Módulo de comportamento inteligente: estudo de caso em Jogo Tático

Rodrigo Marques de Bem
Curso de Jogos Digitais
Universidade Franciscana
Santa Maria, Brasil
rodrigomarques.debem@gmail.com

Alexandre de Oliveira Zamberlan
Curso de Jogos Digitais
Universidade Franciscana
Santa Maria, Brasil
alexz@ufn.edu.br

***Resumo*—O trabalho tem como foco o projeto, a modelagem e a implementação de um módulo que utiliza a técnica de Inteligência Artificial conhecida como Sistema Multiagentes para um jogo que foi projetado e construído em disciplinas do curso de Jogos Digitais da Universidade Franciscana. Para isso, foi realizada uma revisão bibliográfica sobre jogos digitais e Inteligência Artificial, com apoio de análise e discussão de trabalhos relacionados. O jogo do estudo de caso é um *Tatical* RPG. Dessa forma, o trabalho usa pesquisa exploratória com revisão bibliográfica seguido de estudo de caso. A metodologia de construção do jogo tem peculiaridades específicas de jogos eletrônicos. Como ferramentas de apoio para gestão de atividades e controle de versão de código foram utilizadas Trello e GitHub, respectivamente. A *engine* Unity e a linguagem C# são as ferramentas de desenvolvimento do jogo e do módulo de comportamento inteligente.**

Index Terms*—RPG, sistema multiagentes, unity, *multiplayer

I. Introdução

A Inteligência Artificial (IA) possui várias áreas e técnicas para a construção de sistemas inteligentes. Uma justificativa de usar IA em jogos eletrônicos é garantir uma experiência diferenciada para o jogador (uma partida, um novo desafio). Isto é, imprevisibilidade do comportamento dos personagens não controlados pelo jogador humano (*No Playable Character* - NPC) e mapeamento do padrão de jogada do jogador humano, devem garantir que a experiência do jogo seja única, ou mais aproximada da realidade desejada.

A aplicação de IA em jogos digitais pode ser vista como um campo de estudo em constante transformação. O uso de aplicações baseadas em regras, presentes nos primeiros jogos eletrônicos para simular comportamentos com características inteligentes são dificilmente aceitos hoje em dia como IA. A evolução da IA tem como resultado disponibilizar jogos que usam avançadas técnicas de IA para construir a aprendizagem e o raciocínio em personagens virtuais [1]. E essa evolução só aconteceu devido ao incremento dos recursos computacionais como memória e processador dos dispositivos que executam os jogos (*smartphones*, computadores, consoles, entre outros).

O estilo *Tactical* RPG é caracterizado pelos movimentos táticos dos jogadores (*players*) para se chegar à vitória. As partidas são disputadas em um mapa, na forma de *grid* (matriz) e são divididas em turnos. Os jogadores não realizam as jogadas ao mesmo tempo, um deve esperar que o outro realize um movimento, para que ele então possa jogar, como em uma partida de xadrez. Diferente do RPG tradicional em que o jogador pode explorar mapas e cenários, no *tatics* a ênfase da-se na batalha e nas estratégias. As principais inspirações para o desenvolvimento deste trabalho são os títulos (todos pertencendo ao gênero RPG tático):

- *Final Fantasy Tactics*[1];
- *Into the Breach*[2];
- e a franquia de jogos digitais *Fire Emblem*[3].

Dessa forma, o objetivo geral do trabalho é modelar, implementar e avaliar um módulo de comportamento inteligente usando Sistemas Multiagentes, em um jogo do gênero RPG tático.

Para que o objetivo geral seja alcançado, identificaram-se alguns objetivos específicos:

- pesquisar técnicas de IA aplicadas em RPG (trabalhos relacionados);
- identificar em RPG eventos, situações ou comportamentos que possam receber um módulo de comportamento inteligente;
- identificar as principais técnicas de IA e implementá-las em exemplos nos ambientes Visual Studio e Unity.

Este estudo é uma pesquisa exploratória com revisão bibliográfica apoiada em estudo de caso (jogo *Tactical* RPG). As ferramentas para o projeto e a implementação do módulo são:

- Trello (gestão de atividades);
- Github (versionamento de código);
- Linguagem de programação C#;
- *Engine* Unity 2019;
- Bibliotecas adicionais da Unity 2019.

Para auxiliar na compreensão, o texto está dividido em 5 seções. Na Seção II, são apresentados conceitos de Jogos no contexto RPG e de Inteligência Artificial, bem como os trabalhos relacionados, com foco em estudos que utilizaram RPG e/ou técnicas de Inteligência Artificial. Na Seção III, são discutidas a proposta e a descrição deste trabalho, além da metodologia e das ferramentas utilizadas. Por fim, as Considerações Finais e as Referências Bibliográficas.

[1]https://www.finalfantasy.com/
[2]http://subsetgames.com/itb.html
[3]https://fireemblem.nintendo.com/three-houses/

II. Revisão bibliográfica

Nesta seção, busca-se apresentar conceitos, relações, aplicações e trabalhos envolvendo jogos no estilo RPG *tactics* e técnicas de sistemas de comportamento inteligente.

Assim como no RPG tradicional, no *tatics* é possível jogar com diferentes classes de personagens, como guerreiros, magos, *orcs*, entre outros. De maneira geral, os jogadores ganham pontos de experiência ao vencer os combates, o que os deixam mais fortes como uma espécie de recompensa pelas vitórias. Algumas batalhas apresentam diferentes condições de vitória, como por exemplo, derrotar todos os inimigos presentes no mapa da partida ou conseguir sobreviver até um determinado número de turnos. Entre uma partida e outra, os jogadores podem equipar seus personagens ou até mesmo trocar de classes, mas essa última possibilidade depende do jogo, pois nem todos oferecem essa alternativa.

A. Técnicas de Inteligência Artificial para Sistemas de Comportamento Inteligente

A Inteligência Artificial (IA) pode ser categorizada em duas dimensões: centrada nos seres humanos e centrada na racionalidade. A primeira tem como base o comportamento inteligente (perceber, raciocinar, aprender e atuar). Já a segunda, baseia-se em modelos matemáticos [2].

Das inúmeras técnicas da área de IA, é possível citar e contextualizar nas dimensões definidas em [2], [3]:

- métodos de busca ou técnica de *pathfinding* ou movimentação (abordagem racionalista);
- sistemas multiagentes (sistemas baseados no comportamento de seres humanos);
- redes neurais (abordagem racionalista);
- lógica *fuzzy* ou difusa (abordagem racionalista);
- processamento da língua natural (abordagem racionalista);
- algoritmos genéticos (como métodos de busca avançados na abordagem racionalista).

A ideia é focar em Sistemas Multiagentes, pois é a técnica utilizada no trabalho.

1) Sistemas Multiagentes: Sistemas multiagentes (SMA), de acordo com [2], [3], são aqueles que conseguem perceber o ambiente em que estão inseridos com o uso de sensores ou outros dispositivos e agir sobre ele. Dessa maneira, os agentes são capazes de perceber o ambiente e se adaptarem as mudanças. Além disso, há o conceito de sociedade ou ambiente, em que agentes interagem entre si em um ambiente comum. Por meio dessa interação entre os agentes e com o ambiente, surgem um comportamento global inteligente [2], [4], [5]. Ainda segundo [2] um agente é um sistema computacional que está situado em um ambiente e que é capaz de agir autonomamente neste ambiente para atingir seus objetivos. A percepção do ambiente tem como finalidade representar os sentidos humanos como visão, audição e olfato, de modo que um agente possa perceber um ambiente de forma similar à humana. Como por exemplo, um jogador se aproxima de um inimigo de maneira silenciosa, este só é identificado quando se encontra no raio de ação do agente [1]. Um ponto destaque de SMA é a habilidade social de cada elemento da sociedade (agente) que pode interagir com outros elementos (enviar mensagens, enviar comandos, realizar questionamentos, etc).

Dessa forma, os agentes têm algumas características importantes:

- Autônomo: independente para com seus objetivos;
- Pró-ativo: possui uma meta;
- Reativo: reage rapidamente ao ambiente;
- Social: interage de maneira social e cooperativa com o restante do grupo.

Já agente pode ser classificado como [6]:

- reativo - a escolha da ação está diretamente situada na ocorrência de um conjunto de eventos ativadores que ele percebe no ambiente (captados por seus sensores ou por mensagens enviadas por outros agentes);
- cognitivo - a escolha da ação é um processo explícito e deliberativo. A ação pode ser escolhida por função de utilidade e realizada por meio de um plano e uma representação simbólica do ambiente. Um agente cognitivo é um agente racional que possui alguma representação explícita de seu conhecimento e objetivos.

O funcionamento de um agente racional pode ser visualizado por meio da Figura 1.

Em [7], foi realizada uma comparação das ferramentas de modelagem e implementação de SMA. Registra-se que a *engine* Unity possui quase todas as características dos ambientes de programação de SMA, por se tratar de um ambiente com programação orientada a objetos e a eventos. Ou seja, Unity garante que personagens e/ou agentes criados e programados na *engine* respondam aos estímulos (eventos) do ambiente e de outros elementos (personagens e/ou agentes). Por exemplo, na Unity há a opção *istrigger*, que serve como tratadora de eventos.

B. Jogos e a IA

Algoritmos de Inteligência Artificial, como métodos de busca, foram implementados nos primeiros jogos digitais da história e são formados por movimentos aleatórios, como algoritmos de perseguição e fuga [8]. No caso do jogo digital PACMAN (Figura 2) a técnica de método de busca A* é passível de ser implementada, pois os fantasmas devem encontrar o melhor caminho que os leve até o jogador (PACMAN). Na opinião de [9], A* é um dos algoritmos *pathfinding* mais comuns em jogos. Ele é de grande utilidade, pois pode-se alterar seu funcionamento, por meio da modificação ou alteração das heurísticas. Já para [1] heurísticas podem considerar elementos que se relacionam com as mecânicas do jogo, mas que, sozinhas, não representam o caminho mais adequado a ser utilizado na hora de realizar-se uma jogada.

O jogo eletrônico *Half-life 2* (Figura 3) além de aplicar a técnica de máquina de estados finitos, também utiliza a técnica de Sistemas Multiagentes. Nota-se que é natural modelar a problemática de combate entre entidades, abordada no jogo, uma vez que é visível a necessidade de que as entidades

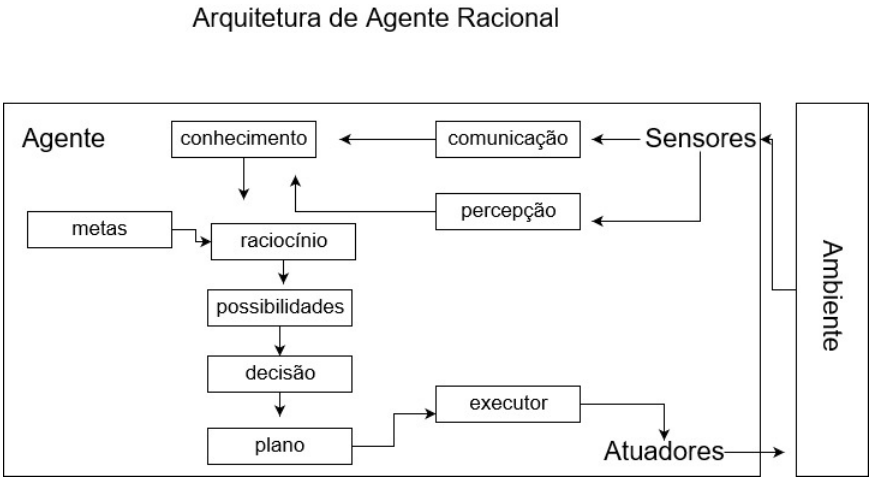


Figura 1. Diagrama do plano de execução de um agente racional (adaptado de [6]).

Figura 2. Gameplay do jogo Pac-Man.

precisam efetuar algum tipo de comunicação para que possam coordenar suas ações e trocar informações [1].

Figura 3. Half-Life 2.

Jogos de estratégia (*Civilization* de 1991, por exemplo) (Figura 4) foram os primeiros a trabalhar com técnicas de IA para jogos, uma vez que jogos deste gênero precisam de um módulo de comportamento inteligente apurado para que ofereçam uma boa jogabilidade, pois necessitam que o computador controle grupos de personagens com estratégias e táticas complexas [8]. Conforme [10], uma variação dos jogos de estratégia são os jogos de estratégia em tempo corrente, onde todas as ações acontecem no momento (ao contrário de outros jogos de estratégia, que ocorre em turnos). O módulo de comportamento inteligente para este tipo de jogo digital deve buscar caminhos (*pathfinding*) para centenas de unidades ao mesmo tempo.

Figura 4. Civilization 1.

C. Mecânica de Jogo

As mecânicas de um jogo eletrônico são toda e qualquer tipo de ação que o jogador pode realizar durante uma partida. A grosso modo, elas podem ser caracterizadas como sendo verbos que ditam as ações que o jogador pode fazer no jogo, como por exemplo, pular, atirar, esconder, caminhar, dirigir, atacar, enfim, o número de mecânicas depende do gênero do jogo e da narrativa do mesmo. Elas podem ser utilizadas como ferramentas de imersão o que proporciona uma experiência mais completa ao jogador. [11] conceitua as mecânicas como sendo os processos e as regras do jogo. Elas descrevem o objetivo do jogo e como os jogadores podem ou não atingi-lo e o que acontece quando eles agem na direção do objetivo. Para [12] mecânica é algum tipo de interação que o jogador pode desempenhar ou algum objetivo do *gameplay*. Por exemplo

mover plataformas, abrir portas, escalar cordas, deslizar no gelo, etc [12].

D. Trabalhos Relacionados

Para [13] RPG não é um jogo competitivo, mas focado na diversão com uso de inteligência e a imaginação,em cooperação com os demais jogadores para encontrar as melhores respostas para as situações apresentadas pela aventura. Para isso, SMA pode ser uma alternativa que garanta experiência diferenciada, pois com o uso de agentes (personagens) é possível promover a relação entre NPC, e NPC e jogadores humanos, por exemplo.

Em [14], é escrito que jogos de RPG são organizados para observar o comportamento dos jogadores. O objetivo deste tipo de jogo é geralmente testar hipóteses, ou de maneira mais geral, responder uma questão científica. O ambiente não é necessariamente realista. A motivação dos jogadores é alimentada pelo o que pode ser ganho no jogo. Este tipo de jogo foca no aprendizado individual e coletivo, com ênfase no comportamento coletivo a ser adotado para resolver um problema comum. Mais uma vez, a ideia de uso de SMA poderia gerar comportamentos autônomos dos personagens, interação social entre eles, por exemplo. Jogos baseados em agentes não impõe nenhum tipo de restrição adicional em comparação com a mediação do jogo de interpretação de papéis, onde os participantes devem seguir as regras do ambiente, as regras sociais e seus papéis. As regras sociais são combinadas pelo uso de primitivas que permitem aos participantes seguir livremente qualquer estratégia. Ou seja, o comportamento implementado nos agentes, logo seus papéis, serão obedecidos, sem a necessidade de intermediação.

Já para [15], em jogos de tabuleiro, algoritmos como árvores de busca podem enumerar exaustivamente estados futuros para avaliar a ação, devido à natureza tática do jogo e por que as ramificações são relativamente pequenas.Os jogos modernos apresentam uma variedade de agentes, cada um possuindo um vasto número de ações diversas, enquanto que peças de jogos de tabuleiro são definidas principalmente, apenas pelos tipos de movimentos que elas podem realizar.Além disso, todos os agentes presentes nos jogos modernos são capazes de agir simultaneamente no tempo corrente (*online*), em contraste com os jogos de tabuleiro baseados em turnos.

A enorme pluralidade dos tipos de jogos eletrônicos, pode acarretar em uma vasta interpretação do que que pode ser classificado de IA para jogos. Para alguns desenvolvedores, a interação entre a interface gráfica do jogo com o usuário pode ser considerada como IA, já para outros, algoritmos de movimentação e colisão também podem ser vistos como formas de IA [16]. Um jogo pode ter sim comportamentos inteligentes, mas Inteligência Artificial é algo ainda muito além do imaginado. Isso ocorre, porque para haver inteligência (como a de um humano), é necessário ser aprovado no Teste de Turing [2] e obrigatoriamente aprender durante o jogo.

Finalmente, um dos principais benefícios que a utilização de IA oferece ao desenvolvimento de jogos digitais é o elemento diversão. Os agentes controlados pela máquina em um jogo eletrônico devem dispor de inteligência e emular erros humanos e apresentar diferentes tipos de personalidades. Além disso, eles podem propiciar vários níveis de dificuldade ao jogador, para que ele sinta-se desafiado [17]. Segundo [8], a IA em jogos pode aumentar a experiência de imersão em um jogo digital,o que acarreta em uma melhor jogabilidade por parte do jogador. Personagens dotados de inteligência deixam jogos do gênero *single-player* mais atraentes, e podem acrescentar mais diversão também em jogos *multi-player* (para vários jogadores).

E. Metodologias para construção de jogos

A exemplo da indústria de software, em que há uma grande variedade de metodologias e técnicas empregadas para a construção de programas computacionais, na área dos jogos, também há inúmeros métodos utilizados para criar-se um jogo eletrônico. Dentre as metodologias ágeis aplicadas, uma delas é a Scrum, utilizada no desenvolvimento deste trabalho.

O método Scrum começa com o *Product Owner* ou (Dono do Produto), ele é o agente responsável que determina o que deve e não deve fazer parte do produto final. Além de definir os requisitos do projeto, o *Product Owner* elabora uma lista com todas as exigências e implementações que o produto final deve apresentar, o que se chama de *Product Backlog*. O *Product Backlog* necessita obedecer uma ordem de prioridades, por exemplo, em um jogo eletrônico, uma das primeiras prioridades do projeto seria a implementação das mecânicas básicas do jogo, pois sem elas, não teria como testá-lo. Após definir-se quais são os requisitos que devem fazer parte do *Product Backlog*, a equipe de desenvolvimento planeja os *Sprints*, que são períodos de tempo pré-determinados, que podem ser mensais ou semanais, onde se divide as tarefas presentes no *Product Bakclog*. Por exemplo, no desenvolvimento de um jogo digital, o primeiro *Sprint* pode ser dividido por áreas de atuação, como a artística,*game design*, programação, etc. Após o término de cada *Sprint*, é realizada um retrospectiva para analisar e discutir o que pode ser melhorado no próximo. Outra característica presente no Scrum é a realização de reuniões diárias, que servem para atualizar a equipe de desenvolvimento a respeito do andamento do projeto. Elas duram em média 15 minutos e são feitas de pé. Há também da metodologia Scrum a figura do *Scrum Master*. Este profissional é responsável por supervisionar o trabalho do restante da equipe de desenvolvimento do projeto, além de definir a quantidade de *sprints* que devem ser realizados durante o mês ou semana. Também é função dele facilitar quaisquer problemas que possam surgir no caminho da equipe de trabalho e assegurar que os métodos e princípios do Scrum sejam mantidos e cumpridos.

A equipe de desenvolvimento foi formada pelo aluno e pelo professor orientador. Entretanto, como este estudo está associado com atividades de outras disciplinas, os alunos integrantes do projeto nas outras disciplinas também foram parte da equipe de desenvolvimento. E o orientador também comportou-se como *scrum master*.

Para a modelagem do Sistema Multiagente de um jogo, existe a metodologia *Prometheus*. Ela gera documentação dos vários aspectos e especificações da criação de um sistema com o uso de uma abordagem orientada a agentes [18]. Na opinião de [18], uma metodologia de projeto de construção de SMA faz parte de um conjunto de métodos, processos e ferramentas, cuja modelagem deve conter os objetivos do sistema, papéis e interações dos seus elementos.

F. Estudo de Caso

Estudo de caso é um método de pesquisa utilizado para se aprofundar em um determinado tema. Ele serve para investigar um fenômeno (caso) em seu contexto no mundo real, coletar e analisar dados a respeito do caso em questão. Ele pode ser categorizado como um estudo empírico que busca determinar ou testar uma teoria que sirva como base para uma futura investigação [19].

O jogo que serve como estudo de caso para o presente trabalho (Sah'Loon) foi desenvolvido durante o 5º semestre do Curso de Jogos Digitais da Universidade Franciscana. O início do processo de desenvolvimento deu-se nas disciplinas Desenvolvimento de Jogos Digitais 3 e Projeto Integrador 3 (PIJ 3). Em Desenvolvimento de Jogos Digitais, realizou-se um *brainstorm* para pensar qual seria o gênero do jogo e as mecânicas básicas. Nesta etapa, também foi formada a equipe de construção do projeto. Ela foi composta por alunos e, cada um deles foi responsável por uma área específica dentro do processo de criação do jogo: arte, *game design* e programação. Por sua vez, a disciplina PIJ3 serviu para pesquisar a respeito do público alvo do jogo e quais seriam as principais referências do gênero. Nesta disciplina, realizou-se o planejamento de como o jogo deveria ser construído, quais recursos ele deveria apresentar e como seriam realizados os *playtests* com o público alvo do jogo. Registra-se que na disciplina Desenvolvimento de Jogos Digitais3 foram implementadas as ideias discutidas em PIJ3, como as mecânicas dos personagens, conceitos de *level design*, modelos 3D dos *tiles*, personagens, cenários e protótipos de interface gráfica.

No que tange ao *gameplay* do jogo, sem o módulo de IA, ele pode ser jogado por dois jogadores (players) ao mesmo tempo. Cada um deles pode controlar um personagem, com o objetivo de derrotar o adversário. A temática principal para a produção do jogo é uma mistura de *western* americano com elementos da literatura do escritor Lovecraft[4], famoso por escrever contos de mistério e terror.

Os personagens do jogo são o Sheriff, o Xamã, o Espantalho, o Padre, o Tesla e o Louco. Buscou-se fazer uma mescla entre estas duas fontes de inspiração para o tema do jogo. O público-alvo são pessoas com baixa visão, ou que apresentam daltonismo ou algum tipo de deficiência motora. Dessa forma, o cenários e os componentes do jogo têm cores com alto contraste, silhueta marcada, dimensões maiores (como mostram as Figuras 5 e 6). Acredita-se que os jogos eletrônicos possam ser uma porta de entrada para a acessibilidade, quanto para o universo do entretenimento. A caracterização *western* é representada pelos *tiles*[5] que formam uma espécie de deserto, compostos por cactos, riachos, montanhas, *saloons*, uma pequena igreja e trilhos de trem. A parte mais obscura da ambientação do jogo, inspirada em Lovecraft é representada por cemitérios, criptas, caixões e árvores de galhos retorcidos. O principal objetivo da ambientação é proporcionar ao jogador uma experiência de imersão e oferecer ao jogo uma identidade visual.

III. Metodologia e proposta de trabalho

Esta pesquisa é exploratória com revisão bibliográfica apoiada em estudo de caso (projeto de jogo). Ela é utilizada com o intuito de ajudar o pesquisador a ter uma maior familiaridade com o objeto do estudo e oferecer técnicas mais adequadas para o prosseguimento da pesquisa. Segundo [20] pesquisa exploratória é quando a pesquisa se encontra na fase inicial e tem como principal finalidade, proporcionar mais informações sobre o assunto que é investigado. Ela orienta na fixação dos objetivos do trabalho e na formulação de hipóteses. De maneira geral, ela assume as formas de pesquisas bibliográficas e estudos de caso [20].

Para a construção do módulo foram utilizadas as ferramentas:

- Trello (gestão de atividades);
- Github (versionamento de código);
- *Prometheus Design Tools* (PDT);
- Linguagem de programação C#;
- *Engine* Unity 2019;
- Bibliotecas adicionais da Unity 2019.

Para a modelagem do módulo SMA utilizou uma adaptação da metodologia *Prometheus*, baseada em quadros de tratamento de planos.

IV. Resultados

O jogo construído possui as interfaces presentes nas Figuras 5 e 6. No cenário do jogo, o humano pode escolher diferentes ângulos de visualização e iniciar a partida, passando o turno (vez) para o módulo SMA.

Figura 5. Tela de *Tela inicial do jogo*

[4]http://www.sitelovecraft.com/hplovecraft.php

[5]Os *tiles* (os espaços do cenários em que os personagens se movimentam) são distribuídos pelo *grid* do jogo de forma aleatória.

Figura 6. Tela de *gameplay*

A. *Descrição do funcionamento do módulo de comportamento inteligente*

Nesta parte do trabalho, buscou-se detalhar o que o módulo faz e como ele se comporta dentro do jogo do estudo de caso.

Neste trabalho, há dois pontos importantes de modelagem:

- Características e funcionalidades dos personagens ou jogadores não humanos: os personagens controlados pelo computador, que apresentam um módulo de comportamento inteligente têm como objetivo principal proporcionar ao jogador um desafio de como se ele jogasse uma partida contra um adversário humano. Os jogadores não humanos realizam as jogadas de acordo com os movimentos que o jogador humano executa durante a partida. Eles buscam realizar as melhores jogadas possíveis com o intuito de vencer a partida e também de aumentar a dificuldade encontrada pelo jogador humano durante os duelos. O jogo é dividido em dois times: os Lunáticos (representados pelos personagens Tesla, Padre e Louco) e os Piromaníacos (representados pelos personagens Xerife, Xamã das Chamas e o Espantalho). O jogador humano sempre joga com o time dos Lunáticos, enquanto o computador fica com a equipe dos Piromaníacos. Os personagens controlados pelo jogador não humano mantém as mesmas características e mecânicas, como movimentação, ataques e habilidades que eles possuem ao serem controlados pelo jogador humano.
- Como se modelou as características e funcionalidades: a Figura 7 mostra a modelagem clássica de uma SMA, levando-se em consideração os planos dos agentes, o que ativam esses planos (eventos ativadores internos ou externos), contexto ou condição de execução de um plano, e a base de fatos ou crenças atualizadas.

Sendo assim, no processo de modelagem de um jogo com o uso de Sistemas Multiagentes, o levantamento de requisitos equivalente seria o mapeamento de planos que um agente pode executar, quais os eventos ativadores desses planos, as condições ou contextos para que um plano entre em execução, conforme ilustrado na Figura 7.

No processo de modelagem do módulo como um todo, é fundamental identificar: i) características e funcionalidades dos personagens ou jogadores não humanos (o que é esperado que os personagens tenham de comportamento; o que o módulo deve agregar ao jogo); ii) como se pretende modelar as características e funcionalidades (se for método de busca, quais estados, quais regras de transição, quais restrições, qual a estrutura de visitados; se for SMA, quais crenças, quais planos; quais restrições; quais os mecanismos de comunicação);

Identificação do Agente	Evento Ativador (interno ou externo)	Contexto (É a condição da seleção)	Plano	Atualização Base Conhecimento
Personagem	Posição da peça adversária **Ou** número de unidades de locomoção disponíveis	Posição não ocupada **Ou** Peça adversária dentro do raio de ação	Atacar as peças adversárias	Posição ocupada **Ou** Peça inimiga atacada

Figura 7. Figura que ilustra quadro de modelagem SMA.

De acordo com a metodologia *Prometheus*, foram realizados algumas etapas:

- **especificação do sistema e suas principais funcionalidades**: os objetivos do módulo com SMA é garantir que os NPC tenham um comportamento mais natural possível, ou seja, realizem interação (comunicação ou negociação) entre os demais NPC, percebam o ambiente e os personagens controlados pelo jogador humano. Além disso, sejam adaptáveis as jogadas do adversário. Por exemplo, os personagens controlados pelo jogador humano estão em posições diferentes dos NPC. Entretanto, quando um NPC perceber um personagem sob controle humano pode executar um plano de abordagem individual ou coletiva (com seus parceiros). Registra-se que NPC modelados e implementados pela teoria SMA devem ser autônomos, proativos, adaptáveis e com habilidades sociais (comunicação);
- **organização dos agentes em papéis e descrição de suas relações**: a relação de papéis (funcionalidades) no jogo é:
  - personagem Xamã: atacar os inimigos; aumentar o número de vidas dos companheiros de time;
  - personagem Sheriff: atacar os inimigos; bloquear os caminhos dos inimigos; combinar jogadas com o personagem Espantalho;
  - personagem Espantalho: combinar jogadas com o personagem Sheriff.
- **descrição dos eventos, planos e dados**: veja as Figuras 8, 9 e 10 para compreender os papéis associados aos planos de cada agente.

B. *Especificação dos planos e dos sub-planos dos NPCs*

Nesta parte do trabalho, mostra-se os planos de cada personagem não controlado pelo jogador humano e como eles os realizam.

- **Xamã (atacar os inimigos)**
  - 1º Passo: disparar projétil na direção do inimigo (mesma linha ou mesma coluna);
  - 2º Passo: diminuir um ponto de ação do Xamã.

Identificação do Agente	Evento Ativador (interno ou externo)	Contexto (É a condição da seleção)	Plano	Atualização Base Conhecimento
Xamã	O inimigo está na mesma linha ou na mesma coluna do Xamã **OU** O inimigo está até quatro linhas ou colunas de distância de onde o Xamã pode se movimentar	O inimigo não está em uma coluna ou em uma linha diferente do Xamã **E** O inimigo não está a uma distância maior do que o número máximo de linhas ou colunas do que o Xamã pode se mover **E** Um companheiro de time não está entre o inimigo e o Xamã	Atacar os personagens inimigos (Disparar arma)	Inimigo atacado + Inimigo perdeu uma vida + Inimigo eliminado do turno caso tenha atingido zero vida
Xamã	Vida dos companheiros de time menor do que três e está na mesma linha ou coluna do Xamã	Número de vidas do companheiro de time não ser igual a três	Aumentar o número de vidas dos companheiros de time	Companheiro ganhou uma vida
Xamã	O Xamã está na mesma linha ou na mesma coluna do inimigo **OU** Está dentro da distância máxima onde o inimigo pode se mover	Sem condição (contexto)	Atacar Xamã	Xamã atacado + Xamã perdeu uma vida + Xamã eliminado do turno caso tenha atingido zero vida

Figura 8. Figura que ilustra quadro de modelagem do personagem Xamã (Fase de especificação do modelo dentro do PDT.

Identificação do Agente	Evento Ativador (interno ou externo)	Contexto (É a condição da seleção)	Plano	Atualização Base Conhecimento
Sheriff	O inimigo está na mesma linha ou na mesma coluna do Sheriff **OU** O inimigo está até três linhas ou colunas de distância de onde o Sheriff pode se mover	O inimigo não está em uma coluna ou em uma linha diferente do Sheriff **E** O inimigo não está a uma distância maior do que o número máximo de linhas ou colunas do que o Sheriff pode se mover **E** Um companheiro de time não está entre o inimigo e o Sheriff	Atacar os inimigos (ao mesmo tempo em que o Sheriff ataca o inimigo, ele já bloqueia o caminho de movimentação)	Inimigo atacado + Inimigo perdeu uma vida + Inimigo eliminado do turno caso tenha atingido zero vida
Sheriff	O inimigo pode se movimentar na direção do Sheriff **OU** O inimigo está na mesma linha ou na mesma coluna do Sheriff	O Inimigo não está em uma linha ou coluna diferente do Sheriff **E** Não há um companheiro de time entre o inimigo e o Sheriff	Bloquear o caminho de movimentação do inimigo	Caminho bloqueado
Sheriff	Espantalho está na mesma linha ou na mesma coluna dos inimigos e o Sheriff pode se mover para essa mesma linha ou coluna **OU** Espantalho está na mesma linha ou na mesma coluna do Sheriff	O Espantalho não está em uma linha ou coluna diferente do Sheriff **E** O Espantalho não se encontra em uma distância maior do que o número máximo de linhas ou colunas que o Sheriff pode se mover	Combinar a jogada com o Espantalho	Jogada combinada
Sheriff	O Sheriff está na mesma linha ou na mesma coluna do inimigo **OU** O Sheriff Está dentro da distância máxima onde o inimigo pode se mover	Sem condição (contexto)	Atacar Sheriff	Sheriff atacado + Sheriff perdeu uma vida + Sheriff eliminado do turno caso tenha atingido zero vida

Figura 9. Figura que ilustra quadro de modelagem do personagem Sheriff.

Identificação do Agente	Evento Ativador (interno ou externo)	Contexto (É a condição da seleção)	Plano	Atualização Base Conhecimento
Espantalho	O Espantalho está na mesma linha ou na mesma coluna do inimigo e o Sheriff pode se mover para essa mesma linha ou coluna **OU** O Espantalho está na mesma linha ou na mesma coluna do Sheriff	O Espantalho não está em uma linha ou coluna diferente do Sheriff **E** O Espantalho não se encontra em uma distância maior do que o número máximo de linhas ou colunas que o Sheriff pode se mover	Combinar a jogada com o Sheriff	Jogada combinada + O Espantalho não pode ser atacado pelos inimigos
Espantalho	O Espantalho está na mesma linha ou na mesma coluna do inimigo **OU** O Espantalho está dentro da distância máxima onde o inimigo pode se mover	Sem condição (contexto)	Atacar Espantalho	Espantalho atacado + Espantalho perdeu uma vida + Espantalho eliminado do turno caso tenha atingido zero vida

Figura 10. Figura que ilustra quadro de modelagem do personagem Espantalho.

- **Xamã (aumentar o número de vidas dos companheiros de time)**
  - 1º Passo: disparar projétil na direção do companheiro de time (mesma linha ou coluna);
  - 2º Passo: entretanto, ao invés de diminuir, este disparo aumenta o número de vida do companheiro de time.
  - 3º Passo: diminuir um ponto de ação do Xamã.
- **Sheriff (atacar os inimigos)**
  - 1º Passo: disparar projétil na direção do inimigo (mesma linha ou mesma coluna);
  - 2º Passo: bloquear o caminho de movimentação do inimigo;
  - 3º Passo: diminuir um ponto de ação do Sheriff.
- **Sheriff (bloquear os caminhos dos inimigos)**
  - 1º Passo: bloquear caminho do inimigo (mesma linha ou mesma coluna);
  - 2º Passo: diminuir ponto de ação do Sheriff.
- **Sheriff (combinar jogada com o Espantalho)**
  - 1º Passo: conferir se o Espantalho encontra-se na mesma linha ou coluna do Sheriff;
  - 2º Passo: conferir se os inimigos encontram-se na mesma linha ou coluna do Espantalho;
  - 3º Passo: Caso o Espantalho esteja na mesma linha ou coluna do Sheriff e dos inimigos, combinar jogada com o Espantalho.
- **Espantalho (combinar jogada com o Sheriff)**
  - 1º Passo: movimentar na direção do Sheriff (mesma linha ou mesma coluna);
  - 2º Passo: diminuir ponto de ação do Espantalho;
  - 3º Passo: ordenar que o Sheriff dispare o projétil.

Como já mencionado (seção II-A1), Unity é um ambiente propício à programação de SMA, pois garante autonomia e comunicação dos agentes. Entretanto, o ambiente Unity usa a linguagem C# como ferramenta de desenvolvimento, pois dessa forma os agentes implementados seguem o modelo da orientação a objetos, em que cada agente é uma classe ou *script*[6], contendo atributos e métodos (planos ou subplanos). Sendo assim, a implementação orientada a objetos permite adicionar mais jogadores não humanos se necessário, pela propriedade de reuso.

C. Implementação do SMA

As Figuras que seguem ilustram trechos de códigos em C# ou Unity que implementam planos de agente.

A Figura 11 representa a implementação do código da função que ativa a movimentação do personagem não humano (NPC) Sheriff. A linha de código 95 utiliza o método *Vector3.MoveTowards* da própria Unity para movimentar o NPC para a direção que foi sorteada.

Após locomover-se, se o Sheriff não encontrou no *grid* nenhum personagem controlado pelo jogador humano, ele volta a movimentar-se para uma nova posição (Figura 12).

Caso o Sheriff encontre os personagens controlados pelo jogador humano, ele executa o disparo do projétil. Na linha

[6]Em C# é comum usar a expressão Script para representar uma classe, pois todo Script criado herda automaticamente da classe MonoBehaviour, logo gerando ideia de herança e subclasse [21]

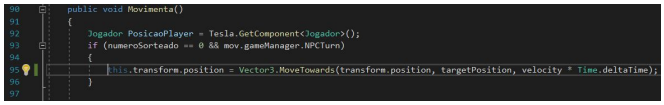


Figura 11. Função para movimentar o Sheriff.

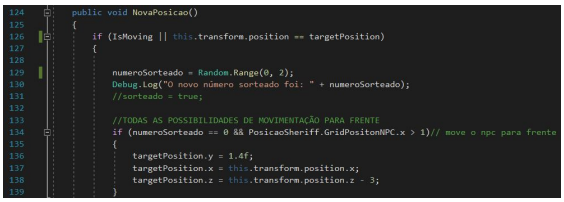


Figura 12. Método para sortear nova posição o NPC.

67 do código (Figura 13), está a condição para detectar se o NPC encontrou algum personagem controlado pelo jogador humano. Se a condição for satisfeita, ele atira no alvo.

Figura 13. Função para disparar o projétil do NPC.

V. Conclusões

Ao final deste trabalho, foram apresentados e discutidos conceitos de jogos que aplicaram técnicas de Inteligência Artificial, principalmente jogos que utilizaram Sistemas Multiagentes. Para isso, há uma seção com trabalhos relacionados. O texto apresentou definições e embasamentos a partir de pesquisadores e obras das áreas da Ciência da Computação e de Jogos Digitais. Buscou-se, ao longo do texto, apresentar a relação de IA (ou sistemas de comportamento inteligente) com jogos eletrônicos, bem como suas características, vantagens, tecnologias, metodologias e resultados.

Outra consideração importante é sobre a *engine* Unity em relação a implementação de personagens com comportamento inteligente, baseado na teoria de Sistemas Multiagentes. Como discutido, agentes têm autonomia, iniciativa, flexibilidade e habilidade social (capacidade de comunicação) e todas essas características não são fáceis de implementar em um ambiente que não seja destinado à SMA. Porém, a Unity, por meio de seus recursos, garante sim essas características. Ou seja, um personagem pode comportar-se autonomamente, pode responder reativamente ou deliberativamente (uso de planos) a estímulos do ambiente ou de outros personagens, por exemplo. Dessa forma, registra-se que os objetivos assumidos foram atingidos.

Para os agentes que representam os jogadores NPC, foram modelados o Sheriff, o Xamã e o Espantalho (planos e crenças). Contudo, para o protótipo do jogo com o módulo SMA, os agentes Sheriff e Xamã foram completamente implementados.

Acredita-se que há alguns trabalhos futuros a serem realizados:

- implementação e testes do agente Espantalho;
- modelagem e implementação de algum método de busca para auxiliar no percurso de jogadores (agentes) NPC.

Finalmente, há apêndice com um manual que descreve as mecânicas dos personagens e como o jogador pode jogar. Ou seja, são instruções para jogar pela primeira vez.

Referências

[1] B. Ribeiro, F. Lucchese, M. Rocha, and V. Figueiredo, "Inteligência artificial em jogos digitais."

[2] P. Norvig and S. Russell, *Inteligência Artificial: Tradução da 3a Edição*. Elsevier Brasil, 2014, vol. 1.

[3] P. R. Norvig and S. A. Intelligence, *A modern approach*. Prentice Hall, 2002.

[4] Y. Demazeau and J.-P. Müller, *Decentralized Ai*. Elsevier, 1990.

[5] M. d'Inverno, P. Howells, S. Montagna, I. Roeder, and R. Saunders, "Agent-based modeling of stem cells," in *Multi-Agent Systems: Simulation and application*, ser. Computational analysis, synthesis, and design of dynamic models series, A. M. Uhrmacher and D. Weyns, Eds. Boca Raton, FL, USA: CRC Press, 2009, ch. 13, pp. 389–418.

[6] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[7] A. O. Zamberlan, "Sistema multiagente para avaliação do efeito de aglomeração em nanopartículas poliméricas," Ph.D. dissertation, UFN - Universidade Franciscana, 2018.

[8] A. Kishimoto, "Inteligência artificial em jogos eletrônicos," *Academic research about Artificial Intelligence for games*, 2004.

[9] E. Borges, J. Nader, and M. Menegas, "Inteligência artificial aplicada ao jogo pacman."

[10] P. Tozour, "The evolution of game ai," *AI game programming wisdom*, vol. 1, pp. 3–15, 2002.

[11] J. Schell, *The Art of Game Design: A book of lenses*. AK Peters/CRC Press, 2014.

[12] S. Rogers, *Level Up! The guide to great video game design*. John Wiley & Sons, 2014.

[13] M. L. M. Machado, D. G. de Souza, J. A. Souza, G. Dandolini, and R. A. Silveira, "Rpg: Uma abordagem empregando sistemas multiagentes," *RENOTE*, vol. 2, no. 1, 2003.

[14] P. Guyot and S. Honiden, "Agent-based participatory simulations: Merging multi-agent systems and role-playing games," *Journal of Artificial Societies and Social Simulation*, vol. 9, no. 4, 2006.

[15] C. T. Tan and H.-l. Cheng, "A combined tactical and strategic hierarchical learning framework in multi-agent games," in *Proceedings of the 2008 ACM SIGGRAPH symposium on Video games*. ACM, 2008, pp. 115–122.

[16] D. M. Bourg, "Glenn seeman," in *AI for Game Developers*. O'Reilly, 2004.

[17] B. Schwab, *AI game engine programming*. Nelson Education, 2009.

[18] L. Padgham and M. Winikoff, *Developing intelligent agent systems: A practical guide*. John Wiley & Sons, 2005, vol. 13.

[19] R. K. Yin, *Estudo de Caso-: Planejamento e métodos*. Bookman editora, 2015.

[20] C. C. Prodanov and E. C. de Freitas, *Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição*. Editora Feevale, 2013.

[21] J. Hocking, *Unity in action: multiplatform game development in C#*. Manning Publications, 2018.

Sah'Loon

Instruções

Time dos Lunáticos

Tesla

Alcance da movimentação:

Até 3 unidades

Para atirar com o personagem Tesla, clique sobre ele para selecioná-lo e clique no alvo com o botão direito do mouse para efetuar o disparo. Este personagem dispara o projétil apenas se ele estiver na mesma linha ou coluna do NPC (inimigo).

Padre

Alcance da movimentação:

Até 3 unidades

Com o Padre selecionado, o jogador pode colocar uma cruz no tabuleiro do jogo. Esta cruz pode ser colocada na esquerda, na direita, atrás ou na frente dos personagens. Caso o inimigo encoste nela, ela perde um ponto de vida. Para colocá-la, clique sobre o Padre para selecioná-lo e com o botão direito do mouse clique na posição do tabuleiro que deseja fincá-la. Não é possível colocar a cruz na diagonal dos personagens.

Jogada Especial

Caso o jogador esteja com o personagem Tesla selecionado e se houver uma cruz na direita, na esquerda, na frente ou atrás dele, o jogador pode clicar sobre ela com o botão direito do mouse para efetuar o disparo da arma e, ao entrar em contato com a cruz, o projétil se divide em quatro partes e se elas colidirem contra o inimigo, ele perde um ponto de vida.

Time dos Piromaníacos

Sheriff

Alcance da movimentação:

Até 3 unidades

O Sheriff tem como mecânica principal disparar o lança chamas nos personagens controlados pelo jogador humano. Se a chama colidir contra eles, eles perdem um ponto de vida. Além disso, o disparo do lança chamas deixa um rastro de fogo pelo tabuleiro do jogo, onde os personagens controlados pelo jogador humano não conseguem se locomover.

Xamã

Alcance da movimentação:

Até 4 unidades

O Xamã tem como principal função atirar nos personagens controlados pelo jogador humano e também, caso ele perceba que o Sheriff perdeu pontos de vida, ele pode aumentar a vida dele em um ponto, através do disparo de sua arma.

Regra Geral

Os personagens podem apenas se locomover e realizar a mecânica específica de cada um, uma vez por turno. Cada time tem quatro pontos de ação por turno.

Mecânica de locomoção dos personagens

Para mover o personagem controlado pelo jogador humano, clique sobre ele e depois clique no espaço do tabuleiro que deseja que ele vá.

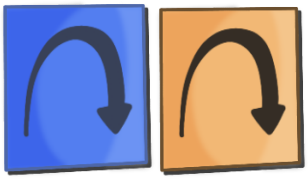
Condição de Vitória

Vence a partida o time que conquistar três estrelas. Elas são conquistadas quando os personagens dos times são eliminados dos turnos, ou seja, quando algum deles perde os três pontos de vida.

Botões

Ao clicar neste botão, o jogador volta para a tela inicial do jogo.

Este botão serve para girar a câmera principal do jogo. Ao clicar nele, o jogador consegue jogar por outros ângulos.



Ao clicar neste botão, o jogador passa o turno dele para o outro time. Caso o turno seja do time dos Piromaníacos e se os NPCS já terminaram de jogar, basta o jogador clicar no botão para o turno voltar para ele.