos, com visuais extremamente simples, exibidos em osciloscópios. Em 1952 foi desenvolvido um jogo chamado de *OXO*, sendo uma versão eletrônica do jogo da velha e, em 1958, foi criado o *Tennis for Two*, que simulava uma partida simplificada de tênis. Esses primeiros jogos eletrônicos foram criados de forma experimental em laboratórios (KENT, 2001; ROGERS, 2013).

Os *arcades*, como ficaram conhecidos, eram grandes gabinetes com monitores de tubo responsáveis por executar os primeiros videogames comerciais. Essas máquinas também eram conhecidos como fliperamas. Em 1971 foi desenvolvido o *Computer Space*, o primeiro jogo *arcade*, mas esse modelo de videogame ganhou muita força a partir do lançamento de *Pong*, em 1972. A partir desse ponto, a indústria dos *arcades* cresceu vertiginosamente e foram desenvolvidos diversos jogos com estilos diferentes e dispositivos de entrada e saída diversificados com o objetivo de aumentar a imersão e a interatividade com os jogos, ganhando ainda mais reputação nos anos 1980, quando se popularizaram os estabelecimentos que também ficaram conhecidos como *arcades* ou fliperamas, com o objetivo de disponibilizar, de forma paga, as máquinas com os jogos para os clientes (ROGERS, 2013).

A partir do final dos anos 1970 também surgiram os consoles, dispositivos que, assim como os *arcades*, tinham o propósito de executar os jogos eletrônicos. Entretanto, os consoles eram muito menores em tamanho e uma pessoa comum poderia tê-los em sua casa, além de que, diferente da maioria dos *arcades*, grande parte dos consoles podiam executar vários jogos diferentes. Enquanto os *arcades* começaram a decair em popularidade com o avanço tecnológico, os consoles se mantiveram no mercado, continuando populares até os dias atuais. Também foram criados os consoles portáteis, com tamanhos ainda menores, possíveis de caberem nas mãos do usuário e de serem utilizados em qualquer lugar (ROGERS, 2013).

O surgimento dos computadores pessoais, ainda na década de 1970, que se popularizaram constantemente ao longo das décadas seguintes, também foi revolucionário para a indústria dos jogos, porque, com esses dispositivos, tornou-se possível não apenas executar diversos videogames, mas também era possibilitou que programadores criassem novos jogos nas suas próprias casas, resultando no surgimento de vários estilos de jogos eletrônicos inovadores (ROGERS, 2013).

O cenário atual dos videogames é extremamente vasto e diversificado. Desde o surgimento dos primeiros jogos eletrônicos houve amplos avanços em *hardware* e *software* que possibilitaram

que os videogames também apresentassem evoluções consideráveis, tanto na qualidade audiovisual quanto nas mecânicas e regras que os regem. Existem vários gêneros de videogames, como jogos de estratégia, ação e simulação (APPERLEY, 2006). Vários critérios diferentes podem ser adotadados para classificar o gênero de um jogo. Podem ser consideradas as suas mecânicas, que podem variar de acordo com diversos outros fatores, o seu objetivo, o gênero da sua história, caso haja uma ou qualquer outra característica que o diferencie.

2.1.2 NPCs

Poucos elementos são obrigatórios dentro de um videogame. Cada jogo pode apresentar aspectos diferentes de outros jogos, sem seguir muitas restrições. Contudo, existem conceitos que são comuns no mundo dos jogos e, dentre eles, estão os NPCs. Os Personagens Não-Jogáveis, ou NPCs, são personagens dentro de jogos que, como o nome indica, não são controlados por nenhum jogador. Apesar de serem extensivamente aplicados aos jogos eletrônicos, o conceito de NPC surgiu antes dos jogos digitais. NPCs são comumente utilizados em jogos de RPG (*Role Playing Game*) de mesa, sendo controlados por uma pessoa responsável por comandar o jogo e a narrativa (mestre), interagindo com os personagens controlados pelos jogadores (WARPEFELT, 2016). Nos jogos eletrônicos, o papel de controle do jogo foi passado para a máquina, incluindo o comando dos NPCs, utilizando algoritmos que definem as suas ações para cada situação prevista ou técnicas de inteligência artificial. NPCs podem ter diferentes funções dentro de um jogo, como auxiliar o personagem em seu objetivo, desempenhar um papel no desenvolvimento do enredo do jogo, atuar como oponentes para o jogador ou até mesmo agir somente como parte do ambiente (WARPEFELT, 2016; PIHLGREN et al., 2016).

Um grande desafio ao desenvolver NPCs é garantir um comportamento convincente. Para isso é necessário que o personagem controlado pela máquina esteja alinhado às expectativas do jogador, proporcionando experiências mais realistas e garantindo a sensação de imersão (WARPEFELT, 2016).

2.1.3 Desenvolvimento de Jogos Eletrônicos

O desenvolvimento de jogos é uma área multidisciplinar, exigindo conhecimento não só da Informática, mas também de outras áreas do conhecimento, como artística e musical, bem como profissionais qualificados em testar os jogos, gerentes de projeto, projetistas de jogos (*Game Designer*), entre outros (GUIMARÃES, 2009).

Para que o processo de desenvolvimento de um jogo ocorra da forma mais consistente possível, minimizando a quantidade de erros necessitando de correção nas etapas mais avançadas do desenvolvimento, deve-se haver um bom planejamento inicial e bom conhecimento dos processos de criação do jogo (ROLLINGS; MORRIS, 2003).

Rollings e Morris (2003) definem cinco principais pontos que devem ser considerados

na especificação de um jogo, são elas: as principais características do jogo, a jogabilidade, a interface de usuário, as regras e o *Level Design*.

As principais características são o que vão diferenciar um jogo de todos os outros. Portanto, a definição dessas características é um bom ponto de início da especificação. Rollings e Morris (2003) separam as características de um videogame em três categorias, sendo, a primeira delas, a categoria das características essenciais para o jogo, que estão relacionadas com a ideia principal do jogo, sem as quais o jogo não poderia funcionar da forma planejada. A segunda categoria engloba recursos que melhoram a experiência do jogador mas não têm uma grande interferência no modo como o jogo é jogado. Na terceira categoria, estão as características que não contribuem com a melhoria da experiência do videogame e que não são necessárias para o seu funcionamento, sendo uma categoria de características que devem ser evitadas.

Após descrever as características principais, é necessário especificar como essas eles impactarão na jogabilidade do jogo. Esta envolve as decisões que o jogador deverá fazer ao jogar um videogame e as opções que ele poderá escolher ao lidar com as situações apresentadas.

A interface de usuário tem um papel importante na comunicação entre o jogador e a máquina. Uma boa interface deve auxiliar o jogador, sem deixar a impressão de que ele está sendo restringido pelo sistema. Segundo Ramos (2004), qualquer *software* que não consiga interagir com o usuário de uma maneira eficiente, está fadado ao fracasso.

As regras, em um videogame, devem existir para impedir com que os jogadores tomem decisões que não estão de acordo com a proposta do jogo, eventualmente, fazendo com que a experiência do jogador, ou de outros jogadores, perca a qualidade. Devem ser estabelecidas regras que impeçam o jogador de realizar algumas ações ou que o convençam de que ele terá mais benefício ao escolher alguma outra opção.

O *Level Design* é a parte do desenvolvimento de videogames em que se elabora o cenário em que o jogador será inserido, definindo as possibilidades de interação com o ambiente, o objetivo que deve ser alcançado pelo jogador naquela fase específica e os meios pelos quais ele pode concluir esse objetivo (JOHANSSON, 2014). Segundo Co (2006), *Level Designers* criam o espaço e ambiente pelo qual você se move e delineiam a experiência enquanto joga o videogame. Portanto, um *Level Designer* é alguém que necessita uma variedade de habilidades, podendo projetar vários tipos de ambiente, cada um deles com comportamentos específicos que devem ser levados em consideração para melhorar a experiência do jogador, ajudando-o a entender com clareza o seu objetivo e o que lhe é permitido fazer para alcançá-lo (JOHANSSON, 2014).

2.1.3.1 *Game Engines*

Game Engines, ou *Engines* de Jogos, são programas que auxiliam no desenvolvimento de jogos eletrônicos. Embora não haja um conjunto específico de recursos que um programa deva oferecer para ser considerado uma *Game Engine*, é possível listar os recursos mais comuns

encontrados nelas, dentre eles, estão a *Engine* de renderização, o gerenciamento de *inputs* recebidos de dispositivos de entrada, a *Engine* de física, o sistema de animações para os elementos do jogo e ferramentas para a criação de Inteligência Artificial para os NPCs.

2.2 Inteligência Artificial

Russell e Norvig (2010) apresentam oito definições de Inteligência Artificial (IA), as dividindo em quatro categorias: pensamento humano, pensamento racional, ação humana e ação racional. A primeira e a segunda categoria se relacionam a processos de pensamento e raciocínio, já a terceira e a quarta têm foco no comportamento. A primeira e a terceira categoria medem a qualidade da IA com base na fidelidade ao desempenho humano, enquanto a segunda e a quarta fazem essa medição com base em um conceito ideal de inteligência, chamado de racionalidade. Russell e Norvig (2010) definem um sistema como racional se ele toma a decisão correta de acordo com as informações que possui. Todas as quatro categorias vem sendo estudadas, utilizando diferentes métodos, mostrando que a IA é uma área bastante versátil e que inclui uma grande variedade de abordagens.

O primeiro trabalho reconhecido como IA foi realizado por McCulloch e Pitts (1943), onde propôs-se um modelo de neurônios artificiais. No entanto, o termo "Inteligência Artificial" só foi formalizado em 1956, por John McCarthy em uma conferência na Universidade de Dartmouth, onde também foi apresentado o primeiro programa de IA, por Allen Newell e Hebert Simon (RUSSELL; NORVIG, 2010; PATEL; JAISWAL, 2021).

Por ser bastante abrangente, podendo apresentar resultados que comparam-se ao desempenho humano ou até o superam, a Inteligência Artificial pode ser aplicada em diversas áreas, algumas das principais são o reconhecimento de voz e imagem, veículos autônomos, robótica, tradução automática, previsão de tempo, mineração de dados e jogos (RUSSELL; NORVIG, 2010; BORANA, 2016).

2.3 IA aplicada a jogos eletrônicos

Na área de desenvolvimento de jogos eletrônicos, o nome "Inteligência Artificial" é utilizado de uma forma um pouco diferente do que no meio acadêmico (KISHIMOTO, 2004). Enquanto na academia costumamos chamar de IA apenas algumas técnicas específicas, que são o foco do estudo dessa área, no desenvolvimento de jogos, é denominado como IA qualquer método utilizado para controlar agentes que devem demonstrar certa inteligência, mesmo que sejam métodos simples e determinísticos. Nesse trabalho, o termo será utilizado da mesma forma que no desenvolvimento de jogos (KISHIMOTO, 2004; BOURG; SEEMAN, 2004).

Existem diversas técnicas utilizadas para o controle de NPCs nos videogames e é importante notar que um mesmo agente pode utilizar diversas técnicas combinadas para definir

seu comportamento completo. A seguir, veremos algumas das técnicas mais utilizadas.

2.3.1 Algoritmos Determinísticos e Aleatoriedade

Algoritmos Determinísticos definem comportamentos predeterminados. Apesar de não definirem os movimentos exatos do NPC, como os Padrões de Movimento fazem, os Algoritmos Determinísticos especificam métodos para que o agente realize determinadas ações, podendo levar em consideração o estado dos demais elementos do ambiente em que se encontra (LAMOTHE, 1999).

O conjunto de condições e ações a serem realizadas quando as condições forem cumpridas pode ser chamado de regra e uma IA composta por diversas regras que regem o comportamento de um agente pode ser denominada como um Sistema Baseado em Regras (SBR). A Figura 1 representa uma regra em um SBR, onde são avaliados os atributos de energia do oponente e do agente, além da distância entre eles, decidindo, assim, se a ação "procurar" será executada pelo agente (KARLSSON, 2005).

Figura 1 – Exemplo de uma regra em um SBR.

```
SE (distancia do inimigo > 100
    E energia do agente > 50
    E energia do inimigo < 50)
ENTAO procurar
```

Fonte: Adaptado de Karlsson (2005).

Exemplos de Algoritmos Determinísticos são os modelos de perseguição, onde o NPC tem um alvo definido e move-se em sua direção, e algoritmos de fuga, onde o NPC deve se mover na direção oposta à do alvo. Essa movimentação pode ser feita de modo que o agente apenas se movimente em linha reta em direção ao alvo ou na direção oposta bem como podem haver outras condições sobre o movimento do NPC, como fazer curvas suavizadas para simular realismo e desviar de obstáculos durante o percurso (LAMOTHE, 1999). Também é comum programar sequências de ações que o personagem deve seguir, como percorrer uma rota predefinida repetidamente, simulando um comportamento de patrulha.

Porém, a aplicação dos Algoritmos Determinísticos não se limita a esses casos. É possível adotar esse tipo de algoritmo para realizar qualquer ação que tenha condições específicas para acontecer, como mercadores de itens em um jogo de RPG.

Também é possível utilizar comportamentos aleatórios. Por exemplo, é possível que o agente escolha direções aleatórias para percorrer, havendo, normalmente, limites que devem ser respeitados. A aleatoriedade também pode ser usada para determinar a ação que será executada pelo NPC, escolhendo uma dentre um conjunto de ações possíveis (LAMOTHE, 1999).

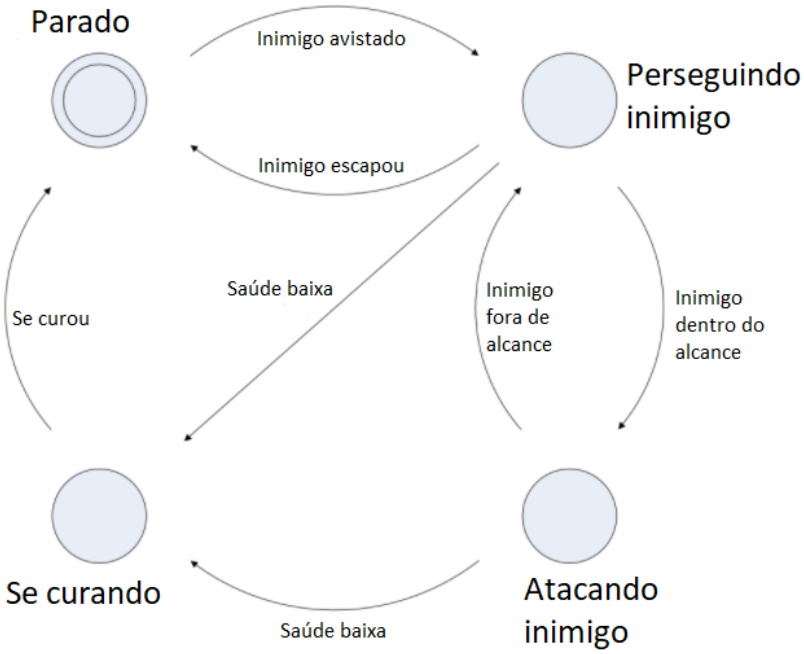
2.3.2 Máquinas de Estados Finitas

Máquinas de Estados Finitas (MEF) são máquinas abstratas que definem estados nos quais o NPC pode estar e transitar entre. Cada estado especifica um comportamento diferente para o agente e também os estados que podem ser acessados a partir dele, com as condições para que a transição para os outros estados aconteça (KISHIMOTO, 2004; BOURG; SEEMAN, 2004).

As MEF são utilizadas desde jogos antigos até aqueles mais recentes porque são fáceis de implementar e de entender, além de apresentarem resultados satisfatórios (KISHIMOTO, 2004). Essa técnica permite que os agentes de um jogo ganhem mais complexidade, mantendo a simplicidade no desenvolvimento e não requisitando muito poder de processamento a mais, em relação às técnicas apresentadas anteriormente.

A Figura 2 demonstra uma representação de uma Máquina de Estados Finita com quatro estados. O primeiro, "Parado", é o estado inicial e pode transicionar para o "Perseguindo inimigo" caso o NPC aviste o inimigo. O estado "Perseguindo inimigo" pode transicionar para "Parado" caso o inimigo escape, para o "Se curando" caso a saúde do NPC esteja baixa ou para o "Atacando inimigo", caso o inimigo esteja dentro do alcance do seu ataque. O estado "Se curando" sempre transicionará para "Parado" após o agente terminar o processo de cura. E o estado "Atacando inimigo" pode transicionar para o "Perseguindo inimigo" caso o inimigo saia do alcance do ataque do NPC e para "Se curando", caso sua saúde esteja baixa (KARLSSON, 2005).

Figura 2 – Exemplo de representação de uma Máquina de Estados Finita.



Fonte: Adaptado de Karlsson (2005).

É possível notar que, dentro de cada estado, ainda podem haver implementações com

outras técnicas de IA. Por exemplo, dentro do estado "Perseguindo inimigo" pode haver uma técnica de *Pathfinding* para que o NPC encontre uma rota até o inimigo perseguido e lide com os obstáculos no percurso. Outro cenário é quando o agente se encontra no estado "Atacando inimigo", pois podem ser usadas outras técnicas dentro desse estado para que o inimigo defina a forma como será feito o ataque.

2.3.3 Algoritmos de Busca

Algoritmos de Busca Heurística são amplamente usados como um método de IA para a resolução de problemas, com o objetivo de encontrar uma solução ótima ou aproximada. Para problemas muito complexos, encontrar a solução ótima pode levar muito tempo, sendo mais viável utilizar uma solução de custo mais alto, porém encontrada rapidamente. Por isso, existem várias técnicas de Busca Heurística, permitindo que seja escolhida uma técnica com base na qualidade da solução apresentada e no tempo de busca pela solução (HANSEN; ZHOU, 2007).

Nos videogames, esses algoritmos costumam ser usados para encontrar rotas entre um agente e um ponto de destino, considerando os elementos do cenário e desviando dos obstáculos. Este processo é chamado de *Pathfinding*. Para esse propósito, o Algoritmo de Busca mais utilizado é o $A*$ (LAMOTHE, 1999; DALMAU, 2004).

O algoritmo $A*$ percorre um grafo direcionado onde cada nó representa um estado e, partindo de um nó que representa o estado inicial, cada nó leva a outros nós que representam estados atingíveis a partir do próprio estado, de forma subsequente, até chegar em um nó que representa o estado final, que é o resultado buscado.

Para percorrer o grafo, o algoritmo $A*$ mantém uma lista dos nós acessíveis a partir do que já foi explorado no grafo, que é inicializada contendo apenas o nó inicial. O algoritmo, então, avalia os nós contidos na lista de acordo com a métrica $f(n)$ calculada por $f(n) = g(n) + h(n)$, onde n é o nó avaliado, $g(n)$ é o custo para chegar até o nó n a partir do nó inicial e $h(n)$ é o custo estimado para percorrer do nó n até um nó que contenha o estado buscado. O custo estimado entre um nó avaliado e um nó que represente o estado buscado pode ser calculado de diversas formas, ficando a critério de quem implementar escolher um método apropriado para o cálculo (EDELKAMP; SCHRODL, 2011; DUCHON et al., 2014).

Após o cálculo de f para cada nó na lista de nós acessíveis, o nó com o menor valor de f é explorado e removido da lista. Então, é verificado se o nó explorado representa o estado buscado e, em caso afirmativo, o algoritmo pára, tomando o caminho entre o nó inicial e o nó explorado como a solução final. Caso o nó explorado não represente o estado buscado, seus nós sucessores são adicionados à lista de acessíveis e o mesmo processo se repete, verificando o valor f dos nós da lista e escolhendo um novo nó para explorar (EDELKAMP; SCHRODL, 2011; DUCHON et al., 2014).

Uma forma de desenvolver um algoritmo de *Pathfinding* utilizando o $A*$ seria dividindo

a área onde a rota é buscada em células, com uma célula que contém a posição inicial e outra contendo a posição final. Cada estado representaria uma célula, com o caminho até ela e o valor de f, onde g poderia ser calculado pela soma das distâncias euclidianas entre os pontos centrais de cada par de células adjacentes pertencentes ao caminho percorrido, e o valor de h, pela distância euclidiana entre os pontos centrais da célula representada pelo estado atual e da célula final.

2.3.4 Raciocínio Baseado em Casos

O Racicínio Baseado em Casos, ou RBC, é uma técnica que propõe a resolução de novos problemas relembrando-se de situações anteriores similares e armazenando as informações adquiridas na resolução do problema atual para utilizar em situações futuras (GUIMARÃES, 2009). Essas situações são chamadas de casos e são mantidas em uma base de dados que é atualizada toda vez que um novo problema é processado.

O RBC conta com quatro elementos básicos, sendo eles a Representação de Conhecimento, que acontece no formato de casos que descrevem experiências concretas, a Média de Similaridade, que indica se um caso anterior é similar ao novo problema, a Adaptação, que busca fazer uma melhor adequação ao novo problema a partir de casos anteriores, e o Aprendizado, onde o problema atual é armazenado na base de dados como um novo caso, podendo ser utilizado na solução de problemas posteriores (GUIMARÃES, 2009).

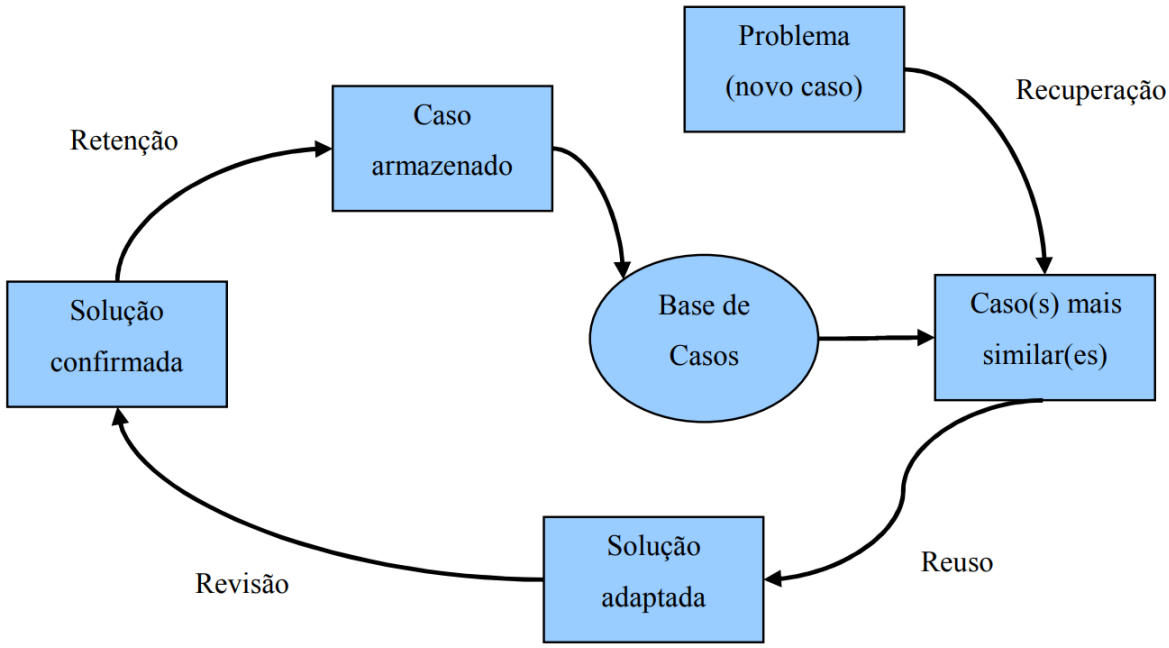
Segundo Wangenheim e Wangenheim (2003), o modelo mais aceito na representação do RBC é o proposto por Aamodt e Plaza (1994), composto por quatro tarefas principais, recuperar casos similares ao problema atual da base de dados, reutilizar esses casos na resolução do problema, revisar a solução encontrada e reter informações do problema atual, armazenando-as como casos na base de dados. A Figura 3 demonstra o funcionamento típico do ciclo do RBC, realizando as tarefas citadas.

Conforme representado na ilustração, quando um novo problema precisa ser resolvido, é feita a busca na base de dados por casos similares, sendo, essa, a tarefa de recuperação, podendo recuperar casos com similaridades diferentes em relação ao problema. Em seguida, a tarefa de reutilização consiste em criar uma solução adaptada a partir dos casos recuperados. A tarefa de revisão busca avaliar, através de métricas adequadas ou simulações, a qualidade da solução encontrada na tarefa anterior aplicada ao problema atual. Por fim, a retenção acontece ao armazenar o problema, como um caso, na base de dados para utilização futura.

Um exemplo da aplicação do RBC em jogos eletrônicos é o trabalho de Guimarães (2009), onde foi desenvolvido um jogo de nave evolutivo utilizando essa técnica, também fazendo uso das Máquinas de Estados. O objetivo era que o agente inteligente apresentado no jogo pudesse evoluir seu comportamento conforme mais casos são adicionados ao sistema do RBC.

O *videogame* desenvolvido por Guimarães (2009) tem uma nave que pode ser controlada

Figura 3 – Ciclo de resolução de problemas do RBC.



Fonte: Adaptado de Aamodt e Plaza (1994).

ou assistida pelo jogador. Quando controlada pelo jogador, ela armazena dados que são utilizados pelo RBC para operar o modo assistido, onde a IA controla a nave. O objetivo da nave é se defender e contra atacar agentes inimigos que, por sua vez, atacam a nave.

2.3.5 Planejamento de Ações Orientado a Objetivo

O GOAP (*Goal-Oriented Action Planning*) é uma técnica que tem como base um método conhecido como Planejamento STRIPS (*STanford Research Institute Problem Solver*) (BJARNOLF et al., 2008). O STRIPS consiste em objetivos e ações, onde objetivos são estados que queremos atingir e ações têm precondições e efeitos (ORKIN, 2006; STUDIAWAN; HARIADI; SUMPENO, 2018).

Os estados, para o STRIPS, são conjuntos de informações que descrevem o contexto em que o método é aplicado. A realização de ações é o meio para atingir um estado objetivo. Porém, para que uma ação possa ser executada, suas precondições, que consideram as informações do estado atual, devem ser cumpridas. Os efeitos de uma ação são descritos como uma lista de modificações ao estado atual, podendo remover informações que não são mais verdadeiras do estado ou adicionar novas informações (ORKIN, 2006).

Orkin (2006) cita três benefícios do planejamento. O primeiro é o desacoplamento de ações e objetivos, pois não há nenhuma restrição sobre quais ações devem ser realizadas para chegar a um objetivo. Isso permite que o conjunto de ações de um agente seja modificado,

podendo adicionar novas ações, sem um grande aumento da complexidade do sistema. O segundo benefício é o comportamento em camadas, que permite organizar comportamentos simples em camadas, com diferentes ações e objetivos, e, então, conseguir um comportamento complexo quando o NPC utilizar todas as camadas simultaneamente. Por fim, o planejamento dinâmico permite ao agente buscar soluções diferentes assim que uma tentativa falhar.

O GOAP difere do STRIPS em quatro pontos, dentre eles, está a adição de um custo por cada ação executada. A técnica utiliza o método de busca $A*$ para formular os planos, considerando cada estado como um nó no grafo gerado pelo $A*$ e as ações como o elemento que conecta cada par de nós, utilizando os custos das ações como métrica para a busca (ORKIN, 2006).

Outra mudança entre o STRIPS e o GOAP envolve a representação dos efeitos e precondições das ações. No GOAP, ambos são representados como vetores de valores com tramanho fixo que representam um estado, dessa forma, tornando trivial a tarefa de encontrar uma ação que satisfaz um objetivo ou precondição de outra ação (ORKIN, 2006).

A terceira diferença é a adição de precondições procedurais, que são funções que fazem verificações personalizadas que vão além das precondições representadas nos vetores de tamanho fixo. Isso é feito pois não é prático que as representações de estados contenham toda a informação do ambiente, portanto, as precondições procedurais permitem fazer validações com informações que não estão contidas na representação padrão (ORKIN, 2006).

A última mudança adiciona efeitos procedurais, de forma similar à anterior. Os efeitos procedurais são onde as ações são executadas efetivamente. Ao invés de alterar o estado atual imediatamente, o GOAP primeiro executa os efeitos procedurais, que demoram um determinado tempo para finalizarem a execução e, após isso, altera o estado (ORKIN, 2006).

Segundo Orkin (2006), o GOAP adapta o STRIPS para funcionar de forma prática em um videogame em tempo real, fazendo com que o planejamento seja mais eficiente e controlado, mantendo os benefícios do STRIPS.

2.3.6 Árvores de Comportamento

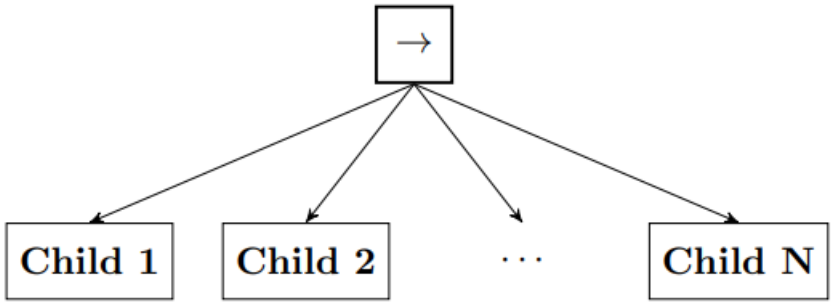
Árvores de Comportamento (*Behaviour Trees*, ou BTs) são estruturas hierárquicas onde cada nó indica uma ação a ser feita. As ações representadas por cada nó, além da raiz, indicam possíveis partes das ações do seu nó pai e outros nós superiores.

Uma BT inicia sua execução da raiz e envia sinais, chamados de *ticks*, com uma determinada frequência para os seus nós filhos. Um nó é executado sempre que recebe um *tick*, podendo retornar os valores "executando", se sua execução estiver em progresso, "sucesso", se a execução foi bem sucedida, ou "falha", em caso de uma execução mal sucedida (COLLEDANCHISE; OGREN, 2019).

As folhas de uma BT são chamadas de nós de execução e podem ser classificadas como ação ou condição. Os demais nós são chamados de nós de controle de fluxo e podem ser classificados como nó de sequência, recuo, paralelo e decorador (COLLEDANCHISE; OGREN, 2019).

Nós de sequência enviam *ticks* para seus filhos, da esquerda para a direita, até que um deles retorne "falha" ou "executando", atribuindo esse valor como retorno do nó de sequência, ou até que todos os filhos retornem "sucesso", atribuindo "sucesso" ao nó de sequência também. Se um filho do nó de sequência retornar "falha" ou "executando", ele não enviará mais *ticks* aos demais filhos (COLLEDANCHISE; OGREN, 2019). A Figura 4 mostra um exemplo de nó de sequência.

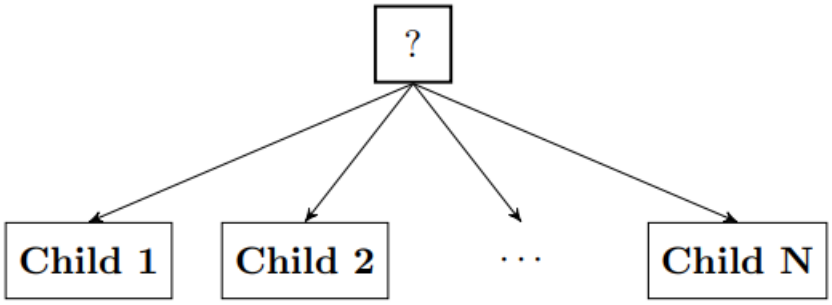
Figura 4 – Representação visual de um nó de sequência.



Fonte: Colledanchise e Ogren (2019).

Nós de recuo, como o representado na Figura 5, funcionam de forma parecida aos nós de sequência, porém, encerram a execução quando um dos nós filhos retorna "executando" ou "sucesso" ou quando todos os filhos retornam "falha" (COLLEDANCHISE; OGREN, 2019).

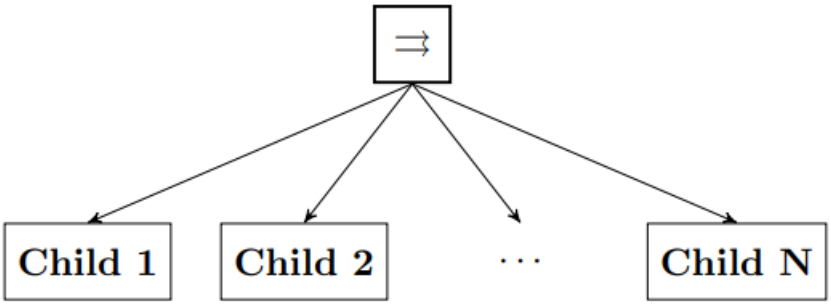
Figura 5 – Representação visual de um nó de recuo.



Fonte: Colledanchise e Ogren (2019).

Nós paralelos enviam *ticks* para todos os seus filhos. retornando "sucesso" caso haja, pelo menos, M filhos com esse retorno, "falha", caso haja, pelo menos, $N - M + 1$ filhos com esse valor de retorno e "executando" se nenhuma das outras condições seja cumprida. Onde N é o número de filhos do nó paralelo e M é um valor definido pelo usuário (COLLEDANCHISE; OGREN, 2019). A Figura 6 exemplifica um nó paralelo.

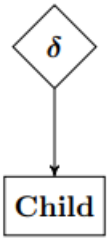
Figura 6 – Representação visual de um nó paralelo.



Fonte: Colledanchise e Ogren (2019).

A Figura 7 demonstra um nó decorador, que pode possuir apenas um filho e retorna o valor de seu nó filho de acordo com uma regra definida pelo usuário, além de poder ter uma regra que define quando o nó decorador enviará *ticks* para o nó filho. Por exemplo, um decorador de inversão inverte o valor de "sucesso" ou "falha" do seu filho. Um decorador com número máximo de tentativas só permitirá que seu filho retorne "falha" um número determinado de vezes, parando de enviar *ticks* após esse número ser atingido (COLLEDANCHISE; OGREN, 2019).

Figura 7 – Representação visual de um nó decorador, onde δ representa as regras definidas.



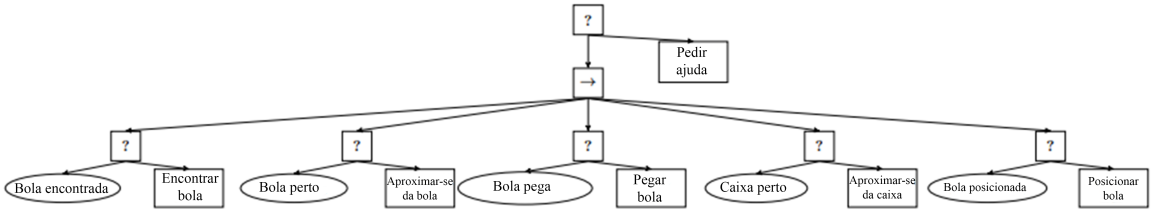
Fonte: Adaptado de Colledanchise e Ogren (2019).

Os nós de execução de ação disparam um comando ao receberem *ticks*, retornando "sucesso" caso a ação for completada corretamente, "falha" caso a ação falhe e "executando" enquanto estiver em progresso. Nós de condição checam uma proposição lógica e podem retornar apenas "sucesso" quando o valor da proposição for verdadeiro e "falha" quando o valor for falso (COLLEDANCHISE; OGREN, 2019).

É possível notar que o comportamento geral de uma Árvore de Comportamento envolve a realização de uma busca em profundidade, sempre visitando os nós da esquerda para a direita e respeitando as regras de cada nó de controle de fluxo, continuando ou interrompendo a busca em cada ramificação da árvore. A busca é feita até que seja retornado um valor de "sucesso" ou "falha" para a raiz.

A Figura 8 exemplifica uma Árvore de Comportamento que tem como objetivo pegar uma bola e posicioná-la dentro de uma caixa. Os nós de execução em formato de elipse são nós de condições e os nós de execução com formato de retângulo são nós de ação.

Figura 8 – Exemplo de Árvore de Comportamento.



Fonte: Adaptado de Colledanchise e Ogren (2019).

Segundo Colledanchise e Ogren (2019), Sekhavat (2021), algumas das vantagens das Árvores de Comportamento são a modularidade, que permite alterar partes da árvore sem prejudicar o funcionamento do resto, e a facilidade de representação gráfica, que ajuda a planejar as BTs e explicá-las.

2.3.7 Planejamento com Redes de Tarefas Hierárquicas

Assim como o GOAP, as Redes de Tarefas Hierárquicas (*Hierarchical Task Networks*, ou HTNs) também são um método de planejamento de ações, considerando os objetivos a alcançar. As tarefas em uma HTN são ações que podem ser primitivas ou compostas. As tarefas primitivas possuem seu comportamento definido por completo em si mesmas, possuindo precondições e efeitos, de forma similar às ações do GOAP. Já as compostas são definidas por outras tarefas primitivas ou compostas que estão abaixo dela no sistema hierárquico (ONTAÑÓN; BURO, 2015).

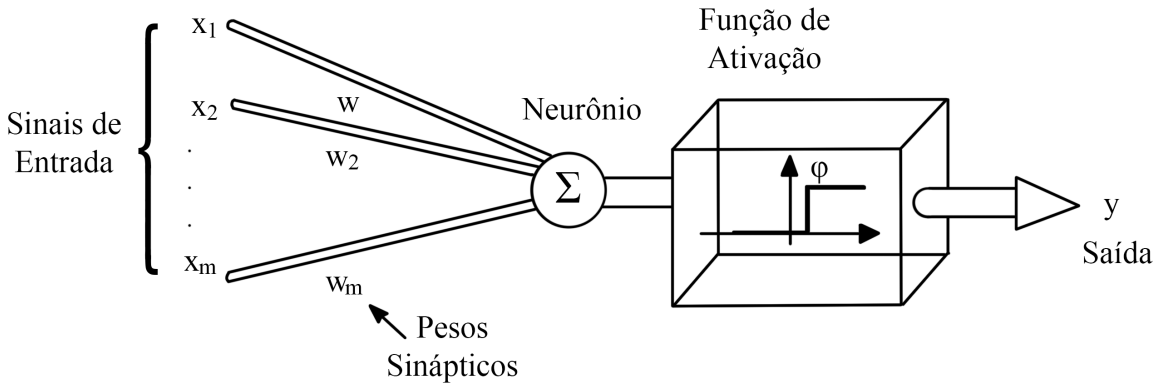
Os objetivos em uma HTN também funcionam de forma parecida com o GOAP, podendo definir as prioridades para cada objetivo de acordo com o estado atual do ambiente.

Uma HTN acaba formando estruturas de árvores muito parecidas com as Árvores de Comportamento, porém, as HTN são árvores geradas em tempo real, baseando-se nos objetivos atuais e no estado do sistema.

2.3.8 Redes Neurais Artificiais

A abordagem das Redes Neurais Artificiais (RNAs) baseia-se no funcionamento do cérebro humano, formando redes de nós, chamados de neurônios, que são os principais elementos das RNAs. A Figura 9 demonstra um modelo matemático de um neurônio de uma Rede Neural Artificial.

Figura 9 – Modelo de neurônio.



Fonte: Adaptado de Furtado (2019)

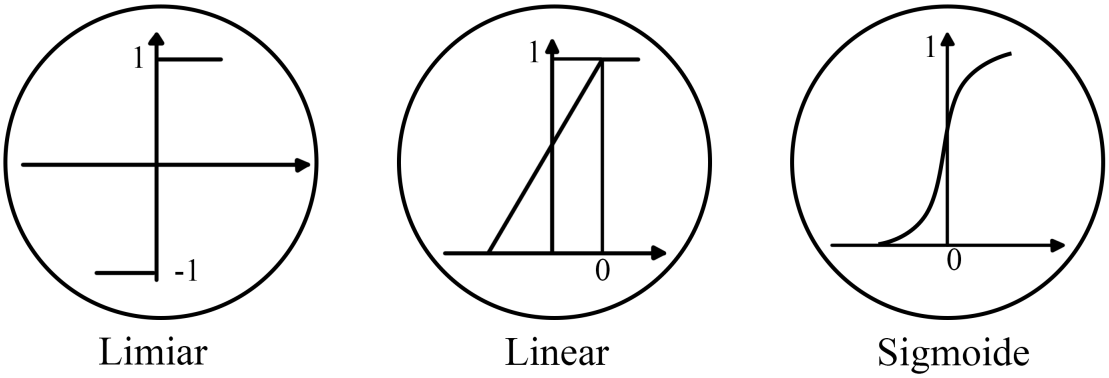
Cada conexão entre dois neurônios é chamada de sinapse, por onde um neurônio envia o seu valor de saída e outro recebe esse valor como uma das entradas. Para cada sinapse $j \in [1..m]$ o neurônio recebe uma entrada x_j, que é multiplicada pelo peso sináptico w_j e, em seguida, somada com as outras entradas já multiplicadas pelos seus respectivos pesos. O neurônio também recebe uma entrada fictícia com valor 1 e com um peso, para manter maior fidelidade ao modelo biológico (FURTADO, 2019; RUSSELL; NORVIG, 2010).

Após realizar a soma ponderada dos valores de entrada, o resultado passa pela função de ativação, que determina o valor de saída do neurônio. A função de ativação pode variar de abordagem para abordagem, porém as mais usadas são as funções de limiar, sigmoide e linear (FURTADO, 2019).

Ao aplicar uma RNA no controle de um NPC de um videogame, pode-se informar à rede, através de neurônios de entrada, dados sobre o estado atual do ambiente do jogo. Esses dados serão processados através da rede até que cheguem em neurônios de saída, que poderão informar ao NPC a ação que ele deverá tomar.

A Figura 11 exemplifica essa situação com uma Rede Neural que recebe, nos neurônios de entrada, informações sobre a distância entre o avatar do jogador e o NPC, a energia atual do NPC e do avatar do jogador e a quantidade de munição que o NPC possui. A partir dessas informações,

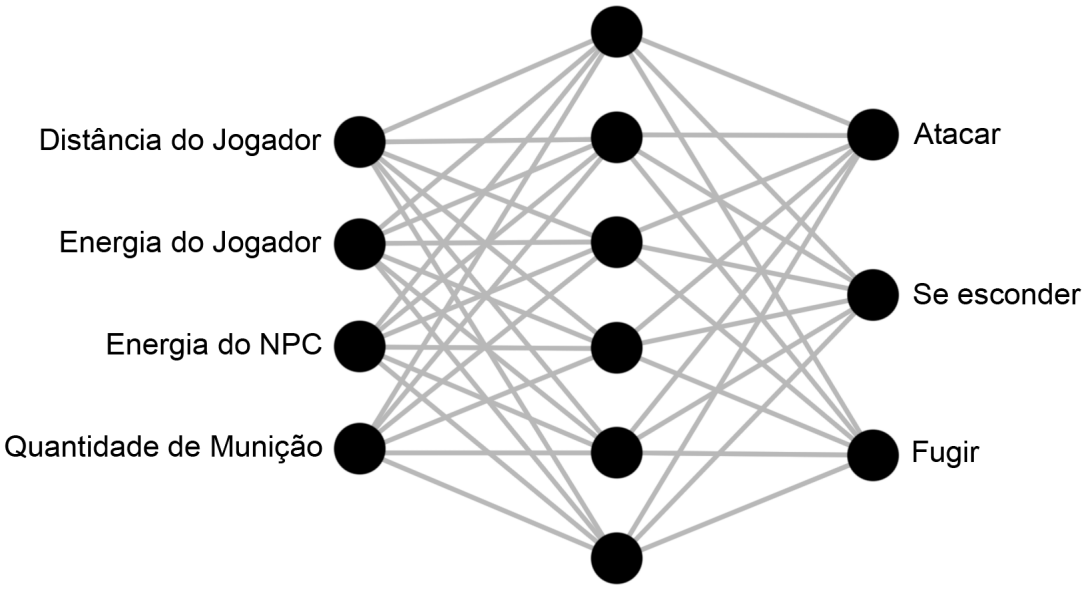
Figura 10 – Funções de Ativação.



Fonte: Adaptado de Furtado (2019)

a rede realiza o processo de ativação dos neurônios intermediários e obtém um resultado, que é expressado nos neurônios de saída, definindo uma ação dentre as válidas, podendo atacar o avatar do jogador, se esconder ou fugir do local.

Figura 11 – Exemplo de RNA para controle de NPC.



Fonte: do autor.

3
Metodologia

Para o desenvolvimento deste trabalho, foram reunidas as referências bibliográficas utilizadas para embasar a fundamentação e os conceitos apresentadas nos capítulos anteriores. Também foi feita a pesquisa do referencial sobre os métodos de Inteligência Artificial utilizados em videogames, falando um pouco sobre cada método.

Em seguida, foi feita a busca por material de referência sobre os jogos analisados e a IA aplicada neles. As informações sobre a implementação da IA foram obtidas a partir de palestras feitas pelos desenvolvedores em eventos como a Game Developers Conference (2022), a maior conferência de desenvolvedores de jogos comerciais, que acontece anualmente. Também foram obtidas informações de entrevistas com desenvolvedores ou artigos escritos pelos mesmos ou por especialistas da área.

A análise foi realizada agrupando os jogos pelos métodos utilizados neles. Foi feito um estudo sobre como os métodos de IA foram aplicados em cada jogo, considerando os objetivos que pretendia-se alcançar, bem como as adequações que foram necessárias para obter os resultados desejados.

Os critérios para escolher os jogos analisados envolveram a data de lançamento do jogo, priorizando os mais recentes, as técnicas utilizadas nos jogos, focando em analisar uma maior variedade de técnicas, e a capacidade de encontrar informações sobre o desenvolvimento da IA do jogo. Outro critério analisado foi se o jogo trouxe alguma inovação para a área de IA nos *videogames*, destacando-se por utilizar as técnicas de IA de forma inovadora ou aplicando alguma técnica pela primeira vez, como é o caso de *F.E.A.R.* com o Planejmaneto de Ações Orientado a Objetivo.

4
Resultados

Neste capítulo são apresentadas alguns jogos cujas características foram identificadas através da metologia apresentada. Dada à abrangência de jogos desenvolvidas, serão aprofundados apenas alguns jogos, focando naqueles que apresentaram estratégias inovadoras ou que contribuíram para o controle inteligente de NPCs.

4.1 *DOOM 2016*

DOOM (BETHESDA, 2016) é um jogo de tiro em primeira pessoa e plataforma desenvolvido pela *id Software* e publicado pela *Bethesda*, lançado em 2016, com um estilo de combate rápido, onde o jogador é incentivado a se expor para atacar os inimigos e a ficar em movimento constantemente (CAMPBELL, 2018).

A IA em *DOOM* emprega uma estrutura de Máquina de Estados Hierárquica (MEFH), uma técnica utilizada para diminuir a complexidade presente em MEFs comuns, pois, com um grande número de estados e, consequentemente, um grande número de transições entre eles, a dificuldade de entendimento e de desenvolvimento do sistema também aumenta. MEFHs buscam resolver esse problema dividindo os estados possíveis de uma MEF em grupos menores, dessa forma, tornando possível trabalhar em cada grupo como se fossem máquinas de estados menores, lidando com poucos estados e transições, além de algumas transições específicas que levam para um outro grupo (THOMPSON, 2018).

Também podemos entender uma HSFM como uma Máquina de Estados que possui outras Máquinas de Estados dentro de cada um de seus estados. Assim, os estados da MEFH definem comportamentos de forma genérica e as MEFs dentro deles controlam o comportamento de forma mais específica.

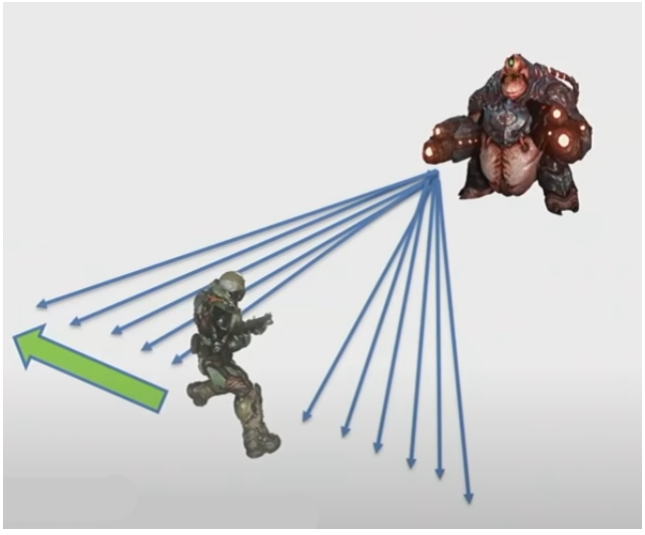
Com a estrutura da MEFH definida, ainda é preciso definir as condições e as formas de analisá-las para que cada transição de estados seja realizada, além do comportamento presente

dentro de cada estado. Campbell (2018) explica alguns desses funcionamentos, que utilizam algoritmos determinísticos.

O primeiro sistema explicado pelo autor são as reações dos NPCs aos ataques que sofrem. A frequência que uma reação é executada por um NPC é definida de acordo com o impacto que ela tem no jogo, acontecendo frequentemente caso tenha pouco impacto e raramente, caso tenha um impacto grande. A forma como uma reação impacta no jogo se dá pela interrupção das ações do NPC, podendo variar entre pequenas reações que não interrompem a ação atual, mas podem atrapalhar na precisão de um ataque, e reações que, além de interromper por completa a ação do NPC, ainda o deixam temporariamente imobilizado.

As ações de ataque dos NPCs que envolvem o disparo de projéteis possuem um comportamento que os força a errar o alvo caso o personagem controlado pelo jogador esteja em movimento. Isso acontece para que o jogador tenha o incentivo para continuar sempre se movendo, garantindo o estilo de jogo rápido que *DOOM* propõe. Ao tentar acertar o avatar do jogador em movimento, o NPC atira em ângulos próximos ao que o acertaria, como mostrado na Figura 12, a variação no ângulo é determinada aleatoriamente, porém, existem probabilidades diferentes para ângulos que resultariam em um projétil mais próximo ou mais distante do *player*. Essas probabilidades são alteradas de acordo com o nível de dificuldade do jogo, fazendo com que os projéteis cheguem mais perto do jogador em dificuldades maiores, aumentando a frequência com que ele é atingido (CAMPBELL, 2018).

Figura 12 – Possíveis direções de um projétil atirado por um NPC em *DOOM*, com o jogador em movimento.



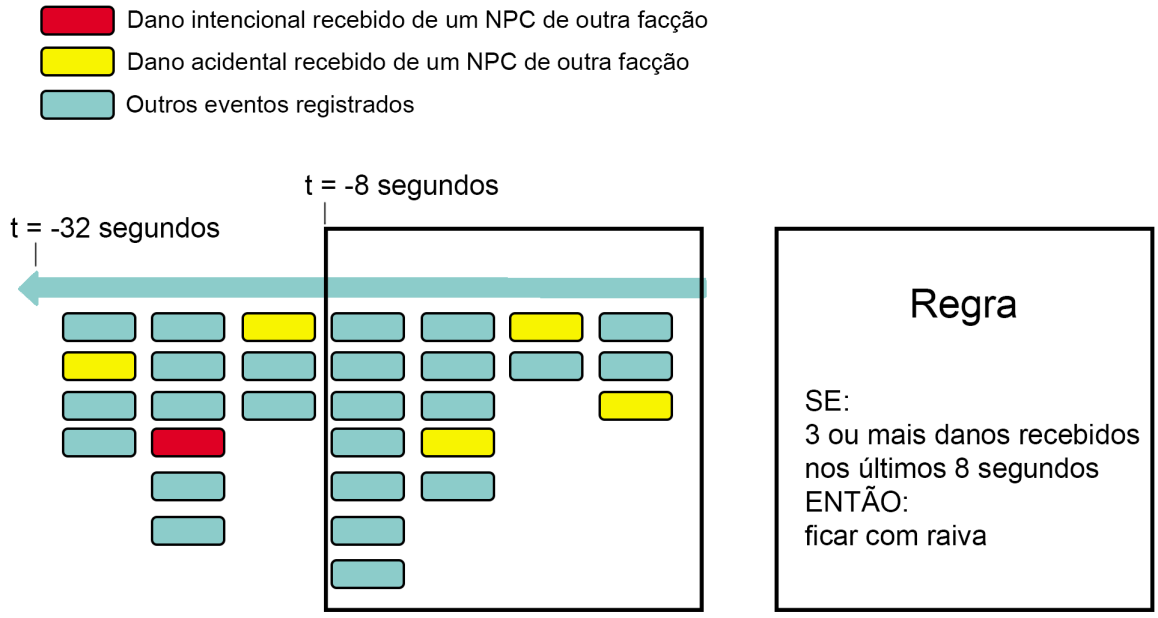
Fonte: Campbell (2018)

DOOM também possui um sistema que limita a quantidade de inimigos realizando ataques ao mesmo tempo, impedindo que todos fiquem repetindo o mesmo movimento e que o jogador seja pressionado por todos os oponentes simultaneamente. Para poder utilizar um

determinado ataque, um NPC precisa requisitar uma ficha ao sistema, que, por sua vez, mantém uma quantidade limitada de fichas disponíveis. Quando o sistema não tem mais fichas, o agente não poderá realizar o ataque desejado, sendo forçado a escolher outra ação, como outro tipo de ataque ou se movimentar pelo ambiente (CAMPBELL, 2018).

Os inimigos no *game* também podem iniciar brigas entre si, pois eles pertencem a diferentes facções. A princípio, as facções focam os ataques apenas ao jogador, porém, de acordo com eventos como um NPC de uma facção ser atingido por um golpe acidental ou proposital de um NPC de outra facção, uma briga entre elas pode ser desencadeada. O jogo armazena esses eventos ao longo do tempo e utiliza um sistema baseado em regras para determinar quando iniciar um confronto entre facções. A Figura 13 mostra um exemplo de regra, que faz o agente direcionar sua raiva para outra facção caso tenha sido atingido por disparos de NPCs pertencentes a ela nos últimos 8 segundos. Caso o jogador chame a atenção de um NPC que está brigando com outra facção, por exemplo, com um ataque, ele irá voltar sua atenção para o *player* por um tempo, mas depois voltará a brigar com a outra facção (CAMPBELL, 2018).

Figura 13 – Eventos armazenados ao longo do tempo e regra que faz um NPC direcionar seus ataques para NPCs de outra facção em *DOOM*.



Fonte: Adaptado de Campbell (2018)

Outro jogo que compartilha de algumas das técnicas utilizadas em *DOOM*, como a Máquina de Estados Hierárquica e alguns dos comportamentos determinísticos, é *Rage* (BETHESDA, 2011), um jogo de tiro em primeira pessoa e ação, também desenvolvido pela *id Software* e publicado pela *Bethesda*, lançado em 2011 (THOMPSON, 2018).

4.2 *F.E.A.R.*

F.E.A.R. (*First Encounter Assault Recon*) é um jogo de tiro em primeira pessoa produzido pela *Monolith Productions* (MONOLITH, 2022) e lançado em 2005. Durante o jogo, o jogador enfrenta diversos inimigos diferentes, mas que usam a mesma abordagem para as suas IAs.

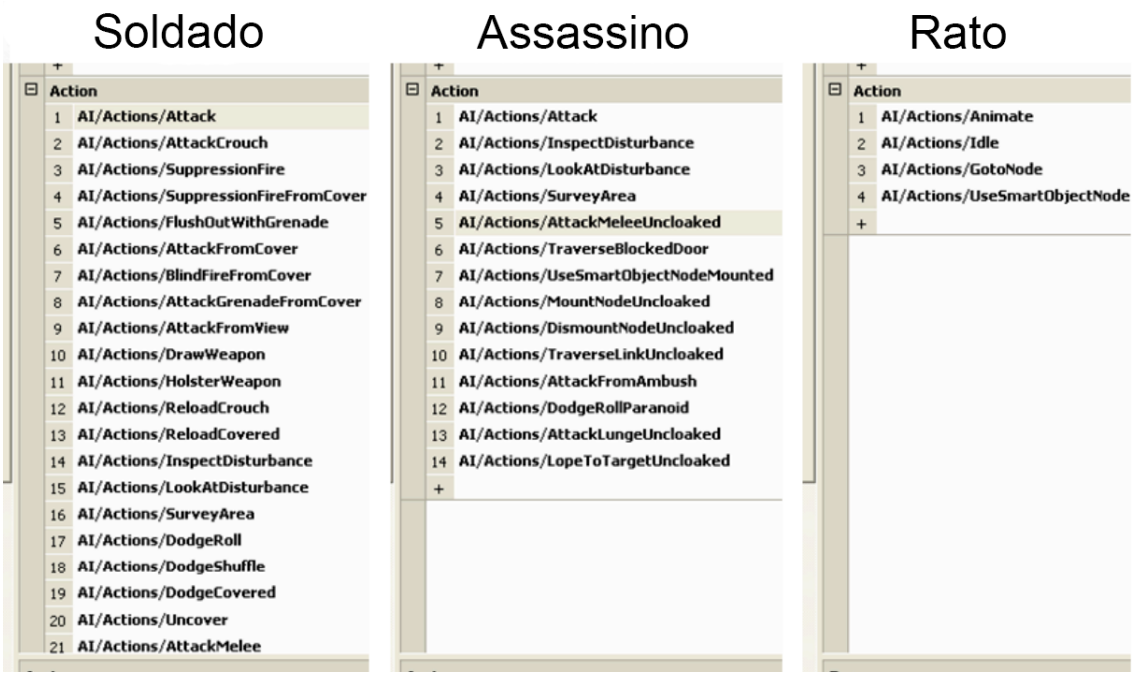
Os desenvolvedores utilizaram, como principal método de controle dos inimigos, o Planejamento de Ações Orientado a Objetivo, sendo o primeiro videogame comercial a implementar esse método (LONG, 2007). O sistema do jogo contém diversos objetivos que são distribuídos entre os NPCs, podendo atribuir mais de um objetivo para um único agente. Cada objetivo possui funções para calcular a sua prioridade no estado atual do jogo, por exemplo, se um NPC caracterizado como um soldado receber os objetivos "MatarInimigo" e "Patrulhar" e ele não possuir nenhuma informação de que o jogador está por perto, o objetivo "MatarInimigo" terá prioridade zero, e "Patrulhar" terá uma prioridade maior, contudo, se o soldado obtiver a informação de que o jogador está por perto, "MatarInimigo" terá sua prioridade aumentada (ORKIN, 2006).

Para que um NPC consiga atingir os seus objetivos, ele terá diversas ações disponíveis, com as quais o GOAP construirá um plano. Porém, ele precisa ser revalidado durante a execução, já que o estado do jogo pode ser alterado durante a execução pelas ações do jogador ou de outros NPCs, impedindo que o plano seja concluído, sendo necessário realizar o planejamento novamente (THOMPSON, 2020a). A Figura 14 mostra ações que podem ser realizadas por três NPCs diferentes. Na primeira coluna são apresentadas as ações possíveis para um character da classe Soldado. A coluna do meio representa a lista de ações aplicáveis a um Assassino enquanto na terceira coluna são listadas as ações de um personagem do tipo Rato. Através da ilustração é possível perceber a diferença de complexidade entre os tipos de classes apresentadas.

Ao executar as ações, é utilizada uma Máquina de Estados Finita. A MEF utilizada contém apenas três estados, "ExecutarAnimação", "Usar Objeto Inteligente" e "Se Movimentar", como demonstrado na Figura 15. O sistema transiciona entre os estados de acordo com a ação que está sendo executada. Além disso, os agentes em *F.E.A.R.* utilizam o algoritmo de busca *A** para formar rotas para moverem-se de um ponto a outro (ORKIN, 2006).

O comportamento descrito acima para os NPCs em *F.E.A.R.* explica o funcionamento individual de cada agente. Contudo, o jogo também conta com mecânicas para realizar ações em grupo, chamados de comportamentos de esquadrão. Comportamentos de esquadrão são controlados por um sistema global que, periodicamente, tenta agrupar os NPCs em esquadrões, com base na proximidade entre eles. Ao formar um esquadrão, o sistema dará uma ordem que fica a critério da IA individual dos agentes para decidir se será seguida ou não. Quando um comportamento de esquadrão é iniciado, ele pode ser concluído com sucesso ou ser interrompido por alguma mudança no estado do jogo, fazendo com que cada NPC volte ao planejamento individual. Orkin (2006) explica que o sistema de esquadrões funciona de maneira determinística,

Figura 14 – Ações do GOAP disponíveis para três NPCs diferentes no jogo *F.E.A.R.*.



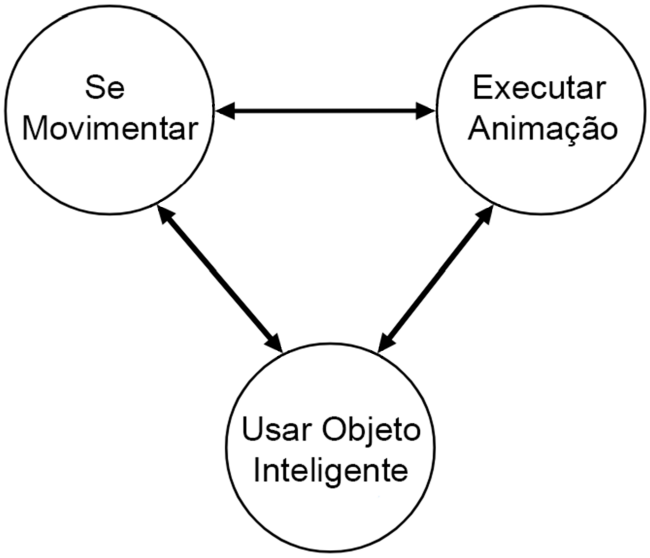
Fonte: Adaptado de Orkin (2006)

sendo uma implementação relativamente simples. Porém, a interação entre os comportamentos de esquadrão e os comportamentos individuais gera uma impressão de que os inimigos possuem uma maior complexidade.

A IA implementada no videogame demonstra resultados interessantes durante o jogo. Os inimigos avançam em grupo quando estão em grande número, interagem com objetos do cenário para criar locais para se esconderem, tentam buscar esconderijos com frequência, jogam granadas para encurralar o jogador e realizam outros comportamentos que passam uma impressão realista sobre os NPCs (BRUDVIG, 2006). Esse resultado não é obtido apenas pela IA aplicada no jogo, mas, também, se deve a um bom *design* de personagens e de ambiente (ORKIN, 2006).

O GOAP é uma técnica utilizada em diversos outros jogos conhecidos, dentre eles, estão *F.E.A.R. 2: Project Origin* (MONOLITH, 2022), de 2009, *Middle-Earth: Shadow of Mordor* (MONOLITH, 2022), de 2017, também, desenvolvido pela *Monolith Productions*, sendo um RPG que usa o GOAP no controle dos *Orcs*, os inimigos presentes no *game*, e *S.T.A.L.K.E.R.: Shadow of Chernobyl* (GSC, 2022), um jogo de tiro em primeira pessoa e sobrevivência que foi desenvolvido pela *GSC Game World* e lançado em 2007 (CHAMPANDARD; KUMAR, 2008; THOMPSON, 2020a).

Figura 15 – MEF utilizada nos NPCs do jogo *F.E.A.R.*.



Fonte: Adaptado de Orkin (2006)

4.3 *Alien: Isolation*

Alien: Isolation, desenvolvido pela *Creative Assembly* (CREATIVE ASSEMBLY, 2022) e lançado em 2014, é um jogo de aventura e terror, onde o jogador enfrenta um inimigo principal, conhecido como *Xenomorfo*, que é considerado invencível e pode derrotar o jogador em apenas um golpe, portanto, o objetivo em relação ao NPC é evitar ser pego por ele.

Por ser o elemento principal do jogo, o *Xenomorfo* necessita de um comportamento que o faça parecer natural e ameaçador. Para isso, o jogo gerencia o NPC através de dois sistemas, a IA do próprio agente e um sistema macro, chamado de diretor, que possui as informações de tudo o que está acontecendo no jogo, incluindo a posição e progresso atual do jogador (THOMPSON, 2017; CALEGARI, 2021).

O diretor aponta, periódicamente, a localização aproximada do jogador ao *Xenomorfo*, garantindo que o NPC não fique muito tempo sem aparecer para perseguí-lo. Outra função do sistema macro é medir o nível de ameaça que o jogador está enfrentando, baseando-se em informações como a distância entre o *player* e o inimigo, se o jogador tem uma linha de visão com o NPC e a capacidade dele alcançar o *player* rapidamente. Quando o nível de ameaça atinge um nível muito alto, porém o jogador não foi detectado pelo *Xenomorfo*, o NPC é enviado para outro lugar, permitindo que o jogador avance com os objetivos do jogo (THOMPSON, 2017; CALEGARI, 2021).

A IA presente no principal inimigo do jogo utiliza uma árvore de comportamento que, segundo Thompson (2017) e Thompson (2020b), possui mais de 100 nós no total e 36 nós no primeiro nível após a raiz, que definem o tipo de comportamento que o NPC irá realizar, sendo

que cada um desses comportamentos pode acontecer de formas diferentes, pois são controlados pelos nós abaixo deles na hierarquia da BT. As ações do *Xenomorfo* são focadas na busca pelo jogador e, para isso, o NPC patrulha os corredores do ambiente, reage e investiga os ruídos que ele detecta e vasculha as salas do cenário (CALEGARI, 2021).

Um recurso interessante da árvore de comportamento do *Xenomorfo* é a capacidade de desbloquear comportamentos em ramificações que iniciam bloqueadas. O desbloqueio de uma ação do NPC ocorre de acordo com as ações do jogador, porém, quando o *Xenomorfo* consegue capturar o jogador, o progresso para a liberação de partes da BT não é contado, ou seja, a árvore só desbloqueia novos comportamentos em situações onde o *player* foi bem sucedido em escapar, impedindo-o de usar sempre o mesmo truque. Um exemplo de comportamento que inicia bloqueado na IA do *Xenomorfo* é a busca pelo jogador em armários, que é desbloqueada após o mesmo se esconder dessa forma repetidas vezes (THOMPSON, 2017; WHEELER, 2018; CALEGARI, 2021). Segundo Thompson (2017), esse recurso passa a impressão de que o NPC está aprendendo conforme as experiências que vive.

O *Xenomorfo* também possui um sistema de *pathfinding* com um algoritmo de busca que usa sensores do NPC, que simulam sua visão e audição, como heurística (THOMPSON, 2017).

A Inteligência Artificial aplicada ao inimigo central de *Alien: Isolation* atinge um resultado satisfatório ao tentar criar uma atmosfera de tensão para o jogo. Segundo a análise de Wheeler (2018), feita para o portal *The Review Geek*, a IA do *Xenomorfo* é o que diferencia o *videogame* de outros jogos de terror.

A principal técnica utilizada no controle do *Xenomorfo*, a árvore de comportamento, também é utilizada em diversos outros jogos comerciais, dentre eles, estão *Halo 2* (STEAM, 2020a), de 2004 e (STEAM, 2020b), de 2007, ambos jogos de tiro em primeira pessoa e aventura, produzidos pela *Bungie*, *Bioshock* (2K, 2007), um jogo de tiro em primeira pessoa e RPG, produzido pela *2K Games* e lançado em 2007 e *Spore* (EA, 2009), um jogo de simulação, desenvolvido pela *Maxis* e lançado em 2012 (ISLA, 2005; AGIS; GOTTIFREDI; GARCIA, 2020; LIM; BAUMGARTEN; COLTON, 2020).

4.4 *Horizon Zero Dawn*

Horizon Zero Dawn é um RPG de ação em mundo aberto, produzido pela *Guerrilla Games* (GUERRILLA, 2022) e lançado em 2017. Dentro do *game*, existem diversas criaturas robóticas que são inspiradas em animais. Essas criaturas habitam o cenário do jogo com um comportamento passivo, porém, também podem entrar em combate com o jogador.

Cada NPC possui uma IA individual, sendo chamados de agentes individuais. Mas também existem agentes de grupos, que ajudam a definir o comportamento de NPCs coletivamente. Cada agente individual pertence a um agente de grupo, que, por sua vez, pode possuir outros agentes

de grupos ou um ou mais indivíduos que comanda (GUERRILLA, 2017; THOMPSON, 2019).

Cada agente de grupo armazena informações que podem ser interessantes para os NPCs que o pertencem. Isso ajuda no desempenho do jogo, pois não é necessário que cada indivíduo armazena essas informações, e também permite que as informações sejam compartilhadas entre todos os agentes individuais do grupo, dessa forma, possibilitando que o grupo inteira reaja a acontecimentos que foram observados por um único NPC (THOMPSON, 2019).

Os agentes em *Horizon Zero Dawn* utilizam a técnica de Planejamento com Redes de Tarefas Hierárquicas. Os NPCs individuais podem formular planos para realizar ações específicas, como visitar novas áreas, atacar um humano ou fugir. Agentes de grupo podem executar planos que ditam os objetivos que os indivíduos pertencentes a ele devem alcançar, podendo, também, alterar a estrutura do grupo (GUERRILLA, 2017; THOMPSON, 2019).

Apesar da influência dos agentes de grupo, a IA individual de cada criatura ainda tem bastante influência sobre seu comportamento e os objetivos que ela irá buscar. Ao mesmo tempo, agentes de grupo mantém os indivíduos coordenados, reagrupando-os quando necessário ou até associando funções para cada NPC, para que eles possam apoiar o grupo (GUERRILLA, 2017; THOMPSON, 2019).

Outro jogo que utiliza o Planejamento com HTN é *Killzone 2* (GUERRILLA, 2022), também desenvolvido pela *Guerrilla Games* e lançado em 2009, um jogo de tiro em primeira pessoa tático. *Killzone* foi o primeiro jogo no qual a *Guerrilla Games* implementou as HTNs e, a partir do seu lançamento, a desenvolvedora passou a utilizar essa técnica em outros de seus jogos (THOMPSON, 2019).

4.5 Franquia *Forza*

Forza é uma série de videogames de corrida distribuídos pela *Xbox Game Studios* (MICROSOFT, 2022) . O primeiro jogo da franquia, *Forza Motorsport* foi lançado em 2005 e introduziu o recurso conhecido como *Drivatar*, uma Inteligência Artificial desenvolvida com Redes Neurais que tem como objetivo simular o comportamento de um jogador ao conduzir os carros do jogo, que continuou a ser implementada em todos os outros jogos da franquia (THOMPSON, 2021).

Cada jogador possui o seu próprio *Drivatar*, que tenta aprender com o seu comportamento para, então, substituí-lo em corridas de outros jogadores. O *Drivatar* não tenta reproduzir ações que o *player* executou previamente e, sim, busca ajustar sua RNA para poder tomar suas decisões a partir das entradas que o algoritmo recebe, porém, tentando tomar as mesmas decisões que o jogador tomaria em uma mesma situação. Esse funcionamento permite que a IA simule o comportamento do jogador em situações que ele ainda não presenciou e com carros que ele nunca pilotou (THOMPSON, 2021).

Outra vantagem do uso das Redes Neurais na franquia é a capacidade de lidar com uma grande quantidade de variáveis que influenciam os movimentos dos carros. Ao conduzir um veículo, o jogador ou o *Drivatar* terá que considerar elementos como o peso e a potência do veículo, o tipo dos seus pneus, o clima e muitos outros, criando uma enorme variedade de situações possíveis. Seria muito difícil, para os desenvolvedores, criar regras lógicas para gerenciar todas essas situações, contudo, com as RNAs, a solução é mais simples, pois basta informar todos os elementos que a IA deve considerar aos neurônios de entrada da rede e, então, ela mesma irá decidir como proceder (THOMPSON, 2021).

A estrutura da RNA é a mesma para todo *Drivatar*, porém, o processo de aprendizado é feito modificando os pesos das entradas de cada neurônio da Rede Neural. Para que a rede possa ser treinada, é necessária uma base de dados com informações sobre o comportamento do jogador. Para popular essa base, o jogo divide as pistas de corrida em segmentos e os classifica em diferentes categorias, dessa forma, armazenando dados sobre o comportamento do jogador para cada categoria de segmento de pista. O sistema também leva em consideração as datas em que as informações da base foram adquiridas, priorizando as mais recentes, pois o jogador pode alterar a forma como joga, fazendo com que os dados antigos não sejam tão fiéis ao seu comportamento atual (THOMPSON, 2021).

4.6 Histórico da aplicação de IA nos Videogames

As primeiras técnicas de IA utilizadas em jogos eletrônicos eram compostas por processos bastante rústicos, havendo, até, jogos que não possuiam NPCs para serem controlados e, portanto, não utilizavam nenhuma técnica de IA para isso. Com o passar do tempo e com a evolução da tecnologia, os NPCs deixaram de ser controlados, apenas, por simples Algoritmos Determinísticos e passaram a utilizar Máquinas de Estados e, posteriormente, outras técnicas ainda mais complexas.

As Tabelas 1 e 2 exibem um histórico de técnicas utilizadas em jogos eletrônicos. Até o ano de 2001, a Tabela 1 contém as mesmas informações apresentadas por Schwab (2009). Porém, as informações presentes na Tabela 2, que focam em jogos publicados a partir do ano de 2004 até o presente, foram preenchida com base nos jogos analisados nesse trabalho.

4.7 Considerações Finais

Observamos uma grande variedade de técnicas para o controle de NPCs e, ao analisar suas aplicações em *videogames*, percebemos que, na maioria dos casos, as abordagens são aplicadas de forma combinada.

Dentre os métodos observados, a maioria é utilizada com frequência na indústria dos jogos eletrônicos, com exceção apenas do Raciocínio Baseado em Casos, que não foi implementado

Tabela 1 – Histórico da utilização das técnicas de IA em jogos - Parte 1. Adaptado de Schwab (2009).

Ano	**Descrição**	**Técnica Utilizada**
1962	Primeiro jogo de computador, *SpaceWar*, para 2 jogadores.	Nenhuma
1972	Lançamento do jogo *Pong*, para 2 jogadores.	Nenhuma
1974	Jogadores tinham que atirar em alvos móveis em *Pursuit* e *Qwak*.	Algoritmos Determinísticos
1975	*Gun Fight*, personagem com movimentos randômicos.	Algoritmos Determinísticos
1978	*Space Invaders*, contém inimigos com comportamento definido por padrões.	Algoritmos Determinísticos
1980	O jogo *Pac-man* conta com movimentos padronizados dos inimigos, porém cada fantasma (inimigo) tem uma "personalidade" sobre o modo em que caça o jogador.	Algoritmos Determinísticos
1990	*Herzog Wei*, primeiro jogo de estratégia em tempo real. Continha um algoritmo de busca impreciso.	Máquinas de Estados
1993	*Doom*, primeiro jogo de tiro em primeira pessoa, primeiro jogo desenvolvido em linguagem de alto nível (linguagem C) e possibilitando partidas em rede.	Máquinas de Estados
1996	*BatleCruiser: 3000AD*, primeiro jogo comercial desenvolvido com redes neurais.	Redes Neurais
1997	*Deep Blue* realizou partidas contra o campeão mundial de xadrez, Garry Kasparov.	Algoritmos de Busca e Aprendizado de Máquina.
1998	*Half-Life*, jogo de tiro em primeira pessoa, visto como o título de melhor IA até sua época.	Algoritmos Determinísticos e Máquinas de Estados.
2001	*Black & White*, contém inimigos que aprendem com decisões feitas pelo jogador.	Redes neurais.

em nenhum dos jogos comerciais analisados, e das Redes Neurais Artificiais, que está presente numa quantidade relativamente pequena de *videogames*.

Apesar de não detectarmos a aplicação de RBC em jogos comerciais, a técnica aparece em jogos desenvolvidos em trabalhos científicos. Contudo, mesmo que apresentem bons resultados, suas aplicações se limitam a jogos muito simples. Dessa forma, não podemos afirmar que o RBC teria resultados satisfatórios em jogos comerciais complexos. Outro fator limitante da aplicação do RBC é a necessidade de um treinamento inicial, que gere, ao NPC, casos suficientes para que ele tenha um comportamento interessante já no início do jogo. Acreditamos entretanto, que a estratégia pode ser empregada localmente em parceria com outras abordagens mais avançadas.

Algoritmos Determinísticos estão presentes em quase todos os jogos, seja como método predominante de controle de NPCs ou apenas em funcionalidades pontuais. Em jogos mais simples, como, por exemplo, os de plataforma, é possível que toda a IA do jogo seja construída apenas com Algoritmos Determinísticos pois, normalmente esses jogos não se propõem a simular um comportamento realista.

As Máquinas de Estados Finitas também são muito comuns nos jogos, pois ajudam na organização do sistema, dividindo comportamentos em módulos, que são os estados da MEF. Contudo, em alguns jogos mais recentes, usar apenas a MEF não foi suficiente, já que a complexidade da IA desejada no comportamento dos NPCs acaba fazendo com que a Máquina de Estados contenha muitos estados e transições entre eles, aumentando a dificuldade de gerenciar o

Tabela 2 – Histórico da utilização das técnicas de IA em jogos - Parte 2.

Ano	**Descrição**	**Técnica Utilizada**
2004	*F.E.A.R.*, um jogo de tiro em primeira pessoa onde os inimigos planejam sequências de ações para desempenhar contra o jogador, além de executarem comportamentos em esquadrão.	Planejamento de Ações Orientado a Objetivo (ORKIN, 2006).
2004	*Halo 2*, um jogo de tiro em primeira pessoa e aventura.	Árvore de Comportamento (CHAMPANDARD; KUMAR, 2008).
2007	*S.T.A.L.K.E.R.: Shadow of Chernobyl*, um jogo de tiro em primeira pessoa e sobrevivência.	Planejamento de Ações Orientado a Objetivo (CHAMPANDARD; KUMAR, 2008).
2007	*Halo 3*, um jogo de tiro em primeira pessoa e aventura.	Árvore de Comportamento (CHAMPANDARD; KUMAR, 2008).
2007	*BioShock*, um jogo de tiro em primeira pessoa e RPG.	Árvore de Comportamento (CHAMPANDARD; KUMAR, 2008).
2009	*F.E.A.R. 2: Project Origin*, um jogo de tiro em primeira pessoa.	Planejamento de Ações Orientado a Objetivo (THOMPSON, 2020a).
2009	*Killzone 2*, um jogo de tiro em primeira pessoa tático.	Planejamento com Redes de Tarefas Hierárquicas (THOMPSON, 2019).
2011	*Rage*, um jogo de tiro em primeira pessoa e ação.	Máquinas de Estados Hierárquicas e Algoritmos (THOMPSON, 2018).
2011	*Spore*, um jogo de simulação.	Árvore de Comportamento (LIM; BAUMGARTEN; COLTON, 2020).
2014	*Alien: Isolation*, um jogo de aventura e terror.	Árvore de Comportamento (CHAMPANDARD; KUMAR, 2008).
2016	*DOOM 2016*, um jogo de tiro em primeira pessoa e plataforma.	Máquinas de Estados Hierárquicas e Algoritmos Determinísticos (THOMPSON, 2018).
2017	*Middle-Earth: Shadow of Mordor*, um jogo de RPG de ação.	Planejamento de Ações Orientado a Objetivo (THOMPSON, 2020a).
2017	*Horizon Zero Dawn*, um RPG de ação em mundo aberto, com criaturas com comportamentos individuais e em bando.	Planejamento com Redes de Tarefas Hierárquicas (THOMPSON, 2019).
2005 - Presente	*Forza*, uma série de jogos de corrida onde os NPCs aprendem a simular o comportamento dos *players*.	Redes Neurais Artificiais (THOMPSON, 2020a).

desenvolvimento. As Máquinas de Estados Hierárquicas são uma alternativa para esses casos, pois permitem a mesma divisão em módulos, porém, em mais níveis.

Algoritmos de Busca costumam desempenhar uma função específica dentro dos *videogames*, o *Pathfinding*, por esse motivo e, também, por não existir outro método que dispute por essa função, eles acabam sendo empregados em muitos jogos.

Planejamento de Ações Orientado a Objetivo é um método que foi adaptado especi-

ficamente para controlar NPCs em jogos. Uma das suas vantagens é o fato de que as ações não são ligadas aos objetivos, possibilitando que o sistema possa escolher qualquer ação para cumprir um objetivo, e não apenas ações predefinidas para isso. Contudo, mesmo que o GOAP possa tomar suas próprias ações, as decisões do desenvolvedor ainda têm um grande peso no seu funcionamento, pois os custos definidos para cada ação e a forma como os objetivos são priorizados faz com que exista uma predisposição para que algumas ações sejam tomadas em casos especificos.

Uma das técnicas que mais se popularizou nos últimos anos são as Árvores de Comportamentos. As BTs têm uma semelhança com as MEFHs, pois os níveis mais altos da hierarquia especificam comportamentos de forma genérica e, quanto mais baixo for um nó na árvore, mais específico é o comportamento definido por ele. Diferente do GOAP, os objetivos presentes na BT definem quais ações podem ser tomadas para cumprí-los. BTs também são fáceis de serem modificadas, pois a alteração de um nó só afeta os nós abaixo dele na hierarquia, dessa forma, é possível alterar, remover ou adicionar nós sem ter que alterar o resto da árvore.

O Planejamento com Redes de Tarefas Hierárquicas se assemelha bastante com o GOAP e com as Árvores de Comportamento, pois usa os mesmos princípios de ações e objetivos do GOAP e forma estruturas parecidas com as BTs, porém essa técnica ainda apresenta diferenças em relação a ambos. Comparado ao GOAP, os NPCs desenvolvidos com HTNs têm menos liberdade de decisão, já que as tarefas primitivas a serem realizadas já são escolhidas com base nas tarefas acima dela na hierarquia, porém, isso também garante mais controle na hora do desenvolvimento. Em comparação às BTs, as HTNs possuem um sistema diferente de escolha das decisões a tomar, uma vez que as primeiras tomam decisões com base na estrutura da árvore enquanto as HSF, o fazem com base nos objetivos e suas prioridadades.

As Redes Neurais Artificiais não estão entre as técnicas mais populares no controle de NPCs pois uma das maiores vantagens em utilizá-las, a capacidade de aprendizado e adaptação, não é algo que está dentro da proposta da maioria dos *videogames*. Outro fator é a dificuldade de implementação. Até mesmo *Alien: Isolation*, que tenta passar a impressão de que o inimigo está aprendendo com as ações do jogador, não optou por esse método.

5
Conclusão

Neste trabalho foi feito um levantamento das técnicas de Inteligência Artificial mais comuns aplicadas ao controle de characteres não jogáveis em jogos eletrônicos.

O conjunto de técnicas observadas varia bastante de acordo com o objetivos dos desenvolvedores. Quanto mais fiel à realidade se pretende criar o jogo, mais avançada e complexa é a estratégia inteligente usada. Observamos que métodos básicos como as máquinas de estado finita e algoritmos determinísticos têm uma aplicabilidade bastante restrita. Isso foz com que sua aplicação seja viável para jogos mais simples ou que sejam aplicadas em conjuntos com outras técnicas.

As técnicas que possibilitaram simular comportamentos mais inteligentes foram as árvores de comportamento, o planejamento de ações com base no objetivo buscado e não com base no objeto e aplicação de redes hierárquicas de tarefas. Justamente por serem abordagens mais avanaçadas e complexas elas foram propostas apenas recentemente e são mais indicadas para jogos mais complexos, como F.E.A.R., *Alien: Isolation*, DOOM 2016 e *Horizon Zero Dawn*. Geralmente tais estratégias são desenvolvidas pela equipe do estúdio e então incluídas nos *engines*.

De todas as aplicações das técnicas vistas, nenhuma oferece liberdade ao NPC para tomar decisões de forma completamente autônoma, ou seja, os desenvolvedores conseguem manter o controle para que os NPCs se comportem de uma forma que atenda aos seus requisitos. Dessa forma, é possível reproduzir resultados diferentes com a mesma técnica de IA, pois os resultados são muito dependentes das variações dentro da própria técnica ou de características desatreladas a ela, como o design de personagem e de ambiente. Por esse motivo, é difícil diferenciar as técnicas analisando somente os resultados que elas proporcionam. Entretanto, notamos várias diferenças de implementação dessas técnicas, envolvendo características como a modularidade e capacidade de se adaptar a sistemas muito complexos.

Um dos possíveis fatores a serem considerados pelas empresas de desenvolvimento de

jogos na escolha das técnicas de IA utilizadas para o controle de NPCs são as *Game Engines* que elas utilizam. É comum que empresas tenham uma *engine* própria, que proporciona ferramentas para o desenvolvimento da IA nos NPCs, portanto, vemos empresas utilizando abordagens semelhantes em vários de seus jogos.

A pesquisa realizada neste trabalho representa um estudo mais abrangente que está em estado incipiente. Há ainda margem espaço para aprofundamento maior no tema. Para explorar a área de desenvolvimento de IA para controle de NPCs em jogos eletrônicos com maior compreenção e análise, seria interessante o desenvolvimento de implementações das técnicas apresentadas e comparações entre os diferentes métodos de IA, analisando os resultados no comportamento dos NPCs e o desempenho computacional apresentado por eles, estimando, por exemplo, quais técnicas demandam mais tempo de processamento.

Outra frente a ser explorada consiste na extensão do levanatamento realizado neste trabalho, abordando possíveis novas técnicas e *videogames* que não apresentados neste estudo, bem como inovações surgidas a partir da data presente.

Referências

2K. *BioShock*. 2007. Disponível em: <https://www.2k.com/en-US/game/bioshock/>. Acesso em: 14 ago 2022. Citado na página 33.

AAMODT, A.; PLAZA, E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, v. 7, n. 1, p. 39–59, Ago 1994. Citado 2 vezes nas páginas 18 e 19.

AGIS, R. A.; GOTTIFREDI, S.; GARCIA, A. J. An event-driven behavior trees extension to facilitate non-player multi-agent coordination in video games. *Expert Systems with Applications*, v. 155, n. 10, p. 1–15, Out. 2020. Citado na página 33.

APPERLEY, T. Genre and game studies: Toward a critical approach to video game genres. *Simulation & Gaming*, v. 37, n. 1, p. 6–23, Mar. 2006. Citado na página 12.

BATTAIOLA, A. L. Jogos por computador – histórico relevância tecnológica e mercadológica, tendências e técnicas de implementação. In: *Anais da XIX Jornada de Atualização em Informática (JAI)*. [S.l.]: SBC, 2000. v. 2, p. 83–122. Citado na página 10.

BETHESDA. *Rage*. 2011. Disponível em: <https://bethesda.net/pt/games/product/RA1CSTPCDENA>. Acesso em: 14 ago 2022. Citado na página 29.

BETHESDA. *DOOM*. 2016. Disponível em: <https://bethesda.net/pt/game/doom-2016>. Acesso em: 14 ago 2022. Citado na página 27.

BJARNOLF, P. et al. Threat analysis using goal-oriented action planning. Ago. 2008. Citado na página 19.

BORANA, J. *Applications of Artificial Intelligence & Associated Technologies*. Dissertação (Monografia) — Jodhpur National University, Jodhpur, Rajasthan, India, 2016. Citado na página 14.

BOURG, D. M.; SEEMAN, G. *AI for Game Developers*. 1. ed. Sebastopol, Califórnia, Estados Unidos da América: O'Reilly, 2004. Citado 2 vezes nas páginas 14 e 16.

BRUDVIG, E. *F.E.A.R. Review: Prepare to be scared.* 2006. Disponível em: <https://www.ign.com/articles/2006/10/25/fear-review-2>. Acesso em: 13 jul 2022. Citado na página 31.

CALEGARI, E. Z. *Análise da Jogabilidade e da IA Presente em The Sims e Alien: Isolation*. Dissertação (Monografia) — Universidade do Sul de Santa Catarina, Florianópolis, Santa Catarina, 2021. Citado 2 vezes nas páginas 32 e 33.

CAMPBELL, K. L. e J. *Embracing Push Forward Combat in DOOM*. 2018. Disponível em: <https://youtu.be/2KQNpQD8Ayo>. Acesso em: 16 jul 2022. Citado 3 vezes nas páginas 27, 28 e 29.

CHAMPANDARD, A. J.; KUMAR, M. *Interview: Inside The AI Of S.T.A.L.K.E.R.* 2008. Disponível em: <https://www.gamedeveloper.com/pc/interview-inside-the-ai-of-i-s-t-a-l-k-e-r-i->. Acesso em: 14 jul 2022. Citado 2 vezes nas páginas 31 e 37.

CO, P. *Level design for games - Creating compelling game experiences*. 1. ed. Berkeley, CA: New Riders, 2006. Citado na página 13.

COLLEDANCHISE, M.; OGREN, P. *Behavior Trees in Robotics and AI: An Introduction*. 1. ed. Flórida, EUA: CRC Press, 2019. Citado 4 vezes nas páginas 20, 21, 22 e 23.

CREATIVE ASSEMBLY. *Creative Assembly*. 2022. Disponível em: <https://www.creative-assembly.com>. Citado na página 32.

DALMAU, D. S.-C. *Core Techniques and Algorithms in Game Programming*. 2. ed. Indianápolis, Indiana, Estados Unidos da América: New Riders, 2004. Citado na página 17.

DUCHON, F. et al. Path planning with modified a star algorithm for a mobile robot. *ScienceDirect*, v. 96, p. 59–69, 2014. Citado na página 17.

EA. *Spore*. 2009. Disponível em: <https://www.spore.com>. Acesso em: 14 ago 2022. Citado na página 33.

EDELKAMP, S.; SCHRODL, S. *Heuristic Search: Theory and Applications*. 1. ed. Amsterdã, Holanda: Elsevier, 2011. Citado na página 17.

FURTADO, M. I. V. *Redes Neurais Artificiais: Uma Abordagem Para Sala de Aula*. 1. ed. Ponta Grossa, Paraná: Atena Editora, 2019. Citado 2 vezes nas páginas 24 e 25.

GAME DEVELOPERS CONFERENCE. *Game Developers Conference*. 2022. Disponível em: <https://gdconf.com>. Acesso em: 14 ago 2022. Citado na página 26.

GSC. *GSC Game World*. 2022. Disponível em: <https://www.gsc-game.com>. Acesso em: 14 ago 2022. Citado na página 31.

GUERRILLA. *The AI of Horirzon Zero Dawn*. 2017. Disponível em: <https://www.guerrilla-games.com/read/the-ai-of-horizon-zero-dawn>. Acesso em: 17 jul 2022. Citado na página 34.

GUERRILLA. *Guerrilla Games*. 2022. Disponível em: <https://www.guerrilla-games.com/games>. Acesso em: 14 ago 2022. Citado 2 vezes nas páginas 33 e 34.

GUIMARÃES, Y. R. *Um Jogo de Nave Evolutivo*. Dissertação (Monografia) — UNIOESTE – Universidade Estadual do Oeste do Paraná, Cascavel, Paraná, 2009. Citado 2 vezes nas páginas 12 e 18.

HANSEN, E. A.; ZHOU, R. Anytime heuristic search. *Journal Of Artificial Intelligence Research*, v. 28, n. 1, p. 267–297, 2007. Citado na página 17.

ISLA, D. *GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI*. 2005. Disponível em: <https://www.gamedeveloper.com/programming/gdc-2005-proceeding-handling-complexity-in-the-i-halo-2-i-ai>. Acesso em: 15 jul 2022. Citado na página 33.

JOHANSSON, T. *Level design in open worlds - Should you think like an architect?* Dissertação (Monografia) — Instituto de Tecnologia de Blekinge, Karlskrona, Suécia, 2014. Citado na página 13.

KARLSSON, B. F. F. *An Artificial Intelligence Middleware for Digital Games*. Dissertação (Monografia) — PUC-RIO - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Rio de Janeiro, 2005. Citado 2 vezes nas páginas 15 e 16.

KENT, S. L. *The Ultimate History of Video Games*. 1. ed. New York, New York: Three Rivers Press, 2001. Citado 3 vezes nas páginas 8, 10 e 11.

KISHIMOTO, A. *Inteligência Artificial em Jogos Eletrônicos*. 2004. Disponível em: <http://www.karenreis.com.br/pdf/andre_kishimoto.pdf>. Acesso em: 04 jun 2022. Citado 2 vezes nas páginas 14 e 16.

LAMOTHE, A. *Tricks of the Windows Game Programming Gurus – Fundamentals of 2D And 3D Game Programming*. 1. ed. Indianápolis, Indiana, Estados Unidos da América: Sams Publishing, 1999. Citado 2 vezes nas páginas 15 e 17.

LIM, C.; BAUMGARTEN, R.; COLTON, S. *Evolving Behaviour Trees for the Commercial Game DEFCON*. Dissertação (Monografia) — Imperial College London, Londres, Inglaterra, 2020. Citado 2 vezes nas páginas 33 e 37.

LONG, E. *Enhanced NPC Behaviour using Goal Oriented Action Planning*. Dissertação (Monografia) — University of Abertay Dundee, Dundee, Escócia, 2007. Citado na página 30.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, Illinois , Estados Unidos da América, v. 5, p. 115–133, 1943. Citado na página 14.

MICROSOFT. *Forza*. 2022. Disponível em: <https://forza.net>. Acesso em: 14 ago 2022. Citado na página 34.

MONOLITH. *Monolith Productions*. 2022. Disponível em: <https://lith.com/our-games/>. Acesso em: 14 ago 2022. Citado 2 vezes nas páginas 30 e 31.

ONTAÑÓN, S.; BURO, M. Adversarial hierarchical-task network planning for complex real-time games. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, p. 1652–1658, 2015. Citado na página 23.

ORKIN, J. Three states and a plan: The a.i. of f.e.a.r. In: *Proceedings of the Game Developers Conference 2006*. [S.l.: s.n.], 2006. Citado 6 vezes nas páginas 19, 20, 30, 31, 32 e 37.

PATEL, M. S.; JAISWAL, M. *Introduction to Artificial Intelligence*. 1. ed. Navamuvada, Lunawada, India: RED'SHINE Publication, 2021. Citado na página 14.

PIHLGREN, G. G. et al. *Realistic NPCs in Video Games Using Different AI Approaches*. 2016. Citado na página 12.

PURCHIO, L. *Rumo aos US$ 200 bi: estratégias da indústria de games para crescer mais*. 2021. Disponível em: <https://veja.abril.com.br/economia/rumo-aos-us200bi-as-estrategias-da-industria-de-games-para-crescer-mais/>. Citado na página 8.

RAMOS, H. *Design de Interface*. Dissertação (Monografia) — Universidade Federal De Minas Gerais, Belo Horizonte, Minas Gerais, 2004. Citado na página 13.

ROGERS, S. *Level UP: um Guia para o Design de Grandes Jogos*. 1. ed. São Paulo, SP, Brasil: Blucher, 2013. Citado 2 vezes nas páginas 10 e 11.

ROLLINGS, A.; MORRIS, D. *Game Architecture and Design: A New Edition*. Nova Iorque, Estados Unidos da América: New Riders, 2003. Citado 2 vezes nas páginas 12 e 13.

RUSSELL, S.; NORVIG, P. *Inteligência Artificial*. 3. ed. São Paulo, São Paulo: GEN LTC, 2010. Citado 3 vezes nas páginas 8, 14 e 24.

SCHWAB, B. *AI Game Engine Programming*. 2. ed. Boston, Massachusetts, Estados Unidos da América: Course Technology, 2009. Citado 2 vezes nas páginas 35 e 36.

SEKHAVAT, Y. A. Behavior trees for computer games. *International Journal of Artificial Intelligence Tools*, v. 26, n. 2, 2021. Citado na página 23.

STEAM. *Halo 2*. 2020. Disponível em: <https://store.steampowered.com/app/1064270/Halo_2_Anniversary/>. Acesso em: 14 ago 2022. Citado na página 33.

STEAM. *Halo 3*. 2020. Disponível em: <https://store.steampowered.com/app/1064271/Halo_3//>. Acesso em: 14 ago 2022. Citado na página 33.

STUDIAWAN, R.; HARIADI, M.; SUMPENO, S. Tactical planning in space game using goal-oriented action planning. *Journal on Advanced Research in Electrical Engineering*, v. 2, n. 1, Abr. 2018. Citado na página 19.

THOMPSON, T. *The Perfect Organism: The AI of Alien: Isolation*. 2017. Disponível em: <https://www.gamedeveloper.com/design/the-perfect-organism-the-ai-of-alien-isolation>. Acesso em: 14 jul 2022. Citado 2 vezes nas páginas 32 e 33.

THOMPSON, T. *Cyber Demons | The AI of DOOM (2016)*. 2018. Disponível em: <https://www.gamedeveloper.com/design/cyber-demons-the-ai-of-doom-2016->. Acesso em: 16 jul 2022. Citado 3 vezes nas páginas 27, 29 e 37.

THOMPSON, T. *Behind The AI of Horizon Zero Dawn (Part 1)*. 2019. Disponível em: <https://www.gamedeveloper.com/design/behind-the-ai-of-horizon-zero-dawn-part-1->. Acesso em: 17 jul 2022. Citado 2 vezes nas páginas 34 e 37.

THOMPSON, T. *Building the AI of F.E.A.R. with Goal Oriented Action Planning*. 2020. Disponível em: <https://www.gamedeveloper.com/design/building-the-ai-of-f-e-a-r-with-goal-oriented-action-planning>. Acesso em: 13 jul 2022. Citado 3 vezes nas páginas 30, 31 e 37.

THOMPSON, T. *Revisiting the AI of Alien: Isolation*. 2020. Disponível em: <https://www.gamedeveloper.com/design/revisiting-the-ai-of-alien-isolation>. Acesso em: 14 jul 2022. Citado na página 32.

THOMPSON, T. *How Forza's Drivatar Actually Works*. 2021. Disponível em: <https://www.gamedeveloper.com/design/how-forza-s-drivatar-actually-works>. Acesso em: 15 jul 2022. Citado 2 vezes nas páginas 34 e 35.

WANGENHEIM, C. G. von; WANGENHEIM, A. von. *Raciocínio Baseado em Casos*. 1. ed. Barueri, São Paulo: Malone Ltda, 2003. Citado na página 18.

WARPEFELT, H. *The Non-Player Character – Exploring the believability of NPC presentation and behavior*. Estocolmo, Suécia: [s.n.], 2016. Citado na página 12.

WHEELER, G. *Alien: Isolation PS4 Review*. 2018. Disponível em: <https://www.thereviewgeek.com/alienisolation-ps4review/>. Acesso em: 15 jul 2022. Citado na página 33.