PRINCIPAIS TÉCNICAS DA INTELIGÊNCIA ARTIFICIAL PARA JOGOS

Marcelo Buscioli Tenorio
Edgar Nalin Alves
Yuri Correa Reis

RESUMO
O presente trabalho teve como objetivo principal estudar e apresentar as principais técnicas da inteligência artificial usadas nos jogos eletrônicos e desenvolver um protótipo na plataforma de desenvolvimento Unity Game Engine, onde personagens que são controlados pelo computador utilizam uma das técnicas apresentadas, Padrões de Movimento, com a finalidade de atacar o personagem controlado por humano. As principais técnicas estudadas foram Maquinas de Estado Finito, Path-Finding, Padrões de Movimento, Sistemas Baseados em Regras, Lógica Fuzzy, Redes Neurais e Algoritmos Genéticos. No protótipo foram criados dois inimigos que são controlados pelo computador e utilizam a inteligência artificial para atacar o jogador controlado por humano, o primeiro inimigo ataca de longe, dependendo da distância do jogador controlado por humano, e o segundo inimigo ataca de perto, também dependendo da distância do jogador controlado por humano. O método do trabalho utilizado foi o dedutivo, a natureza da pesquisa básica, a forma de abordagem qualitativa e o procedimento técnico utilizado foi a pesquisa bibliográfica.
Palavras-chave: Jogos Eletrônicos. Protótipo. IA.

ABSTRACT
The main objective of this work was to study and present the main techniques of artificial intelligence used in electronic games and to develop a prototype in the Unity Game Engine development platform, where characters that are controlled by the computer use one of the techniques presented, Patterns of Motion, with the purpose of attacking the human-controlled character. The main techniques studied were Finite State Machines, Path-Finding, Patterns of Motion, Rules-Based Systems, Fuzzy Logic, Neural Networks and Genetic Algorithms. In the prototype were created two enemies that are controlled by the computer and use the artificial intelligence to attack the player controlled by human, the first enemy attacks from far, depending on the distance of the player controlled by human, and the second enemy attacks closely, also depending the distance of the player controlled by human. The method used was the deductive, the nature of the basic research, the qualitative approach and the technical procedure used was the bibliographic research.
Keywords: Electronics Games. Prototype. AI**.**

1. INTRODUÇÃO

De acordo com Ciriaco (2008) Inteligência Artificial (IA) é um ramo da ciência da computação que se propõe a elaborar dispositivos que simulem a capacidade humana de raciocinar, perceber, tomar decisões e resolver problemas, enfim, a capacidade de ser inteligente.

O objetivo da IA para jogos é fornecer um comportamento para objetos ou entidades, cujo comportamento simule um movimento de inteligência, por exemplo, no caso de um jogo de RPG (*Role-Playing Game*), pode ser um movimento de ataque de um NPC (*Non Player*

Characters, personagens do jogo que não são controlados pelo humano). Por exemplo, se o jogador passar perto de um NPC, o movimento de ataque pode ser realizado com a utilização de técnicas de Inteligência Artificial.

Com o desenvolvimento da tecnologia, surgiu uma nova técnica da Inteligência Artificial, a qual é utilizada até hoje, que são as Máquinas de Estado Finito. Elas descrevem os comportamentos dos NPCs. Com o passar do tempo, os jogadores queriam mais desafios para que os jogos se tornassem mais complicados, então algoritmos foram criados para realizarem uma busca de melhor caminho entre dois pontos no mapa, melhorando principalmente jogos de estratégia em tempo real.

No começo dos jogos eletrônicos, o fator mais importante era em relação aos gráficos tridimensionais. Com a grande evolução dos computadores domésticos, os usuários estão cada vez mais satisfeitos com os gráficos, tornando assim, nos dias de hoje, a IA um dos mais importantes elementos de um jogo.

Para um jogo possuir inteligência, na implementação do jogo deve haver técnicas da IA que serão as responsáveis por ditar os comportamentos dos NPC's, como atacar, andar, pegar um objeto, ou até mesmo comportamentos aleatórios, que são pré-programados, mas o movimento irá depender das escolhas que o personagem humano fizer.

As principais técnicas da IA utilizadas no desenvolvimento de jogos são: Máquinas de Estado Finito, é a técnica mais usada para expressar o comportamento dos personagens de um jogo, onde ocorrem as mudanças de ações dos personagens; *Path-Finding*, é a técnica que verifica o melhor caminho que o personagem vai percorrer; Padrões de Movimento, definem os movimentos de pernas, braços, gestos, enfim, animação do personagem e ações de ataque e defesa; Sistemas Baseados em Regras, permite a modelagem de comportamentos complexos, mas exige um poder de processamento alto; Lógica *Fuzzy*, complementa a tradicional lógica booleana, representando valores que estão entre o verdadeiro e falso, por exemplo, expressões do tipo pouca vida; Redes Neurais Artificiais, técnica responsável pelo aprendizado adquirido com personagem humano, como resultado gera um comportamento diferente para surpreender o jogador; Algoritmos Genéticos, esta técnica desenvolve diferentes objetos ou indivíduos a partir de um DNA virtual (PATHFINDING, online, 2013).

Devido aos grandes avanços da tecnologia para o desenvolvimento de jogos, principalmente relacionados à Inteligência Artificial, este trabalho tem sua importância ao estudar e informar as principais técnicas da IA que são utilizadas pelos desenvolvedores de jogos, como essas técnicas funcionam e porque são as mais utilizadas.

O trabalho em questão tem como objetivo principal estudar as técnicas da IA aplicada nos jogos, mostrando o quanto a área de jogos eletrônicos pode ser beneficia com a Inteligência artificial

O objetivo específico é o desenvolvimento de um protótipo simples, onde personagens similares aos de jogos eletrônicos utilizam inteligência artificial. A técnica de IA usada no protótipo é baseada na técnica Padrões de Movimento. Existem dois tipos de inimigos no protótipo, o primeiro inimigo ataca de longe e o segundo inimigo ataca de perto, estes ataques são acionados quando o jogador controlado por humano entra na distância programada em relação aos inimigos. O jogador controlado por humano possui uma movimentação e realiza ataque. O ambiente de desenvolvimento é o *Unity Game Engine*.

2. METODOLOGIA

O método utilizado foi o dedutivo, a natureza da pesquisa é básica e a forma de abordagem do problema é qualitativa. O objetivo da pesquisa é exploratório, o procedimento técnico utilizado é a pesquisa bibliográfica e a pesquisa documental.

O protótipo está desenvolvido na plataforma Unity Game Engine versão 5, utilizando a linguagem de programação Javascript. Os scripts foram feitos com base na técnica de IA, chamada padrões de movimento (ALVAREZ, 2004).

O protótipo foi iniciado com a criação do terreno, onde ficam o Jogador e os Inimigos com a inteligência artificial. Foram feitas imperfeições no solo para simular montanhas e morros e em seguida foram adicionados árvores e gramas no ambiente. Para que o terreno pareça mais real, foram colocadas texturas de uma grama bem fina no solo e outra textura nas montanhas para parecer pedra.

Na próxima etapa foi feito o Jogador, personagem controlado por humano. A modelagem do Jogador foi feita de forma simples utilizando esferas e cubos da Unity 5 (GASPAROTTO, 2015). Nele foi inserido um *script* para movimentação nos sentidos: ir para frente, ir para trás, ir para os lados e rotacionar.

3. RESULTADOS

No Quadro 1 pode-se ver um trecho do código fonte relacionado à movimentação do Jogador na linguagem JavaScript. Se o usuário que controla o Jogador pressionar e não soltar uma das teclas “w, s, a, d, q, e”, o Jogador irá realizar uma movimentação para frente, para trás, para a esquerda, para a direita, girar para esquerda e girar para a direita, respectivamente conforme as teclas mencionadas.

Quadro 1: Script Movimento do Jogador

```
if(Input.GetKey("w"))
{
        corpo.velocity = velocidade*transform.forward;
}
if(Input.GetKey("s"))
{
        corpo.velocity = -velocidade*transform.forward;
}
if(Input.GetKey("a"))
{
        corpo.velocity = -velocidade*transform.right;
}
if(Input.GetKey("d"))
{

        corpo.velocity = velocidade*transform.right;
}

if(Input.GetKey("q"))
{
        transform.Rotate(0,-velocidadeVirar,0);
}
if(Input.GetKey("e"))
{
        transform.Rotate(0,velocidadeVirar,0);
}
```

Fonte: Elaborado pelo autor

Ainda dentro do Jogador foi feito a arma, onde a arma mira na direção do ponteiro do mouse e apertando a tecla espaço do teclado a arma atira uma bala. Esta bala é instanciada em um objeto invisível próximo a arma, somente existindo quando a tecla espaço for pressionada. A bala é destruída se tocar em algum objeto, terreno ou depois de percorrer determinada distância sem acertar algo.

No Quadro 2 pode-se observar o trecho do código fonte da arma onde a bala é instanciada no objeto invisível identificado no como a variável inicioBala.

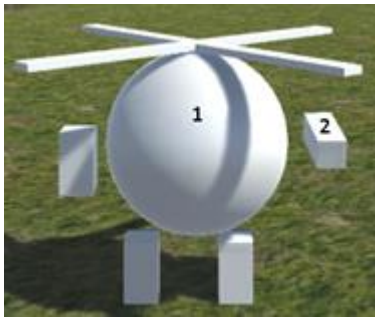
Quadro 2: Script Arma

```
if (Input.GetKeyDown(KeyCode.Space))
{
        Instantiate(bala,inicioBala.position,transform.rotation);
}
```

Fonte: Elaborado pelo autor

Na Figura 1 pode-se ver o Jogador marcado pelo número 1 e a arma marcada pelo número 2.

Figura 1: Jogador e arma.



Fonte: Elaborada pelo autor

O Quadro 3 mostra um trecho do código fonte da arma, onde exibe a parte relacionada a movimentação da arma em relação a posição do mouse.

Quadro 3: Movimento mouse

```
raio = visao.ScreenPointToRay(Input.mousePosition);

if (Physics.Raycast(raio,colisao,1000)){
transform.LookAt (colisao.point);
}
else
{
semcolisao = visao.transform.position + raio.direction*500;
transform.LookAt(semcolisao);
}
```

Fonte: Elaborado pelo autor

A variável raio recebe a localização do ponteiro do mouse na tela do computador. Em seguida, verifica se o mouse está tendo colisão com algo que tenha massa, no caso o terreno, árvores ou os inimigos. Se tiver colisão, a mira é exata na direção e se não tiver colisão, no caso olhando para o horizonte ou o céu. A mira da arma é direcionada para 500 metros à frente da mira do mouse. O objetivo disto é para dar profundidade na mira da arma, fazendo a movimentação dela ficar bem melhor quando estiver olhando para perto ou longe.

Para o Jogador possuir física foi colocado nele um componente chamado *Rigidbody*, que o Unity 5 disponibiliza, com isso o Jogador passa a ter massa e sofrer a força da gravidade. Todos os objetos do protótipo possuem o componente *Rigidbody*.

Em seguida foi criado o Inimigo1 (Figura 2) representando um personagem controlado pelo computador, a modelagem do Inimigo1 também foi feita de forma simples utilizando esferas e cubos do Unity 5. O Inimigo1 é um personagem que contém o script com inteligência artificial, baseado na técnica Padrões de Movimento para realizar movimentos pré-programados com o objetivo de simular uma inteligência. Quando o Jogador está dentro da distância determinada em relação ao Inimigo 1, não importando qual lado ou direção seja,

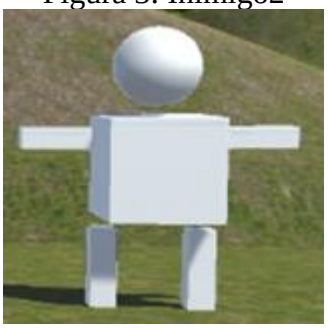
o Inimigo 1 irá atirar uma bala no Jogador, se o Jogador sair desta distância determinada o Inimigo1 para de atirar (RUAIDHRI, 2013).

Figura 2: Inimigo1

Fonte: Elaborada pelo autor

Logo na sequência foi criado o Inimigo2 (Figura 3), com a modelagem feita de forma simples utilizando o Unity 5. O script com inteligência artificial do Inimigo2 também é baseado na técnica Padrões de Movimento para realizar movimentos pré-programados. A base deste script é semelhante ao script do Inimigo1, com a diferença de que o Inimigo2 anda na direção do Jogador para atacar se o Jogador entrar na distância determinada. Se o jogador sair desta distância, o Inimigo2 para de perseguir e fica parado no lugar, só voltando a se movimentar quando o Jogador entrar na distância programada do script novamente.

Figura 3: Inimigo2

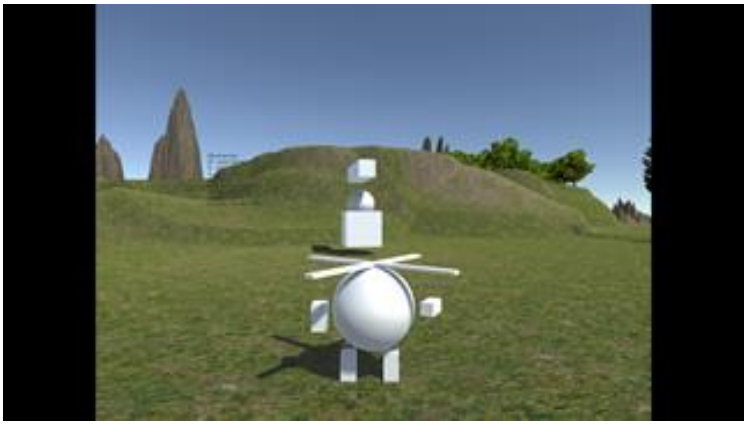


Fonte: Elaborada pelo autor

O protótipo apresenta uma implementação simples da inteligência artificial, onde personagens utilizam técnicas para demonstrar uma movimentação. Esta movimentação só ocorre quando o Jogador está a certa distância dos inimigos.

Quando o Jogador está fora da distância que executa a movimentação do Inimigo1, o Inimigo1 fica parado (Figura 4).

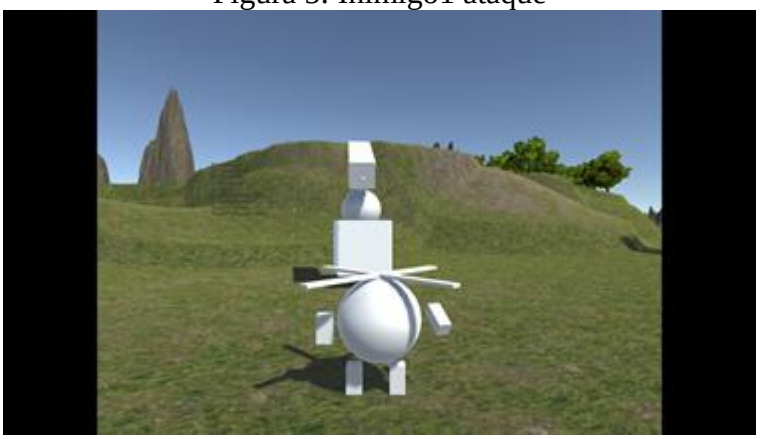
Figura 4: Inimigo1 parado



Fonte: Elaborada pelo autor

Quando o Jogador se aproxima do Inimigo1 e entra na distância determinada no script, o Inimigo1 começa a atacar o Jogador saindo balas da sua arma, no caso a arma do Inimigo1 é o seu retângulo superior. As balas saem do centro (Figura 5).

Figura 5: Inimigo1 ataque



Fonte: Elaborada pelo autor

No Quadro 4 pode-se ver um trecho do código fonte chamado InteligenciaArtificial, onde este script é responsável por toda a movimentação do Inimigo1, com o intuito de dar Inteligência para ele.

Quadro 4: Script InteligenciaArtificial

```
function Start () {
 tempoBala = 0.5;
}

function Update () {

if(!Heroi)
{
 Heroi = GameObject.Find("Jogador").transform;
}
else
{
        Distancia = Vector3.Distance(transform.position,Heroi.position);
        if(Distancia<40)
        {
                AcertaHeroi(Ba,Ar,Acer);
                veloBala += Time.deltaTime;
                if(veloBala>tempoBala)
                {
                        Fogo(Acer, Ar);
                        veloBala = 0;
                }
        }
        else
        {
                veloBala=0;
        }
}
}

function AcertaHeroi(Base: Transform, Arma: Transform,Acerta: Transform){
        Acerta.LookAt(Heroi);
        Base.rotation.eulerAngles.y = Acerta.rotation.eulerAngles.y;
        Arma.rotation.eulerAngles.x = Acerta.rotation.eulerAngles.x;
}

function Fogo (Acerta: Transform, Arma: Transform)
{
        Instantiate(Bala, Acerta.position, Arma.rotation);
}
```

Fonte: Elaborada pelo autor

A função Start tem como objetivo executar tudo de uma vez, logo quando for acionado o script, neste caso a função Start está determinando para a variável tempoBala a quantidade de balas que o Inimigo1 dispara por segundo, como pode ser visto, a cada meio segundo dispara uma bala.

A função Update executa o que estiver dentro dela o tempo todo, toda vez que o processador do computador executar o script, irá ser executado toda a programação existente, quanto mais rápido o processador, mais vezes a função Update será executada. A variável Heroi recebe o Jogador. Na sequência a variável Distancia recebe o valor da distância entre o Inimigo1 e o Jogador. Vector3 significa os eixos x, y e z, onde eixo x representa esquerda e direita, o eixo y representa para cima e para baixo e o eixo z para frete e para trás. Se a distância do inimigo para o jogador for menor que 40 metros no protótipo, executa a função AcertaHeroi e a variável veloBala recebe a velocidade da bala em segundos. Se a variável

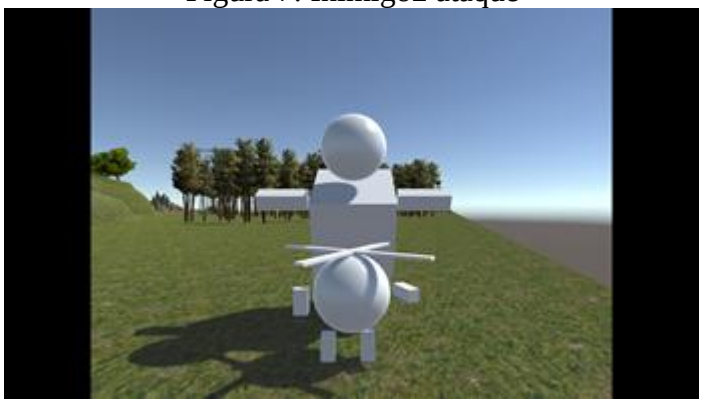
veloBala for maior que a variável tempoBala, executa a função Fogo e zera o tempo da variável veloBala. Se a distância do Inimigo1 para o Jogador for maior que quarenta metros, variável veloBala recebe zero.

A função AcertaHeroi é responsável por mover a arma na direção do Jogador. A variável Acerta.LookAt(Heroi) mira na variável Heroi que é o Jogador. Em sequência, as variáveis Base e Arma são executadas para a arma do Inimigo1 se mover conforme a posição do Jogador.

A função Fogo é responsável por instanciar a bala na arma do Inimigo1, instanciando a bala de acordo com a posição da variável Acerta e a rotação da variável Arma.

A inteligência artificial do Inimigo2 é parecida com a do Inimigo1, com a diferença de não atacar de longe. Se o Inimigo2 alcançar o Jogador, ele irá atacar o Jogador. O ataque é simples, somente encosta no Jogador. A Figura 7 representa o ataque do Inimigo2 ao Jogador.

Figura 7: Inimigo2 ataque



Fonte: Elaborada pelo autor

No Quadro 5 pode-se ver um trecho do código fonte do Script InteligenciaArtificial2, onde este script é responsável por toda a movimentação do Inimigo2, com o intuito de dar inteligência para ele. Este script é semelhante ao do Inimigo1.

Quadro 5: Script InteligenciaArtificial2.

```
function Start () {
Corpo = GetComponent(Rigidbody);
}

function Update () {
if(!Heroi)
{
 Heroi = GameObject.Find("Jogador").transform;
}
else
{
		Distancia = Vector3.Distance(transform.position,Heroi.position);
		if(Distancia<50)
		{
			AcertaHeroi(Ba,Ar,Acer);
			Corpo.velocity = Ar.forward*10;
		}
}
}

function AcertaHeroi(Base: Transform, Arma: Transform,Acerta: Transform){
Acerta.LookAt(Heroi);
Base.rotation.eulerAngles.y = Acerta.rotation.eulerAngles.y;
Arma.rotation.eulerAngles.x = Acerta.rotation.eulerAngles.x;
}

function Fogo (Acerta: Transform, Arma: Transform)
{
		Instantiate(Bala, Acerta.position, Arma.rotation);
}
```

Fonte: Elaborado pelo autor

Na função Start a variável Corpo está recebendo o componente *Rigidbody*, onde este componente representa o Inimigo2. Na função Update a variável Heroi recebe o Jogador. A variável Distancia recebe o valor da distância entre o Inimigo2 e o Jogador. Se o Jogador estiver a menos de 50 metros, executa a função AcertaHeroi e o Inmigo2 representado no código fonte como a variável Corpo recebe velocidade igual a 10 na direção para frente. Como a função AcertaHeroi mira no Jogador o tempo todo, o Inimigo2 sempre vai em direção ao Jogador. As funções AcertaHeroi e Fogo são idênticas ao Script InteligenciaArtificial, citados anteriormente.

4. CONSIDERAÇÕES FINAIS

O trabalho mostrou as principais técnicas da IA aplicada nos jogos eletrônicos, apresentando o quanto a área de jogos eletrônicos pode beneficiar-se com esta área de estudos. Com a intenção de mostrar aplicabilidade da IA, o trabalho também apresenta o desenvolvimento de um protótipo.

A IA é uma grande aliada aos jogos eletrônicos, proporcionando aos criadores e desenvolvedores novas possibilidades de desafios e estilos de jogos diferentes. A tendência é que os jogos fiquem cada vez melhores com o aperfeiçoamento das técnicas existentes da inteligência artificial e com novas descobertas nesta área.

Algumas técnicas de inteligência artificial foram analisadas, apresentando suas características gerais e suas funções em jogos eletrônicos. As técnicas estudadas foram: Maquinas de Estado Finito, Path-Finding, Padrões de Movimento, Sistemas Baseados em Regras, Lógica Fuzzy, Redes Neurais e Algoritmos Genéticos (TANSCHEIT, .

O protótipo desenvolvido atendeu a expectativas gerando resultados satisfatórios e permitindo a reprodução de uma das técnicas estudadas - Padrões de Movimento - gerando uma movimentação e ataque de inimigos controlado pelo computador contra o jogador controlado por humano. A movimentação e ataques ocorrem quando o jogador controlado por humano se aproxima dos inimigos com inteligência artificial. Ainda no protótipo foi desenvolvido o terreno, contendo morros, montanhas, árvores e gramas. Para diferenciar o que é terra, grama ou pedra, foi adicionado texturas. O Jogador controlado por humano contém um script de movimentação e outro de ataque.

REFERÊNCIAS

ALVAREZ, Miguel Angel. *O que é Javascript*, 2004. Disponível em: http://www.criarweb.com/artigos/184.php. Acessado em: 07/11/2015.

CIRIACO, Douglas. *O que é Inteligência Artificial?*, 2008. Disponível em: http://www.tecmundo.com.br/intel/1039-o-que-e-inteligencia-artificial-.htm. Acessado em: 07/04/2013.

GASPAROTTO, Henrique Machado. *Unity 3D: Introdução ao desenvolvimento de games*. Disponível em: http://www.devmedia.com.br/unity-3d-introducao-ao-desenvolvimento-de-games/30653. Acessado em: 07/11/2015.

PATHFINDING. Online. Disponível em: http://www.dotnetperls.com/pathfinding. Acessado em: 02/07/2013.

RUAIDHRI. *Black and White Gameplay Screen*, 2013. Disponível em: http://pixelbedlam.co.uk/kaiju-video-games-big-creatures-in-a-small-city-the-best-examples-of-a-niche-genre/black-and-white-gameplay-screen/. Acessado em: 04/06/2014.

TANSCHEIT, Ricardo. *Sistemas Fuzzy*. Tutorial sobre Sistemas Fuzzy. Rio de Janeiro. Disponível em: http://www2.ica.ele.puc-rio.br/Downloads%5C41/LN-Sistemas%20Fuzzy.pdf. Acessado em: 07/11/2015.