Um Jogo de Estratégia Aplicando Técnicas de Inteligência Artificial

Franthiesca V. da Silva e Diana F. Adamatti

Abstract— **This work deals with the utilization of Artificial Intelligence (AI) applied to Electronic Games. Firstly, it presents a brief introduction of eletronic games development. In a second moment, the definition and comparison of AI and Game AI, in the scope of this work, are presented. To exemplify and apply the acquired knowledge, a prototype of a strategy game was developed using the AI techniques.**

***Index Terms*— Artificial Intelligence, Game AI, Real-Time Strategy**

I. INTRODUÇÃO

NA academia, a área de Inteligência Artificial (IA) é um tema "rico" para estudo e pesquisa. Na área de jogos, é o que os deixa mais perto da realidade, que torna as personagens, sejam elas pessoas, animais, carros ou aviões, etc., mais verossímeis.

Mas existem diferenças entre a IA Acadêmica (ou Tradicional) e a IA para jogos (Game AI). E a principal diferença entre elas é o que cada uma busca: a acadêmica visa principalmente buscar solução para problemas difíceis e tenta imitar e entender alguns comportamentos humanos através de agentes inteligentes. Já na IA para jogos o objetivo é a diversão, e o que importa são as impressões do usuário quanto ao resultado gerado [19]. Para a Game AI não importa a resposta dada pelo sistema inteligente nem como ele funciona internamente. O que interessa é como o sistema atua, e não o que ele pensa [22].

O principal objetivo deste artigo é desenvolver um jogo de estratégia utilizando técnicas tradicionais de IA, tornando-o inteligente e de alto nível para o usuário.

Este artigo está dividido em 6 seções. Na seção 2 são apresentados os jogos eletrônicos. A seção 3 trata sobre a Inteligência Artificial e as principais técnicas aplicadas a jogos. Na seção 4 está a proposta desse trabalho, apresentando o jogo propriamente dito. Já a seção 5 trata dos detalhes de implementação do jogo. E finalmente, na seção 6 estão as conclusões do trabalho e propostas de trabalhos futuros.

Franthiesca V. da Silva é mestranda do Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal do Rio Grande (PPGMC/FURG). E-mail: thiescavaladao@gmail.com

Diana F. Adamatti é professora doutora do Centro de Ciências Computacionais e do Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal do Rio Grande (C3/PPGMC/FURG). E-mail: dianaadamatti@furg.br

II. JOGOS ELETRÔNICOS

A. Definição de Jogo

"Um jogo é uma atividade ou ocupação voluntária (podendo ser física ou mental), exercida dentro de limites determinados de tempo e espaço, segundo regras livremente criadas, mas absolutamente obrigatórias, dotado de um fim em si mesmo, e cuja finalidade é recreação" [9].

Um jogo eletrônico é um jogo concebido em formato de software, conectado a um dispositivo de vídeo e outro dispositivo de entrada de dados, que permite ao jogador interagir com ele Weiszflog,2004 apud [6].

B. História dos Jogos Eletrônicos

O precursor dos jogos eletrônicos foi, em 1958, William Hinginbothan que, utilizando um osciloscópio, criou um protótipo de um jogo de tênis, chamado Tennis for Two. O primeiro jogo considerado eletrônico, de fato, foi o *Spacewar!*, um jogo de guerra espacial desenvolvido por Steve Russel, estudante do Massachusetts Institute of Technology (MIT), em 1961 [4].

Em 1967, Ralph Baer fez o primeiro jogo interativo para televisão, chamado *Chase* [1]. Em 1970, a Magnovax comprou a idéia de Baer e baseou-se nela para criar o vídeo game *Odyssey*, considerado o primeiro console da história, dois anos mais tarde [7]. Ainda em 1970, o jogo *Spacewar!* começou a ser implementado em uma versão para fliperama por Nolan Bashnell, chamada de *Computer Space*. A empresa Nutting Associates comprou o jogo e em 1972 aparece o primeiro fliperama da história. Nesse ano, Bashnell junto com Ted Dabney, funda sua própria empresa, a *Atari*, que em seu primeiro ano, com ajuda do engenheiro Al Alcorn lança o famoso jogo *Pong* [10], [1]. Nos anos seguintes, várias outras empresas seguem o exemplo da Atari e entram no mercado, entre elas estão a Taito, Midway e Namco.

Em 1976, a empresa Farchild lança o primeiro console que utilizava cartuchos, o Channel F. Foi um dos primeiros sistemas eletrônicos a usar o recém inventado *microchip*.

Ainda nesse ano a Warner Communications compra a Atari. Em 1977, é lançado o Studio II, da RCA, com quatro jogos na memória principal e jogos extras em cartuchos; é lançado o Atari 2600 [21], [1] e Shigeru Miyamoto (criador do Mario e do Donkey Kong) entra para a Nintendo [7]. No fim da década de 70 são lançados vários jogos que ficaram famosos, como: *Football*, *Lunar Lander* e *Asteroids* da Atari e *Space Invaders* da Midway. Em 1979, surge a empresa Capcom.

Os anos 80 foram o auge dos jogos de fliperamas. E foi quando surgiram os primeiros consoles de 8-bits: Famicon, da Nintendo, conhecido como NES (Nintendo Entertainment System) e Master System, da SEGA. Além do lançamento dos jogos Donkey Kong, Pac-man e Tron, foi fundada a empresa Activision e iniciou-se movimentação no setor de jogos para computador com a On-Line Systems.

Em 1983, devido ao grande número de jogos ruins no mercado e o lançamento dos computadores pessoais aconteceu a "*Crash* dos Jogos", crise que afetou e levou à falência grande parte das empresas de videogames. A Atari, fortemente afetada pela crise reage com o console Atari 7800. No final da década, o mercado de jogos começa a se erguer novamente. Em 1989, a SEGA apresenta o Mega Drive, primeiro console de 16 bits e a Nintendo lança o Game Boy, primeiro videogame portátil [21].

A década de 1990 começa com o *Super Mario Bros. 3*, o jogo de cartucho mais vendido no mundo e a Nintendo surge com o sucessor do NES, o SNES (Super Nintendo Entertainment System). No ano seguinte a SEGA apresenta o primeiro jogo do personagem Sonic [4]. E as duas empresas se tornam líderes do mercado. [21] Nesse ano é lançado o primeiro jogo em 3D para computador, o *Wolfstein 3D*.

Nos anos seguintes, segue-se a batalha dos bits. Para concorrer com o SNES, de 16-bits, a SEGA lança o Sega Saturn de 32-bits. Após, em 1994, surge o PlayStation, da Sony, que revolucionou com gráficos e jogabilidade superiores. Em 1996, a Nintendo apresenta o Nintendo 64, com 64-bits, trazendo franquias já existentes, tais como Mario, Zelda e Donkey Kong Villiegas,2000 apud [21].

A década de 2000 já começa com o lançamento do PlayStation 2 da Sony, "quebrando" as vendas do Dreamcast da Sega, lançado no ano anterior. O PlayStation 2 tinha 128-bits e usava como mídia CD ou DVD. Com o fim do Dreamcast, a Sega passou apenas a produzir jogos [21], [4].

Nessa época, a Microsoft decide entrar na competição e coloca seu console Xbox na disputa [10]. E, em seguida, a Nintendo lançou seu novo console portátil, o Game Boy Advance. Em 2004, surge outro portátil da empresa, o Nitendo Dual Screen ou Nintendo DS e o portátil da Sony, o PSP [4].

Atualmente estão concorrendo fortemente no mercado, o PlayStation 3 da Sony, o Xbox 360 da Microsoft, com seu inovador acessório, o Kinect, e o Nintendo Wii, o primeiro console que trouxe a possibilidade de jogar com os movimentos do corpo. Os consoles da próxima geração já estão em desenvolvimento e, em um futuro próximo, serão lançados o sucessor do Xbox 360 e o PlayStation 4.

C. Tipos de Jogos

Existem diversas divisões para os jogos. Aqui, apresenta-se a mais comumente utilizada.

Jogos de Ação

Em geral, neste gênero, o jogador emerge em um mundo caótico onde seu objetivo é a sobrevivência para alcançar o final do jogo [13]. Neste gênero estão incluídos os jogos de Tiro, sejam eles em primeira (do inglês *First Person Shooter* ou FPS) ou terceira pessoa e os jogos de luta.

Nos FPS, o jogador é colocado na perspectiva do personagem, atrás de uma arma, na tela só aparecem as mãos do personagem [22]. Podem ser citados como exemplo os jogos Doom, Call of Duty: Modern Warfare 3 e Battlefield Bad Company 2. Em jogos de terceira pessoa (*Third-Person Shooter* ou TPS), segundo Vieira (2005) [22] a perspectiva normalmente é atrás do jogador. São exemplos de TPS a série Resident Evil e a série Max Payne. O sub-gênero de luta enfatiza o combate *"corpo-a-corpo"* entre dois jogadores. Em geral, são baseados em artes marciais como os jogos da série Mortal Kombat.

Jogos de Aventura

É uma estória interativa em que o jogador é colocado no papel do personagem principal e ele deve superar obstáculos para alcançar um determinado objetivo. Focam na resolução de quebra-cabeças e em exploração e coleção de objetos [3], [13]. Alguns exemplos de jogos de aventura são Super Mario e Prince of Persia.

Jogos de Estratégia

Jogos de estratégia enfatizam habilidades de pensamento e planejamento para alcançar a vitória. Este é o tipo de jogo que estamos focando neste trabalho, por isso ele será mais detalhado na próxima seção.

Jogos de Simulação

Tem como objetivo simular atividades ou comportamentos da vida real. Abrange um grande número de sub-gêneros, tais como: simuladores de corrida, de voo e de cidades, gerenciamento e construção, e até simuladores de vida real [13], [22]. Alguns exemplos desse tipo são: Fligth Simulator, Need For Speed: Shift, The Sims 3 e Sim City.

Jogos de Interpretação de Personagem

Jogos de Interpretação de Personagens, ou RPG (do inglês, *Role-playing game*), tem origem nos jogos de mesa e tabuleiro que tem como base a criatividade e a cooperação entre os jogadores.

Neste gênero o jogador também assume o papel do protagonista, mas agora em uma estória bem mais complexa em um mundo *aberto*, onde pode-se andar livremente para qualquer direção desejada, e onde as ações do jogador influenciam no rumo da história, podendo ela ter vários finais diferentes. Em geral há muita interação com os NPC (*Non-Player Characters*) e é possível customizar a aparência e

habilidades do protagonista. Tem como uma das principais características a evolução do personagem ao longo do jogo, que pode ganhar novas habilidades, armas, poderes, etc. [13], [3], [5]. Alguns exemplos de jogos desse tipo são The Legend of Zelda, Diablo II e Dragon Age: Origins.

Jogos de Esporte

São jogos que simulam esportes do mundo real [13]. Neste gênero, geralmente o mundo do jogo e as regras exercidas neste mundo são bastante conhecidos dos jogadores [3]. São exemplos: Pro Evolution Soccer e NBA.

Jogos On-Line

Jogos on-line na verdade englobam todos os outros tipos de jogos. O que os diferencia é poder jogá-los através da internet podendo interagir com outros jogadores que também são reais [13]. Atualmente há uma grande popularização desse gênero.

D. Jogos de Estratégia

O conceito de estratégia originou-se das estratégias militares e hoje é aplicado nas mais diversas áreas, desde o entretenimento até o contexto de negócios [6]. A estratégia militar lida com o planejamento e condução de campanhas, o movimento e divisão de forças, cooperativismo, entre outras características.

Em jogos de estratégia, em geral, o jogador controla um exército e precisa usar sua estratégia militar e sua habilidade de coletar e gerenciar recursos para tomar ou defender territórios, ou destruir inimigos, de forma a alcançar a vitória [13], [22].

Jogos de estratégia podem ser de dois tipos: baseados em turno ou em tempo real. Nos baseados em turno, cada jogador tem um determinado tempo para montar seu planejamento e executar suas ações. Um jogando de cada vez, tal como na maioria dos jogos de tabuleiro, como xadrez e o jogo WAR. São exemplos de jogos baseados em turno: Civilization e Total War. Já em jogos em tempo real, conhecidos como RTS (*Real-Time Strategy*), a todo o momento o jogador tem que pensar e reavaliar suas estratégias e executar suas ações. Todas as ações do jogador e dos adversários ocorrem ao mesmo tempo [22], [15]. Alguns exemplos de RTS's são os jogos da série Age of Empires, Stronghold e Starcraft.

III. INTELIGÊNCIA ARTIFICIAL

A. Definição de IA

É difícil encontrar um consenso entre os autores sobre o que é Inteligência Artificial (IA). O dicionário Oxford [23] diz que a IA corresponde a uma área de pesquisa sobre computadores simulando o comportamento inteligente. Já, segundo Millington (2006), é a capacidade dos computadores de executarem tarefas que humanos e animais podem realizar.

B. IA e Jogos Eletrônicos

Atualmente os jogadores querem ser desafiados e querem que a cada jogada tenham uma experiência nova e única. Os oponentes tem que ser atrativos e devem ter comportamentos verossímeis. Por isso, é cada vez mais necessário o uso de IA em jogos digitais. A Tabela 3.1 apresenta a utilização da IA em jogos eletrônicos.

Tabela 3.1: Linha de Tempo do uso de IA em Jogos [19], [2].

IA Utilizada	**Descrição**	**Ano**
Sem IA	Spacewar! – Primeiro jogo de computador. Para 2 jogadores	1962
	Pong – Versão eletrônica do tênis de mesa	1972
Padrões	Inimigos começam a aparecer	1974
	Space Invaders – Inimigos são padronizados, mas atiram de volta. Considerado o primeiro jogo "clássico" com níveis, score 47, controles simples a dificuldade crescente no decorrer do jogo	1978
	Pac-Man – Fantasmas com movimento padronizado, mas cada fantasma possui uma personalidade	1980
	Um microcomputador vence um jogador profissional de xadrez pela primeira vez	1983
	Karate Champ (Data East, 1984) – um dos primeiros jogos de luta um contra um, com o computador como adversário	1984
Máquinas de Estado Finito	Herzog Zwei (TechnoSoft, 1990) - O primeiro RTS a surgir e o mundo experimenta pela primeira vez uma péssima implementação do algoritmo de pathfinding.	1990
	Doom (id Software, 1993) – início oficial da era dos FPS	1993
Várias Técnicas	BattleCruiser: 3000AD (Take Two Software, 1996)– Primeiro uso de redes neurais em um jogo comercial	1996
	Deep Blue – derrota o atual campeão de xadrez Gary Kasparov	1997
	Half Life – A inteligência artificial em jogos encontra-se em seu auge. O jogo faz grande uso de linguagens de script.	1998
	Black & White (Lionhead Studios, 2001) – Utiliza criaturas que usam aprendizado por reforço e observação.	2001
	F.E.A.R – Utiliza técnica de Plannig	2005

Neste contexto, funde-se um pouco da IA Tradicional com a Game AI. Muitas das técnicas contidas na Tabela 1 são de IA Tradicional, porém sua implementação cria comportamentos apropriados somente dentro do contexto do jogo Tozour,2002 apud [10]. Por isso, elas podem ser consideradas Game AI.

Muitas outras técnicas utilizadas pelos desenvolvedores de jogos não são consideradas técnicas de IA. Porém especialistas e acadêmicos concordam que houve uma rápida evolução no desenvolvimento de IA na indústria de jogos e que, em termos de resultados e soluções práticas para certos problemas, ela parece estar muito à frente do meio acadêmico. Novas técnicas de IA Tradicional podem levar anos sendo formuladas e pode-se levar mais tempo ainda criando estudos formais para prová-las Woodcock,1999 apud [10].

Jogos recentes cada vez mais começam a utilizar de técnicas mais avançadas de IA para deixar seus jogos mais atrativos. E o uso frequente de outras técnicas de Game AI, como scripts para determinar comportamento, talvez façam com que no futuro ela seja mais bem aceita pela área acadêmica.

C. *Técnicas de IA Aplicadas a Jogos*

Máquinas de Estado Finito

Máquinas de Estado Finito, do inglês *Finite State Machines* (FSM), também conhecidas como Autômatos Finitos, são estruturas lógicas compostas por um conjunto finito de estados e regras de transição entre estes estados [18].

FSM não é considerada uma técnica de IA, tradicionalmente, mas pode sim ser considerada uma técnica de inteligência artificial dentro da Game AI.

Sipser (2007) [20] apresenta uma definição formal para as Máquinas de Estado Finito. Uma FSM é uma 5-tupla (Q, Σ, δ, q_0, F), onde:

1. Q é um conjunto finito conhecido como os **estados**,
2. Σ é um conjunto finito chamado o **alfabeto**
3. δ: Q x Σ → Q é a **função de transição**,
4. $q_0 \in Q$ é o **estado inicial**, e
5. F "que está contido ou é igual a" Q é o **conjunto de estados de aceitação**.

A função de transição é responsável por mapear o estado atual da FSM e um dado de entrada pertencente ao alfabeto para um novo estado. A cada interação um dado de entrada é lido, se houver uma função de transição correspondente com o estado atual e a entrada, obtem-se o novo estado. Esse processo continua até que não haja mais estados de entrada para serem lidos ou chegue-se ao estado de aceitação.

No contexto de jogos, as FSM são usadas para definir os estados em que um personagem pode se encontrar e situações que o farão mudar de estado. O personagem apresentará um determinado comportamento dentro do jogo a partir de seu estado atual [10]. Ações e comportamentos estão associados a cada estado [12]. Além disso, a FSM não necessita reconhecer a linguagem, pois seu objetivo é apenas gerar o comportamento dos personagens [15].

Lógica Fuzzy

Na Lógica Fuzzy é possível mapear informações imprecisas ou vagas da mesma forma como os seres humanos costumam fazer [14], [5]. Por exemplo, se não se sabe ao certo a altura de alguém pode-se dizer que ela é *muito baixa*, *baixa*, *mais alta* ou *alta*.

É baseada na Teoria dos Conjuntos Fuzzy, em que é possível que um elemento pertença apenas parcialmente a um conjunto. Grau de Pertinência é o nome de um valor que diz o quanto um elemento pertence a um determinado conjunto [16], [18].

A Lógica Fuzzy permite que termos linguísticos, ou variáveis linguísticas, como *baixa*, *alta*, *muito longe*, etc., possam ser processados numericamente. Para tal, é necessário convertê-los em funções de pertinência, que atribuirão os graus de pertinência de cada termo de Barros e Bassanezi,2006 apud [11]. Na Figura 3.1, é possível ver o exemplo de uma função de pertinência, onde a variável temperatura apresenta dois termos linguísticos (valores fuzzy).

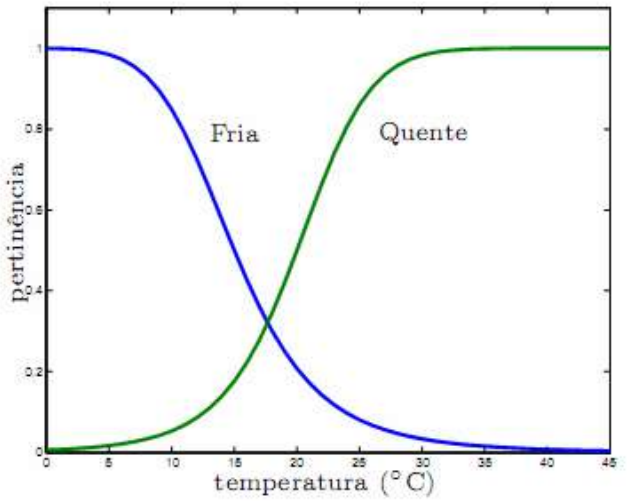


Figura 3.1: Exemplo de uma função de pertinência [14].

Máquinas de Estados Fuzzy

Pode-se usar a Lógica Fuzzy juntamente com as Máquinas de Estado, gerando então uma Máquina de Estados Fuzzy ou FuSM (do inglês, *Fuzzy State Machine*).

A definição formal de uma Máquina de Estados Fuzzy é igual a de uma Máquina de Estados Finitos. Mas, nela, as regras de transição podem conter valores fuzzy para determinar ou não a mudança de estado. Sendo assim, é possível ter uma máquina de estados em que os estados não estão só *ativos* ou *inativos*, mas também podem estar *muito ativos* ou *pouco ativos* [18], [17].

Como a Lógica Fuzzy permite situações intermediárias, uma FuSM pode não se encontrar exatamente em um único estado por vez, mas pode estar em vários estados ao mesmo tempo [18].

Um jogo com FuSM diminui a previsibilidade da tomada de decisão e torna cada NPC com comportamento mais individual.

Flocking

O Flocking é um algoritmo que foca em técnicas para coordenação de movimentos em grupo. É muito utilizado para gerenciamento do comportamento de rebanhos, bandos e bots em jogos de FPS e em jogos de estratégia [3].

No Flocking, cada membro do grupo precisa guardar informações de outros membros que estão a certa distância dele, dentro de um determinado raio. Este raio é o *raio de vizinhança* Buckland,2005 apud [5].

Com essas informações é possível seguir as 3 regras principais, ilustradas pela Figura 3.2, para implementação do algoritmo [19], [12]:

- **Separação:** os membros do grupo devem manter certa distância um do outro, de forma a evitar colisões.
- **Alinhamento:** o grupo deve mover-se junto, por isso é necessário que cada unidade do grupo, mantenha sua direção alinhada à direção dos membros vizinhos.
- **Coesão:** o grupo deve permanecer unido e o movimento de um membro deve se dar em relação à posição de seus vizinhos. Por isso, deve-se calcular a posição média da vizinhança e o resultado será a nova posição da unidade.

No Flocking, para seguir as regras, é necessário apenas ter o estado atual dos membros do grupo. A falta de necessidade de guardar informações anteriores para sua execução faz com que ele seja uma boa opção em relação à memória [3].

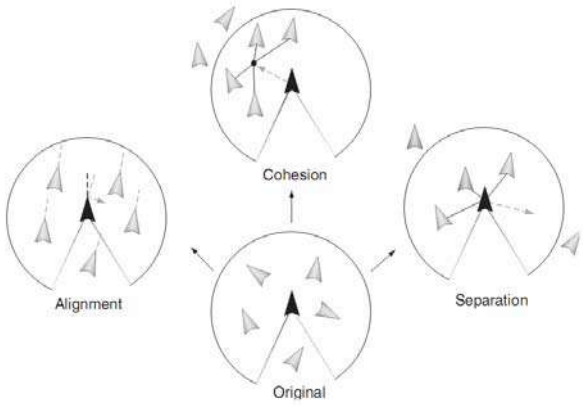


Figura 3.2: Três regras do Flocking [19].

IV. Proposta do Jogo

A ideia deste trabalho é desenvolver um jogo de estratégia, em tempo real, aplicando técnicas de Inteligência Artificial, em um ambiente gráfico 2D.

A. Definição do Jogo

O jogo é formado por duas equipes, controladas pelo computador: a Equipe A e a Equipe B, uma jogando contra a outra.

A estratégia baseia-se no controle de recursos e na descoberta do melhor momento para atacar o inimigo.

A primeira equipe que destruir todo o exército inimigo e destruir todos os edifícios centrais do adversário ganha o jogo.

B. Elementos do Jogo

Os elementos do jogo são: Soldado, Trabalhador, Recursos, Edifícios e Cenário.

Eles podem ser divididos em elementos de Ambiente ou Caracteres.

Elementos do Jogo

- Recursos: Os recursos são constituídos de **pedra, madeira** e **comida.** Eles são necessários para criação de novos edifícios e unidades, tanto soldados quanto trabalhadores. Eles podem ser extraídos dos seguintes quatro objetos presentes no cenário: Pedra, para obter pedra, Árvore para obter madeira, e Vaca ou Planta para obter comida.
- Edifícios: Existem dois tipos de edifícios, o **Edifício Central** e o **Quartel**. O quartel é onde podem ser criados novos soldados e o Edifício Central é onde ficam armazenados os recursos e onde é possível criar novos trabalhadores.
- Mapa: É onde todos os elementos aparecem e interagem. Alguns podem ser fixos como as Plantas e outros podem andar livremente como os Trabalhadores.

Caracteres

- Soldado: Os soldados são responsáveis pelas batalhas contra os inimigos. Ele podem manter suas posições, caminhar ao longo do mapa, realizar ataques, sejam eles individuais ou em grupo, e podem morrer.
- Trabalhador: Os trabalhadores são os elementos mais complicados e operantes do jogo. Eles são responsáveis pela coleta e armazenamento dos recursos, necessários para evolução do exército, pela construção e reparo de edifícios. Além disso, eles são capazes de fugir dos inimigos e até de se defender.

C. IA Utilizada no Jogo

Máquinas de Estado Finito

Dado um evento interno é possível transitar pelos estados presentes nas quatro Máquina de Estados Finitos desenvolvidas para o jogo: três gerais, uma para cada elemento principal do jogo, e uma para o grupo. No caso da FSM de grupo, é para um grupo de soldados.

A FSM dos Edifícios, visualizada na Figura 4.1, é a mais simples. Inicialmente um edifício pode ser construído, depois ele passa para o estado PRONTO. Desse estado é possível ficar num ciclo indeterminado de ser atacando e ser reparado, até o momento em que ele é totalmente destruído, chegando ao estado DESTRUIDO que eliminará o elemento Edifício do cenário, e a máquina é finalizada.

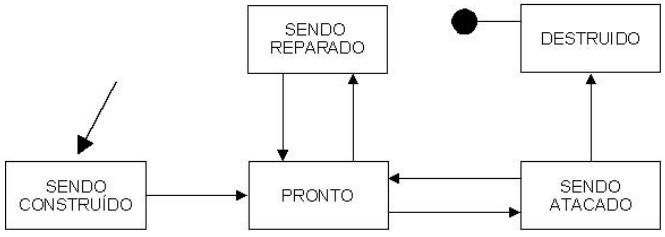


Figura 4.1: FSM dos Edifícios

A Figura 4.2 apresenta a FSM dos Soldados. Como pode ser visto, os soldados começam no estado NASCENDO, e logo depois de ser criado passa para o estado PARADO, onde ficará aguardando ocorrer algo que ative a transição e o leve a mudar de estado. Se ele detectar um inimigo próximo a ele, poderá passar para o estado ATACANDO, onde ele entrará em combate com um ou mais inimigos, sendo um inimigo qualquer trabalhador, soldado ou edifício do time oposto. Durante o combate, o soldado pode ter sua *vida* zerada, passando para o estado MORTO e sendo eliminado do cenário, ou pode vencer o combate voltando ao estado PARADO ou CAMINHANDO. Neste caso, por exemplo, ele pode vencer o combate e decidir voltar para perto de seus edifícios. Quando ocioso por muito tempo, o soldado pode decidir ir para o estado CAMINHADO e consequentemente mudar para uma nova posição.

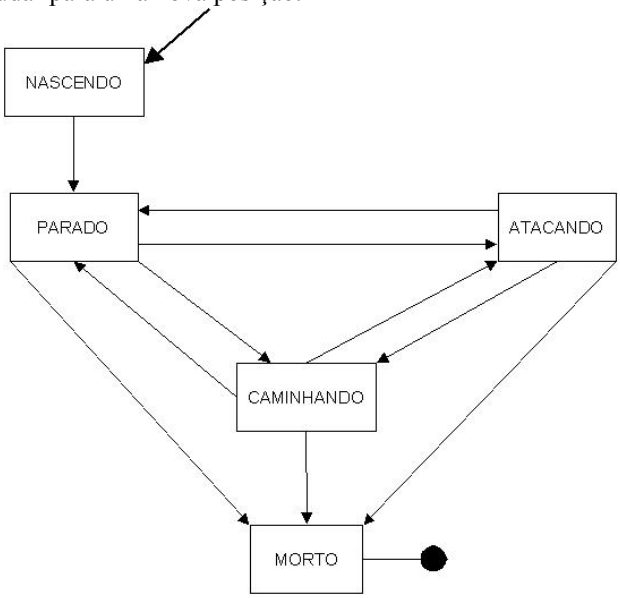


Figura 4.2: FSM dos Soldados

A FSM do trabalhador (Figura 4.3) também começa no estado NASCENDO. Quando parado, o trabalhador pode decidir buscar recursos. Neste caso, pode coletar e armazenar esses recursos, ou pode construir ou reparar um edifício, quando ele julgar necessário ou quando o edifício tiver sido atacado, mas não destruído, respectivamente. Em qualquer desses estados, com exceção do NASCENDO, o trabalhador pode perceber um inimigo próximo. Ele então tentará fugir. Caso o inimigo esteja muito próximo ele tentará atacar para se defender, mas inevitavelmente morrerá, pois sua força e vida são muito pequenas. Caso consiga fugir, voltará ao estado PARADO, onde avaliará novamente qual será sua decisão.

A FSM do grupo (Figura 4.4) é executada somente quando é necessário realizar um ataque em grupo, que se dará quando vários soldados estiverem mudando ao mesmo tempo para o estado ATACANDO. Depois de iniciada a FSM, será acionado um "aviso" para que os soldados possam juntar-se para executar o algoritmo de *flocking*. Eles, então, irão se locomover em direção aos inimigos e atacarão em conjunto.

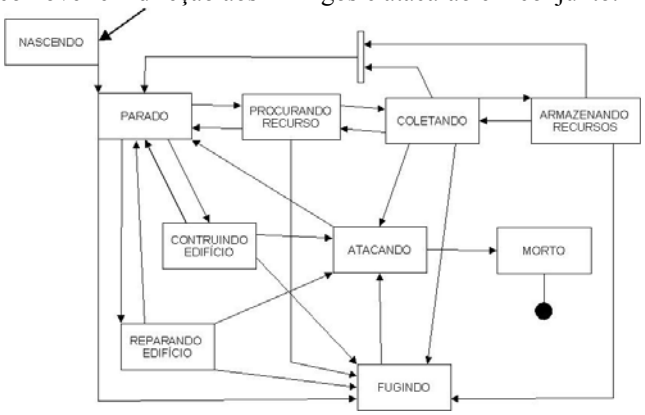


Figura 4.3: FSM dos Trabalhadores

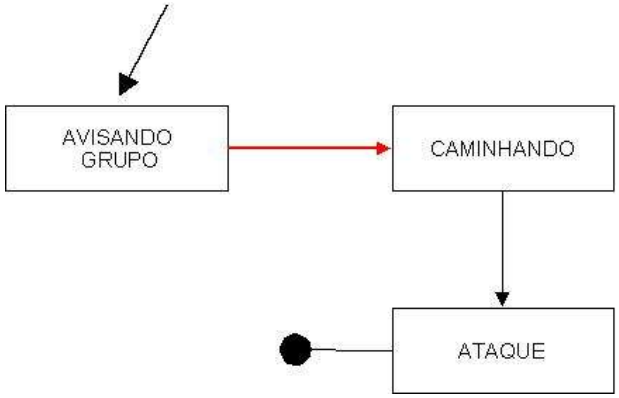


Figura 4.4: FSM do Grupo

Flocking

No jogo, o *flocking* é usado para gerenciar o movimento e o ataque em grupo executado pelos soldados. Quando vários soldados decidem atacar o adversário ao mesmo tempo é chamada a função de flocking que permitirá que eles façam uma movimentação ordenada e conjunta em direção ao alvo de ataque.

O *Flocking* é aplicado na máquina de estados do grupo, e é chamado logo depois que todos os soldados foram avisados do ataque em grupo. Esse momento pode ser visto na Figura 4.4, na seta em vermelho.

V. IMPLEMENTAÇÃO E TESTE DO JOGO PROPOSTO

A. *Tecnologias Utilizadas*

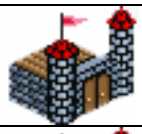
Para o desenvolvimento do jogo foi utilizada a linguagem de programação C++, juntamente com a biblioteca gráfica ALLEGRO, nos ambientes de desenvolvimento DEV-C++ e CodeBlocks. Foi adotada a programação orientada a objetos, a partir do uso de classes e objetos da linguagem.

ALLEGRO é uma biblioteca de software livre e código fonte aberto para desenvolvimento de jogos eletrônicos. É multi-plataforma e fornece suporte para gráficos 2D e 3D, manipulação de imagens, produção de texto, saída de áudio, manipulação de arquivos, entrada de dados pelo teclado e mouse, entre outros [8].

Para o desenvolvimento do jogo fora utilizadas várias funções básicas da biblioteca além da técnica de double buffering. A função *allegro_init*, por exemplo, é responsável por inicializar a biblioteca declarando algumas variáveis globais e reservando memória. São utilizadas também as funções *set_color_depth* e *set_gfx_mode*, responsáveis pelas configurações básicas de vídeo, *create_bitmap* e *load_bitmap* responsáveis por criar e carregar as imagens que serão usadas no jogo, e a função *draw_sprite*, que desenha as imagens carregadas na tela.

Para representar o cenário e elementos do jogo, juntamente coma a biblioteca ALLEGRO, foram utilizados arquivos de bitmap que estão listados na Tabela 5.1.

Tabela 5.1: Bitmaps do Jogo

Imagem	Descrição da Imagem
	Trabalhador da Equipe A
	Trabalhador da Equipe B
	Soldado da Equipe A
	Soldado da Equipe B
	Edifício Central Equipe A
	Edifício Central Equipe B
	Quartel Equipe A
	Quartel Equipe B
	Vaca
	Árvore

B. Classes Gerais do Jogo

As classes gerais do jogo são: *Ambiente, Soldado, Edificio, Trabalhador, Recurso, Pedra, Arvore, Vaca* e *Planta.* Estão representadas na Figura 5.1.

A classe *Ambiente* é onde tudo acontece. Todas as demais classes e principais funções de IA são executadas a partir da classe *Ambiente*. Nela é onde são inicializados todos os elementos gráficos referentes a cada classe e ao mapa do jogo.

Todas as classes, exceto a *Ambiente*, apresentam as variáveis **int x** e **int y** que são responsáveis por guardar as coordenadas de cada elemento dentro do jogo, seja ele fixo ou móvel. Já **int vida** e **int vida_max**, presentes nas classes *Trabalhador, Soldado* e *Edifício*, são responsáveis por guardar o valor da vida que o elemento tem no momento e a vida com a qual ele nasce, respectivamente. Caso a *vida* seja menor que a *vida_max*, poderá ocorrer alguma mudança de estado dos elementos, sendo que se ela chegar à zero, significa que o Trabalhador ou Soldado foi morto, ou o Edifício foi destruído, levando a sua exclusão do cenário.

Todas a classes que geram recursos possuem a variável **quantidade**, que diz qual a quantidade de recursos que pode ser extraída dela. Caso a quantidade chegue a zero o elemento desaparece do mapa.

C. Máquina de Estados Finito

Como dito na seção anterior, existem três máquinas de estados gerais e uma do grupo. Elas são representadas pelas seguintes funções:

- void percorre_estados_edificios();
- void percorre_estados_soldado()
- void percorre_estados_trabalhador()
- void percorre_estados_grupo();

As três máquinas de estado gerais são inicializadas logo depois dos elementos do jogo, e permanecem rodando até que o jogo seja finalizado. Já a FSM do grupo, válida para um grupo de soldados, só é executada quando determinada ação ocorre dentro do jogo.

Quando a máquina de estados do grupo é acionada, a máquina de estado de cada soldados é salva em seu estado anterior. Terminada a execução do grupo (*Flocking*), cada um volta a executar a sua máquina de estados, e assim por diante. Dentro de cada máquina de estado estão as demais funções de suma importância para a IA do jogo, tal como, detecção de inimigos e movimentação.

D. Flocking

O *Flocking* se baseia nos três conceitos explicados na seção III-C.

Como os bitmaps dos soldados não mudam, independentemente para qual lado ele está indo, para manter a direção é simplesmente passada a coordenada de destino para todos os soldados envolvidos no flocking. A coordenada de destino será o local onde ocorrerá o ataque e poderá se obtida através da função de detecção de inimigo.

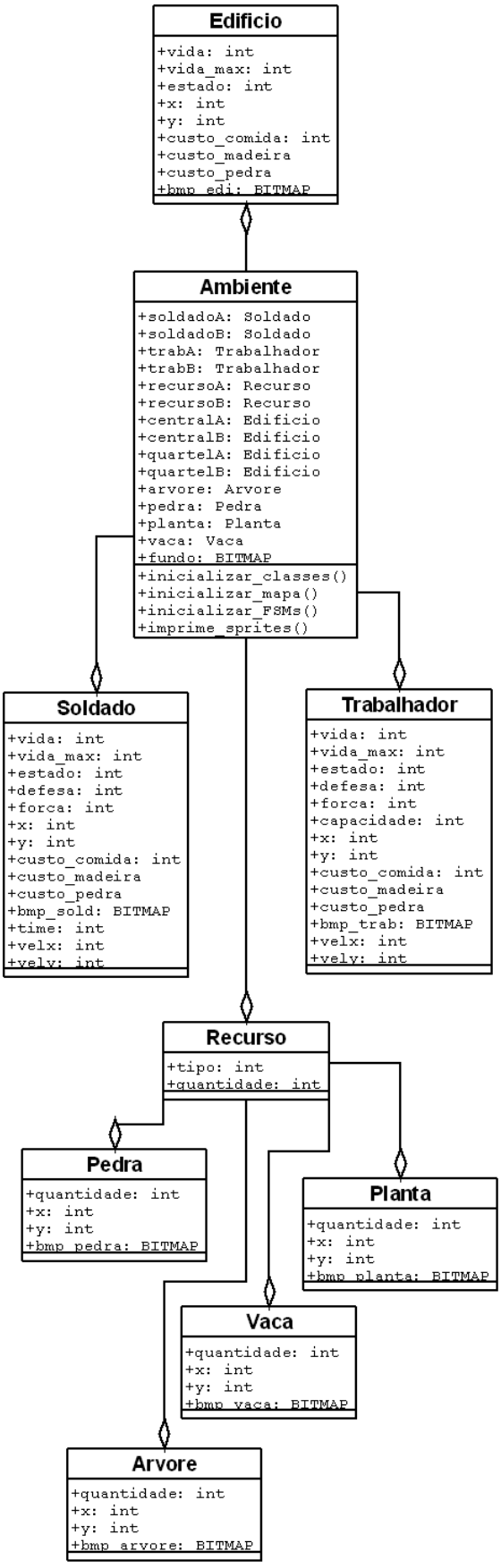


Figura 5.1: Diagrama de Classes do Jogo

O soldado perceberá o inimigo caso ele se aproxime de sua área de detecção. A função responsável por verificar esta aproximação é a *detecta_inimigo_sold.* A área de detecção é um raio em volta do soldado. Então, caso algum inimigo penetre nesse raio, a função retornara um valor positivo, e o soldado imediatamente mudará seu estado para ATACANDO, como visto na Figura 5.3.

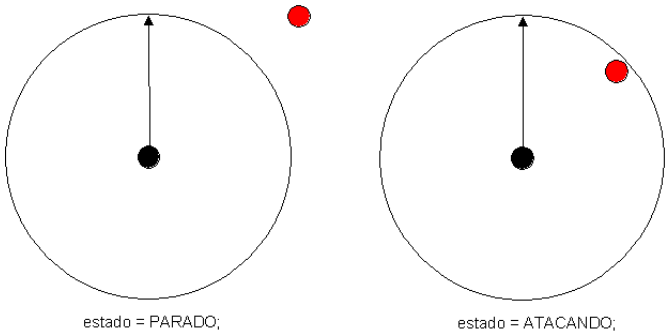


Figura 5.3: Exemplo de detecção de um inimigo

No caso de metade dos soldados ou mais estiverem no estado ATACANDO ao mesmo tempo será chamada a FSM do grupo, onde eles irão tentar se comunicar e executar o flocking.

Tendo a coordenada os soldados começam a se mover em direção a ela e começam a se aproximar uns dos outros. Dada uma maior aproximação cada soldado examina a sua vizinhança e caso ele esteja muito perto de seu vizinho ele irá mudar o sentido do movimento, se mover por alguns ciclos neste sentido, e retomar o sentido original, ligeiramente mais afastado de seu vizinho. Quando ele faz este movimento, pode se aproximar de outro soldado, que então repetirá o mesmo processo.

Alcançada as coordenadas de destino, a FSM do grupo é finalizada e é dado o prosseguimento com as máquinas de estado individuais.

E. Interfaces e Testes Realizados

Na Figura 5.4, é possível ver a interface do jogo ao iniciá-lo.

Figura 5.4: Interface inicial do jogo.

Ainda na configuração inicial é possível visualizar as

equipes A, à esquerda, e B, à direita (Figura 5.5).

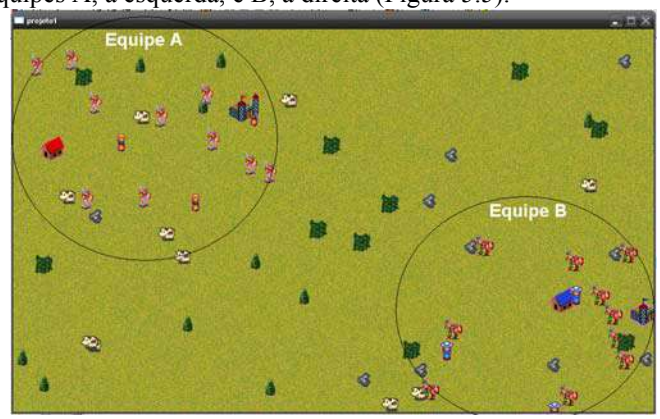


Figura 5.5: Equipe A e Equipe B

Na Figura 5.6 é possível ver um grupo de soldados iniciando o processo de *flocking*.

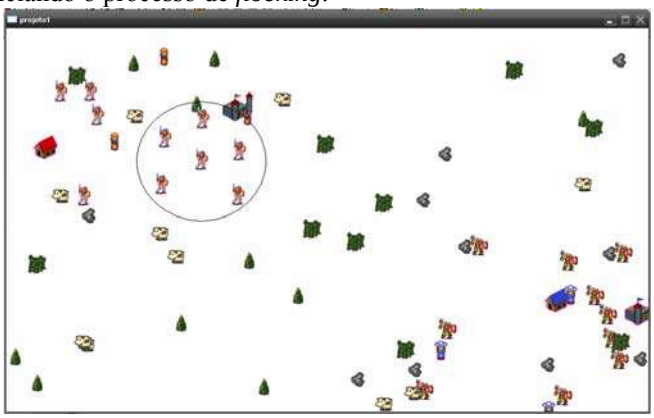


Figura 5.6: Soldados em *flocking*

VI. CONCLUSÃO

Com o aumento do nível de exigência dos jogadores e da capacidade de processamento dos computadores, vai ser cada vez mais exigido um avanço da inteligência artificial aplicada em jogos. E devido ao seu crescimento e valorização, a tendência é que a Game AI, seja gradualmente melhor aceita e incorporada à área acadêmica. É possível que no futuro tenha-se agentes inteligentes que aprendam com maneira de jogar dos jogadores e deixem os jogos mais desafiadores. Schwab (2009) [19] afirma que os agentes dentro dos jogos, inclusive, poderão ser dotados de emoções e personalidades bem definidas. Esse avanço na IA em jogos, juntamente com os novos equipamentos e tecnologias, como o Kinect, que são apenas um começo para esse novo tipo de iteratividade, deixarão os jogos eletrônicos muito mais imersivos e, espera-se, divertidos.

A ideia de criar um jogo para este trabalho foi fazer um pouco do que os jogos fazem para nós, ou seja, deixar a experiência e o aprendizado mais divertidos e interessantes. Não foi possível aplicar todas as técnicas desejadas, porém isso não diminuiu em nada o conhecimento adquirido ao longo de seu desenvolvimento. O jogo serve como um protótipo do que ainda pode ser realizado com técnicas conhecidas de IA, além da Game AI.

Futuramente, espera-se incluir as Máquinas de Estado Fuzzy ao jogo, o qual era inicialmente desejado para este trabalho, mas infelizmente não foi possível.

Outro objetivo futuro é aprofundar os testes sobre o jogo, realizando testes de desempenho computacional e uma análise comparativa detalhada do funcionamento do jogo executando as técnica de IA separadamente.

REFERÊNCIAS

[1] M. Bellis, Computer and Video Game History. Disponível em <http:// http://inventors.about.com/library/inventors/blcomputer_videogames.htm > Acessado em 18 de Novembro de 2011.

[2] A. J. Champandard, An Overview of the AI Architecture Inside the F.E.A.R. SDK. Disponível em < http://aigamedev.com/open/article/fear-sdk/>. Acessado em 05 de Novembro de 2011.

[3] M. Da Luz, Mairlo. Desenvolvimento de Jogos de Computador. 2004. 117 f. Projeto Final de Graduação. Departamento de Matemática e Computação –Universidade Federal de Itajubá (UNIFEI). Itajubá, 2004

[4] R. Demaria, J. L. Wilson, High Score! The Illustrated History of Eletronic Games, 2nd Editon. Emeryville: McGraw-Hill/Osborne. 2004.

[5] E. Fujita, Algoritmos de IA para Jogos. Projeto Final de Graduação. Departamento de Computação. Universidade Estadual de Londrina (UEL). Londrina, 2005.

[6] P. Ghemawat, A Estratégia e o Cenário de Negócios. Porto Alegre: Bookman, 2007.

[7] G. G. Granada, Framework de Técnicas Inteligentes Voltado a Jogos Eletrônicos. Projeto Final de Graduação. Universidade Federal do Rio Grande. Rio Grande, 2011.

[8] J. S. Harbour, Game Programming All in One, Third Edition. Thomson Course Technology PTR. Boston, 2007.

[9] H. Japiassu, D. Marcondes, icionário Básico de Filosofia. Jorge Zahar Editora: Rio de Janeiro, 1996.

[10] A. Kishimoto, Inteligência Artificial em Jogos Eletrônicos. São Paulo, 2004.

[11] C. A. Madsen, A. FURG Smart Games: Um Framework para Jogos Inteligentes. Universidade Federal do Rio Grande. Qualificação de Dissertação de Mestrado em Modelagem Computacional. Rio Grande, 2011.

[12] I. Millington, Artificial Intelligence for Games. San Francisco: Elsevier. 2006.

[13] F. C. Morais, Desenvolvimento de Jogos Eletrônicos. Belo Horizonte, 2009.

[14] L. A. Mozelli, Controle Fuzzy para Sistemas Takagi-Sugeno: Condições Aprimoradas e Aplicações. Dissertação de Mestrado. Universidade Federal de Minas Gerais. Belo Horizonte, 2008.

[15] S. Rabin, AI Game Programming Wisdom. Hingham: Charles River Media. 2002.

[16] S. Rezende, O. Sistemas Inteligentes. Barueri: Manole.2003.

[17] P. Ribeiro, IA e Jogos. Seminario 1 – Mestrado em Informática e Sistemas – Universidade de Coimbra, 2003.

[18] G. L. Santos, dos. Máquinas de Estados Hierárquicas em Jogos Eletrônicos. 2004. 54f. Tese de Mestrado. Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Rio de Janeiro, 2004.

[19] B. Schwab, AI Game Engine Programming. Hingham: Charles River Media. 2009.

[20] M. Sipser, Introdução à Teoria da Computação. São Paulo: Thomson Learning. 2007.

[21] R. Tengan, IA em Jogos. Projeto Final de Graduação. Faculdade de Tecnologia de Praia Grande. Praia Grande, 2006.

[22] V. Vieira, Revolution AI Engine: Desenvolvimento de um motor de Inteligência Artificial para a criação de jogos eletrônicos. 75f. Projeto Final de Graduação. Centro de Informática – Universidade Federal de Pernambuco. Recife, 2005.

[23] S. Wehmeier, Oxford Advanced Learner's Dictionary. Oxford: Oxford University Press. 2000.