UNIVERSIDADE DO SAGRADO CORAÇÃO

BRUNO ROBERTO BRICCE

APLICAÇÃO DO ALGORITMO KNN PARA CONTROLE DE MOVIMENTOS DE NPCS EM UM AMBIENTE DINÂMICO (JOGO)

BAURU
2015

BRUNO ROBERTO BRICCE

APLICAÇÃO DO ALGORITMO KNN PARA CONTROLE DE MOVIMENTOS DE NPCS EM UM AMBIENTE DINÂMICO (JOGO)

Trabalho de conclusão de curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas da Universidade do Sagrado Coração, como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Me. Patrick Pedreira Silva.

BAURU
2015

Bricce, Bruno Roberto

B8493a

Aplicação do algoritmo KNN para controle de movimentos de NPCS em um ambiente dinâmico (jogo) / Bruno Roberto Bricce. -- 2015.

59f. : il.

Orientador: Prof. Me. Patrick Pedreira Silva.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. Inteligência Artificial. 2. K-NN. 3. NPC. 4. Jogo Eletrônico. I. Silva, Patrick Pedreira. II. Título.

BRUNO ROBERTO BRICCE

APLICAÇÃO DO ALGORITMO KNN PARA CONTROLE DE MOVIMENTOS DE NPCS EM UM AMBIENTE DINÂMICO (JOGO)

Trabalho de conclusão de curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas da Universidade do Sagrado Coração, como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Me. Patrick Pedreira Silva.

Banca examinadora:

_____
Prof. Me. Patrick Pedreira Silva
Universidade do Sagrado Coração

_____
Prof. Dr. Elvio Gilberto da Silva
Universidade do Sagrado Coração

_____
Prof. Me. Henrique Pachioni Martins
Universidade do Sagrado Coração

Bauru, 07 de dezembro de 2015.

Dedico este trabalho a todos que me incentivaram a cursar uma graduação.

AGRADECIMENTOS

Primeiramente a Deus, por estar do meu lado sempre e principalmente nos momentos mais difíceis.

Agradeço também a minha família por acreditarem em mim e me apoiarem nas minhas decisões por mais difíceis que foram.

Agradeço a minha namorada e meus amigos que me apoiaram sempre a continuar a graduação e nunca desistir.

Ao meu orientador Prof. Me. Patrick Pedreira Silva, pela paciência e motivação que me proporcionou para realizar este trabalho.

E a todos os Mestres e Doutores que me passaram seus conhecimentos durante todos esses anos.

“Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista. Se você quer chegar à onde a maioria não chega, faça o que a maioria não faz.”

(Bill Gates)

RESUMO

A Inteligência Artificial (IA) faz com que a máquina tenha a capacidade de criar estratégias, aprender, reconhecer padrões ou encontrar as melhores soluções possíveis, simulando a forma de raciocínio de um ser humano. Nesse projeto foi desenvolvido um jogo na linguagem JavaScript utilizando o motor gráfico Unity 3D, com objetivo de aplicar o algoritmo KNN para movimentação de um personagem não controlado (NPC). O jogo foi feito em 2D, com interface simples e clara, para qualquer tipo de jogador (usuário), poupando de uma longa aprendizagem para começar a jogar. O objetivo do jogo é armazenar as jogadas do usuário e realizar o aprendizado de máquina para que no fim o jogador possa ver como o NPC se sairá jogando sozinho, com base nas jogadas realizadas anteriormente, ou seja, quanto mais o usuário jogar mais o computador irá aprender sobre o jogo. Através de testes realizados nesse projeto foi possível verificar que o algoritmo K-NN implementado no NPC condiz com o que foi proposto. Este trabalho também apresenta a importância de avaliar formas de inteligência artificial para jogos eletrônicos.

Palavras-chave: Inteligência Artificial. K-NN. NPC. Jogo Eletrônico.

ABSTRACT

The Artificial Intelligence (AI) causes the machine has the ability to strategize, learn, recognize patterns and find the best possible solutions, simulating the way of thinking of a human being. In this project we developed a game in JavaScript language using the Unity 3D graphics engine, in order to apply the KNN algorithm for moving an uncontrolled character (NPC). The game was done in 2D, simple and clear interface for any type of player (user), saving a long apprenticeship to start playing. The goal is to store the user moves and perform machine learning that at the end the player can see how the NPC will do playing alone, based on the moves made earlier, that is, the more you more you play the computer will learn about the game. Through tests conducted in this project it observed that the K-NN algorithm implemented in NPC matches what has been proposed. This work also shows the importance of evaluating forms of artificial intelligence for computer games.

Keywords: Artificial Intelligence. KNN. NPC. Electronic game.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A Inteligência Artificial (IA) faz com que a máquina tenha a capacidade de criar estratégias, aprender, reconhecer padrões ou encontrar as melhores soluções possíveis, simulando a forma de raciocínio de um ser humano, porém este raciocínio é implementado em um computador. (RUSSEL; NORVIG, 2003).

No âmbito de jogos computacionais a Inteligência Artificial tem forte presença, permitindo uma interação mais dinâmica ao jogador, transmitindo a sensação de competitividade entre homem e máquina. (SANTAELLA; FEITOZA, 2009). Essa é certamente uma característica desejável, até mesmo porque, pela própria evolução dos jogos digitais, que antes tinham um alto grau de previsibilidade, atualmente contam com jogadores que cada vez mais exigem um nível de dificuldade compatível com a de um ser humano. É comum a utilização de IA em jogos que possuem NPCs (Non Player Characters), que são personagens que têm movimentação própria, ou seja, não são controlados pelo jogador (humano). Isso vem ocorrendo devido à necessidade cada vez maior de realismo para o usuário, pois se não existe uma inteligência nos NPCs, o jogo poderá se tornar monótono, limitando a experiência do jogador em relação aos desafios de um jogo. (OSORIO, 2007).

Uma das formas de se modelar os NPCs como agentes inteligentes para a resolução de problemas é encará-los como uma entidade que procura raciocinar, tentando atingir sua meta, por meio de ações inteligentes. Uma das técnicas que podem ser usadas para atingir esse objetivo é o Algoritmo KNN ou Nearest Neighbour Retrieval (Vizinho Mais Próximo). Trata-se de uma técnica muito utilizada para comparação e estimativa de casos existentes. Através de uma base de casos de ações ocorridas, por exemplo, o KNN consegue supor qual seria o resultado de determinado movimento apenas comparando o que já foi feito anteriormente.

Segundo Lagemann (1998), um Raciocínio Baseado em Casos, acontece principalmente ao acumularem-se novas experiências em sua memória e na correta indexação dos problemas. Em jogos eletrônicos ele pode ser usado para leitura do cenário e aprendizado do computador, o que motiva pesquisas e implementações envolvendo esta técnica.

Deste modo, o estudo da IA aplicada aos jogos torna-se relevante do ponto de vista acadêmico, por permitir a criação de jogos muito mais interativos e interessantes.

2 OBJETIVOS

Apresenta-se nas seções a seguir o objetivo geral e os objetivos específicos da pesquisa.

2.1 OBJETIVO GERAL

Aplicar o Algoritmo K-NN - Vizinho Mais Próximo (Nearest Neighbour Retrieval) para controle de movimentos de NPCs em um ambiente dinâmico (jogo).

2.2 OBJETIVOS ESPECÍFICOS

a) Pesquisar e definir os conceitos que envolvem o Algoritmo K-NN e sua utilização no desenvolvimento de jogos;
b) Pesquisar e definir etapas que envolvem a modelagem e o desenvolvimento de um jogo digital;
c) Criar um conceito para o game, definindo seu cenário e os NPCs;
d) Modelar a inteligência dos NPCs, utilizando o Algoritmo K-NN;
e) Abordar a integração da engine Unity 3D/2D com outras ferramentas de desenvolvimento;
f) Analisar o desempenho dos NPCs através do Algoritmo K-NN aplicada no resultado de suas ações.

3 INTELIGÊNCIA ARTIFICIAL E SUAS APLICAÇÕES

Segundo Fernandes (2005), a palavra inteligência vem do latim *inter* (entre) e *legere* (escolher). Tendo como significado aquilo que dá o poder ao ser humano escolher entre uma coisa e outra, permitindo que realize uma tarefa de forma eficiente.

Já a palavra artificial vem do latim também *artificiale*, que significa algo não natural, produzido pelo homem. Ou seja, Inteligência Artificial (IA) é um tipo de inteligência criada pelo homem para dotar os computadores de algum tipo de habilidade que simula o raciocínio humano.

Ainda segundo Fernandes (2005), o objetivo da Inteligência Artificial é o estudo e a modelagem da inteligência tratada como fenômeno. A inteligência é algo extremamente complexo, resultado de milhões de anos de evolução humana e entendê-la não é uma tarefa fácil. Uma das maiores áreas onde se aplica IA é a robótica, onde cientistas tentam fazer com que as máquinas possam interagir com o meio assim como os seres humanos.

De acordo com Russel e Norvig (2003), Inteligência Artificial é uma das ciências mais recentes do planeta, iniciada logo após a Segunda Guerra Mundial, em 1956.

Atualmente a Inteligência Artificial abrange uma enorme variedade de subcampos, desde a área de uso geral, como aprendizado e percepção, até tarefas específicas como jogos, demonstração de teoremas matemáticos, criação de poesia e diagnóstico de doenças.

Para Feigenbaum (1992 citado por FERNANDES, 2005), IA é a parte da Ciência da Computação voltada para o desenvolvimento de sistemas de computadores inteligentes, isto é, sistemas que exibem caraterísticas que se associam com inteligência no comportamento humano, como por exemplo a compreensão de uma linguagem ou resolução de um problema.

Para Ganascia (1993 citado por FERNANDES, 2005), os principais modelos de Inteligência Artificial são: Algoritmos Genéticos, Programação Evolutiva, Lógica Fuzzy, Sistemas Baseados em Conhecimento, Raciocínio Baseado em Casos, Programação Genética e Redes Neurais. A seguir são apresentados cada um desses modelos:

- **Algoritmos Genéticos:** é um modelo de aprendizado de máquina, inspirado no livro Origem das Espécies, escrito pelo naturalista inglês Charlie Darwin (1809-1882), criador da teoria evolucionista. Os Algoritmos Genéticos foram criados por Holland (1975) e objetivam emular operadores genéticos (específicos como cruzamento, mutação e reprodução) da mesma forma como são observados na natureza. Isso é feito através criação de indivíduos dentro da máquina, simulando uma evolução, seleção e reprodução, surgindo uma nova população.
- **Programação Evolutiva:** campo da IA concebido por Fogel em 1960, assemelha-se aos algoritmos genéticos, sendo que dá maior ênfase na relação comportamental entre parentes e seus descendentes. As soluções para os problemas são obtidas por meio de tentativas e transmitidas para a nossa população.
- **Lógica Fuzzy:** denominada lógica difusa ou lógica nebulosa, foi estruturada por Lofti Zadeh, na Universidade da Califórnia, no ano de 1965. Serve para representar, manipular e modelar informações imprecisas.
- **Sistemas Baseados em Regras:** são sistemas que implementam comportamentos inteligentes de especialistas humanos, utilizando-se de regras.
- **Programação Genética:** é um campo de estudo voltado para a construção de programas que visam imitar o processo natural da genética, trabalhando com métodos aleatórios.
- **Raciocínio Baseado em Casos:** é uma área de estudo da Inteligência Artificial que utiliza uma grande biblioteca de casos para consulta e resolução de problemas. Os problemas atuais são resolvidos através da recuperação e consulta de casos já solucionados e da consequente adaptação das soluções encontradas.
- **Redes Neurais:** É considerada uma classe de modelagem de prognóstico que trabalha por ajuste repetido de parâmetro. Estruturalmente uma Rede Neural consiste em um número de elementos interconectados (chamados “neurônios”) organizados em

camadas que aprendem pela modificação da conexão firmemente conectada as camadas.

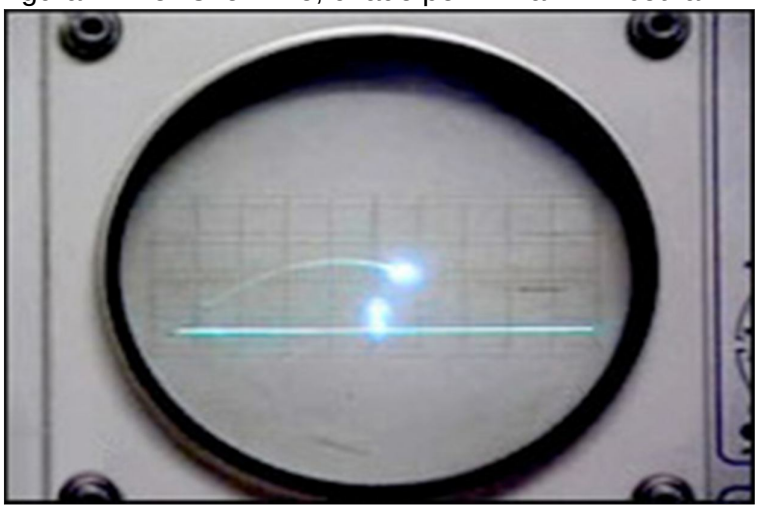
- **Algoritmo KNN (K-Nearest Neighbour):** O Algoritmo KNN ou Nearest Neighbour Retrieval (Vizinho mais próximo) é uma técnica simples que pode resolver um problema se baseando na sua distância com os casos existentes. A ideia chave desse modelo é que as propriedades de qualquer ponto da entrada x especifico tem probabilidade de serem semelhantes às propriedades de pontos na vizinhança de x. O algoritmo KNN calcula os K-vizinhos mais próximos a X, e classifica-o como sendo da classe que aparece com maior frequência dentre os seus K-vizinhos.

4 HISTÓRICO DOS JOGOS ELETRÔNICOS

O surgimento do vídeo game não tem data exata e pode até ser um assunto muito polêmico, o primeiro aparelho surgiu em 1958, quando William Higinbotham resolveu criar algo atrativo para a exposição permanente do Brookhaven National Laboratory (BNL) em Columbia nos Estados Unidos.

Utilizando um computador analógico, mais precisamente um monitor de osciloscópio, Higinbotham inventou um jogo de tênis interativo (Figura 1) para dois jogadores, onde era possível controlar o saque da bolinha e o momento da rebatida. Não havia placar exposto na tela, muito menos inicio ou fim de jogo, era somente uma interação de ação e reação. Sua experiência foi um sucesso, ganhando atualizações no futuro, como uma tela mais ampla e alteração na gravidade. Muitas pessoas visitavam o BNL para ver de perto o controle de objetos em uma tela. (LUZ, 2010).

Figura 1 - Tenis for Two, criado por William Hinbotham.



Fonte: Pacheco (2013).

Por nunca ter enfatizado seu jogo na área comercial ou continuar a criar aplicações mais complexas, muitas pessoas afirmam que Higinbotham não foi o criador do vídeo game, pois seu invento não passava de um experimento.

Em 1960, no Instituto de Tecnologia de Massachussets (MIT) surgiu o pioneiro dos jogos eletrônicos, Steve Russel. Ele era membro de um clube de entusiastas em eletrônica e modelismo, onde juntamente de seus amigos utilizava um TX-O computador para fins militares, que tinha um monitor de vídeo acoplado.

Em pouco tempo Russel e seus amigos se tornaram hábeis programadores e dominaram o equipamento. Com a chegada de um novo computador, o PDP-1, que era baseado em transistores e muito menor que seus anteriores, Russel resolveu desenvolver algo que tinha se tornado uma ideia fixa em sua cabeça, um jogo que fosse interativo e permitisse usar o monitor do computador. (LUZ, 2010).

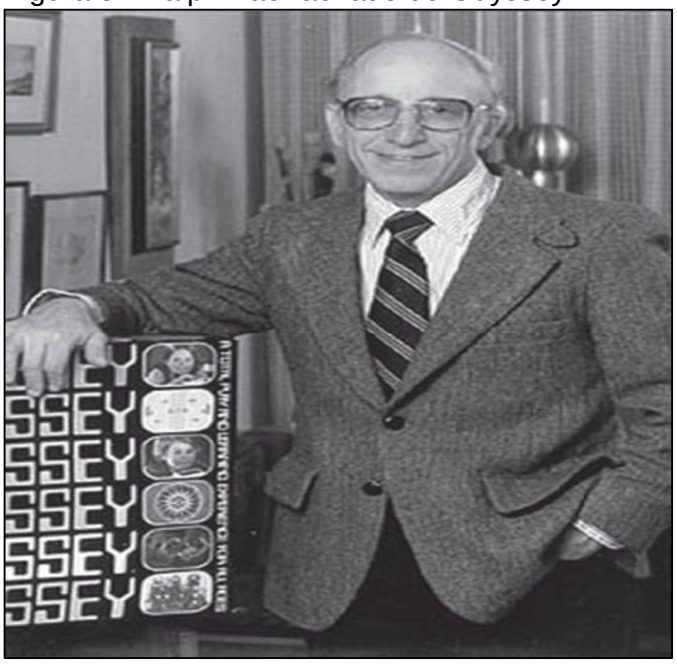
Durante seis meses de programação, eles desenvolveram o “Spacewar!” (Figura 2), uma batalha espacial entre duas naves, que controladas por quatro interruptores de alavanca (switch) podiam atirar torpedos entre si. Nasceu então o primeiro jogo. (LUZ, 2010).

Figura 2 - “Spacewar!” criado por Steve Russel.

Fonte: (UOL JOGOS, 2004)

Outra data que pode ser marcada como lançamento do vídeo game foi 1951, proposta pelo engenheiro chamado Ralph Baer (Figura 3), considerado até hoje o pai do conceito de vídeo game. Aos 29 anos de idade trabalhando em um fabricante de televisores, Baer teve a ideia de usar a televisão de uma maneira interativa, com algum tipo de jogo em sua tela. Seu chefe na época detestou a ideia e Baer a arquivou por mais de 15 anos. (LUZ, 2010).

Figura 3 - Ralph Baer ao lado do Odyssey.



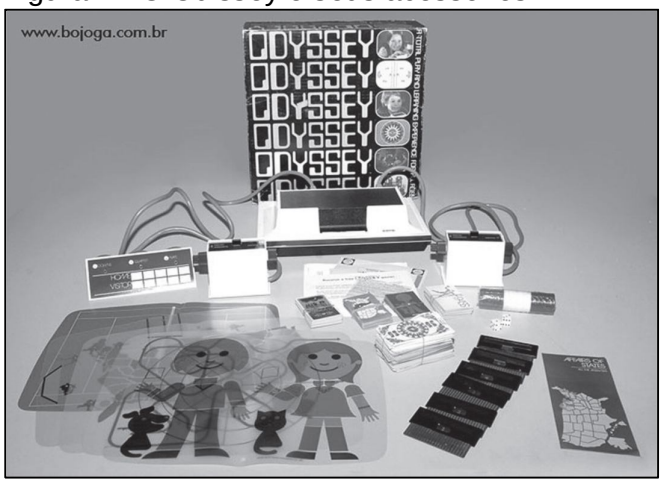
Fonte: Luz (2010, p. 23).

Em 1966, Baer era engenheiro chefe de uma empresa fabricante de equipamento militar e, então resolveu retomar sua antiga ideia, colocando um de seus funcionários para trabalhar o conceito que ele havia desenvolvido. Em pouco tempo, havia um aparelho capaz de gerar um rudimentar jogo de perseguição em uma tela, composto por dois pequenos pontos que se perseguem um ao outro.

Em dois anos, Baer e sua equipe conseguem criar um jogo rudimentar de pingue-pongue e suas variantes como Hóquei e Voleibol, jogos de perseguição e conceitos inovadores como a da pistola ótica (light gun) que podia apontar e atirar em objetos na tela.

Sua criação foi licenciada pela Magnavox® em 1971 e transformou-se no Odyssey® (Figura 4). Por isso, Baer é considerado o pai do vídeo game, pois deu forma de produto de mercado à sua criação e criou o conceito e vídeo game como mídia de entretenimento na sala de estar. (LUZ, 2010).

Figura 4 - O Odissey e seus acessórios.



Fonte: Luz (2010, p. 24).

Até a década de 1970 o computador era enxergado apenas como um processador de números. O Spacewar! ® foi o primeiro registro de uso do computador com uma interface gráfica simbólica, influenciando inclusive o desenvolvimento das interfaces gráficas (graphic user interfaces – GUI) nos anos 1970 no Centro de Pesquisa da Xerox em Palo Alto (Xerox Palo Alto Research Center). (LUZ, 2010).

Enquanto isso, Nolan Bushnell adapta o “Spacewar!” de Russel, criando o “Computer Space”, o primeiro arcade do mundo. Como os mainframes eram caros e ocupavam muito espaço, Bushnell criou uma máquina exclusivamente para se jogar o “Spacewar!”. (UOL JOGOS, 2004).

Bushnell decide, então, criar sua própria empresa em 1972, acreditando que um dos pontos para o fracasso do Computer Space® foi a pobre divulgação feita pela Nutting. Em busca de um nome e após algumas tentativas, Bushnell chega a uma palavra tirada do jogo oriental Go. A palavra é o equivalente a um cheque mate, no xadrez. Ele acha perfeito e batiza sua empresa: Atari®. (LUZ, 2010).

A Atari começou apenas com três funcionários, o próprio Bushnell, um engenheiro contratado e uma recepcionista. A empresa tinha ênfase na criação de pinballs, porém Nolan não teria desistido dos vídeo games. Em pouco tempo eles desenvolveram um jogo que foi batizado de Pong, um simulador de tênis de mesa. O sucesso foi tanto que empresa quadruplicou e finalmente o primeiro vídeo game foi vendido pelo mundo todo. (LUZ, 2010).

Em 1978, a empresa japonesa Taito lançou o primeiro jogo que trazia um desafio para os jogadores: o Space Invaders®. A mesma empresa também foi responsável por revolucionar a parte interativa dos jogos, pois seus personagens tinham animações. O sucesso foi tanto que a Casa da Moeda do Japão precisou a fabricar lotes extras de moedas de 100 ienes, pois os arcades viralizaram em todo o país. (LUZ, 2010).

O sucesso do Space Invaders® mostrava como o público estava amadurecido e esperava por títulos com novos temas, novos desafios. Até 1978 a grande maioria dos jogos ou era composta por simulações de esportes ou batalhas em que o jogador lutava contra um oponente humano. Nesse jogo seu oponente era a própria máquina, impiedosa, cuja única certeza era lhe derrotar em um momento ou outro. O lançamento de Space Invaders® (Figura 5) marcou o início de uma nova era de jogos criativos e inovadores e a tensão passou a fazer parte integrante do vídeo game. (LUZ, 2010).

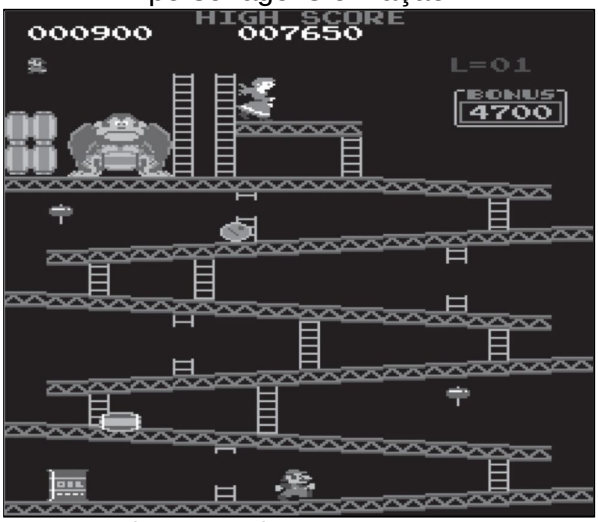
Figura 5 - Space Invaders.

Fonte: Luz (2010, p. 34).

A década de 80 foi marcada por vários outros nomes como Pacman®, Street Fighter, Pitfall, “E.T” entre outros.

Assim como Pacman®, o Donkey Kong® (Figura 6) se tornou fenômeno de mídia, aparecendo em programas de tevê e gerando uma série de itens licenciados, o que proporcionou à Nintendo® duas de suas maiores franquias Mario® e o próprio Donkey Kong®. (LUZ, 2010).

Figura 6 - Tela do Donkey Kong® com seus personagens em ação.



Fonte: Luz (2010, p. 42).

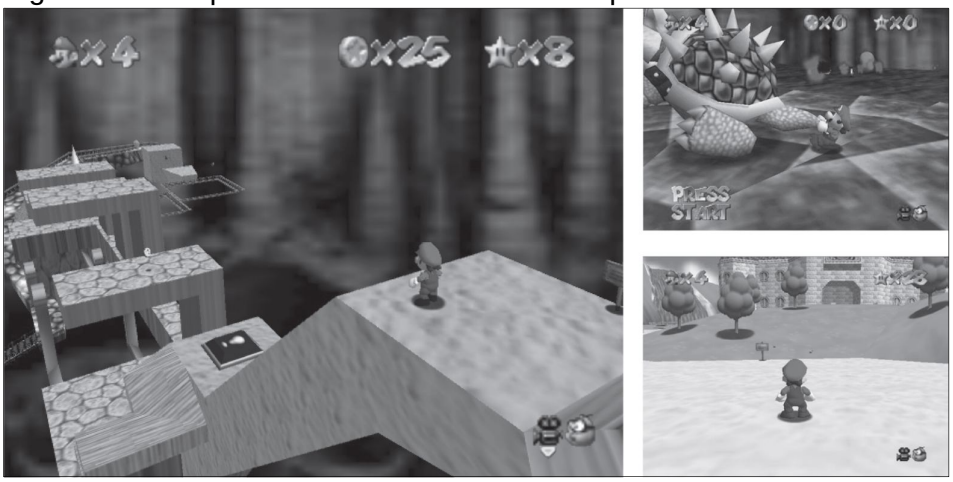
Na década de 90 a Nintendo vem com seu novo console chamado Super Nintendo, o modelo americano do Super Famicon lançado apenas no Japão. Lançaram então grandes títulos para o console como *Turtles in Time, Street Fighter 2, Super Mario World, Rock and Roll Racing*, etc.

Começa então uma batalha entre a Nintendo e a Sega. Com mais títulos a sua mão a Nintendo sai na liderança, porém diversos títulos saem exclusivamente para Mega Drive, como o *Sonic Hedgehog* e *Altered Beast.*

Após a era 16 bits, começam os consoles com gráficos de terceira dimensão (3D), sendo eles o Playstation, Nintendo 64, que ainda utilizava cartuchos, e o Sega Saturn. Lançado pela Sony, o Playstation foi sucesso garantido, seus jogos eram em CD-ROM e alcançavam todos os tipos de públicos. (LUZ, 2010).

Com medo de se perder no tempo a Nintendo lança jogos marcantes para seu console Nintendo 64, como Super Mario 64 (Figura 7), Mario Kart, Star Fox 64 e até a série POKEMON, animação japonesa lançada na época.

Figura 7 - O Super Mario 64® definiu o estilo plataforma em 3D.



Fonte: Luz (2010, p. 68).

A partir de 1998 começaram a surgir os consoles 128 bits, abrindo o mercado foi lançado o Dreamcast pela Sega. Apesar de ter um processamento ótimo e seus jogos terem mais de 1 gigabyte de tamanho, o vídeo game foi deixado de lado quando a Sony apresentou o novo Playstation 2, com seu poderoso processamento gráfico. (LUZ, 2010).

Com os vídeo games invadindo as casas de todo o planeta, a Microsoft resolveu desenvolver sua própria plataforma, baseada na tecnologia dos computadores. Surge então o Xbox, com placa gráfica da nVidia, disco interno de 8 gigabytes e placa ethernet para partidas online utilizando banda larga, superando então o console da Sony. (LUZ, 2010).

Depois de aguardar seus concorrentes entrarem no mercado, em 2001 a Nintendo lança seu console com arquitetura 128 bits, o Game Cube, que contava com processador IBM Power PC e chip gráfico da ATI. Mas como sempre apenas os exclusivos desenvolvidos pela própria empresa fizeram sucesso com o console, não conseguindo disputar no quesito de variedades de jogos.

Na disputa pelos melhores vídeo games e que atendessem melhor seus clientes, as empresas começaram a desenvolver consoles portáteis, como o PSP (Playstation Portátil) e o Nintendo DS, mais famosos da época. Esses entraram no mercado para atender o público que não tinha tempo para jogar os então chamados de consoles de mesa. (LUZ, 2010).

Continuando na linha do tempo, o mercado de jogos foi marcado pelo lançamento da sétima geração de consoles lançados pelas principais empresas do

ramo. O Playstation 3 da Sony e o Xbox360 da Microsoft mudaram o conceito de vídeo games, pois eles iam além de jogar, eram um centro de entretenimento na sua sala de estar, contando com reprodução online de vídeos, músicas e fotos a partir da sua televisão e em alta definição. (LUZ, 2010).

A Nintendo apesar de não entregar um produto que competisse com os gráficos e funções dos concorrentes, inovou nos controles, trazendo o Nintendo Wii. Os controles convencionais se transformaram em sensores de movimentos que colocavam o jogador dentro dos jogos, tendo uma imersão que até então nenhum outro vídeo game trazia. (LUZ, 2010).

Em 2010 mais uma tecnologia veio para mudar e inovar o vídeo game caseiro, o Kinect (Figura 8) trouxe a evolução ao captar os movimentos dos jogadores sem a necessidade de controles ou sensores conectados ao corpo. Fabricado pela gigante Microsoft, o Kinect era exclusivo para Xbox 360, sendo aprovado pelos jogadores, mas não muito pelos desenvolvedores da época, o aparelho não rendeu muitos títulos de sucesso, mas não deixou de impressionar com seus lançamentos. (LUZ, 2010).

Figura 8 - Kinect, sensor de movimento lançado Microsoft em 2010.

Fonte: Figueiredo (2013).

Em 2012 o mercado de jogos eletrônicos passa para uma nova geração. Os principais fabricantes mostram seus novos vídeo games, capazes de transmitir a realidade com seus gráficos reais e processamento poderoso. O Xbox One da Microsoft traz junto com ele um novo sensor Kinect, dessa vez muito mais explorado e capaz de reconhecer movimentos específicos. (LUZ, 2010).

O Playstation 4 da japonesa Sony é o forte concorrente do Xbox, também com hardware poderoso, ele traz títulos que impressionam na qualidade gráfica e na jogabilidade com seu controle DualShock 4. (LUZ, 2010).

A Nintendo não apostou apenas nos gráficos, mas na inovação dos controles, lançando o Nintendo WiiU, agora com movimentos realizados a partir de um gamepad com tela sensível ao toque. (LUZ, 2010).

5 INTELIGÊNCIA ARTIFICIAL APLICADA A JOGOS

A definição de jogos de computador, segundo Battaiola (2000 citado por GALDINO, 2007, p. 6),

> [...] pode ser definido como um sistema interativo que permite ao usuário experimentar, uma situação de conflito. Quase sempre, como plano de fundo dessa situação, existe um enredo, que define do que se trata o jogo, as regras que o jogador deve seguir e os objetivos que ele deve se esforçar para atingir.

Para os desenvolvedores de jogos eletrônicos, as aplicações computacionais de IA e o significado do termo no contexto de jogos são diferente dos encontrados no meio acadêmico. Para distinguir a Inteligência Artificial utilizada em jogos e no meio acadêmico, os desenvolvedores adotaram o termo Game AI (Artificial Intelligence). (FUNGE, 2004 citado por KISHIMOTO, 2004).

A diferença segundo Schwab (2004 citado por KISHIMOTO, 2004) entre IA Acadêmica e IA para jogos é o que cada uma tem como objetivo. No primeiro caso, objetivo da IA é buscar a solução para problemas caracterizados por serem extremamente difíceis, exemplo disso pode ser o reconhecimento facial e de imagens, entender e construir agentes inteligentes.

Na Inteligência Artificial para jogos, o intuito é utilizá-la para diversão. Segundo Schwab (2004 citado por KISHIMOTO, 2004) isso se deve pelo fato que os jogos eletrônicos são negócios, onde os consumidores querem comprar algo que proporcione diversão e não se importam como o personagem do jogo foi feito, apenas como ele age dentro do contexto.

Um dos problemas que podem ser encontrados na IA utilizada na indústria de jogos eletrônicos é a grande variedade de gêneros dos jogos existentes e os comportamentos dos personagens, o que pode resultar numa ampla interpretação do que é considerada IA para jogos. (BOURG, 2004 citado por KISHIMOTO, 2004).

De acordo com Tozour (2002 citado por KISHIMOTO, 2004) é vergonhoso que a Game IA seja chamada e considerada Inteligência Artificial, já que no campo de IA para jogos é necessário criar agentes e comportamentos apropriados num determinado contexto, embora a adaptabilidade da inteligência humana nem sempre é necessária para se produzir tais comportamentos. No início a programação de IA era mais focada na jogabilidade e não no comportamento exibido pelo computador,

que muitas vezes não se mostrava nada inteligente. Com o tempo a programação de IA foi evoluindo trazendo diversos tipos de comportamentos para os jogos eletrônicos, como mostra a Figura 9.

Figura 9 - Linha do tempo da IA em jogos.

Ano	Descrição	IA 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*.	Padrões de movimento
1975	*Gun Fight* lançado, personagens com movimentos aleatórios.	Padrões de movimento
1978	*Space Invaders* contém inimigos com movimentos padronizados, mas também atiram contra o jogador.	Padrões de movimento
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.	Padrões de movimento
1990	O primeiro jogo de estratégia em tempo real, *Herzog Wei*, é lançado. Junto, os jogadores puderam noticiar uma péssima busca de caminho.	Máquina de estados
1993	*Doom* é lançado como primeiro jogo de tiro em primeira pessoa.	Máquina de estados
1996	*BattleCruiser: 3000AD* é publicado como o primeiro jogo a utilizar redes neurais em um jogo comercial	Redes neurais
1998	*Half-Life* é lançado e analisado como a melhor IA em jogos até a época, porém, o jogo utiliza IA baseada em *scripts*.	Máquina de estados / *Script*
2001	O jogo *Black & White* é alvo da mídia a respeito de como as criaturas do jogo aprendem com as decisões feitas pelo jogador. Utiliza redes neurais, *reinforcement* e *observational learning*.	Diversos

Fonte: Kishimoto (2004, p. 4).

5.1 INÍCIO DA INTELIGÊNCIA ARTIFICIAL EM JOGOS

Muitos programadores do início da era de jogos eletrônicos implementavam padrões de movimentos ou movimentos repetitivos e/ou aleatórios para os personagens controlados pelo computador (como Galaga e Donkey Kong) como sendo a inteligência existente no jogo.

Esse fato foi principalmente causado pela falta de memória e limitação existente na velocidade de processamento. (SCHWAB 2004 citado por KISHIMOTO, 2004). Os jogos de estratégia como, por exemplo, Civilization de 1991 estão entre os pioneiros em IA para jogos, uma vez que esses gêneros de jogos necessitam de

uma boa IA para que sejam jogáveis, pois requerem que o computador controle grupo de personagens com estratégias e táticas complexas. Uma extensão dos jogos de estratégia são os jogos de estratégia em tempo real, onde toda a ação acontece em tempo real (ao contrário de outros jogos de estratégia, que ocorre em turnos). A IA para esse gênero de jogo deve realizar buscas de caminhos (pathfinding) para centenas de unidades em tempo real. (TOZOUR, 2002 citado por KISHIMOTO, 2004).

Dentro deste contexto, tratando-se da Game IA, ou seja, umas ramificações da inteligência artificial que caracteriza a mesma, como já explicado anteriormente, existem tópicos ou técnicas, para buscar soluções para desenvolvimentos dos jogos. Algumas das principais destacadas por Karlsoon (2005 citado por GALDINO, 2007) são:

- **Máquinas de Estado Finito:** a forma comumente utilizada para determinar e definir o comportamento dos personagens dentro de um jogo. Sendo formada por uma série de estados, que sofrem transação segundo regras, representando ações que os personagens poderão executar no ambiente do jogo.
- **Path-Finding:** é a busca de caminho entre dois pontos do cenário. Para isso utilizam-se algoritmos específicos, como o A*, que determinam os caminhos a serem percorridos para o desvio de possíveis obstáculos, inimigos etc.;
- **Padrões de Movimentos:** técnicas e meios que determinam o comportamento durante a movimentação dos objetos.
- **Sistemas baseados em regras:** Conjunto de parâmetros e regras definidas, de forma que os parâmetros processem as regras e definam uma saída para uma ação do personagem.
- **Lógica Fuzzy:** a lógica fuzzy ou difusa é uma lógica que admite valores intermediários entre o falso e o verdadeiro, como o “talvez”. Esse tipo de lógica analisa as sentenças de forma incerta e procura representar as tomadas de decisão humana.

Ainda concluindo sua explicação, Galdino (2007), afirma que essas áreas embora sejam utilizadas na IA voltada para pesquisa acadêmica, sempre irão

contribuir para a Game IA assim como o contrário também é valido, visto que a solução de um problema na área de games possa ser utilizada futuramente para solucionar ou melhorar métodos de busca na área de pesquisa.

5.2 NON-PLAYER CHARACTER (NPC)

Non-Player Character (NPC), são personagens que não são controlados pelo jogador em jogo eletrônico. Eles existem nos jogos multiplayer, ou seja, com vários jogadores e também singleplayer, onde só há um jogador, sendo que é nesse estilo ele ganha mais destaque, pois é o adversário do jogador é controlado pela máquina.

Para que um jogo possa se tornar mais realístico em termo do comportamento dos personagens, os NPCs devem ter um nível de inteligência, que varia geralmente do gênero do jogo e da dificuldade selecionado pelo jogador. Antigamente essa inteligência era implementada através de regras por meio da estrutura if-then-else, mas além de tornar mais complexo a mudança em algum comportamento do personagem ainda trazia uma precariedade em comportamentos mais complexos. Desta maneira identificou-se que para o NPC ser capaz de interagir de forma mais real e atrativa, dentro de um ambiente de jogo era necessário buscar novas técnicas que fizessem com que eles tivessem a capacidade de agir de forma racional e aprender com suas ações. (BORGES; BARREIRA; SOUZA, 2009).

Os NPCs possuem algumas características que estabelecem se eles são inteligentes, como por exemplo, serem autônomos. Um agente autônomo é o que consegue operar com completa autonomia, decidir como relacionar os dados obtidos com ações de modo que seus objetivos sejam atingidos com sucesso. (MAES, 1995 citado por BORGES; BARREIRA; SOUZA, 2009). Assim pode-se dizer que agentes autônomos são aqueles capazes de perceber o ambiente através dos seus sensores e operar com completa autonomia, sem qualquer intervenção, buscando sempre atingir seus objetivos da melhor forma possível.

Outra característica de agente de um NPC são os Emocionais Hedonistas. O Hedonismo segundo Gomes (2007 citado por BORGES; BARREIRA; SOUZA, 2009) é uma teoria ou doutrina filosófico-moral que coloca como maior valor, o prazer e a satisfação individual. Dessa maneira os agentes hedonistas irão tomar decisões baseando-se no prazer e na dor e estas se transformarão em emoções, no qual cada emoção irá codificar um sub objetivo no agente. A ideia de usar agentes

emocionais hedonistas para a abordagem de comportamentos de personagens está na característica de considerar emoções como base para tomada de decisão. Um personagem que tenha emoções, ou simula uma, consegue se aproximar de um comportamento de um outro jogador, e não como um simples personagem de computador.

Conclui-se então que essas definições tem um ponto em comum: atingir objetivos. Assim para se alcançar esses objetivos o NPC deve seguir uma estratégia, que juntamente do jogador humano, trará mais realidade e um nível qualidade maior para o jogo.

6 ALGORITMO K-NN

Segundo Fernandes (2005), o algoritmo K-NN ou Nearest Neighbour Retrieval (Vizinho mais próximo) é uma técnica simples que pode resolver um problema se baseando na sua distância com os casos existentes.

Segundo Russel e Norvig (2003), a ideia chave desse modelo é que a propriedade de qualquer ponto de entrada especifica tem probabilidade de serem semelhantes às propriedades de pontos na vizinhança. Esse algoritmo se dá durante a fase de teste/classificação, onde o algoritmo faz uso dos K-vizinhos mais próximos, para estimar a classe de um novo padrão X, o algoritmo KNN calcula os K-vizinhos mais próximos a X e classifica-o como sendo da classe que aparece com maior frequência dentre os seus K-vizinhos.

De acordo com Lagemann (1999 citado por FERNANDES, 2005, p. 43), “[...] os aspectos de definição e identificação dos índices é fator fundamental para uma recuperação de sucesso.”. Garantindo estes aspectos, a técnica de busca irá indicar em qual região do espaço de busca o problema em questão está inserido, sendo o próximo passo encontrar os casos mais parecidos usando comparação e valorização.

Cabe destacar duas características importantes do algoritmo k-NN: a regra de classificação e a função que calcula a distância entre dois pontos (instâncias). A regra de classificação informa de que modo o algoritmo vai tratar a importância de cada um dos k elementos selecionados – os k mais próximos. Assim, procura-se classificar x atribuindo a ele o rótulo representado mais frequentemente dentre as k amostras mais próximas. Já a função de distância serve para medir a distância entre dois elementos de forma a poder identificar quais são os mais próximos.

Para efetuar o cálculo do algoritmo KNN podem ser utilizadas diversas medidas de distância, mas de acordo com Russel e Norvig (2003), a distância euclidiana (Figura 10) é a mais simples para ser usada, principalmente em casos bidimensionais. A fórmula é representada da seguinte maneira:

D = Distância euclidiana.

X e Y = pontos de entrada de dados que serão calculados.

∑ = somatória.

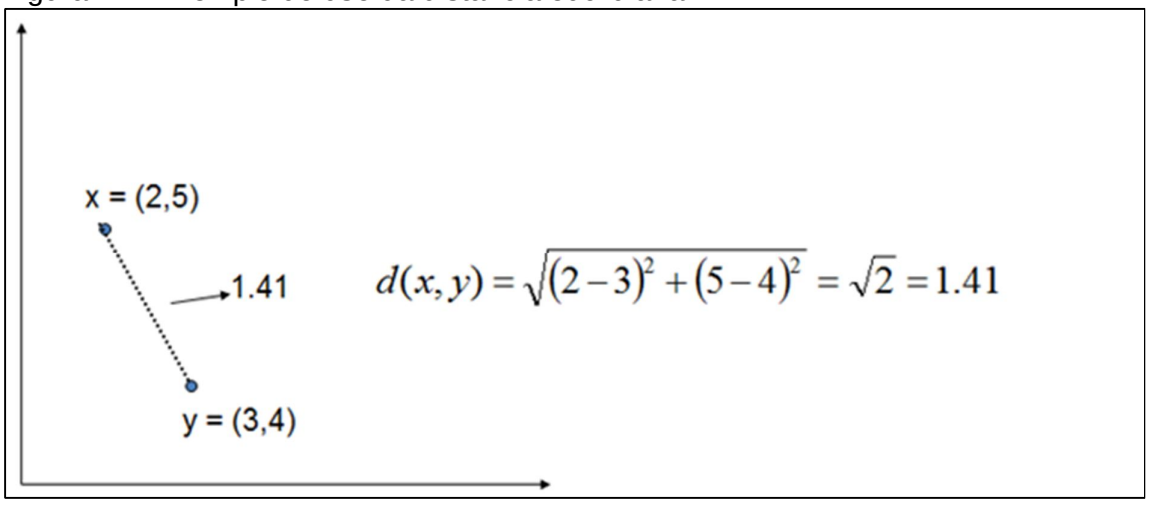
Figura 10 - Fórmula da distância euclidiana.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Fonte: Elaborada pelo autor.

A Figura 11 exemplifica o cálculo realizado (distância euclidiana) considerando dois pontos. Observa-se que o ponto “y” tem dois valores que estão sendo subtraídos dos valores do ponto “x”, elevando a segunda potência, somando e tirando a raiz quadrada. Dessa maneira se encontra a distância entre esses dois pontos “x” e “y”, podendo utilizar esse valor para similaridade entre os vizinhos.

Figura 11 - Exemplo do uso da distância euclidiana



Fonte: Elaborada pelo autor.

7 PLANEJAMENTO DE JOGOS

Segundo Clua e Bittencourt (2005) como qualquer outro software a produção de um jogo computadorizado, seja 2D ou 3D, precisa passar por um processo de desenvolvimento. No caso das aplicações de entretenimento digital, tais com os jogos, é necessário tratar dos aspectos artísticos neste processo, como modelagem e design. Assim para o desenvolvimento de um jogo 3D são utilizadas as seguintes etapas:

a) Confecção do Design Bible;
b) Produção de áudio e imagens 2D;
c) Modelagem 3D;
d) Desenvolvimento dos artefatos computacionais. Escolha ou desenvolvimento da Engine;
e) Integração dos aspectos artísticos com os aspectos computacionais.

7.1 DESIGN BIBLE

Segundo Clua e Bittencourt (2005), assim como um roteiro é primordial para produção de um filme, nos jogos também é necessário se ter um documento que possua todas as suas especificações. O nome desse documento é Design Bible e nele possui todas as instruções para futuros desenvolvedores do jogo, sendo tão importante que apenas estando pronto pode-se começar a produção do jogo. Para produção do jogo é necessário os seguintes elementos designados no Design Bible.

7.1.1 Roteiro

Se assemelha aos roteiros de filmes, sendo um item fundamental para o processo de criação, principalmente para agradar e convencer os investidores que o produto tem potencial. É nesse item que o jogo mostra o diferencial em relação aos outros, pois mostrará seu gênero e onde o jogador se encaixará na história. (CLUA; BITTENCOURT, 2005).

7.1.2 Game design

Entende-se por game design a conceituação artística do jogo. Hoje em dia, dada a complexidade das histórias e dos ambientes elaborados, essa etapa é necessária ser escrita por um artista, que irá expor as principais características dos cenários, esboços dos personagens, descrição das texturas, mapas e descrições das fases (denominadas, level design). (CLUA; BITTENCOURT, 2005).

7.1.3 Game play

Nessa etapa do Design Bible será abordada a jogabilidade. Entendem-se por jogabilidade as regras do jogo e o balanceamento das regras (game balancing), aqui ficará claro que o jogo é divertido e irá proporcionar desafios interessantes ao jogador. Uma etapa muito importante para os programadores. (CLUA; BITTENCOURT, 2005).

7.1.4 Interface gráfica

Pode-se dividir a interface gráfica em ingame e outgame. A primeira consiste na instrumentação disponível durante o jogo e é responsável pela entrada de dados do jogador para a aplicação. A interface outgame é a forma de apresentar a introdução do jogos, sua configuração, instruções, carregar um jogo salvo anteriormente, entre outras operações de suporte durante todo o jogo. A melhor interface é aquela que passa desapercebida para o jogador, permitindo que o mesmo possa focar-se no desenrolar da história e das ações. (CLUA; BITTENCOURT, 2005).

7.2 PRODUÇÃO DE ÁUDIO E IMAGENS 2D

Além de utilizar as engines para criação de jogos, é necessário produzir sons e imagens 2D. Para criação de sons existem diversas ferramentas profissionais, dentre as mais conhecidas são o Cubase e o SoundForge, ambas são pagas. Como alternativa gratuita pode ser utilizado o programa Audacity, que permite criação de

trilhas e efeitos sonoros, porém não tem tantos recursos com o SoundForge, por exemplo. No aspecto de áudio, a OpenAL desenvolvida pela Loki Software, trate-se de uma alternativa de implementação de som bastante interessante, pois permite adicionar som 3D aos jogos, com uma biblioteca estruturada na forma de máquina de estados que oferece um conjunto de primitivas de alto nível para manipulação de sons. (CLUA; BITTENCOURT, 2005).

Os jogos 3D não são constituídos apenas por modelos tridimensionais, na produção de um jogo é necessário criar imagens bidimensionais, que serão utilizadas como texturas na parte ingame, ou componentes gráficos como botões e janelas no quesito outgame. Como alternativa de software, o GIMP é uma ótima escolha para produção desse conteúdo, fornecendo funcionalidades semelhantes a do Photoshop, como pincéis, trabalhar com camadas, uso de máscaras, efeitos de luz e outras coisas. (CLUA; BITTENCOURT, 2005).

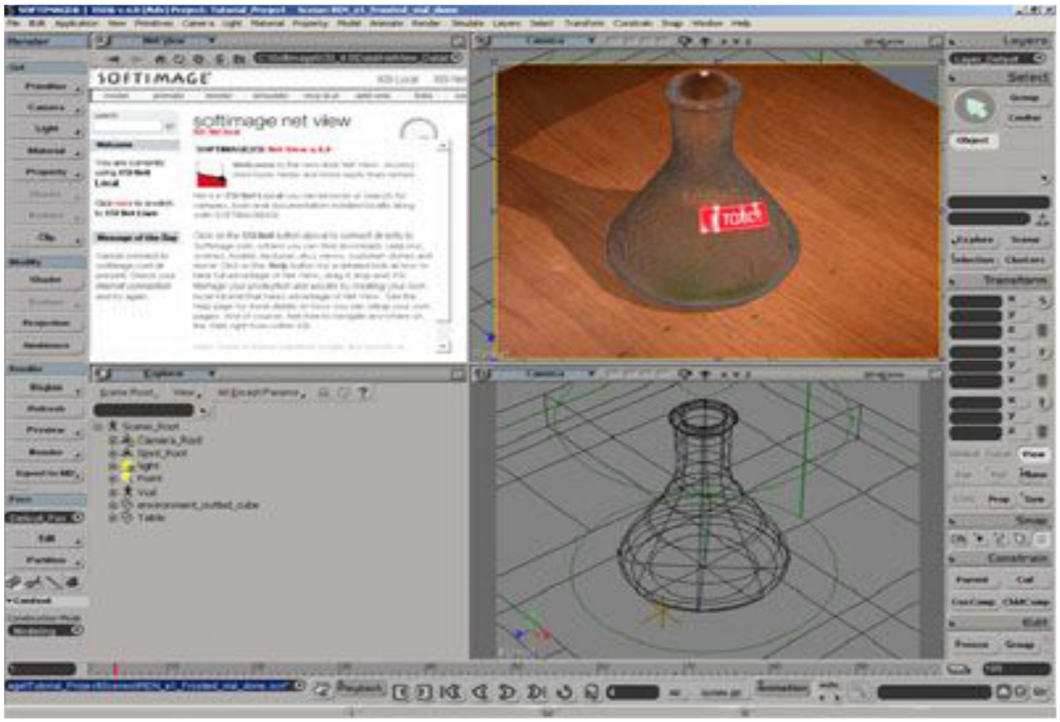
As texturas faladas anteriormente, consistem em imagens que representarão algum material especifico, quando aplicado sobre um modelo, por exemplo, madeiro, concreto, plástico, etc. Algumas imagens de texturas são fotografias de materiais reais, esse procedimento aumenta o realismo das imagens que serão utilizadas sobre os modelos.

7.3 MODELAGEM 3D

Dentro da produção do jogo, uma equipe ficará responsável por criar objetos geométricos tridimensionais para utiliza-los como cenários, estruturas e personagens. A geometria de um jogo é dividida em dois tipos: modelagem estrutural e modelagem de elementos dinâmicos. Esta diferença existe pelo fato de que os modelos estruturais por não sofrerem alteração da posição, terão um pré processamento, de maneira a otimizar o processo de renderização. A modelagem estrutural basicamente consiste na criação do cenário em si, o terreno e outros elementos estáticos. (CLUA; BITTENCOURT, 2005).

Dentre os softwares mais utilizados para modelagem de recursos 3D, podemos citar o AutoDesk 3DS MAX, AutoDesk MAYA, AutoDesk Softimage (Figura 12) e o LigthWave.

Figura 12 - Exemplo de modelagem 3D utilizando o software Softimage da Autodesk.



Fonte:CLUA e BITTENCOURT (2005, p. 1331).

Segundo Clua e Bittencourt (2005) esses programas oferecem diversos recursos, o que os torna ótimos para este processo, sendo eles:

- **Ferramentas de modelagem baseadas em polígonos:** A modelagem é feita em polígonos, por isso é necessário que o software ofereça esse recurso de forma fácil e intuitiva.
- **Ferramentas intuitivas para texturização:** A riqueza de uma modelagem se dá pelas texturas, que muitas vezes tem que ser aplicadas em polígonos separados utilizando um mapeamento de texturas.
- **Boas ferramentas para otimização de polígonos:** É comum durante a modelagem criar objetos que possam ter mais polígonos que o jogo pode suportar, por isso o software tem que contar com um ferramenta que diminua o número de polígonos sem perder muita qualidade.
- **Boa interface de visualização:** Essa ferramenta é essencial pois permite que o artista acompanhe o processo em tempo real, sabendo a priori como o mesmo será visto no jogo.

Um software livre e gratuito que pode ser utilizado é o Blender 3D, que oferece funcionalidades de modelagem, animação, renderização, pós-produção e criação de objetos tridimensionais (3D), tendo como o único ponto negativo a sua interface pouco intuitiva. (CLUA; BITTENCOURT, 2005).

7.4 ENGINES

Para desenvolver um jogo, é necessário uma engine que será responsável por lidar com o hardware gráfico, controlar os modelos para serem renderizados, tratar das entradas de dados do jogador, tratar de todo o processamento de baixo nível e outros cosia que o desenvolvedor de jogos normalmente não deseja fazer ou não tem tempo para ficar se preocupando com isso. Uma engine é um componente software capaz de executar um processamento que resulta em um determinada saída, precisando sempre de um front end que irá fornecer as informações de entrada e exibir as informações de saída. (CLUA; BITTENCOURT, 2005).

Existem diversas definições para um engine, mas todas tem os mesmos pontos em comum, como:

a) Permitir que o desenvolvedor possa criar jogos diferentes, usando a mesma engine.
b) Pode reaproveitar com facilidade o código desenvolvido em projetos anteriores.
c) Abstrair a manipulação de APIs.
d) Possibilitar uma fácil integração entre código e modelagem.

Uma engine é composta por diversas ferramentas, cada uma responsável por uma etapa do processo de criação de um jogo. Segundo Clua e Bittencourt (2005), as mais comuns são:

- **Engine Core:** É o coração da engine, responsável por executar a aplicação, renderizar as cenas e manipular objetos da fase, pode ser dizer que a engine core é o sistema operacional do jogo;
- **Engine SDK:** é o código fonte da engine core. Através dele pode-se mudar o funcionamento da engine alterando seus códigos.
- **Level Editors:** Através dele é possível unificar as modelagens feitas em diversos programas, associa-los a programação, inserir códigos

scripts, etc. Em algumas engines também se pode criar modelos tridimensionais.

- **Conversores/ Exportadores:** Os recursos são normalmente feitos em diversos softwares, assim a engine deve fornecer instrumentos para importar estes modelos para o formato específico dela, podendo ser exportados ou importados por plug-ins e até mesmo pelo próprio level editor.
- **Builders:** São ferramentas inclusas no level editor para algumas operações de pré processamento de objetos.
- **Linguagem Script:** Grande parte do desenvolvimento da lógica do jogo e da IA aplicada nos elementos dinâmicos serão feitas através de scripts e não diretamente sobre a engine core. Cada engine possui a sua linguagem de programação, sendo as mais comuns o C# e Javascript.

Conclui-se então que de acordo com Clua e Bittencourt (2005) a engine é composta por várias “sub-engines” onde cada uma é responsável por tratar o tipo de problema envolvido em jogos, como renderização, física, som e inteligência artificial.

7.6 UNITY 3D

Segundo Coelho (2010), Unity 3D é um software utilizado para a criação de jogos e aplicações interativas que permitem a visualização de ambientes tridimensionais em tempo-real. A partir de ferramentas como Blender ou AutoDesk 3D Studio Max, é possível importar modelos 3D e assim criar cenários e personagens. Criado em 2005, o Unity era exclusivo para a plataforma Macintosh, mas desde então a equipe vem atualizando o programa que passou a ser disponível para Windows também.

O Unity 3D permite a criação de cenários em tempo-real, podendo mover objetos e luzes de acordo com o gosto do usuário (Figura 13). Também é possível criar interações, animações, definir os controles que serão utilizados no jogo e muito mais ferramentas que uma engine pode oferecer.

Figura 13 - Exemplo de um cenário no UNITY 3D.

Fonte: COELHO (2010, p. 36).

8 PROCESSO DE TESTES PARA JOGOS

O processo de testes de um jogo não envolve apenas a procura por falhas, mas também a jogabilidade e aceitação dos usuários, por esse motivo, os testes são distribuídos entre todas as fases do seu desenvolvimento. Os testes de funcionalidade e bug fix possuem as características das abordagens clássicas da Engenharia de Software. Existem, porém uma classe de testes que é particular ao desenvolvimento de games: o chamado Playtest. (JUNIOR; NASSU; JONACK, 2002).

No Playtest os programadores analisam a aceitação do game e a reação dos jogadores. Um Playtest pode ser aberto, aonde é disponibilizada uma versão do jogo para download ou marca-se um dia e local para testes ou fechado, com testadores previamente selecionados, podendo ser esses profissionais ou qualquer interessado que possa dar uma opinião para os produtores. O diferencial do uso de profissionais é que além de simplesmente apontar o problema, eles são capazes de dar informações mais detalhadas e sugestões que auxiliam em muito a correção. (JUNIOR; NASSU; JONACK, 2002)

Segundo Junior et al (2002) as abordagens de teste variam conforme a equipe e fase de desenvolvimento, sendo as mais conhecidas e utilizadas:

- **Usuários jogam enquanto programador observa:** Nessa abordagem, o programador não tem contato algum com o jogador. Ele apenas mantem-se nas proximidades fazendo anotações das reações e dificuldades que o jogador encontra durante a seção de testes. Essa metodologia se mostra eficiente na descoberta de dificuldades de jogabilidade e de partes tediosas ou com grande dificuldade em um game.
- **Usuários jogam enquanto programador faz perguntas:** Conforme o jogador vai avançando no jogo, o programador vai lhe questionando sobre aspectos do jogo, dificuldades e o que ele sente enquanto joga. Funciona bem para jogos onde a ação seja constante e dinâmica, uma vez que o programador não teria tempo de captar todas as reações do jogador se não lhe fizesse perguntas.

- **Usuários jogam e fazem relatórios:** Muito utilizada em Playtests de grande porte, essa abordagem exige que o usuário jogue e faça um relatório do que achou do jogo. Nesse modelo se encaixam bem os “testadores profissionais”, que já estão acostumados a avaliar jogos. Durante o processo de testes, os desenvolvedores devem estar a par do problema de vazamento de informações.
- **Testes internos de jogabilidade:** Além da ajuda dos usuários, as empresas também têm seus próprios testadores, esses são chamados Game Designers. Esses profissionais são responsáveis por encontrar bugs e problemas, testar a dificuldade do jogo, evitar que existam trapaças, etc. Tudo o que esses profissionais realizam no jogo devem ser redigidos em um Plano de Testes, assim toda equipe consegue acompanhar o que já foi feito e testado.

9 APRENDIZAGEM DA MÁQUINA

Existem diversas abordagens diferentes de IA para solucionar uma grande variedade de problemas. Uma dessas abordagens é o aprendizado de máquina, uma das áreas mais relevantes dentro da IA. O aprendizado é qualquer processo no qual um sistema melhora seu desempenho através da experiência. (ARRUDA, 2015).

A aprendizagem automática ou Aprendizado de Máquina (Machine Learning) é um sub-campo da Inteligência Artificial dedicado ao desenvolvimento de algoritmos e técnicas que permitam ao computador aprender, isto é, que permitam ao computador aperfeiçoar seu desempenho em alguma tarefa, adquirindo conhecimento de forma automática. Enquanto que na Inteligência Artificial existem dois tipos de raciocínio: o dedutivo (no qual se deduz a verdade da conclusão se as premissas forem verdadeiras) e o indutivo (que extrai regras e padrões de grandes conjuntos de dados). O Aprendizado de Máquina só se preocupa com o indutivo. (ARRUDA, 2015).

De acordo com Arruda (2015) surgiu da ideia de criar programas que aprendam um determinado comportamento ou padrão automaticamente a partir de exemplos ou observações. A ideia é que um programa de computador tome decisões baseado em experiências acumuladas através da solução bem sucedida de problemas anteriores. Assim, guarda alguma relação com o aprendizado humano, já que seres humanos (e outros animais) são capazes de generalizar a partir de exemplos.

Deste modo, o objetivo do Aprendizado de Máquina é programar computadores para aprender um determinado comportamento ou padrão automaticamente a partir de exemplos ou observações. Hoje em dia muitas aplicações utilizam algoritmos de Aprendizado de Máquina, incluindo sistemas para prever o comportamento de clientes a partir de dados de compras, reconhecer faces ou voz, ou extrair conhecimento de dados biológicos. (ARRUDA, 2015).

Os algoritmos de Aprendizado de Máquina procuram padrões dentro de um conjunto de dados. Esses algoritmos existem há bastante tempo, mas nunca houve uma quantidade tão grande de dados digitais disponíveis para alimentar esses algoritmos como hoje graças a dois fatores: a informatização em massa e o surgimento da Internet. Atualmente as empresas têm milhares de informações de

milhares de usuários, podendo comparar o seu comportamento com o dos demais. Muitas outras áreas tiveram esse aumento de informações disponíveis como os serviços, o que fez aumentar o interesse da academia e do mercado nas técnicas de aprendizado de máquina, inclusive sua aplicação na área de jogos. (ARRUDA, 2015).

10 TRABALHOS CORRELATOS

No trabalho de Paula et al (2006), os autores descrevem a criação de uma arquitetura para construção de jogadores automáticos para a classe dos jogos de estratégia baseados em impérios. A arquitetura proposta (com três camadas) foi construída através de técnicas de raciocínio baseado em casos (RBC), mais especificamente planejamento baseado em casos (PBC). O objetivo da arquitetura foi povoar o mundo do jogo com jogadores automáticos. A primeira camada é responsável pela recuperação dos planos mais semelhantes ao estado atual do jogador automático e na aplicação do plano no ambiente do jogo. Esta base de planos é construída a partir da captura das ações realizadas por jogadores humanos. Caso haja falha no plano, a segunda camada, denominada camada reparadora, permite a correção do plano e a anotação do possível motivo do erro. A terceira camada, por sua vez, é responsável por a partir da informação de erro preenchida pela segunda camada prevenir possíveis erros antes de ocorrerem. O ambiente de aplicação desta arquitetura foram os jogos online multiplayer devido à necessidade de captura de planos. Os resultados obtidos no teste demonstraram a aplicabilidade da arquitetura explicitando que os hábitos do grupo foram associados ao estilo do agente.

Na pesquisa realizada por Borges, Barreira e Souza (2009), os autores procuraram demonstrar a interação entre jogo e jogador. Os jogos de computadores são softwares de entretenimento que devido ao seu constante crescimento vem focando mais na interatividade e jogabilidade do jogo, na tentativa de trazer oponentes e aliados cada vez mais inteligentes. Um dos responsáveis por essa interação são os personagens controlados pelo jogo, os NPC (Non Player Character), que atuam de forma inteligente para assim trazer mais realidade com o seu comportamentos que a cada dia fica mais complexo.

O trabalho de Machado (2012) avalia ambientes complexos e dinâmicos, onde Sistemas Especialistas e Máquinas de Estados deixam de ser soluções viáveis. Algoritmos Genéticos, da mesma forma, necessitam de certo conhecimento do domínio do problema para a função de avaliação. Redes Neurais então aparecem para solucionar todos os problemas, se não fossem as representações pouco compreensíveis geradas por seu treinamento e seu tempo de processamento consideravelmente alto. Sua tese apresenta uma solução para agentes inteligentes

imersos neste tipo de ambiente, o algoritmo NC-RLA. Caracterizada por um comportamento evolutivo e um motor baseado em classificadores numéricos, ela garante a evolução e adaptação do sistema em aplicações com exigência de processamento em tempo real. Através de experimentos em um jogo de estratégia e em uma simulação de vida artificial, foi demonstrada sua capacidade de gerar comportamentos emergentes no domínio dos jogos eletrônicos. Apesar de alcançar soluções piores em comparação com o Algoritmo Genético, seu desempenho e independência de informações do ambiente comprovaram seu potencial por meio da incorporação do classificador M5P.

11 METODOLOGIA

Este trabalho de pesquisa foi dividido em duas etapas: fundamentação teórica e desenvolvimento de software. Na fundamentação teórica (capítulos 3, 4, 5, 6, 7, 8 e 9), foram abordadas, teorias, algoritmos e ferramentas computacionais necessárias ao desenvolvimento do jogo proposto. A metodologia adotada nesta etapa consiste, basicamente, no estudo dos conceitos, teorias, algoritmos e métodos computacionais relacionados ao tema desta pesquisa: desenvolvimento de jogos utilizando a ferramenta gratuita Unity 3D e algoritmo KNN.

Este levantamento bibliográfico foi baseado em consultas à literatura especializada, incluindo: livros e artigos científicos, bem como consultas a sites da Internet. Basicamente são abordados os seguintes tópicos: histórico dos jogos e Game IA (breve histórico sobre jogos computacionais e para consoles com o intuito de mostrar a evolução destes jogos e das técnicas de inteligência artificiais utilizadas), algoritmo K-NN (conceitos de Inteligência Artificial com foco em algoritmos aprendizado de máquina com o intuito de contextualizar e mostrar a sua utilidade para o projeto), Planejamento de Jogos (caracterizando os passos para criação de um jogo e abordando características do motor de jogos, Unity 3D).

A metodologia utilizada na etapa de desenvolvimento do jogo envolveu as etapas seguintes: Definição das especificações do jogo, produção artística, definição da engine e integração dos elementos artísticos e computacionais.

Segundo Clua e Bittencourt (2005), a parte de especificação do jogo é uma das partes mais importantes no desenvolvimento, assim como não é possível criar um filme antes de ter um roteiro bem elaborado, é impossível desenvolver um jogo sem antes documentar todas as suas especificações. Nesta etapa são definidos os seguintes elementos: roteiro (definição história e estilo do jogo), game design (conceituação artística do jogo, envolvendo principais características dos cenários, personagens, elementos comportamentais e descrições das fases), jogabilidade (regras e desafio proporcionados aos usuários), interface gráfica (modos de interação do usuário com o jogo). Portanto, nesta etapa, foi realizado todo o planejamento envolvendo os elementos bidimensionais como cenário, objetos e os personagens (avatares). Para o desenvolvimento do conteúdo 2D foram utilizadas modelagens 2D, chamadas Sprites, providas por bases gratuitas e algumas delas fornecidas pela própria ferramenta de desenvolvimento, o Unity 3D.

Na etapa de produção artística foram definidas e utilizadas ferramentas computacionais para a produção de elementos como imagens e áudio utilizados durante o jogo bem como elementos de interface da aplicação tais como, botões, janelas e outros componentes gráficos.

A etapa de definição do engine consistiu na escolha do motor de jogo que é o componente de software responsável por manipular o hardware gráfico, controlando toda a parte de renderização de elementos, bem como, as entradas do usuário e respostas da aplicação. Este elemento de software lida com o processamento de baixo nível, facilitando o desenvolvimento já que abstrai do desenvolvedor de jogos a necessidade de lidar diretamente com elementos complexos envolvidos na arquitetura de jogos. Neste projeto optou-se pelo motor de jogo Unity3D para o desenvolvimento dos scripts de Inteligência Artificial e controle da aplicação. O capítulo 7 descreve a Unity 3D e mostra como esta ferramenta funciona e como a mesma serviu para o apoio e a otimização deste projeto. O projeto foi desenvolvido utilizando Sprites 2D e os scripts para o desenvolvimento dos algoritmos e do controle da aplicação foram programados em linguagem JavaScripit.

O projeto foi desenvolvido em um computador Intel® Core™ i5, 2.5 GHz 2.5GHz, 6GB de memória RAM, disco rígido de 750GB, SO 64 Bits.

11.1 DEFINIÇÕES DAS ESPECIFICAÇÕES DO JOGO

11.1.1 Roteiro

O jogo desenvolvido por esse projeto é do gênero tabuleiro, com sua mecânica baseada em turnos, podendo ser jogado pelo usuário ou pelo computador, no caso onde será aplicada a inteligência artificial.

O jogador controla um herói pelo ambiente do tabuleiro e seu objetivo é reunir todos os diamantes da tela, evitando obstáculos.

Cada movimento feito pelo jogador é registrado, após isso o algoritmo K-NN faz uma comparação entre o cenário atual e os registrados no histórico de jogadas, buscando uma similaridade entre os ambientes para assim verificar qual o melhor caminho a ser utilizado pelo NPC, sempre se baseando no modelo mais próximo do que o jogador efetuou. A hipótese é que a reprodução dos movimentos feitos pelos

jogadores humanos possa contribuir para que o NPC atinja os objetivos no jogo, desde que haja similaridade entre os cenários considerados.

11.1.2 Definição do problema

Os diamantes pertencentes ao rei foram roubados e levados a um castelo abandonado.

Para conseguir cumprir sua missão, foi enviado o melhor herói do reino e ele terá que usar toda sua estratégia, desviando sempre dos obstáculos, fazer o menor caminho.

11.1.3 Personagens

A história conta com um protagonista, o herói, que poderá ser controlado pelo jogador ou pelo computador.

Esse personagem terá que se movimentar pelo tabuleiro em movimentos horizontais e verticais, evitando obstáculos predefinidos no cenário.

11.2 A UTILIZAÇÃO DO ALGORITMO K-NN NO CONTROLE DE MOVIMENTO DO PERSONAGEM

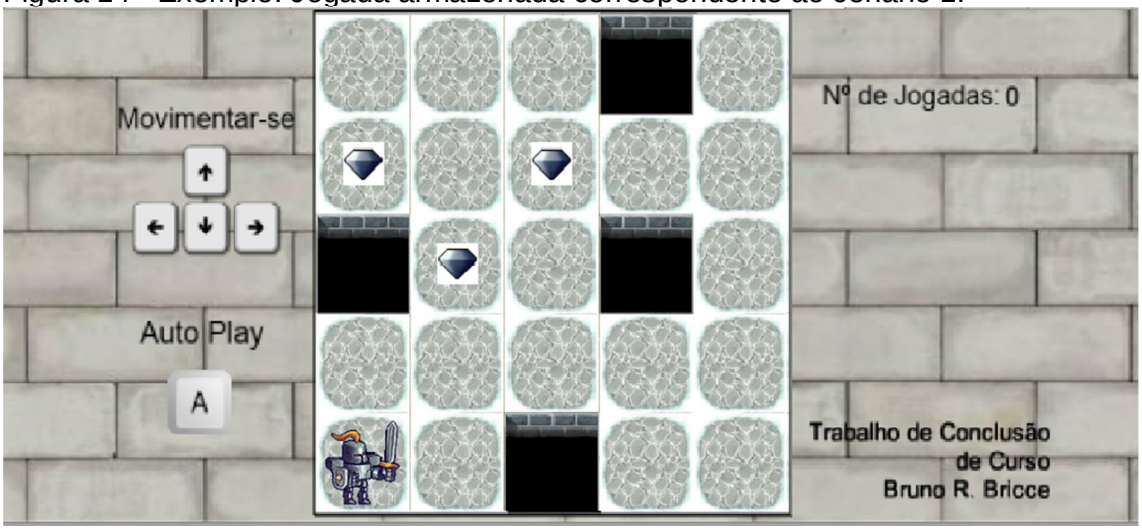
Assim como descrito no capítulo 6, segundo Fernandes (2005), o algoritmo K-NN ou Nearest Neighbour Retrieval (Vizinho mais próximo) é uma técnica simples que pode resolver um problema se baseando na sua distância com os casos existentes.

No jogo o algoritmo KNN se encaixa no Aprendizado de Máquina, pois a partir dos movimentos feitos pelo jogador humano, a máquina irá calcular a distância (similaridade) entre cenários desses para descobrir qual o melhor caminho a ser utilizado.

Detalhando com um exemplo, o jogador humano irá ter as opções de se movimentar na vertical e na horizontal apenas, cada passo que ele der será registrado, após concluir a fase, o jogador terá a opção de solicitar que o computador jogue, selecionando “Auto Play”, através da tecla “A” do teclado. Quanto mais o usuário jogar, mais a máquina irá aprender os seus movimentos em

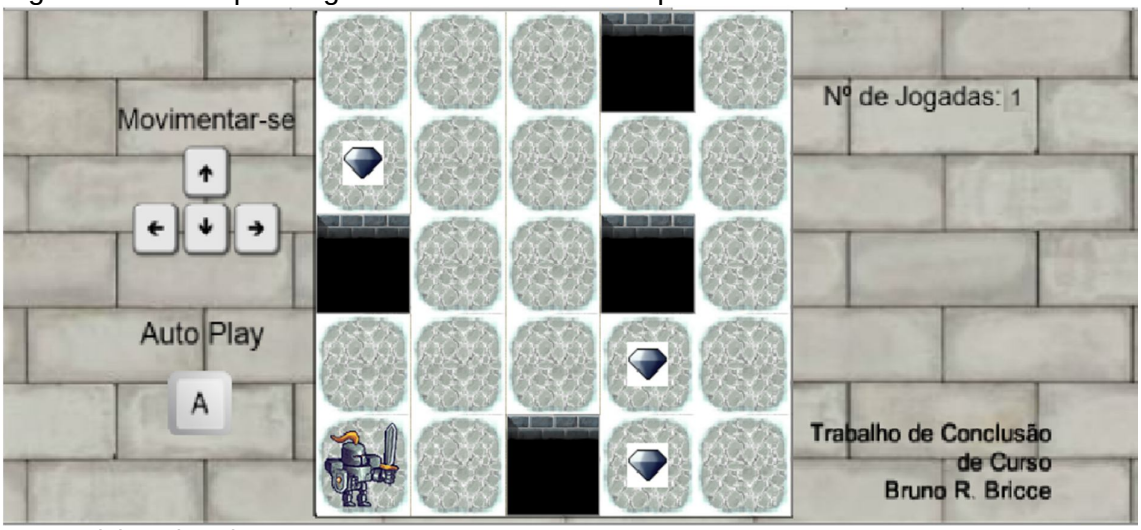
diferentes configurações de cenários, pois na opção do computador jogar, será feita uma comparação utilizando a distância euclidiana para saber qual rota é a mais adequada de acordo com a disposição dos elementos no cenário (diamantes).

Figura 14 - Exemplo: Jogada armazenada correspondente ao cenário 1.



Fonte: Elaborada pelo autor.

Figura 15 - Exemplo: Jogada armazenada correspondente ao cenário 2.



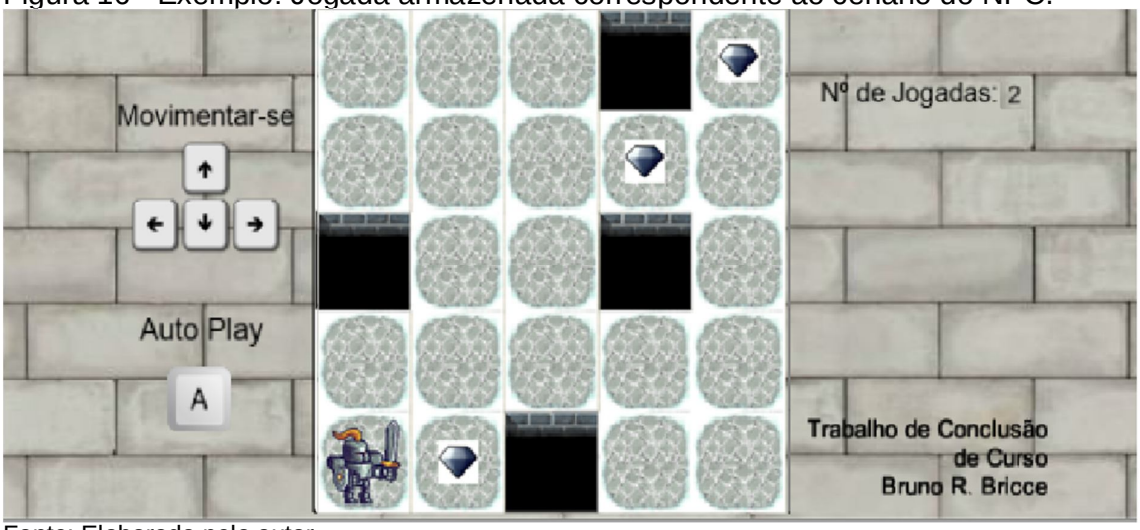
Fonte: Elaborada pelo autor.

Nos exemplos das Figuras 14 e 15 são ilustrados possíveis cenários iniciais encontrados por jogadores, cujos movimentos até a captura com sucesso de todos os diamantes foram armazenados. Cabe destacar que existirá um histórico de jogadas contendo vários cenários e rotas correspondentes armazenados. No código, estas informações vão sendo armazenadas em um Array, à medida que os

jogadores usam a aplicação. Após fechar o jogo, essas informações serão reiniciadas, fazendo com que o jogador inicie novamente a aprendizagem.

Na Figura 16 é mostrado um cenário através do qual o NPC deverá se movimentar automaticamente.

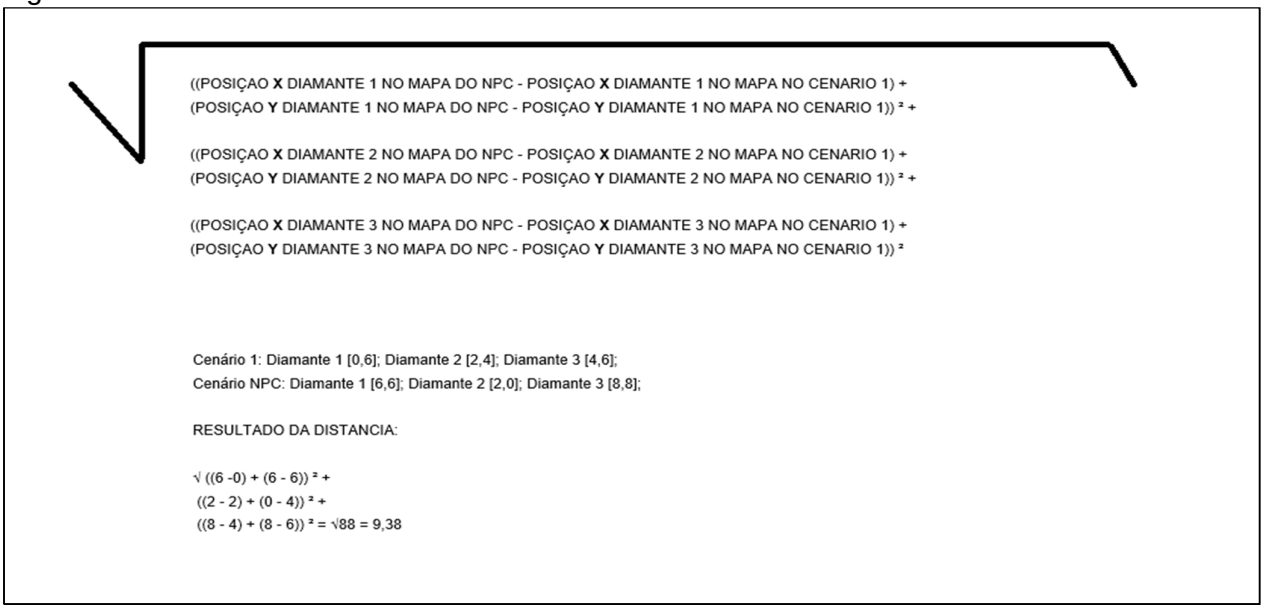
Figura 16 - Exemplo: Jogada armazenada correspondente ao cenário do NPC.



Fonte: Elaborada pelo autor.

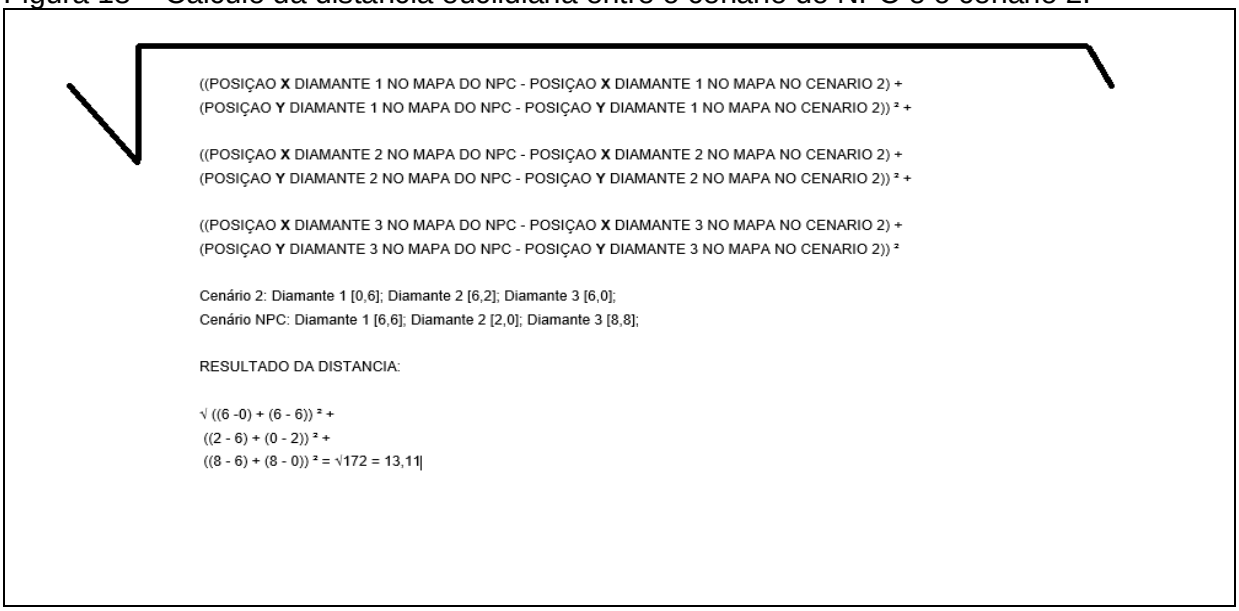
Neste caso, o algoritmo K-NN irá calcular a distância euclidiana entre o cenário atual do NPC (Figura 16) e todos os demais cenários armazenados, calculando a similaridade entre eles. A similaridade aqui adotada irá se basear na localização de cada elemento (diamantes) que compõe o cenário do NPC e os demais. Desta forma, será verificada a posição (x,y) de cada diamante nos dois cenários. Por exemplo, no cenário do NPC (Figura 16) o elemento “diamante 1” encontra-se na posição [6,6] já no cenário 1 (Figura 14) sua posição é [0,6]. A posição dos demais diamantes seria avaliada da mesma maneira e, neste caso a distância euclidiana (descrita na seção 6) calculada entre esses elementos correspondentes nos cenários seria realizada conforme Figuras 17 e 18.

Figura 17 – Cálculo da distância euclidiana entre o cenário do NPC e o cenário 1.



Fonte: Elaborada pelo autor.

Figura 18 – Cálculo da distância euclidiana entre o cenário do NPC e o cenário 2.



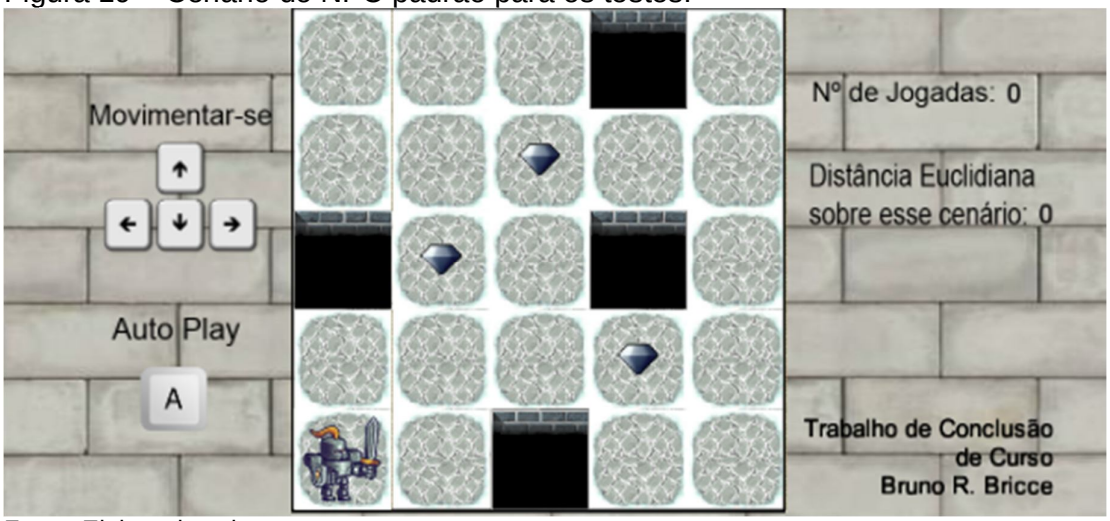
Fonte: Elaborada pelo autor.

Com base nestes resultados, e considerando apenas os dois cenários, o NPC reproduziria os movimentos do jogador correspondentes ao cenário 1, pois a distância sendo menor, caracteriza que os dois mapas tem mais elementos em posições mais similares.

12 RESULTADOS

Para avaliar o algoritmo implementado, foram realizados quatro testes. No primeiro teste foram realizadas e armazenadas 5 jogadas. O segundo teste adicionou mais 20 jogadas, totalizando 25 o terceiro contou com 75 jogadas e o último com 125 jogadas. Este aumento incremental de jogadas visou observar a influência do número de elementos armazenados no histórico na qualidade das rotas e na similaridade entre cenários. Os testes foram planejados com o intuito de verificar se houve o aprendizado do NPC durante as jogadas feitas pelo jogador humano e verificar o potencial da estratégia adotada pelo algoritmo proposto. Como base foi utilizado um cenário ilustrado na Figura 19.

Figura 19 – Cenário do NPC padrão para os testes.



Fonte: Elaborada pelo autor.

12.1 PRIMEIRO TESTE

No teste de número 1 foram realizadas 5 jogadas antes de iniciar o Auto Play. A distância gerada foi de 7,48 entre o cenário do NPC e o mapa mais similar dentre aqueles armazenados. Uma vez que o mapa mais similar tenha sido escolhido, a rota seguida pelo jogador foi, então, reproduzida no mapa do NPC, para verificar se essa sequência de movimentos permitiria atingir o objetivo do jogo (coletar todos os diamantes). Com a reprodução desta rota foram coletados dois diamantes. Pode-se concluir que o número de jogadas armazenadas não foi o suficiente para garantir a coleta de todos os diamantes.

12.2 SEGUNDO TESTE

No teste de número 2 foram realizadas 25 jogadas antes de iniciar o Auto Play do NPC. A distância euclidiana calculada foi de 6,92 entre o cenário do NPC e o mapa mais similar dentre aqueles armazenados. Uma vez que o mapa mais similar tenha sido escolhido, a rota seguida pelo jogador foi, então, reproduzida no mapa do NPC, permitindo a coleta de três diamantes. Pela quantidade de jogadas armazenadas (cinco vezes maior que no teste anterior), o algoritmo se melhor coletando todos os diamantes.

12.3 TERCEIRO TESTE

No terceiro teste foram realizadas 75 jogadas gerando a distância euclidiana (entre o cenário do NPC e aqueles armazenados) no valor de 6,0, o que representa uma melhoria aos dois testes anteriores com relação à similaridade. A rota escolhida neste teste possibilitou também a coleta de dois diamantes.

12.4 QUARTO TESTE

No quarto e último teste foram realizadas 125 jogadas (um incremento de 66% de jogadas com relação ao teste anterior), gerando a menor distância euclidiana dentre todos os testes no valor de 4,89 entre o cenário do NPC e aquele escolhido do histórico. Neste teste, o NPC atingiu totalmente seu objetivo, coletando três diamantes.

12.5 DISCUSSÃO

Após o jogador solicitar que o NPC jogasse no primeiro teste, pode se verificar que o número de jogadas não era o suficiente para gerar cenários bem similares ao do NPC, de modo a permitir que fossem coletados todos os diamantes. Mesmo assim, dois deles (67%) foram coletados.

O segundo teste, além de também ter levado à coleta de 3 diamantes (100%) indica que houve melhor desempenho pela quantidade de mapas armazenados, o que pode ser notado pela diminuição da distância euclidiana.

Os dois últimos testes indicaram melhor desempenho, pois o NPC havia uma maior quantidade de dados para verificação de cada diamante nos mapas anteriores, permitindo, inclusive, que ele recuperasse todos os elementos no último teste. Além disso, observa-se maior similaridade entre os cenários (valor de 4,89) o que nos leva a perceber que existe uma relação direta entre o número de jogadas armazenadas e o valor da similaridade, conforme pode ser visto na Tabela 1 que resume todos os testes realizados.

Analisando todos os testes, foi possível verificar que o algoritmo K-NN implementado no NPC, está condizendo com o que foi proposto, pois a medida que as quantidades de jogadas vão sendo feitas e armazenadas, as similaridades entre os mapas vão aumentando.

A distância euclidiana é calculada toda vez que o jogador pede para o NPC jogar, ativando o Auto Play, dessa maneira pode-se verificar que os valores, apesar de serem diferentes nos mapas jogados, podem ser melhorados à medida que os mapas vão sendo armazenados no histórico de jogadas.

Tabela 1 – Resumo dos testes.

Nº de jogadas	Nº de Diamantes Encontrados	Menor distancia euclidiana (similaridade)
5	2	7,48
25	3	6,92
75	2	6,00
125	3	4,89

Fonte: Elaborada pelo autor.

13 CONSIDERAÇÕES FINAIS

Com base nos resultados obtidos, é possível afirmar que a utilização do algoritmo K-NN no desenvolvimento de jogos pode impulsionar o grau de interatividade destes tipos de aplicação em um novo nível, já que esta técnica de Inteligência Artificial pode ser aplicada em um NPC, utilizando o Aprendizado de Máquina para, assim, identificar movimentos similares e executar uma ação dentro do jogo.

No geral, o projeto atingiu todos os objetivos propostos, conseguindo principalmente mostrar a utilização do K-NN no contexto de jogos e possibilitando mapear o nível de seu desempenho. Conclui-se, portanto, que esse algoritmo possui uma enorme capacidade e pode ser utilizado dentro de jogos com o intuito de simular e resolver os mais diferentes tipos de problemas, principalmente no Aprendizado de Máquina.

Para trabalhos futuros pode ser proposta a melhoria de alguns aspectos gráficos no jogo, como por exemplo, não fazer desaparecer do cenário o ícone do diamante quando existe a colisão com o jogador. Outra proposta seria aumentar o tamanho do cenário e gerar os diamantes com suas identificações ou mostrar qual está sendo coletado no momento, além de incrementar com outros tipos de obstáculos dinâmicos e utilizá-los também nos cálculos de similaridade. Novos testes também devem ser realizados, com quantidades bem maiores de dados armazenados, a fim de detectar mais claramente a influência do tamanho do histórico na qualidade dos movimentos do NPC.

REFERÊNCIAS

A História do Vídeo Game. **UOL JOGOS**, [s.d]. Disponível em: <http://jogos.uol.com.br/reportagens/historia/1961.jhtm>. Acesso em 03 de junho de 2015.

ARRUDA, G. Inteligência Artificial: o que é aprendizado de máquina? **Manual do Usuário**, 2015. Disponível em: <http://www.manualdousuario.net/inteligencia-artificial-aprendizado-de-maquina/>. Acesso em 03 de junho de 2015.

BORGES, D. de M.; BARREIRA, R. G.; SOUZA, J. G. de. Comportamento de personagens em jogos de computador. In: ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO TOCANTINS, 11., 2009, Palmas. **Anais...** Palmas: Centro Universitário Luterano de Palmas, 2009. p. 113-120. Disponível em: <http://www3.ulbra-to.br/eventos/encoinfo/2009/Anais/Comportamento_de_personagens_em_jogos_de_computador.pdf > Acesso em 05 de maior de 2015.

CLUA, E. W. G.; BITTENCOURT, J. R. **Desenvolvimento de Jogos 3D**: concepção, design e programação. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 25., 2005. São Leopoldo. **Anais...** São Leopoldo: UNISINOS, 2005. p. 1313-1356.

COELHO, P. R. P. S; **A construção de visitas virtuais: o caso do Museu de Aveiro**. 2010. 69 f. Universidade de Aveiro. Aveiro, Portugal. Disponível em: <https://ria.ua.pt/bitstream/10773/3785/1/disserta%C3%A7%C3%A3o.pdf>. Acesso em 28 de maio de 2015.

FERNANDES, A. M. da R. **Inteligência Artificial**: noções gerais. 2. ed. Visualbooks Florianopolis, 2005.

FIGUEIREDO, A. Kinect para Xbox 720 tem especificações e melhorias detalhadas por site. **TechTudo**, 2013. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2013/02/kinect-para-xbox-720-tem-especificacoes-e-melhorias-detalhadas-por-site.html>. Acesso em 13 de maio de 2015.

FUNGE, J. D. **Artificial Intelligence for Computer Games***:* an introduction. Natick: AK Peters, 2004.

GALDINO, C. H. S. **Inteligência Artificial aplicada no desenvolvimento de jogos de computadores**. 2007. 14 f. Trabalho apresentado à disciplina Metodologia de Pesquisa aplicado à Comunicação - Universidade Federal de Itajubá, 2007. Disponível em: <http://projetofinal1.googlecode.com/svn/trunk/refer%C3%AAncias/material%20bruto/carlos_galdino.pdf>. Acesso em 05 de maio de 2015.

JUNIOR, A. S. D.; NASSU, B. T.; JONACK, M. A. **Um estudos sobre os processos de desenvolvimento de jogos eletrônicos.** 2002. Universidade Federal do Paraná, Curitiba, 2002. Disponível em: <http://www.ademar.org/texts/processo-desenv-games.pdf>

KISHIMOTO, A. **Inteligência artificial em jogos eletrônicos**. 2004. 10 f. Trabalho apresentado à disciplina Inteligência Artificial - Universidade Presbiteriana Mackenzie, 2004.
Disponível em: < http://pt.slideshare.net/AndreKishimoto/inteligncia-artificial-em-jogos-eletrnicos-48108829?ref=http://kishimoto.com.br/publications.php>. Acesso em 05 de maio de 2015.

LUZ, A. R. da. **Vídeo game**: história, linguagem e expressão gráfica. São Paulo: Blucher, 2010. (Coleção pensando o design).

MACHADO, A. F. V. Uma metodologia de Aprendizagem Baseada em Classificador Numérico Aplicada a Jogos Eletrônicos. 2012. Universidade Federal Fluminense, Niterói, 2012. Disponível em: < http://www2.ic.uff.br/PosGraduacao/Teses/550.pdf>. Acesso em 03 de junho de 2015.

OSORIO. F. et al. **Inteligência artificial para jogos**: agentes especiais com permissão para matar... e raciocinar! 2007. 4 f. Trabalho de Conclusão de Curso (Graduação em Jogos Digitais) - Universidade do Vale do Rio dos Sinos, São Leopoldo, 2007. Disponível em: <http://osorio.wait4.org/publications/Osorio-et-al-SBGames07-Tutorial.pdf>. Acesso em 09 de março de 2015.

PACHECO, A. M. Tennis For Two, o primeiro game da história, completa 55 anos. **GameHall**, 2013. Disponível em: < http://gamehall.uol.com.br/v10/tennis-for-two-o-primeiro-game-da-historia-completa-55-anos/>. Acesso em 03 de junho de 2015.

PAULA, B. C. de; KOERICH, A. L; ENEMBRECK, F**. Agente Inteligente para Jogos de Estratégia Baseados em Turnos**: Uma Abordagem Utilizando Planejamento Baseado em Casos, 2006. Pontifícia Universidade Católica do Paraná – PUCPR. Recife, 2006. Disponível em: < http://www.ppgia.pucpr.br/~alekoe/Papers/ALEKOE-AgenteInteligente-SBGames2006.pdf>. Acesso em 03 de junho de 2015.

RUSSEL, S.; NORVIG, P. **Inteligência artificial**. Tradução: Stuart Russel, Peter Norvig. Rio de Janeiro: Elsevier, 2011.

SANTAELLA, L.; FEITOZA M. **Mapa do Jogo**: a diversidade cultural dos games. São Paulo: Cengage Learning, 2009.