UNIVERSIDADE FEDERAL DE SANTA CATARINA

CAMPUS ARARANGUÁ

Mauricio Teixeira Pereira

Análise de Técnicas de Inteligência Artificial Aplicadas em Jogos Computacionais

Araranguá, dezembro de 2013.

Mauricio Teixeira Pereira

Análise de Técnicas de Inteligência Artificial aplicadas em Jogos Computacionais.

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina, como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Tecnologias da Informação e Comunicação.

Orientadora Profª. Dra. Eliane Pozzebon

Araranguá, dezembro de 2013.

Mauricio Teixeira Pereira

Análise de técnicas de Inteligência artificial aplicadas em jogos computacionais.

Este trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de Bacharel em Tecnologias da Informação e Comunicação, e aprovado em sua forma final pelo Curso de Graduação em Tecnologias da Informação e Comunicação.

Araranguá, dezembro de 2013.

Prof. Vilson Gruber, Dr.
Coordenador do Curso

Banca Examinadora:

Prof.ª Eliane Pozzebon, Dr.ª
Orientadora
Universidade Federal de Santa Catarina

Prof. Anderson Luiz Fernandes Perez Dr.
Universidade Federal de Santa Catarina

Profª Luciana Bolan Frigo, Drª.
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Primeiramente a Deus, por me possibilitar a estudar em uma Universidade Federal;

Minha família, que sempre me desejou boa sorte e apoio para fazer o curso;

A professora Eliane que sempre esteve presente e disposta e fez o possível para contribuir com o trabalho;

Meus colegas de classe, os professores e as demais pessoas que contribuíram para o processo de ensino.

RESUMO

O presente trabalho abrange a área de jogos computacionais e inteligência artificial (IA). O objetivo deste trabalho foi analisar algumas técnicas de Inteligência Artificial utilizadas no desenvolvimento de jogos, identificar as características destas técnicas e propor qual a técnica mais adequada para um jogo específico. Foram analisadas as técnicas de Redes Neurais, Lógica *Fuzzy* e Sistemas Multiagentes. O estudo de caso escolhido foi um jogo de estratégias táticas denominado *Astorian Tears* que está sendo desenvolvido no Laboratório de Tecnologias Computacionais da Universidade Federal de Santa Catarina. O jogo *Astorian Tears* acontece num cenário de fantasia-medieval, no qual o jogador controla as ações de um personagem – avatar -, que é o herói do jogo. O jogador deverá guiá-lo pelo mundo em busca de sua vingança pessoal contra um ditador cruel. Como resultado deste estudo de caso, foram identificadas as características do jogo e sugerida a utilização dos agentes cognitivos para tornar o jogo mais dinâmico e atrativo.

Palavras-chave: Inteligência Artificial, Jogos Computacionais, Estratégias Táticas.

ABSTRACT

This paper covers the area of computer games and artificial intelligence (AI). The objective of this study was to analyze some techniques of Artificial Intelligence used in game development, identify the characteristics of these techniques and propose the most appropriate technique for a specific game. The techniques of Neural Networks, Fuzzy Logic and Multiagent Systems were analyzed. The case study selected was a game called *Astorian Tears* tactical strategies being developed in the Laboratory of Computational Technologies, Federal University of Santa Catarina. The game *Astorian Tears* happens in a fantasy - medieval setting, in which the player controls the actions of a character - avatar - who is the hero of the game. The player must guide him through the world in search of his personal vendetta against a evil dictator. As a result of this case study, the game features were identified and suggested the use of cognitive agents to make the game more dynamic and attractive.

Keywords: Artificial Intelligence, Computer Games, Strategies Tactics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABRAGAMES - Associação Brasileira de Desenvolvedores de Jogos Digitais

IA - Inteligência Artificial

JADE - Java Agent Development Framework

LabTeC - Laboratório de Tecnologias Computacionais

RNAs - Redes Neurais Artificiais

RPG – Role Playing Game

SEBRAE - Serviço Brasileiro de Apoio às Micro e Pequenas Empresa

SMA - Sistemas Multiagentes

Web - Word Wide Web

SUMÁRIO

1 INTRODUÇÃO

Denomina-se a espécie *Homo sapiens* – homem sábio – porque as capacidades mentais dos humanos são importantes para todos. Durante milhares de anos, buscou-se entender como os humanos pensam; isto é, como um "mero punhado de matéria" possa perceber, compreender, prever e manipular um mundo muito maior e mais complicado que ela própria. O campo da Inteligência Artificial (IA) vai ainda mais além: Não tenta apenas compreender, mas também construir entidades inteligentes. (RUSSEL; NORVIG, 2004).

A inteligência artificial é uma das ciências que teve início após a segunda guerra mundial e, atualmente, abrange uma enorme variedade de áreas de uso geral até tarefas específicas, por exemplo, jogos de xadrez, demonstração de teoremas matemáticos, diagnóstico de doenças, etc. (RUSSEL; NORVIG, 2004).

Uma definição mais abrangente, segundo Gomes (2010) é que a IA sistematiza e automatiza tarefas intelectuais e, portanto, é potencialmente relevante para qualquer esfera da atividade intelectual humana (GOMES, 2010).

A partir dos conceitos de Gomes (2010), é possível constatar que os jogos computacionais estão incluídos nesta "esfera da atividade intelectual humana" e que para serem desenvolvidos apresentam certo grau de complexidade, incertezas durante o processo e tomadas de decisões. Para auxiliar no desenvolvimento destes jogos computacionais foram criadas técnicas de Inteligência Artificial para facilitar na

implementação e automatização das tarefas intelectuais exigidas nos jogos.

Como os jogos computacionais são produtos voltados ao entretenimento, os desenvolvedores devem inovar constantemente na elaboração dos jogos para que estes se tornem mais criativos e com ambientes cada vez mais realistas (PASSOS, 2012).

Este trabalho abrange a área de jogos computacionais, principalmente para auxiliar os desenvolvedores na escolha da técnica de IA mais adequada para um determinado jogo.

1.1 OBJETIVOS

Os objetivos deste trabalho são divididos em objetivo geral e objetivos específicos.

1.1.1 Objetivo Geral

Analisar as técnicas de Inteligência Artificial em jogos computacionais, identificar características destas técnicas e propor qual a técnica mais adequada para um jogo específico.

1.1.2 Objetivos Específicos

- Estudar as técnicas de IA aplicadas a jogos computacionais;

- Pesquisar jogos computacionais que utilizam técnicas de IA;
- Identificar características de cada técnica de IA estudada;
- Elaborar uma proposta de aplicação de uma técnica de IA para um jogo específico.

1.2 Problemática

Como escolher uma técnica de IA adequada para um determinado jogo?

Rabin (2012) afirma que o problema do programador de IA para jogos, é criar oponentes divertidos e desafiadores para o entretenimento dos jogadores. Os personagens de um jogo interagem com o ambiente e/ou com adversários. Estas interações dependem de decisões tomadas durante o jogo que são implementadas utilizando técnicas de Inteligência Artificial. Além disso, as técnicas de IA exigem um tempo considerável de desenvolvimento, e os testes devem ser realizados no início do ciclo de desenvolvimento do jogo.

Algumas implicações que dificultam o processo de aplicação da IA num jogo são diagnosticadas por Rabin (2012):

1. A IA deve ser inteligente, porém com falhas intencionais;
2. A IA não deve ter falhas não pretendidas;
3. O desempenho da IA deve estar de acordo com as limitações da CPU e da memória do jogo;

4. A IA deve ser configurável pelos designers/desenvolvedores do jogo;
5. A IA não deve impedir o lançamento do jogo.

Além das cinco implicações que Rabin (2012) cita, no desenvolvimento de um jogo computacional são encontradas inúmeras dificuldades para definir qual a técnica mais adequada ao jogo. Segundo Bataiolla (2000) há grandes dificuldades na implementação da técnica de IA. Uma delas, por exemplo, é fazer a sincronização entre alguns itens do jogo como, por exemplo, a fala e o movimento da boca de um personagem.

Com base nestes argumentos, neste trabalho será desenvolvida uma análise de algumas técnicas de IA para auxiliar o desenvolvedor na tomada de decisão. Além disso, serão identificadas características de cada técnica e apresentada uma proposta de aplicação de uma técnica de IA a um jogo específico.

1.3 Justificativa

A evolução do mercado na área de jogos computacionais é constante e o Brasil, segundo o SEBRAE (2013), já é o quarto maior mercado de games do mundo com cerca de 35 milhões de usuários. Como base, o mercado nacional de games movimentou R$ 5,3 bilhões em 2012, obtendo um crescimento de 32% em relação a 2011.

Paralelo a esta evolução do mercado, aumenta a utilização de técnicas de inteligência artificial embutida nesta classe de software,

provendo realidade e diversão aos jogos disponíveis no mercado. Neste contexto, há uma forte influência dos jogos sobre o desenvolvimento das técnicas de inteligência artificial, a qual será tratada neste trabalho.

A utilização de inteligência artificial em jogos computacionais agrega reais melhorias a estes softwares, conforme Souza (2011) é possível aumentar a experiência e imersão do jogo, melhorando sua jogabilidade. De maneira similar, os jogos exigem uma inteligência elaborada em seus personagens, enredo e cenários. Com isso, surgem alguns problemas a serem resolvidos, o que acaba por desenvolver as áreas de estudo da inteligência artificial.

Em outras palavras, a IA fornece benefícios aos jogos, ao passo que estes fazem com que o campo cresça e se desenvolva. Algumas características consideradas relevantes na implementação da IA, segundo Souza (2011), são:

a. Oponentes devem de alguma forma, propor um desafio ao jogador;
b. Oponentes devem manter o jogo divertido;
c. Oponentes devem perder para o jogador de maneira divertida e desafiadora;
d. Não devem existir caminhos iguais para derrotar a IA da mesma forma que já foi derrotado antes;
e. A IA não deve falhar vergonhosamente ou parecer burra.
f. A maioria dos jogos é em tempo real e por isso deve ter IA que reage em tempo real;

g. A IA do jogo às vezes recebe de 10% a 20% do tempo de *frame;*

h. Designers devem ser capazes de ajustar o nível de dificuldade, configurar a IA e ocasionalmente as interações específicas de *script;*

i. Se o jogo é extensível, jogadores podem modificar ou customizar a IA;

j. As técnicas de IA empregadas não devem por o jogo em risco;

k. Técnicas experimentais devem ser testadas com antecedência no ciclo de desenvolvimento durante a pré produção;

l. Se o alcance da IA evoluir ou mudar nas atualizações, ela deve ser testada antes, a fim de garantir que não se deteriore e produza erros quando lançada para milhões de consumidores.

Com base nestas características relatadas, este trabalho propõe realizar a análise de algumas técnicas de IA para auxiliar os desenvolvedores na escolha da técnica de IA mais adequada para um jogo específico.

1.4 Metodologia

Esta pesquisa pode ser classificada como explorativa e aplicada, uma vez que conhecimentos são gerados e aplicados na prática, visando à solução de determinada problemática. Como parte inicial do projeto, pode-se citar a caracterização do problema de pesquisa. Este remete à aplicação de a inteligência artificial a jogos computacionais. Desta forma pode-se destacar algumas técnicas de IA que são aplicadas em cada jogo, conforme a realidade do mesmo.

Após esta etapa inicial, serão realizadas pesquisas para reunir um referencial bibliográfico, possibilitando um embasamento maior sobre os temas que serão abordados. Com este material, serão estudados mais a fundo as técnicas de IA selecionadas, bem como suas propriedades e capacidades, além do jogo que estas serão aplicadas. Para o embasamento teórico do trabalho, foram utilizadas várias bibliografias que apresentam às definições na área de Inteligência Artificial.

Com a fundamentação necessária extraída, é dado início à aplicação da técnica de IA a um jogo específico.

1.5 Organização do documento

Este documento está organizado em 5 capítulos. O primeiro capítulo apresenta uma introdução do trabalho, objetivos, problemática e metodologia. No segundo capítulo é apresentada uma introdução sobre os jogos computacionais, ferramentas utilizadas para o desenvolvimento e as metodologias utilizadas para este fim. No terceiro capítulo serão

abordadas três técnicas de IA: Redes Neurais Artificiais (RNAs), Lógica *Fuzzy* e Sistemas Multiagentes. Neste capítulo também são apresentados exemplos de jogos que utilizam estas técnicas. No quarto capítulo será apresentado o estudo de caso do jogo de estratégias táticas *Astorian Tears,* onde será sugerida uma técnica de IA para aplicação no jogo estudado. No quinto e último capítulo, serão apresentados os resultados oriundos deste trabalho.

2. JOGOS COMPUTACIONAIS

2.1 Introdução

O desenvolvimento de jogos computacionais difere do desenvolvimento de software tradicional em diversos aspectos. Segundo Passos (2012), para o desenvolvimento de software tradicional, quase 100% dos desenvolvedores é da área técnica. Para o desenvolvimento de jogos computacionais, o número gira em torno de 33% de pessoas nessa área e 67% são pessoas que trabalham com conteúdo. Este conteúdo inclui desenho de paisagens (níveis), modelagem, animação, sendo que essas etapas constituem a parte artística. Essas pessoas podem ter formação em diversas áreas como artes plásticas, arquitetura ou informática.

De acordo com Grubba (1997) *apud* Pereira (2006), os jogos podem ser divididos em dois grupos principais relacionados aos tipos e gêneros. O primeiro grupo relacionado à capacidade de simulação que o jogo promove e podem ser de ação, aventura, simulação, educativo, estratégia, entre outros; e o segundo relacionado com a tecnologia (arquitetura) envolvida e os jogos podem ser de quatro plataformas: arcade, console, computadorizada ou móvel.

Baseado nessa grande quantidade de gêneros e tipos a Associação Brasileira das Desenvolvedoras de Jogos Eletrônicos - ABRAGAMES (2008), afirma que a indústria brasileira de desenvolvimento de jogos vem crescendo no mercado local e também para exportação e, devido a chegada de estúdios internacionais, deve-se expandir ainda mais no país.

Apesar da pirataria, a indústria tem se destacado. Segundo a pesquisa, a indústria de jogos brasileira obtém por meio da exportação uma significativa parte do mercado bilionário. Somando-se hardware e software, o produto nacional bruto é de R$ 87,5 milhões. A exportação nacional de software é de 43%, enquanto que quase 100% do hardware fabricado se destinam ao mercado interno.

Porém há falta de profissionais para tanta procura. A maioria das vagas é para programadores de jogos para redes sociais e plataformas móveis como *smartphones, tablets*, etc. A caça a novos talentos é feita nas Universidades, e as empresas também tem procurado capacitar seus colaboradores com cursos específicos dentro da própria empresa. Para que o Brasil cresça mais na área de desenvolvimento de jogos, é necessário políticas de incentivo a desenvolvimento de softwares e novos talentos (ABRAGAMES, 2013).

Pode-se constatar que 80% dos empregados possuem ensino superior, mas a principal exigência é que o profissional possua experiência. Para isso a ABRAGAMES (2013) recomenda alguns cursos como: design de jogos, engenharia da computação, etc.

Como exemplo referente a aplicação de jogos, foi desenvolvido um jogo educacional para a capacitação de empresas do setor elétrico em São Paulo para quantificar riscos e possíveis erros que podem ser cometidos, evitando que estes possam sofrer lesões graves. Outro ponto importante, é que a empresa aplicava um treinamento de oito horas aos funcionários; e com este jogo educacional, o mesmo treinamento é realizado em apenas quatro horas; ou seja, a metade do tempo. (ABRAGAMES, 2013)

2.2 Metodologias para desenvolver Jogos Computacionais

Uma metodologia pode ser definida como uma abordagem organizada para atingir um objetivo, através de passos e procedimentos pré-estabelecidos além de um roteiro, que possui um processo dinâmico e interativo para o desenvolvimento estruturado de projetos, sistemas ou software, visando a qualidade e produtividade do mesmo (PEREIRA, 2006).

Existem diversas metodologias que são utilizadas para o desenvolvimento de jogos. Algumas delas são: *Scrum* [Lima, 2009], *OriGame* [Santos; Góes; Almeida, 2012], Maiêutica [Silva, Hounsell, Kemczinski, 2007], *Extreme Programming* [Teles, 2013], etc.

Nas seções 2.2.1 e 2.2.2 serão apresentadas duas metodologias denominadas *Scrum* e *OriGame,* respectivamente.

2.2.1 Metodologia *Scrum*

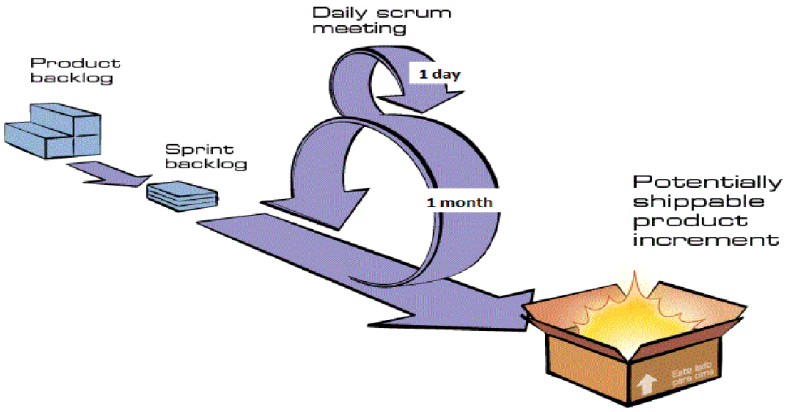
Uma das metodologias mais utilizadas para o desenvolvimento de jogos é o *Scrum,* que segundo Rodrigues e Rost (2007) concluem que o *Scrum* uma metodologia de desenvolvimento ágil, onde há um grande foco no trabalho em equipe; que são auto-gerenciadas e há uma participação ativa do cliente. Uma figura importante é o *Scrum master*, que tem como objetivo e eliminar obstáculos fazendo com que a equipe tenha um melhor desempenho.

Rodrigues e Rost (2007) também afirmam que as metodologias ágeis de desenvolvimento permitem que as mudanças durante o

desenvolvimento do jogo possam ser feitas rapidamente, reduzindo o impacto das mudanças nos projetos e permite que mudanças tardias nos requisitos ou mesmo no escopo do projeto não sejam tão prejudiciais. As entregas do jogo podem ser feitas constantemente em versões preliminares que podem ser entregues ao cliente, que, por sua vez, fica mais satisfeito por que não há necessidade de esperar que o jogo seja totalmente desenvolvido para depois ser entregue. Como o jogo é entregue aos poucos, é possível que o cliente sugira algumas modificações que possam ser entregues na próxima versão e assim sucessivamente, até que o jogo fique completamente pronto (RODRIGUES; ROST, 2007).

Figura 1 - Visão geral da metodologia *Scrum*

Fonte: Adaptada de Lima (2009)



Gomes *et. Al* (2011) afirmam que nesse modelo de desenvolvimento de jogos as empresas tem obtido vários benefícios como a redução de custos e aumento significativo de produtividade pois acrescenta foco, comunicação, clareza e transparência para desenvolvimento de software.

Gomes et al. (2011) afirmam que o *Scrum* possui um conjunto formado por tempo de *Scrum* além de seus papéis associados, *Time-Boxes* (eventos com duração fixa), Artefatos e Regras.

Os Times de *Scrum* são auto-gerenciáveis, e vizam melhorar e flexibilizar a produtividade. Cada Time de Scrum possui três papeis:

1) *Scrum Master*: É um líder que facilita o processo e faz com que ele seja compreendido e acompanhado;

2) *Product Owner*: É responsável por apresentado trabalho feito pelo Time de Scrum, serve como ponte entre o cliente e o fornecedor;

3) *Time*: Tem a função de executar as *tasks* que resultarão em uma *release*.

A rotina de *Scrum* começa com a lista dos requisitos do projeto, que são ordenados por prioridade (*product backlog*). A partir desta lista são formados os requisitos que serão implementados na próxima iteração de desenvolvimento (*sprint backlog*); cada *sprint* dura cerca de 30 dias (dependendo o projeto) e, após seu final, as funcionalidades desenvolvidas são validadas pelo cliente, normalmente (*product owner*) e liberadas, iniciando-se um novo ciclo. (RODRIGUES; ROST, 2007).

Ao final de cada *Sprint* tem-se uma reunião para que seja feita uma revisão (*Sprint Review Meeting ou Daily Scrum meeting*) inde a equipe apresente o resultado alcançado na interação ao *Product Owner*, ocorrendo a inspeção e adaptação das funcionalidades e do projeto. Em seguida, *Scrum Master* conduz a reunião de retrospectiva (*Sprint Retrospective Meeting*), onde os participantes relatam quais foram os impedimentos apresentados durante o *sprint*, isto com o objetivo de

melhorar o processo e/ou produto para a próxima *sprint* até a conclusão do projeto do jogo computacional. (RODRIGUES; ROST, 2007).

O *Scrum* tem algumas características que devem ser analisadas antes de aplicá-lo ao desenvolvimento de um jogo computacional. Algumas pontos importantes são destacados por Gomes *et. Al* (2011):

- O *Scrum* é aplicado para equipes pequenas de, no máximo, sete pessoas;
- Requisitos devem ser conhecidos e estáveis;
- Deve possuir pequenas iterações.

O *Scrum* pode ser utilizado sempre que um grupo de pessoas precise trabalhar em conjunto para atingir um objetivo comum, desde o gerenciamento de projetos de software até tarefas do cotidiano, como organizar uma festa.

2.2.2 Metodologia *OriGame*

De acordo com Santos, Góes e Almeida (2012) a grande diversidade dos jogos existentes e as suas características particulares encontradas nestes jogos de acordo com seus propósitos, a adoção de uma metodologia de desenvolvimento rígida para este tipo de software torna-se inviável.

Isso ocorre por que os jogos nem sempre possuem características em comum. Um jogo *Role-playing Games* (RPG) deve possuir uma

documentação totalmente detalhada onde mostre a história do jogo e a descrição de cada personagem e como estes participam da trama. Já em jogos *First Player Shooter* (FPS) a história pode ser supérflua ou inexistente (SANTOS; GÓES; ALMEIDA; 2012).

Além da categoria a qual o jogo se enquadra (RPG, FPS...) possui outros fatores que também devem ser levados em consideração como a plataforma a qual se propõe; como por exemplo, *web* ou *mobile*; ambos não podem ter a mesma característica áudio-visual devido as limitações de hardware e questões empíricas (SANTOS; GÓES; ALMEIDA; 2012).

A partir disto Santos, Góes e Almeida (2012) proporam a metodologia *OriGame* que utiliza um diagrama que relaciona as etapas existentes na produção de jogos com uma visão geral do processo. A metodologia, no momento, é apenas umas proposta e abaixo segue o detalhamento.

O jogo tem início na concepção da ideia. Após definir, é necessário documentar e avaliar as ideias para verificar a viabilidade de fazer o projeto. Após ter a documentação pronta é necessário desenvolver os detalhes do jogo como o áudio e vídeo integrando as características artísticas ao padrões computacionais para formar o jogo computacional. (SANTOS; GÓES; ALMEIDA; 2012).

A metodologia *OriGame*, surgiu a partir do pressuposto que um jogo começa a partir da concepção de uma ideia e isto se torna o início do processo. Visando a qualidade e o planejamento correto do jogo, a documentação deve ser elaborada apenas após a concepção. A Figura 2 ilustra uma visão geral da metodologia *OriGame* (SANTOS; GÓES; ALMEIDA; 2012).

Reforçando os conceitos onde Santos, Góes e Almeida (2012) afirmam que é necessário primeiro realizar a concepção dos jogos, os mesmo autores explicam sobre o público alvo dos jogos. Por exemplo, se o jogo tem como público alvo os deficientes auditivos, o foco do desenvolvimento será em recursos visuais; e caso o público alvo seja deficiente visual, o foco do jogo será o desenvolvimento de recursos auditivos; salvo exceções.

Outro ponto interessante citado por Santos, Góes e Almeida (2012) referente à produção de áudio e vídeo. A produção de áudio é simples: Basta determinar as características desejadas e produzir ou adquiri-los. A produção do vídeo já tem algumas diferenças básicas como a utilização de recursos 2D e 3D; que podem ser utilizados em conjunto. Além disso, é necessário fazer a integração entre o áudio e o vídeo que é feita na implementação do jogo. Com todo material de áudio e vídeo produzidos, pode ser realizada a produção do jogo; salvo questões administrativas.

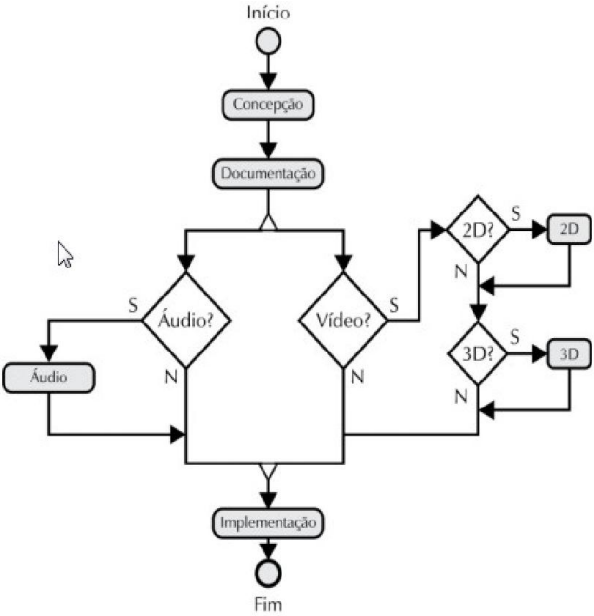
Baseados nestes argumentos, Santos, Góes e Almeida (2012) destacam as três fases necessárias para aplicação da metodologia *OriGame*.

1. Design e projeto: analisados os requisitos do jogo e definidas as características do mesmo;
2. Produção: pautada na criação dos recursos necessários (áudio e vídeo, por exemplo) e aprimoramento da estética;

3. Implementação: União do que foi desenvolvido, codificação e testes.

Figura 2 - Visão geral da metodologia *OriGame*

Fonte: Extraído de Santos, Góes e Almeida (2012)



Cada uma das três fases citadas anteriormente são expandidas em diversos processos e sub-processos, com base em metodologias de desenvolvimento ágil e prototipação. Embora distintas, as três etapas apresentadas não precisam ser sequenciais (modelo cascata), ou seja, não é necessário que toda documentação seja concluída para começar a produção, ou toda a produção para começar a implementação (SANTOS; GÓES; ALMEIDA; 2012).

O processo de desenvolvimento do jogo é bem dinâmico, e mudanças de escopo, recursos e tecnologias podem acarretar em atualizações na documentação, mudanças na produção e refatoramento de códigos; o que se torna mais caro quando estiver mais próximo das etapas finais do desenvolvimento (SANTOS; GÓES; ALMEIDA; 2012).

2.3 Ferramentas para desenvolver Jogos Computacionais

Segundo NEMES (2012), há cada vez mais ferramentas para desenvolvimento de jogos no mercado e o desenvolvimento e distribuição de *games* não são mais apenas das grandes empresas: Qualquer pessoa pode se empenhar em um projeto de jogo.

Seguindo esta ideia, o autor citou quatro ferramentas livres muito utilizadas para o desenvolvimento de jogos eletrônicos. São elas:

- Unity Engine: Grande parte dos jogos em 3D para o navegador é desenvolvida com esta *engine*, que é bastante conhecida dos fãs de games. Além disso, existe um modo totalmente gratuito que possui apenas algumas limitações de gráficos e licença. Uma das grandes vantagens citadas pela autora, é que não é necessário que o desenvolvedor tenha um amplo conhecimento de programação, mas que este é essencial para saber o que acontece atrás dos cenários. (NEMES, 2012).
- Unreal Development Kit (UDK): Este software é utilizado para a criação de games, inclusive o desenvolvimento dos jogos da Epic Games, são desenvolvidos utilizando uma versão deste

software. É uma versão para quem está iniciando no desenvolvimento de jogos eletrônicos e é uma "introdução ao" Unreal Engine 3, ferramenta profissional utilizada para criar os jogos da Epic. Para utilizar este *kit* também é preciso ter uma máquina com grande capacidade de processamento, pois esta ferramenta gráfica é bem completa, podendo criar games extraordinários. O UDK é recomendado para quem já possui um bom conhecimento de criação de games, além de ser necessário ter um bom conhecimento de inglês, pois o aplicativo não possui versão em português (NEMES, 2012).

- Game Maker: É uma ferramenta mais simples, para a criação de jogos que não precisam de muito processamento gráfico (geralmente são 2D), e permite o desenvolvimento sem precisar saber programar (NEMES, 2012).
- KODU Game Lab: É o software mais simples para quem está começando a desenvolver jogos e precisa de uma ajuda adicional. Possui uma maneira simples para indicar comandos (when) e o que será feito após o comando ter sido executado. Por exemplo: Caso você solicite que o personagem se mova um quadro para a direta sempre que a tecla "D" acionada. Para realizar esta tarefa, no campo "When" é necessário colocar o atalho "D" e no campo "Do" é necessário indicar a ação pretendida; ou seja, mover o personagem para o quadro à frente. Esta forma de "programar", facilita muito para quem não conhece sobre programação e possibilita que os iniciantes desenvolvam um simples jogo eletrônico sem grandes dificuldades (NEMES, 2012).

2.4 Considerações do capítulo

Neste capítulo foi feita uma breve introdução aos jogos computacionais, o mercado que vem se expandindo no país, os tipos, os gêneros, as metodologias e algumas ferramentas de desenvolvimento de jogos computacionais.

Procurou-se abordar duas metodologias distintas onde a metodologia *Scrum* é mais consagrada devido aos seus procedimentos adotados para desenvolvimento de quaisquer tipos de software e a metodologia *OriGame* que foi proposta por Santos, Góes e Almeida (2012) que apesar de ser bem elaborada, ainda precisa ser implementada em um jogo computacional.

Quanto às ferramentas para desenvolvimento dos jogos computacionais foram abordadas algumas ferramentas que podem ser utilizadas para este fim. Podemos citar como exemplo de utilização, a UDK e Kodu. A primeira é utilizada mais profissionalmente enquanto a segunda é uma ferramenta mais simples e limitada.

Vale salientar que cada ferramenta pode ter características semelhantes uma a outra; cabendo ao desenvolvedor escolher a ferramenta de sua preferência.

3. TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL APLICADA A JOGOS COMPUTACIONAIS

3.1 Introdução sobre inteligência artificial

A definição deste termo é motivo de discussão entre vários autores da área. Algumas definições de Inteligência Artificial são:

- IA é o estudo de como fazer os computadores realizarem tarefas as quais, até o momento, os homens fazem melhor (Rich, 1994).
- O estudo de computações que tornem possível perceber, raciocinar e agir (Winston, 1992).

É possível perceber que as definições remetem a fazer com que o computador possa "sentir o ambiente" e possa tomar decisões por si só.

Como o computador tem a capacidade de resolver problemas muito mais rápido, (cálculos extremamente complexos, por exemplo) um ser humano levaria um bom tempo para resolver, por outro lado o computador, sem que haja um mecanismo, não consegue distinguir entre uma mesa de madeira e uma mesa de metal; coisa que uma criança de três anos é capaz de fazer. (Rosa, 2011).

Nos jogos computacionais é necessário inserir uma série de artifícios que possam tornar o jogo atrativo e sempre traga surpresas aos jogadores; proporcionando mais diversão (BOSCATTO; NAKAMITI; TOLEDO, 2012).

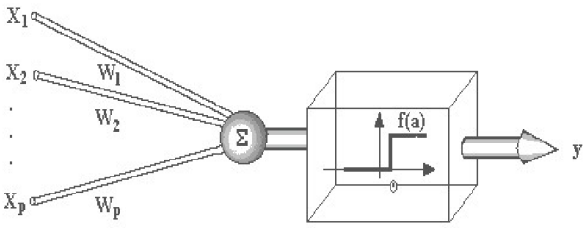
Neste trabalho foram escolhidas três técnicas de IA para serem analisadas. São elas: Redes Neurais Artificiais (RNAs), Lógica *Fuzzy* e Sistema Multiagentes (SMA) que serão detalhas nas próximas seções.

3.2 Redes Neurais Artificiais

O final da década de 1980 marcou o ressurgimento da área de Redes Neurais Artificiais (RNAs), que tiveram início em 1943 quando McCulloch e Pitts desenvolveram um modelo matemático simples de um neurônio. De 1943 em diante, foram desenvolvidos modelos muito mais detalhados e realistas, tanto para neurônios quanto para sistemas maiores no cérebro, levando ao moderno campo da neurociência computacional. Além disso, pesquisadores em IA e estatística ficaram interessados nas propriedades mais abstratas de RNAs, como sua habilidade para executar computação distribuída, evitar entradas prejudiciais e aprender. Embora possuam outros tipos de sistemas semelhantes (redes bayesianas, por exemplo) as RNAs são uma das formas mais populares e efetivas de sistemas de aprendizagem (RUSSEL; NORVIG, 2004).

Figura 3 - Neurônio Artificial McCulloch

Fonte: Extraído de Tatibana e Kaetsu (2013)



Conhecida também por *sistemas de processamento paralelo ou distribuído* e *conexionismo,* estas estruturas relembram, de certa forma, o cérebro-humano. Por não ser baseada em regras, a computação neural se constitui em uma alternativa à computação algorítmica convencional (BRAGA et al*.,* 2007).

A Tabela 1 faz um comparativo entre o cérebro humano e o computador.

Tabela 1 - Comparação Cérebro Humano e Computador

Fonte: Extraído de Tatibana e Kaetsu (2013)

Parâmetro	Cérebro	Computador
Material	Orgânico	Metal e plástico
Velocidade	Milisegundos	Nanosegundos
Tipo de Processamento	Paralelo	Seqüencial
Armazenamento	Adaptativo	Estático
Controle de Processos	Distribuído	Centralizado
Número de elementos processados	10 e 11 à 10 e 14	10 e 5 à 10 e 6
Ligações entre elementos processados	10.000	<10

As RNAs são sistemas paralelos distribuídos compostos por unidades de processamento simples (neurônios artificiais) que calculam determinadas funções matemáticas. Tais unidades estão distribuídas em uma ou mais camadas e interligadas por várias conexões, geralmente unidirecionais. O que determina essas conexões são pesos, os quais armazenam o "conhecimento" adquirido e servem para ponderar a entrada recebida por cada neurônio da rede.

Devido à forma como as RNAs são representadas internamente é obtido um desempenho superior ao dos modelos de técnicas de IA convencionais. Em RNAs, o processo de resolução dos problemas passa inicialmente por uma fase de *aprendizagem,* em que um conjunto de exemplos é fornecido para a rede que por sua vez extrai as características necessárias para representar a informação fornecida. Essas características são utilizadas posteriormente para gerar respostas ao problema (BRAGA et al*.,* 2007).

A capacidade de *aprender* por meio de exemplos e de *generalizar* a informação aprendida é, sem dúvida, o atrativo principal da solução de problemas utilizando RNAs. A capacidade da rede aprender por meio de um conjunto de exemplos e posteriormente dar respostas coerentes para dados não conhecidos, demonstra que a capacidade das RNAs vai muito além do que simplesmente mapear relações de entrada e saída.

As RNAs são capazes de extrair informações não apresentadas de forma explícita através de exemplos e a capacidade de auto-organização e de processamento temporal que, aliada àquelas citadas anteriormente, faz das RNAs uma ferramenta computacional muito atrativa para resolução de problemas complexos (BRAGA et al*.,* 2007).

Russel e Norvig (2004) afirmam que existem duas principais estruturas de RNAs que possibilitam o aprendizado. São elas: Redes de alimentação direta ou redes recorrentes.

- Redes de alimentação direta – Representa uma função de sua entrada atual; desse modo, ela não possui nenhum estado interno além dos pesos propriamente ditos.
- Redes recorrentes. Utiliza duas próprias saídas para alimentar de volta suas próprias entradas; permitindo que possa adquirir memória de curto prazo.

Para Braga et al. (2007), a forma mais comum de utilização das RNAs é o de aprendizado por meio de um conjunto de dados. Dentro dessa visão, as tarefas principais nas quais as RNAs se aplicam são: classificação, categorização (agrupamento ou *clustering*), aproximação, previsão e otimização que serão descritos abaixo conforme o ponto de vista de Braga et al. (2007):

- Classificação: A tarefa de atribuir um padrão desconhecido entre várias classes conhecidas. A resolução de problemas de classificação por meio de RNAs baseia-se por aprendizado supervisionado, ou seja, exemplos de padrões são apresentados às entradas e as classes correspondentes são apresentadas às saídas da rede durante o processo de aprendizado. A rede, por sua vez, deve alterar seus pesos de forma a mapear as

relações entre padrões de entrada e classes correspondentes de saída, tendo por base, os dados do conjunto de treinamento. Assim, a classificação envolve, após o treinamento, atribuir uma das classes conhecidas a um padrão qualquer de entrada.

- Categorização: Envolve a descoberta de categorias ou classes bem definidas nos dados de entrada. Neste, caso, ao contrário da classificação, as classes não são conhecidas anteriormente. A resolução desse tipo de problema envolve aprendizado não supervisionado, ou seja, não há padrões apresentados nas entradas.
- Aproximação: Se caracteriza como aprendizado supervisionado. Em problemas de aproximação, o objetivo é mapear funções contínuas das variáveis de entrada.
- Previsão: Se caracteriza tipicamente pela estimativa de situações futuras com base nos estados atuais e anteriores do sistema a ser modelado**.** A sua solução é através do aprendizado supervisionado.
- Otimização: Se caracteriza pela minimização ou maximização de uma função de custo. Podem ser resolvidos através de modelos recorrentes. (Ex: modelo de Hopfield (VASCONCELOS, 2013). Com o a função de custo deve ser conhecida anteriormente para a caracterização do problema, os dados não se tornam

> necessários à sua resolução. Os valores dos pesos são obtidos através da formulação analítica do problema.

Tatibana e Kaetsu (2013) reforçam os conceitos citados por Braga et. al (2007) que as redes neurais possuem certas características exclusivas de sistemas biológicos e essas características entram em conflito com os tradicionais métodos computacionais. Sistemas de computação baseados em redes neurais tem a capacidade de receber ao mesmo tempo várias entradas e distribuí-las de maneira organizada. Geralmente, as informações armazenadas por uma rede neural é compartilhada por todas as suas unidades de processamento. Característica que contrasta com os atuais esquemas de memória, onde a informação fica confinada em um determinado endereço.

Em um sistema de rede neural, a informação pode parecer ter representação redundante, porém, o fato de que ela se encontre distribuída por todos os elementos da rede significa que mesmo que parte da rede seja destruída, a informação contida nesta parte ainda estará presente na rede, e poderá ser recuperada. Portanto, a redundância na representação de informações em uma rede neural, diferente de outros sistemas, transforma-se em uma vantagem, que torna o sistema tolerante a falhas. Os atributos de uma rede neural, tais como aprender através de exemplos, generalizações redundantes, e tolerância a falhas, proporcionam fortes incentivos para a escolha de redes neurais como uma escolha apropriada para aproximação para a modelagem de sistemas biológicos (TATIBANA; KAETSU, 2013).

O modelo de rede neural tem muitos neurônios conectados por pesos com capacidade de adaptação que podem ser arranjados em uma

estrutura paralela. Este paralelismo reforça os conceitos de tolerância a falhas; pois a falta de alguns neurônios não causa efeitos significantes para o desempenho de todo o sistema.

O principal aspecto na estrutura de redes neurais reside em sua habilidade de adaptação e aprendizagem. A habilidade de adaptação e aprendizagem pelo ambiente significa que modelos de redes neurais podem lidar com dados imprecisos e situações não totalmente definidas. Uma rede treinada de maneira razoável tem a habilidade de generalizar quando são inseridas entradas de dados que não são conhecidos por ela (TATIBANA; KAETSU, 2013).

Tatibana e Kaetsu (2013) afirmam que na área de jogos computacionais, as redes neurais apresentam como característica principal a capacidade de capturar comportamentos e predisposições em determinados padrões. Esta habilidade das redes neurais as tem tornado útil para modelar sistemas não lineares na combinação de controladores não lineares.

3.2.1 Um jogo implementado com Redes Neurais Artificiais

3.2.1.1 Descrição do jogo

Conceição (2009) propõe uma aplicação que consiste em um jogo chamado Jogo da Velha 3D em Pinos. Este jogo é uma variação do famoso Jogo da Velha. O Jogo da Velha 3D em Pinos possui muita semelhança com o jogo da velha jogado no “papel” onde a lógica do jogo é praticamente a mesma: Colocar os símbolos X (xis) ou O

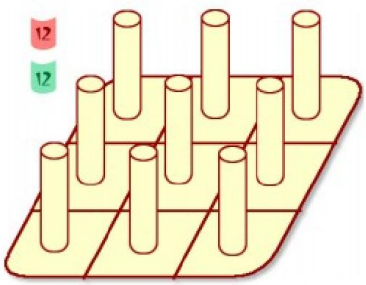
(bolinha) em linha reta (seja horizontal, vertical ou diagonal) de três símbolos do mesmo tipo que determina a pontuação.

3.2.1.2 Como o jogo funciona?

O Jogo da Velha 3D em Pinos é um jogo de tabuleiro para ser jogado por duas pessoas, uma sendo o adversário da outra. Ele consiste em jogadas de pontuação em turnos, ou seja, cada um dos oponentes tem a sua vez de jogar e em sua vez ele tem a chance de marcar pontos. O tabuleiro tem o formato geométrico quadrado (ou similar) em sua superfície, essa superfície é dividida em 9 partes de mesma área, contabilizando assim 9 pequenos quadrados, conforme a Figura 4. No centro de cada quadrado há um pino cilíndrico, contabilizando assim nove pinos distribuídos uniformemente na superfície. O jogo contém, além do tabuleiro, 24 peças com forma de argola e tamanhos iguais, este formato é para que as peças possam ser encaixadas nos pinos. As peças são divididas em duas cores, contabilizando assim, 12 peças para cada jogador.

Figura 4 - Jogo da velha 3D em pinos

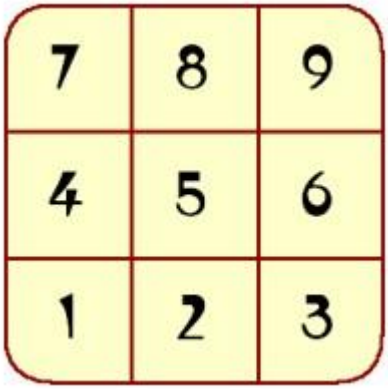
Fonte: Extraído de Conceição (2009)



Como no Jogo da Velha, a pontuação é feita quando um jogador coloca 3 de suas peças em linha reta. No Jogo da Velha há 9 locais onde podemos colocar uma peça X (xis) ou O (bolinha) como na Figura 3.2, esses locais são chamados de casas e estas numeradas de 1 a 9 conforme a Figura 5.

Figura 5 - Jogo da velha – Casas

Fonte: Extraído de Conceição (2009)



Conforme Conceição (2009) o jogo da velha existem 8 formas possíveis de ganhar o jogo. A Figura 6 ilustra as possibilidades possíveis.

Figura 6 - Possibilidades do jogo da velha

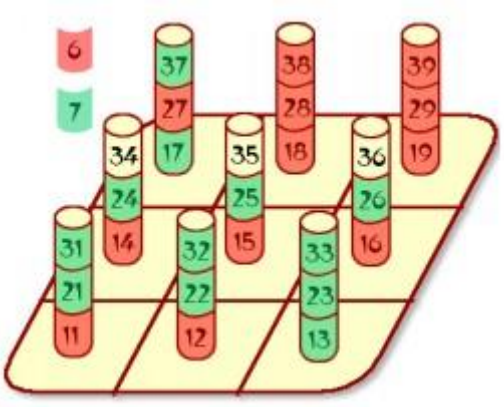
Fonte: Extraído de Conceição (2009)

(1,2,3),(1,4,7),(1,5,9),(2,5,8),
(3,5,7),(3,6,9),(4,5,6),(7,8,9)

Já no Jogo da Velha 3D em Pinos, os jogadores jogam até que as 24 peças se esgotem; e no final são contabilizados os pontos marcados de acordo com as combinações feitas por cada jogador, conforme pode ser visto na Figura 7.

Figura 7 - Exemplo de pontuação jogo da velha 3D em pinos

Fonte: Extraído de Conceição (2009)



Como é possível constatar na Figura 7 que cada pino possui três níveis (1, 2 e 3 – de baixo para cima, respectivamente), é sugestivo que cada pino possa receber três argolas; sendo assim, há 49 combinações possíveis, conforme a Figura 8.

Figura 8 - Possibilidades de pontuação - Jogo da Velha 3D em pinos

Fonte: Extraído de Conceição (2009)

(11,12,13),(11,14,17),(11,15,19),(11,21,31),(11,22,33),(11,24,37),(11,25,39),
(21,22,23),(21,24,27),(21,25,29),(31,32,33),(31,34,37),(31,35,39),(31,22,13),
(31,24,17),(31,25,19),(12,15,18),(12,22,32),(12,25,38),(22,25,28),(32,35,38),
(32,25,18),(13,15,17),(13,16,19),(13,23,33),(13,25,37),(13,26,39),(23,25,27),
(23,26,29),(33,35,37),(33,36,39),(33,25,17),(33,26,19),(14,15,16),(14,25,36),
(14,24,34),(24,25,26),(34,35,36),(34,25,16),(15,25,35),(16,26,36),(17,18,19),
(17,27,37),(17,28,39),(27,28,29),(37,38,39),(37,28,19),(18,28,38),(19,29,39)

3.2.1.3 Estrutura da rede

Conceição (2009) propõe que a rede tem como objetivo aprender a jogar o Jogo da Velha 3D em Pinos. Para isso, ela terá que aprender as regras do jogo como armar as jogadas, pontuar e bloquear pontos.

A estrutura da rede contém três camadas (por causa dos três níveis dos pinos), uma camada de entrada que contêm 27 neurônios (devido a quantidade de posições - 9 - multiplicado pela quantidade de níveis – 3 - de cada pino), uma camada oculta que contêm 49 neurônio (devido a quantidade de pontuações possíveis) e uma camada de saída que contêm 9 neurônios (que é a quantidade de pinos) (CONCEIÇÃO, 2009).

Segundo Conceição (2009), no Jogo da Velha 3D em Pinos, cada posição pode estar em apenas um dos três estados possíveis:

- ✓ Sem peça,
- ✓ Com a minha peça
- ✓ Com a peça do oponente

Para isso serão contabilizadas todas as posições, para cada posição há um neurônio responsável na camada de entrada e cada neurônio receberá um valor. Este valor pode ser:

- ✓ 0 (zero) quando não há peça para aquele neurônio,
- ✓ 1 (um) quando há peça para aquele neurônio e a peça é minha
- ✓ - 1(menos um) quando há peça para aquele neurônio e a peça é do oponente.

As posições onde as peças se encaixam vão de 1 a 27 em uma contagem sequencial pela numeração levando em consideração seu nível, sendo assim, haverá peças do primeiro nível numeradas de 11 a 19; segundo nível numeradas de 21 a 29 e terceiro nível numeradas de 31 a 39. (CONCEIÇÃO, 2009).

No treinamento da rede, foi utilizado o algoritmo de aprendizagem por correção de erro também conhecido como *Backpropagation* (ROISENBERG, 2013), pois se encaixa perfeitamente à estrutura da rede. (CONCEIÇÃO, 2009).

Um dos princípios que Conceição (2009) aborda, é que a rede aprenda a jogar sem que um humano intervenha no jogo para que o treinamento não seja lento. Para isso, foi proposta a criação de mais três redes para que pudessem ser oponentes.

Logo que os oponentes são criados, é necessário que algum deles saiba a regra do jogo. Para isso, foi criado um juiz, que auxilia as redes na disputa.

O papel do juiz é, a cada turno, avaliar a situação do jogo e a escolha da rede. Como o jogo possui 9 pinos, cada um com 3 posições, dando um total de 27 posições, a rede não escolhe em que posição vai jogar e sim em que pino vai jogar, então a rede escolhe um dos 9 pinos para jogar, e o juiz só precisa avaliar as 9 posições possíveis.

- ✓ Melhor posição – O juiz atribui o valor de 1 (um);
- ✓ Pior posição - O juiz atribui o valor de -1 (menos um);
- ✓ Outras posições o valor é proporcional, menor que 1 (um e maior que 1(um).

Para que o juiz saiba a melhor posição em cada turno para a situação, ele varre as 9 posições do tabuleiro. Para cada posição ele analisa as combinações possíveis e verifica se é possível fazer ponto (Ótimo), bloquear ponto do adversário (Ótimo), tentar caminho para um ponto na próxima jogada (Muito Bom), iniciar um caminho de ponto (Bom), colocar uma peça na combinação onde o adversário já tem uma peça (Ruim), colocar uma peça na combinação onde a rede e o adversário já têm uma peça finalizando assim as posições do pino e não fazendo pontuação alguma (Muito Ruim), e por último, colocar uma peça num pino sem espaço vago (Péssimo) (CONCEIÇÃO, 2009).

O juiz também faz análise da posição de onde se está fazendo a jogada, quantas combinações são possíveis fazer com aquela posição mesmo que no momento não façam pontos, isso porque é importante pegar posições estratégicas. O juiz pode analisar o jogo e pontuar cada jogada, pois ele tem conhecimento sobre todas as possibilidades de pontuação e as posições de influência (CONCEIÇÃO, 2009).

Atualmente o juiz considera apenas a situação atual do jogo para criticar a rede, não levando em consideração jogadas passadas e movimentações futuras, por isso é provável que a rede aprenda a não cair em armadilhas (CONCEIÇÃO, 2009).

Para que estas estratégias sejam executadas, existem os algoritmos responsáveis pela execução. O algoritmo do Juiz (Algoritmo 4) é o responsável por qualificar a jogada da rede e utiliza o Algoritmo 2 e o Algoritmo 3 como auxílio buscando neles as combinações possíveis e as porcentagens das posições de influência respectivamente (CONCEIÇÃO, 2009).

Os algoritmos desenvolvidos para o jogo podem ser consultados no trabalho de Conceição (2009).

3.3 Sistemas Multiagentes

A inteligência, como ciência, não consiste em uma característica individual, que pode ser separada pelo contexto social em que se encontra. Um ser humano, por exemplo, não pode se desenvolver adequadamente em ambiente social, se ele ou ela não é cercado por outros seres humanos. Sem um ambiente social adequado, o seu desenvolvimento cognitivo é muito restrito. Aprender a articular seus pensamentos em linguagem em comum torna-se praticamente impossível para alguém que não tenha sido envolvido em uma cultura humana desde os primeiros anos de vida. Em outras palavras, as outras pessoas são indispensáveis para o desenvolvimento cognitivo dos humanos, e o que chamamos de "inteligência" é devido não mais para a base genética que define a nossa estrutura “neurônica”, mas sim para a interação com o mundo que os rodeiam, e ir participar com a sociedade humana (FERBER, 1999).

Para reforçar os conceitos de Ferber (1999), Hübner, Bordini e Vieira (2004) afirmam que o estudo de Sistema Multiagentes (SMA), ao contrário dos paradigmas tradicionais da IA, tem por objetivo o estudo da coletividade entre os vários agentes que podem compor o sistema, e não apenas um agente. Desta forma, deixam de ter atenção as iniciativas de compreender e simular o comportamento humano isoladamente, mas sim entre a interação entre os agentes que formam o sistema e sua organização.

Exemplificando, apesar de um formigueiro ser constituído por seres simples como as formigas, pode-se dizer que o sistema como um todo (o formigueiro, a formiga e as atitudes destas) é muito mais complexo e inteligente do que uma formiga por si só. Isso mostra que a inteligência não está aplicada apenas a um componente, e sim na coletividade do grupo. Entre as características coletivas do SMA incluem-se a interação entre os agentes (linguagens e protocolos), o ambiente e a organização (HÜBNER; BORDINI; VIEIRA, 2004).

Utilizando o exemplo do formigueiro citado anteriormente e aliado aos conceitos apresentados, é possível dizer que um SMA estuda o comportamento organizado de agentes autônomos para resolver problemas que um agente não poderia resolver sozinho (HÜBNER; BORDINI; VIEIRA, 2004).

Segundo Hübner, Bordini e Vieira (2004), o SMA possui duas propriedades que são aparentemente contraditórias, mas que são essenciais:

1. Autonomia: Os agentes devem ser autônomos para realizar tomadas de decisões e além disso o agente tem sua existência independente dos demais.
2. Restrições: Devem existir algumas restrições referente ao comportamento em conjunto dos agentes para que haja um comportamento grupal coeso.

Aparentemente, “uma coisa tranca a outra”. Este é uma das características que não devem ocorrer. O ideal é obter o resultado esperado através da interação entre os agentes até atingir um ponto de equilíbrio.

Durante o desenvolvimento do SMA, duas abordagens são aplicadas para conciliar a autonomia e as restrições.

1. *Top Down* – "Inicia-se definindo os aspectos coletivos, como organização e comunicação, que são refinados até a definição dos agentes." (HÜBNER; BORDINI; VIEIRA, 2004).
2. *Botton up* – Inicia-se definindo os aspectos individuais, relacionados aos agentes. A interação e a organização são definidas do ponto de vista dos agentes.

Em outras palavras, a abordagem *Top Down* prioriza o estudo da coletividade dos agentes, o grupo como um todo; e a abordagem *Botton up* foca na visão de cada agente perante o jogo (HÜBNER; BORDINI; VIEIRA, 2004).

Para compreender melhor como funciona um agente em si, é essencial ter os conhecimentos da abordagem *Botton Up*. Segundo Hübner, Bordini e Vieira (2004), a abordagem *Botton Up* apresenta as seguintes características:

- Os agentes são gerados independentemente de um problema particular;
- A projeção das interações a serem realizadas pelos agentes não são feitas previamente, e deve ser definidos protocolos para serem utilizados em situações genéricas;
- A divisão das tarefas pode ser realizada pelos próprios agentes;

- Não existe um agente controlador central.

A partir destas características, é possível citar algumas vantagens (HÜBNER; BORDINI; VIEIRA, 2004).

- Viabiliza sistemas adaptativos e evolutivos; pode se adaptar a novas situações, tanto pela inclusão e/ou eliminação de novos agentes ao sistema quanto pela mudança da sua organização.

- Sistemas distribuídos: na maioria das situações, o conhecimento, recursos e controle são distribuídos e heterogêneos. Por ser distribuído, o sistema possui um melhor desempenho.

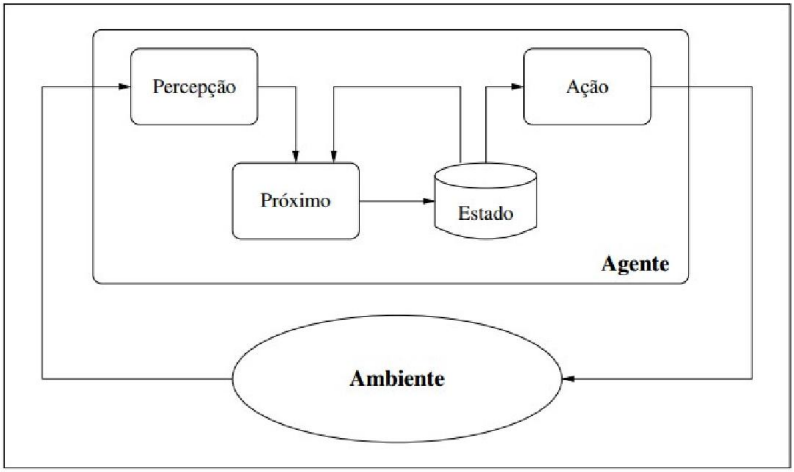
- Sistemas abertos: um agente pode participar de outras sociedades (entrar e sair) conforme necessário. Mesmo que os objetivos sejam diferentes e desenvolvidos por projetistas distintos.

Com isso, Hübner, Bordini e Vieira (2004), afirmam que SMA podem ser utilizados na aplicação de jogos computacionais para aumentar o realismo dos personagens; já que estes devem ser

inteligentes e capazes de se adaptar ao jogo conforme a interação dos personagens.

Figura 9 - Modelo geral de agente

Fonte: Extraído de Hübner, Bordini e Vieira (2004)



Para exemplificar melhor, a Figura 9 apresenta o modelo geral de agente (HÜBNER; BORDINI; VIEIRA, 2004).

Segundo Hübner, Bordini e Vieira (2004) algumas características importantes de um agente e de um SMA como um todo são:

- Percepção: O agente é capaz de perceber alterações no ambiente;
- Ação: As alterações no ambiente são feitas pelos agentes que por sua vez tem uma motivação para atingir os objetivos para qual ele foi criado;
- Comunicação: Comunicar-se com os agentes da comunidade (que compartilham o mesmo ambiente) e

como eles precisam coordenar suas ações, a comunicação é essencial;

- Representação: o Agente possui uma representação simbólica daquilo que acredita ser verdade em relação ao seu ambiente e os outros agentes que compartilham aquele ambiente;
- Motivação: Os agentes são (ou podem ser) autônomos e é essencial que exista uma representação dos seus objetivos ou o que os motiva o agente a almejar algo. Como consequência, o agente interage com o ambiente a fim de atingir seu objetivo;
- Deliberação: Dada a motivação, o agente deve ser capaz de decidir a partir do estado em que se encontra e os estados possíveis que ele pode assumir de acordo com seus objetivos;
- Raciocínio e aprendizagem: Outras técnicas de IA podem ser utilizadas para cada agente aumentando o desempenho desses. Como exemplo, pode-se pegar a característica citada anteriormente (Deliberação). Quando um agente estiver em uma situação a qual já "errou", ele pode ter uma técnica de redes neurais que auxiliará na decisão, já que é possível "aprender" através desta técnica de IA. Porém, esta é uma área ainda não muito explorada e ainda carece de muita pesquisa.

3.3.1 Um jogo implementado com Sistemas Multiagentes

Valente e Rio (2005) desenvolveram um jogo estratégico de palavras que tem o objetivo de montar palavras através da negociação das letras entre agentes que compõem o jogo.

O jogo é composto por três a seis jogadores (que por sua vez são os próprios agentes) onde cada um possui autonomia para tomada de decisões e que sejam capazes de interagir com os outros agentes (conceitos já citados anteriormente) que são reforçados por Valente e Rio (2005). O sistema permitirá a participação de quaisquer agentes que atendam algumas regras específicas e que reconheça as mensagens que são enviadas entre os agentes. A garantia e segurança que as regras estejam sendo cumpridas são realizadas por um agente gestor.

Outra característica importante do jogo, é que os agentes são suficientemente inteligentes para "cooperar" com o agente que tem menos pontos. Quando a distância de pontos entre os agentes atingir uma quantia significativa, os agentes adotam técnicas para não deixar que o último colocado se distancie muito dos demais agentes.

3.3.3.1 Funcionalidades

O jogo proposto por Valente e Rio (2005) baseia-se em um sistema onde, no início, é selecionado os jogadores que participarão do jogo seguindo uma série de regras que são descritas no manual.

Além disso, são apresentados o estado atual e a fase a qual o jogo se encontra de maneira dinâmica e também de todo o processo de

negociação entre os agentes, tornando a verificação da correta execução do jogo mais prática.

O sistema permite a participação de qualquer agente cujos estes sejam compatíveis com a plataforma utilizada no jogo bastando a implementação do protocolo de comunicação utilizado nas negociações e atualizações de estados utilizado para realizar a comunicação entre o gestor e o jogador.

Inicialmente, foram projetados agentes que realizam apenas jogadas aleatórias, em seguida foram criado agentes que sempre realizam os "melhores" negócios, os mais lucrativos que lhe trazia a maior quantidade de pontos instantânea.

Em uma fase posterior, Valente e Rio (2005) desenvolveram agentes com noção de estado de jogo que escolhiam o negócio que proporcionava um melhor estado de jogo, ou seja, quantificava o jogo atual e um jogo após cada uma das jogadas possíveis no momento, executando assim uma pesquisa em largura com um nível de profundidade (Similar a um jogo de xadrez).

Por fim, foram construídos agentes baseados nestes últimos que evoluem de jogo para jogo utilizando um algoritmo de melhoria iterativa, baseado num algoritmo genético; (característica que reforça os conceitos citados anteriormente por Hübner, Bordini e Vieira (2004), onde argumentam que um agente pode ter outra técnica de IA implementada).

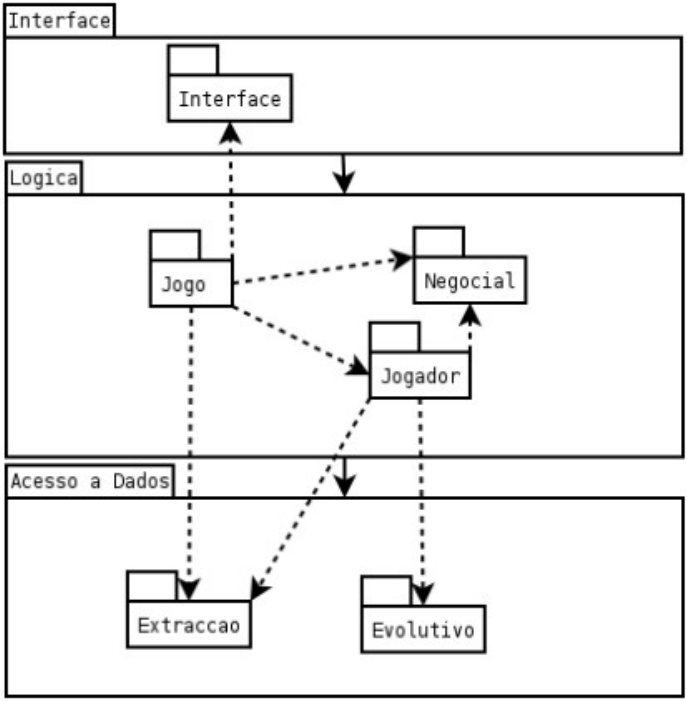
Com intuito de tornar o jogo mais real, foram criados seis jogadores com inteligência inicial distinta, onde cada um possui uma estratégia diferente. Por exemplo, um jogador pode ter uma estratégia

onde deve possuir muitas letras e outro tentar obter sempre muitos pontos. Durante o andamento do jogo, os agentes podem se modificar e adaptar a sua inteligência e por consequência a sua estratégia no jogo, de modo que em cada estratégia distinta, possa obter um maior aperfeiçoamento do modo de aplicá-la (VALENTE; RIO, 2005).

3.3.1.1 Estrutura do jogo

Figura 10 - Arquitetura em três camadas.

Fonte: Extraído de Valente e Rio (2005)



Para obter uma melhor estruturação e organização do jogo, Valente e Rio (2005) optaram por dividir o jogo em módulos, sendo que cada módulo seja responsável por um conjunto de funcionalidades. Para obter um sistema mais flexível, foi adotado uma arquitetura de três camadas (interface, lógica de negócio e acesso a dados), permitindo que possa ser substituído qualquer uma delas sendo necessário manter apenas o canal de comunicação entre as camadas.

Para expressar melhor, a Figura 10 mostra a arquitetura proposta por Valente e Rio (2005).

- Interface – Esta camada é responsável pela apresentação ao utilizador do estado do jogo, visualização das diferentes negociações entre jogadores e por possuir todas as funcionalidades relativas à escolha de jogadores no início do jogo.

- Lógica de negócio – Responsável por todas as funcionalidades referentes ao funcionamento do jogo, desde a execução de jogadas, a sua validação, a comunicação entre os agentes, a execução das regras, etc.

- Acesso a dados - Responsável pelo acesso à informação guardada nos arquivos que acompanham a aplicação, tais como a extração das palavras dos dicionários, a extração e atualização dos genes, bem como o acesso aos dados enviados pelos agentes através da comunicação.

3.3.1.2 Estrutura modular

Como foi citado anteriormente, o jogo é dividido em módulos que são responsáveis por organizar a estrutura do jogo. Abaixo segue a descrição dos módulos conforme Valente e Rio (2005) estruturaram.

- Módulo de interface – São desenvolvidos as funcionalidades referente ao visual do jogo.

Neste módulo é desenvolvido duas interfaces (VALENTE; RIO, 2005):

- Interface inicial: Nela é possível selecionar os jogadores que disputarão o jogo.
- Interface principal: Responsável por possibilitar uma visão geral do jogo – jogadores e seus nomes; pontuação; palavras a serem montadas e as letras disponíveis. Em cada instante serão apresentadas frases que representam cada passo do jogo.

Para desenvolver a interface foi utilizado a biblioteca gráfica, "G" (GeoSoft), "que permite criar objetos gráficos e adicioná-los a cenas criadas". As classes de maior relevância no módulo: *InterfaceInicial* e *InterfaceJogo*.

- *Módulo de jogo* – Implementa as funcionalidades referentes a inicialização do jogo "desde a invocação dos jogadores à chamada dos métodos oferecidos pela interface".

Classes do módulo*: Jogo, EstadoJogo e AgenteGestor.*

- Módulo de jogador – Implementa as funcionalidades relativas aos jogadores desde aquelas que executam os processos de raciocínio e validam as informações utilizadas nos raciocínios.

  Classes do módulo: *Jogador* e *AgenteJogadorInteligente.*

- Módulo negocial – Implementa as funcionalidades de negociação entre os jogadores. Exemplo: Obter negócios possíveis.

  Classes do módulo: *Negocio* e *ListaNegocios.*

- Módulo de extração – Implementa as funcionalidades de extração de palavras do dicionário. Além disso são implementados o sorteio e a atribuição de letras aos jogadores.

  Classes do módulo: *Palavras* e *Letras.*

- Módulo evolutivo: Implementa as funcionalidades referentes a extração e submissão da informação guardada nos arquivos de cada jogador e também as funcionalidades genéticas que devem ser aplicadas aos valores extraídos.

A plataforma utilizada para desenvolvimento do jogo foi a JADE (veja mais em: http://jade-lang.com/).

Além disso, o trabalho de Valente e Rio (2005) aborda mais detalhadamente os detalhes como a comunicação entre os agentes e o gestor; o cálculo de estado de um jogo para um jogador e a escolha do negócio a ser executado a cada momento a colaboração entre os agentes que não são objetos deste trabalho.

3.4 Lógica *Fuzzy*

A lógica *Fuzzy* proporciona que sistemas manipulem de forma eficaz informações incertas ou incompletas, os sistemas baseados em casos – ex: andando ou parado – permite que essas informações sejam reaproveitadas reduzindo a necessidade de processá-las novamente. Como os sistemas baseados em casos reusam casos anteriores adaptando-os às novas situações, o desempenho pode ser muito satisfatório quando soluções melhores podem não ser alcançadas se suas características não estiverem presentes em nenhum caso salvo anteriormente (Boscatto; Nakamiti; Toledo, 2012).

Conforme Rezende (2003) *apud* Rieder e Brancher (2004), o termo *Fuzzy* pode ter diversos significados de acordo com o contexto de interesse, mas, basicamente o conceito básico é de algo vago, incerto. A lógica *Fuzzy* de maneira geral, generaliza a lógica binária para ser utilizada em situações em que haja um alto grau de incerteza (Demasi 2003) *apud* Rieder e Brancher (2004).

Outra ressalva importante citada por Rieder e Brancher (2004) é que a Lógica *Fuzzy* traduz em termos matemáticos a informação

imprecisa expressa por um conjunto de regras linguísticas (RIEDER; BRANCHER, 2004).

Ao contrário dos sistemas lógicos binários, onde os valores devem ser exatos (ex: Ligado (1), desligado (0)), a lógica *Fuzzy* trabalha com valores aproximados, onde os valores podem variar (GOMINE; GUDWIN; TANSCHEIT, 2002). Por exemplo, pode-se considerar que um aluno de uma universidade X diga: "Eu estudo *muito*". Pode-se ser levantado várias incógnitas referente a quantidade de horas por dia/semana que este aluno estuda. *Muito* para uma *pessoa Y* pode ser duas horas de estudo por dia, enquanto para uma *pessoa Z, muito* pode significar 10 horas por dia.

Em outras palavras, a lógica *Fuzzy* trabalha com valores lógicos diferentes. Ao invés de utilizar números, a lógica *Fuzzy* trabalha com variáveis linguísticas (ex: muito, pouco, grande, pequeno, etc.) onde cada termo linguístico é interpretado de maneira diferente. (GOMINE; GUDWIN; TANSCHEIT, 2002).

Outro fator interessante é que a probabilidade na lógica clássica é sempre um valor numérico ou um intervalo entre dois valores. Na lógica *Fuzzy* é possível aplicar probabilidades linguísticas (ex: risco pequeno, risco grande, probabilidade grande, probabilidade pequena) que são interpretados por "números *Fuzzy"* e manipulados por aritmética *Fuzzy* (Kaufmann & Gupta *apud* (GOMINE; GUDWIN; TANSCHEIT, 2002)).

> *"A modelagem e o controle Fuzzy (Lee, 1990) são técnicas para se manusear informações qualitativas de uma maneira rigorosa. Tais técnicas consideram o modo como a falta de*

> *exatidão e a incerteza são descritas e, fazendo isso, tornam-se suficientemente poderosas para manipular de maneira conveniente o conhecimento. A sua utilização em sistemas de controle de processos em tempo real, em computadores ou micro-controladores, é das mais convenientes, dado que, geralmente, não envolvem nenhum problema computacional sério. A teoria de modelagem e controle Fuzzy trata do relacionamento entre entradas e saídas, agregando vários parâmetros de processo e de controle. Isso permite a consideração de processos complexos, de modo que os sistemas de controle resultantes proporcionam um resultado mais acurado, além de um desempenho estável e robusto. A grande simplicidade de implementação de sistemas de controle Fuzzy pode reduzir a complexidade de um projeto a um ponto em que problemas anteriormente intratáveis passam agora a ser solúveis."* (GOMINE; GUDWIN; TANSCHEIT, 2002).

3.4.1 Um jogo implementado com lógica *Fuzzy*

Boscatto, Nakamiti e Toledo (2012) implementaram um jogo que possui um sistema de controle tempo-real para uma nave autônoma onde a estratégia de controle é testada em um cenário similar ao jogo *River Raid* onde a nave deve voar em um local composto por vários obstáculos

sem se chocar com obstáculos que ficam em constante movimentação no cenário, conforme a Figura 11.

A lógica *Fuzzy* é utilizada na elaboração deste jogo para definir as distâncias aproximadas entre a nave e os demais objetos do cenário do jogo.

Figura 11 - Tela do jogo River Raid

Fonte: Extraído de Boscatto, Nakamiti e Toledo (2012)

A arquitetura do sistema é composta por diversos módulos. O módulo XNA, descrito na Figura 12, utiliza as bibliotecas da plataforma de mesmo nome baseada no *framework* .NET com alguns métodos básicos como:

- *Load()* –Carrega os dados para a memória principal;
- *Unload()* – Limpar a memória;
- *Update()* – Atualizar as posições dos objetos;
- *Draw()* – Desenhar a tabela.

O módulo *Preparar Obstáculo* obtém informações do cenário como os obstáculos e as respectivas coordenadas.

O módulo *Atualizar Posição da Nave* obtém informações da movimentação da nave e é dividido em três métodos:

- *MovimentarNave()* –Atualiza a posição da nave principal a partir dos comandos de alteração na direção e intensidade de movimento;
- *ChecarLimites*() – Verifica se a nave não ultrapassa os limites da tela previamente estabelecidos;
- *ChecarColisão*() – Verifica a colisão com os objetos do cenário e a intensidade das colisões observando a sobreposição dos pixels para maior precisão em relação aos objetos não retangulares.

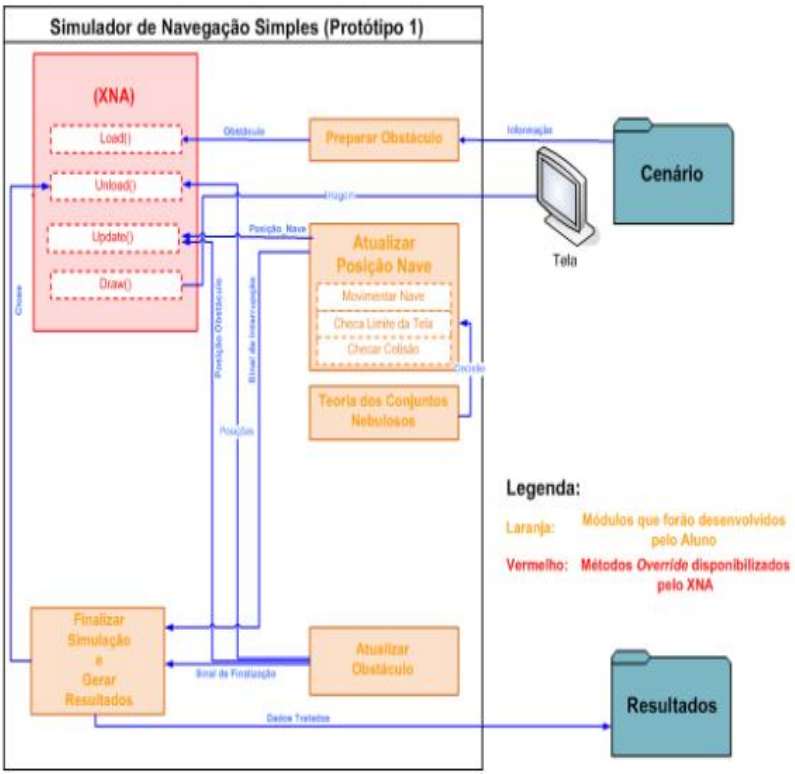
O módulo *Atualizar Obstáculo* tem a função de mover obstáculos da tela para baixo, passando a sensação que o cenário está em constante movimento, fazendo com que a nave pareça se deslocar para frente.

Os módulos *Finalizar Simulação* e *Gerar Resultados* libera a memória, pois os objetos a ocuparam durante a simulação e gera um arquivo texto com um relatório com as informações como o número de objetos ultrapassados e o tempo total de simulação.

Por fim, os módulos *Teoria dos Conjuntos Nebulosos (Fuzzy), Raciocínio Baseado em Casos e Algoritmos Genéticos,* são responsáveis pela orientação da nave de acordo com o cenário.

Figura 12 - Arquitetura do jogo River Raid

Fonte: Extraído de Boscatto, Nakamiti e Toledo (2012).



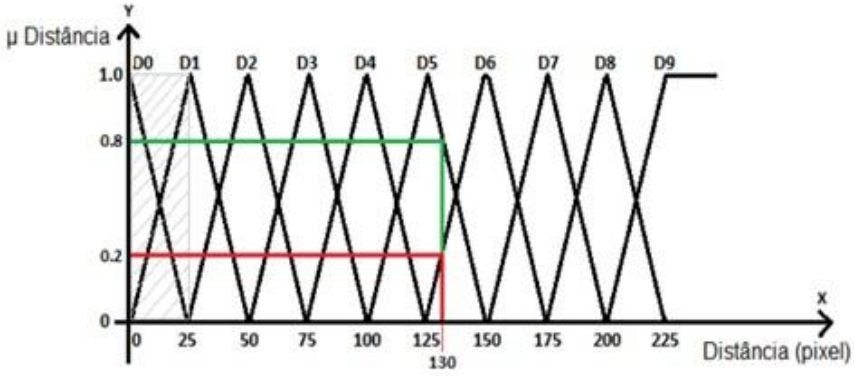
Nota: Neste trabalho será apenas detalhado o módulo de *Teoria os Conjuntos Nebulosos (Fuzzy).*

O módulo *Fuzzy* realiza o treinamento do sistema, gerando a base inicial de casos que será utilizada pelos outros módulos adaptativos.

Basicamente, ele converte as distâncias entre a nave os demais objetos do cenário, usando uma função de pertinência *Distância* mostrada na Figura 13 (BOSCATTO; NAKAMITI; TOLEDO, 2012).

Figura 13 - Representação gráfica da variável Distância.

Fonte: Extraído de Boscatto, Nakamiti e Toledo (2012).



As funções D0 a D9 são utilizadas no lugar de termos linguísticos (Ex: Baixa, Média, Alta, etc). Na Figura 13, um objeto que lista 130 pixels da nave (distância cartesiana) teria como função de pertinência 0,8 D5 e 0,2 D6. A aplicação das regras *Fuzzy* e o cálculo da ação a ser tomada são obtidos através do centro da área, descrito na literatura, incluído em Nakamiti et. Al. [2001] apud Boscatto, Nakamiti e Toledo (2012). A fase de treinamento *Fuzzy* é executada até que gere 50 casos (valor configurável), que servirão de entrada ao mecanismo híbrido

genético-baseado em casos, facilitando sua convergência. (BOSCATTO; NAKAMITI; TOLEDO, 2012).

Conforme os autores deste jogo, em todos os testes em que foram utilizadas as regras de produção *Fuzzy*, o mesmo resultado foi obtido (a nave conseguiu ou não atravessar o cenário sem colisões). Cabe observar que foram utilizados regras de produção estáticas, isto é, não foram geradas novas regras em tempo de execução. Nos testes com o sistema completo, o sistema colidiu com os obstáculos nos cenários mais complexos nas primeiras tentativas. Depois, em todos os casos, o sistema aprendeu uma solução e prosseguiu nos testes sem colidir; com auxílio de uma RNA.

É possível verificar os resultados obtidos por Boscatto, Nakamiti e Toledo (2012) na Tabela 2.

Tabela 2 - Resultados obtidos com a lógica *Fuzzy*

Fonte: Extraído de Bocatto, Nakamiti e Toledo (2012)

Cenário (obstáculos)	*Fuzzy* (nº de tentativas)	Completo (nº de tentativas)
1. Quatro obstáculos dispostos próximos aos cantos da tela (cenário).	10	10
Nove obstáculos divididos em três linhas, alternadamente do lado esquerdo e direito do cenário	10	10
Nove obstáculos dispostos em zig-zag	0	10
Dezoito obstáculos, que		

incluem os obstáculos dos cenários 2 e 3.	0	7
Dezenove obstáculos, sendo catorze vindos do cenário anterior, quatro obstáculos formando pirâmides com os anteriores e mais um obstáculo disposto aleatoriamente.	0	4

A partir dos resultados obtidos, é possível verificar que o módulo *Fuzzy* se comportou melhor que o módulo raciocínio baseado em casos e o módulo algoritmos genéticos; mostrando a capacidade do sistema em aprender a com suas experiências anteriores, mesmo em cenários que possuem muitos obstáculos e poucas alternativas de sucesso. Mais informações sobre o comparativo com a técnica de algoritmos genéticos e raciocínio baseado em casos podem ser consultado no artigo de Bocatto, Nakamiti e Toledo (2012).

3.5 Considerações sobre as características de cada técnica

Neste capítulo foram estudadas três técnicas de IA (Redes Neurais Artificiais, Sistemas Multiagentes e lógica *Fuzzy*) e uma aplicação de jogo a cada técnica e com o estudo foi possível verificar quais as características de cada técnica.

Com RNAs, é possível afirmar que elas tentam simular o comportamento do cérebro humano, realizando aprendizado de acordo com as interações que esta faz.

Já a lógica *Fuzzy* é utilizada para resolver problemas onde não há uma métrica "binária" e as decisões passam a ser tomadas sobre outra perspectiva; como por exemplo: muito, pouco, grande, pequeno, etc. Essas variáveis não-numéricas são trabalhadas para determinar um padrão de decisões que deve ser feito de acordo com um valor aproximado.

Concluindo o capítulo, o SMA são baseados na coletividade e na autonomia para tomada de decisões. O SMA é muito recomendado para aplicações distribuídas onde os agentes que compõem o sistema se comunicam para a tomada de decisões.

Para os resultados ficarem mais visíveis, segue abaixo as características de cada técnica conforme os autores citados no decorrer do capítulo.

Redes Neurais:

- ✓ Permite recuperação de informação;
- ✓ Possui tolerância a falhas;
- ✓ Capaz de realizar aprendizado através de exemplos fornecidos;
- ✓ Generalizações;
- ✓ Distribuição da informação de maneira organizada;
- ✓ Não é baseada em regras.

Sistemas Multiagentes:

- ✓ A inteligência é aplicada não apenas a um agente e sim ao grupo todo;
- ✓ Permite interação entre os agentes;
- ✓ Os agentes devem ser autônomos para realizar tomadas de decisões;
- ✓ Existem algumas restrições referente ao comportamento em conjunto dos agentes para que haja um comportamento grupal coeso;
- ✓ Viabiliza sistemas adaptativos e evolutivos;
- ✓ Podem ser utilizados em Sistemas distribuídos e Sistemas abertos;
- ✓ Outras características: Percepção, Ação, Comunicação, Representação, Motivação, Deliberação, Raciocínio e aprendizagem;
- ✓ É aplicável a jogos computacionais para aumentar o realismo dos personagens; já que estes devem ser inteligentes e capazes de se adaptar ao jogo conforme a interação dos personagens.

Lógica *Fuzzy*

- ✓ Manipula de forma eficaz informações incertas ou incompletas;

- ✓ Como os sistemas baseados em casos reusam casos anteriores adaptando-os às novas situações, o desempenho pode ser muito satisfatório quando soluções melhores podem não ser alcançadas se suas características não estiverem presentes em nenhum caso salvo anteriormente;
- ✓ Possui a capacidade de generalizar a lógica clássica binária de forma a ser aproveitada em situações em que haja um alto grau de incerteza;
- ✓ Traduz em termos matemáticos a informação imprecisa expressa por um conjunto de regras linguísticas
- ✓ Trabalha com valores aproximados;
- ✓ É capaz de resolver processos complexos, de modo que os sistemas de controle resultantes proporcionam um resultado mais acurado, além de um desempenho estável e robusto;
- ✓ Redução da complexidade de um projeto a um ponto em que problemas anteriormente intratáveis passam agora a ser solúveis.

A partir do levantamento destes dados, será abordado no Capítulo 4 como será aplicado a IA a um jogo específico.

4. APLICAÇÃO DA IA A UM JOGO ESPECÍFICO

Neste capítulo será apresentada a contribuição do trabalho a partir do estudo das técnicas de IA além do detalhamento do jogo *Astorian Tears*, desenvolvido pelo LabTeC.

4.1 Descrição do Jogo de estratégias táticas ***Astorian Tears*** (técnica SMA)

O jogo acontece num cenário de fantasia-medieval, no qual o jogador controla as ações de um personagem – avatar -, que é o herói do jogo. O jogador deverá guiá-lo pelo mundo em busca de sua vingança pessoal contra um ditador cruel.

Para isto, o herói poderá encontrar aliados durante sua jornada que poderão se juntar a sua causa, bem como poderá contratar novos recrutas sempre que visitar uma cidade, ou ainda domar monstros para que estes lutem ao seu lado.

4.1.1 Mecânica do jogo

O jogo consiste em duas fases principais. A primeira apresenta para o jogador uma visão do mapa do continente onde o herói se encontra. A partir desta tela, é possível escolher o próximo ponto navegável, através do uso de pontos de interesse, ou ainda o jogador pode decidir por permanecer no ponto onde se encontra e realizar ações de manutenção em sua equipe; que consiste na contratação e/ou

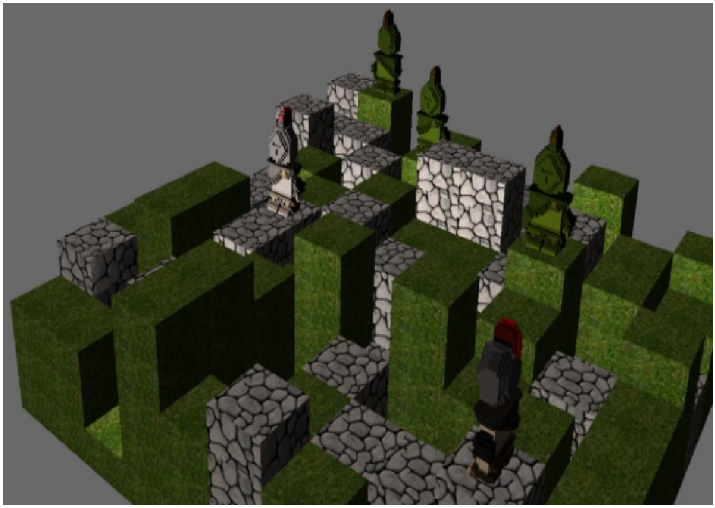
exclusão de membros em seu grupo, a distribuição de pontos de habilidades adquiridos em batalha e a escolha de equipamentos e habilidades utilizadas em combate.

Uma vez que o jogador explora um novo ponto de interesse, este poderá ativar um evento de batalha, que consiste no jogador ter de enfrentar um grupo adversário, que se formará de acordo com o local visitado.

As batalhas do jogo são a segunda e principal etapa do mesmo, onde o jogador deverá controlar seu esquadrão de soldados contra um grupo de adversários. A condição para vitória será alcançada quando o mesmo eliminar todos os oponentes.

Figura 14: Exemplo de figura de combate

Fonte: Extraído de Canella et al. (2013)



Estas batalhas ocorrem em um mapa similar a um tabuleiro tridimensional (conforme pode ser visto na Figura 14), o qual é divido em quadrados, sendo que cada um deles pode ser ocupado por apenas um único personagem por vez.

Ao invés de utilizar turnos para indicar de qual jogador é a vez, será utilizado um sistema em que apenas um personagem estará ativo por vez, sendo que a frequência com a qual este personagem se ativa é determinada pela sua velocidade.

Além da velocidade, existem outros atributos que definem um personagem, os quais afetam suas habilidades em combate, e que são adquiridos conforme este avança em experiência, a qual aumenta proporcionalmente a quantidade de batalhas por ele presenciadas. Estes valores determinarão a quantidade de casas que ele poderá se locomover em um único turno ativo, bem como as formas de ataque que lhe estarão disponíveis, além de influenciar em outros pontos, como a quantidade de dano causada e a quantia máxima de dano que poderá sofrer antes de morrer.

4.1.2 Personagens

O jogo é composto por guerreiros, magos, arqueiros, dragões, monstros, animais, etc. A Figura 15 apresenta um exemplo de personagem do jogo.

Figura 15 - Exemplo de personagem

Fonte: Adaptado de Canella et al. (2013)

4.1.2.1 Ações dos personagens

Por vez, apenas um personagem poderá estar ativo no cenário de turno de batalha. O personagem que estiver ativo poderá realizar três ações principais durante sua vez, demonstradas abaixo, sendo que fora a terceira opção, as demais opções são opcionais em um turno:

- Realizar uma ação de movimento (opcional);
- Realizar uma ação de atuação (opcional);
- Encerrar seu turno.

Ação de Movimento: Esta ação é a que realiza o deslocamento do personagem pelo cenário.

Ação de Encerrar o Turno: Quando esta ação for escolhida, o personagem ativo deve escolher uma direção na qual ele ficará direcionado, onde após a confirmação da mesma seu turno se encerra.

A ação de encerrar o turno é obrigatória para cada vez que o personagem se tornar ativo.

O personagem possui vários estados. Abaixo são descritos os estados possíveis:

Ativo: Aguarda um comando para a transição de estado.

Deslocando: Ocorre enquanto o personagem estiver se deslocando de um ponto a outro.

Saqueando: Acontece ao final de uma ação de movimento, onde o personagem recupera um item, se houver, do novo local onde ele está ocupando.

Atuando: O personagem está realizando uma ação em um alvo, sendo que este alvo pode ser ele mesmo, um aliado, um adversário, ou um objeto do cenário (à exemplo, um personagem morto, ou um baú). As atividades que ele pode executar enquanto atuando estão descritas na sessão de ações do personagem.

Sendo alvo: O estado será ativado sempre que o personagem for alvo de uma ação do tipo atuando, e quando a mesma lhe atingir.

Reagindo: O personagem entra neste estado apenas quando ele possui uma ação de resposta a atuação de outro personagem, seja ele aliado ou não.

Desmaiado: O evento acontece sempre que os pontos de vida do personagem caem a zero, ou quando uma ação causar este efeito.

Morto: Acontece quando o personagem não é reanimado a tempo, enquanto este estava no estado desmaiado. Uma vez que o personagem morra, ele não poderá ser reanimado nunca mais. Caso um personagem chave morra, isto indicará fim de jogo.

Nota: O jogo *Astorian Tears* possui diversas outras características que são descritas no trabalho e que podem ser consultados nas referências.

4.3 Técnica recomendada para o jogo de estratégias táticas *Astorian Tears.*

O jogo *Astorian Tears* possui uma grande quantidade de estados possíveis; e é necessário que haja uma técnica de IA que seja capaz de tratá-las e aplicar uma técnica a estes estados.

Neste caso, é possível atribuir a técnica de SMA onde estes podem ser utilizados para tomada autônoma de decisões; onde os agentes dividirão as tarefas (defender/atacar inimigos) a fim de concluir o objetivo do jogo. Hübner, Bordini e Vieira (2004) afirmam isto no decorrer do trabalho colocando que os agentes não devem ter atenção somente as iniciativas de compreender e simular o comportamento humano isoladamente, mas sim entre a interação entre os agentes que formam o sistema e sua organização.

Hübner, Bordini e Vieira (2004) ainda afirmam que um SMA possui duas características que se encaixam no jogo; que são:

- Autonomia;
- Restrições;

No primeiro item, está de acordo com o personagem que deve percorrer o cenário virtual para derrotar o seu inimigo (ditador cruel) que será localizado ao longo das fases. Além disso, os agentes podem ter autonomia na tomada de decisões. Canella et al. (2013) afirma que o personagem pode encontrar aliados durante sua jornada que poderão se juntar a sua causa (derrotar o ditador cruel), bem como poderá contratar novos recrutas sempre que visitar uma cidade, ou ainda domar monstros para que estes lutem ao seu lado.

Figura 16 - Objetivo do personagem

Fonte: Adaptado de Canella et al. (2013)

Outro fator que reforça a necessidade de utilizar SMA, é que a segunda e principal etapa do jogo, ocorre às batalhas do jogo onde o jogador deverá possuir uma organização do seu esquadrão de soldados contra um grupo de adversários. Neste contexto é explorado o fato de

que os agentes tem uma "missão" em comum e que esta deve ser cumprida. A Figura 16 apresenta os detalhes do objetivo do jogador.

O personagem deve possuir um plano de ação para cumprir uma lista de desejos. Ao analisar a Figura 16, é possível constatar que a sequência de ações a serem tomadas (metas) e cada uma delas possui sub-desejos que também devem seguir a mesma ordem.

> *"Seu primeiro sub-desejo é o de se aproximar de seu inimigo, pois seus ataques não podem alcançá-lo de onde ele se encontra. Para satisfazer esta meta, uma nova lista de sub-desejos deve ser satisfeita. Agora, para que Xico possa se aproximar de seu oponente, ele deverá primeiro identificar em qual direção ele deve avançar e em seguida, estipular qual o menor caminho até seu objetivo, enquanto evita pontos inacessíveis do mapa, como lugares muito altos e/ou muito profundos."* (CANELLA et al., 2013).

Ao término da execução da meta, o personagem verifica se o seu desejo foi concluído e caso contrário, deverá repetir as metas até concluí-las ou elaborar uma nova estratégia, dependendo do caso (o oponente pode possuir uma arma que anule, por exemplo, seu ataque "Bola de fogo") reforçando os conceitos de Hübner, Bordini e Vieira (2004).

Referente ao item Restrições:

Caso o personagem chegue a um estado onde é possível atacar o adversário, ele poderá optar por qual tipo de ataque (magia, ataque

físico) de acordo com as características do adversário. Além disso em hipótese algum o aliado atacará um outro aliado.

4.3.1 Correspondência entre a técnica e o jogo analisado

As características citadas na seção 3.5 e 4.3 são relacionadas na Tabela 3.

Tabela 3 - Características do jogo *Astorian Tears*

Características da técnica SMA	Características do jogo analisado
Permite a interação e cooperação entre os agentes.	O jogo é composto por diversos personagens, como: guerreiros, magos, arqueiros, dragões, monstros, animais, etc Os personagens são representados por agentes que interagem entre sí como adversários ou aliados (cooperam).
Autonomia: Os agentes devem ser autônomos para realizar tomadas de decisões.	Os personagens podem tomar a decisão de realizar um ataque a um outro personagem específico (Ex: o que possui menor quantidade de pontos de vida..
Restrições: Devem existir algumas restrições referente ao comportamento em conjunto dos	Os aliados ao personagem do jogador devem atacar os inimigos

agentes para que haja um comportamento grupal coeso;	para defender o grupo aliado.
Viabiliza sistemas adaptativos e evolutivos.	É possível adquirir pontos de experiência que podem melhorar o ataque e defesa.
Características: Percepção, Ação, Comunicação, Representação, Motivação, Deliberação, Raciocínio e aprendizagem;	O personagem é capaz de perceber o território e agir de acordo com o mesmo. Além disso a comunicação com os outros personagens aliados a fim de derrotar o inimigo.
É aplicável a jogos computacionais para aumentar o realismo dos personagens; já que estes devem ser inteligentes e capazes de se adaptar ao jogo conforme a interação dos personagens.	Percepção do território a qual se encontra (Ex. Mais elevado, mais baixo) e adotar uma estratégia de ataque.
A inteligência é aplicada não apenas a um agente e sim ao grupo todo;	O grupo todo dos agentes/personagens tem o mesmo objetivo que é derrotar o inimigo.
Podem ser utilizados em Sistemas distribuídos e Sistemas abertos;	A distribuição das tarefas (derrotar o inimigo) pode ser realizada na hora de atacar o oponente.

Com este comparativo podemos inferir que é possível aplicar a técnica de SMA ao jogo *Astorian Tears* devido a compatibilidade das características do jogo com a técnica; apresentada na Tabela 3.

5. CONSIDERAÇÕES FINAIS

A utilização de inteligência artificial em jogos computacionais agrega reais melhorias a estes softwares porque é possível aumentar a experiência e imersão do jogo. Neste trabalho, buscou-se as informações referente as técnicas de Inteligência Artificial com intuito de propiciar aos leitores um subsídio para que estes ampliem seu conhecimento na área para poder determinar qual a técnica – das três estudadas – possa ser aplicada ao seu jogo.

Ao longo do estudo, foi possível constatar que cada técnica possui várias características que devem ser levantadas para aplicá-las num jogo específico como por exemplo, a correspondência entre as técnicas e as características do jogo.

A técnica de Redes Neurais Artificiais é utilizada para problemas que simulam o comportamento humano nos jogos, por exemplo, um personagem do jogo que é autônomo e precisa tomar a decisão para vencer um adversário. Neste personagem terá que ser implementado o treinamento da rede neural para auxiliar este personagem na escolha da melhor ação durante o jogo. A entrada dos dados deste treinamento da RNA será alimentada com as diferentes situações que o personagem enfrentará no jogo e quanto maior o volume de dados para treinar a rede neural, consequentemente mais inteligente serão as decisões futuras deste personagem.

A técnica de lógica *Fuzzy* é utilizada quando os problemas não são bem definidos num jogo, por exemplo, situações onde o personagem tem dúvidas quanto a mensurar os dados de entrada para tomar a decisão. Os dados de entrada variam entre 0 e 1, quente ou frio, rápido

ou devagar, alto ou baixo, etc. O personagem terá que calcular o grau de pertinência destes dados para tomar a decisão mais adequada para as ações futuras.

A técnica de Sistema Multiagentes é utilizada quando o problema é distribuído e precisa da cooperação e interação entre os agentes para atingir o objetivo do jogo. Por exemplo, um personagem no jogo que é representado por um agente cognitivo que analisa sua situação coletiva (adversários e/ou aliados) para tomar a decisão. A autonomia do personagem é de sua importância para a mecânica do jogo e a cooperação é importante para solução coletiva e distribuída do problema.

Neste trabalho foi escolhida a técnica de Sistema Multiagentes a ser aplicada ao jogo *Astorian Tears* devido as suas características distribuídas. Neste jogo existe um conjunto de personagens autônomos que devem interagir para atingir um objetivo comum que reforça a necessidade de utilizar SMA; onde os agentes podem ser cada personagem do jogo.

Por fim, é imprescindível analisar em cada jogo as características para decidir qual a melhor ou melhores técnicas de Inteligência Artificial a serem aplicadas.

5.1 Proposta para trabalhos futuros

Nesta seção são listadas algumas propostas para continuação deste trabalho por outros alunos ou pesquisadores interessados pelo tema.

- Implementar o jogo proposto neste trabalho com os Sistemas Multiagentes;
- Aplicar em um estudo de caso onde engloba diversas técnicas, como por exemplo, a técnica hibrida *Neuro-Fuzzy*;
- Definir métricas para a comparação das técnicas a partir das características definidas neste trabalho;
- Analisar as características de outras técnicas de Inteligência Artificial utilizada nos jogos, como Algoritmos Genéticos, Raciocínio Baseado em Casos, etc.

REFERÊNCIAS

BATTAIOLA, A. L. **Jogos por Computador – Histórico, Relevância Tecnológica e Mercadológica, Tendências e Técnicas de Implementação.** São Carlos, 2000. 42p. Disponível em: <http://projetofinal1.googlecode.com/svn/trunk/refer%C3%AAncias/material%20bruto/JogosComputadores.pdf> Acesso em: 25 ago. 2013.

BOSCATTO, A. B.; NAKAMITI, G.; TOLEDO, C. T. In. SBGames. 11., 2012, Campinas. ***Fuzzy* River Raid – Uma abordagem adaptativa para navegação Autônoma.** Brasília, 2012. Disponível em: <http://sbgames.org/sbgames2012/proceedings/papers/computacao/comp-short_11.pdf> Acesso em: 22 set. 2013.

BRAGA, A d. P. et al. **Redes neurais artificiais.** 2ª Ed. Rio de Janeiro: LTC, 2007.

CANELLA B.F. *et al.* **Estratégias Táticas para um Jogo Medieval Astorion Tears.** Laboratório de Tecnologias Computacionais da UFSC: Relatório Técnico, Araranguá: 2013.

CARNEIRO, L. E. S. SBGames. 7., 2008, Pernambuco. **Inserindo Incerteza subjetiva na Inteligência Artificial dos NPC's.** Belo Horizonte: SBC, 2008. Disponível em: <http://www.sbgames.org/papers/sbgames08/ad/papers/p09.pdf> Acesso em: 19 nov. 2013.

CARNIELLO, Adriana; CARNIELLO, Andréia. **Redes Neurais e Jogos.** Disponível em: < http://www.din.uem.br/ia/jogos/redes.htm > Acesso em: 10 nov. 2013.

CONCEIÇÃO, Daniel Tré da. **Redes Neurais Artificiais Aplicadas ao Jogo da Velha 3D em Pinos.** 2009. 85 f. TCC (Graduação) - Curso de Tecnólogo em Informação e Comunicação, Instituto Superior de Tecnologia em Ciência da Computação de Petrópolis, Petrópolis, 2009. Disponível em: <http://www.lncc.br/~borges/doc/Redes Neurais Artificiais Aplicadas ao Jogo da Velha 3D em Pinos.TCC.pdf>. Acesso em: 22 nov. 2013.

FERBER, Jacques. **Multi-Agent Systems**. Great Britian. Pearson: 1999.

GOMES, D. d. S. Inteligência Artificial: Conceitos e Aplicações. In. **Revista de publicações da Faar** nas áreas de Psicologia, Informática, Administração, Direito, Farmácia e Enfermagem. Vol. 1, nº 2. Ago/Dez 2010. Disponível em: <http://www.olharcientifico.kinghost.net/index.php/olhar/article/viewFile/49/37 > Acesso em: 28 mai. 2013.

GOMIDE, F. A. C.; GUDWIN, R. R.; TANSCHEIT, R. In International *Fuzzy* Systems Association World Congress-IFSA95, Tutorials. 6., 1995. **Conceitos fundamentais da teoria de conjuntos *Fuzzy*, lógica *Fuzzy*, e aplicações.** Campinas, 1995. 38p. Disponível em: <ftp://calhau.dca.fee.unicamp.br/pub/docs/gudwin/publications/ifsa95.pdf>. Acesso em: 10 out. 2013.

HÜBNER, J. M.; BORDINI, R. H; VIEIRA, R. **Introducão ao Desenvolvimento de Sistemas Multiagentes com *Jason*.** In Fernando

GOMES, J. E. A. *et. Al.* In. Simpósio Brasileiro de Qualidade de Software. 6., 2011. **Adoção de Metodologias Ágeis para Produção de Jogos Sociais com Times Distribuídos.** Recife, 2011. 8 p. Disponível em: <http://www.researchgate.net/publication/236784175_Adoo_de_Metodologias_geis_para_Produo_de_Jogos_Sociais_com_Times_Distribudos> Acesso em: 27 out. 2013.

KISHIMOTO, André. **Inteligência Artificial em Jogos Eletrônicos.** Disponível em: <http://www.programadoresdejogos.com/trab_academicos/andre_kishimoto.pdf > acesso em: 28 mai. 2013.

LIMA, R. ***Scrum*? O que é? Como funciona?** Inovatividade. Rio de Janeiro, 16 jul. 2009**.** Disponível em: < http://www.inovatividade.com/metodologias/*Scrum*-o-que-e-como-funciona > Acesso em: 18 nov. 2013.

NEMES, Ana. 4 programas que ajudam no desenvolvimento de games. **TecnoMundo**. São Paulo, 8 out. 2012. Disponível em: <http://www.tecmundo.com.br/como-fazer/31025-4-programas-que-ajudam-no-desenvolvimento-de-games.htm#ixzz2j2avshtw> Acesso em: 28 out. de 2013.

OSÓRIO, F. et al. In. SBGames, 6., 2007, São Leopoldo. **Inteligência Artificial para Jogos: Agentes especiais com permissão para matar... e raciocinar!** São Leopoldo, 2004. 20p. Disponível em: <http://osorio.wait4.org/publications/Osorio-et-al-SBGames07-Tutorial.pdf> Acesso em: 16 ago de 2013.

OSÓRIO, F.; VIEIRA R. In. Encontro Nacional da Inteligência Artificial. 11, Rio dos Sinos. **Sistemas Híbridos Inteligentes.** Rio de Janeiro: SBC, 1999.

Disponível em: < http://osorio.wait4.org/oldsite/enia99/enia99.pdf > Acesso em: 19 nov. 2013.

PASSOS, Ketry Gorete Farias dos. **O FLUXO DE INFORMAÇÃO NO PROCESSO DE DESENVOLVIMENTO DE JOGOS ELETRÔNICOS.** 2012. 223 f. Dissertação (Mestrado em Ciência da Informação), Universidade Federal de Santa Catarina, Florianópolis, 2012. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/100760/308890.pdf?sequence=1>. Acesso em: 22 ago. 2013.

PEREIRA, G. A. **Projeto de desenvolvimento de Jogos Computacionais.** 2006. 155p. TCC (Graduação) - Curso de Graduação em Ciência da Computação, Universidade do Estado de Santa Catarina, Joinville, 2006. Disponível em: <http://www.pergamum.udesc.br/dados-bu/000000/000000000002/0000020E.pdf>. Acesso em: 22 nov. 2013.

RABIN, Steve. **Introdução ao desenvolvimento de games, vol. 2.** São Paulo: Cengage Learning, 2012.

RIEDER, R.; BRANCHER, J. D. CONGRESSO IBEROAMERICANO DE INFORMÁTICA EDUCATIVA, 7., 2004, Monterey. **APLICAÇÃO DA LÓGICA *FUZZY* A JOGOS DIDÁTICOS DE COMPUTADOR – A EXPERIÊNCIA DO MERCADÃO GL 1.** Erechim: ., 2004. 10 p. Disponível em: <http://www.niee.ufrgs.br/eventos/RIBIE/2004/comunicacao/com127-136.pdf>. Acesso em: 10 nov. 2013.

RICH, E.; KNIGHT, K. **Inteligência Artificial**. 2.ed. São Paulo: Makron Books, 1994.

RODRIGUES, R; ROST, R. **Metodologia para desenvolvimento ágil de software.** Disponível em: <http://rafaelrgi.files.wordpress.com/2007/11/*Scrum*.pdf> Acesso em: 27 de outubro de 2013.

ROINSENBERG, M. Redes Neurais Artificiais. Disponível em: < http://www.inf.ufsc.br/~mauro/ine6103/slide/cursoredesneurais/sld281.htm>. Acesso em: 21 nov. 2013.

ROSA, J. L. G. **Fundamentos da Inteligência Artificial**. Rio de Janeiro: LTC, 2011. il. ; 24cm.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. 2.ed. Rio de Janeiro: Elsevier, 2004.

SANTOS, R. A.; GÓES, V. A.; ALMEIDA, L. F. d. SBGames. 11., 2012, Brasília. **Metodologia *OriGame*: um processo de desenvolvimento de jogos.** Taubaté: Sbc, 2012. 7 p. Disponível em: <http://sbgames.org/sbgames2012/proceedings/papers/artedesign/AD_Full16.pdf>. Acesso em: 31 out. 2013. (santos goes e Almeida)

SEBRAE. **Brasil tem o maior mercado de games no mundo em 2012.** Disponível em: <http://www.sebrae2014.com.br/Sebrae2014/Alertas/Brasil-tem-o-maior-mercado-de-games-no-mundo-em-2012#.Ug6DMtLFXwx> Acesso em: 16 de agosto de 2013.

SILVA, E. L. d; HOUNSELL, M. d. S.; KEMCZINSKI, A. Metodologia Maiêutica: **Uma proposta metodológica para desenvolvimento de Ambientes Virtuais 3D.** Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/svr/2007/0055.pdf> Acesso em: 26 nov. 2013.

SOUZA, M. d. **Influência dos jogos no campo da Inteligência Artificial.** Florianópolis: 2011. Disponível em: <http://www.ceavi.udesc.br/arquivos/id_submenu/387/marcelo_de_souza.pdf>. Acesso em: 18 nov. 2013.

TAKASHI, I. et. al., editor, *XII Escola de Informática da SBC - Paraná*, chapter 2, pages 51-89. Ed. da UNICENTRO, Guarapuava, PR, 2004. Disponível em: <http://www.das.ufsc.br/~jomi/pubs/2004/Hubner-eriPR2004.pdf> Acesso em 5 nov. 2013.

TATIBANA, Cássia Yuri; KAETSU, Deisi Yuri. **Uma introdução a Redes Neurais.** Disponível em: <http://www.din.uem.br/~ia/neurais/>. Acesso em: 20 set. 2013.

TELES, V. **Extreming Programing XP.** Disponível em: <http://desenvolvimentoagil.com.br/xp/> Acesso em: 26 nov. 2013.

VALENTE, Joana Vieira; RIO, Simão Cardoso Espinheira. **Implementação de um Sistema Multi-Agente para um Jogo Estratégico de Palavras.** Porto: 2005. Disponível em:

<http://paginas.fe.up.pt/~eol/AIAD/TRABALHOS_ANT/SMAjogo_0506.pdf>. Acesso em: 9 nov. 2013.

VASCONCELOS, G. C. **O modelo de HopField.** Disponível em: < http://www.cin.ufpe.br/~gcv/web_lci/Aula-Hopfield.pdf> Acesso em: 20 nov. 2013.

WINSTON, P. H. **Artificial Intelligence.**3 ed. Reading, MA: Addison-Wesley, 1992.