UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

TRABALHO DE GRADUAÇÃO

REVOLUTION AI ENGINE

DESENVOLVIMENTO DE UM MOTOR DE INTELIGÊNCIA ARTIFICIAL PARA A CRIAÇÃO DE JOGOS ELETRÔNICOS

Vicente Vieira Filho <vvf@cin.ufpe.br>

Orientadora: Patrícia Cabral de Azevedo Restelli Tedesco <pcart@cin.ufpe.br>

Recife, Agosto de 2005

“Shoot for the moon. Even if you miss, you'll land among the stars”

— LES BROWN

À memória de Luiz Henrique e Emanuel Vieira.

AGRADECIMENTOS

“Gratitude is the memory of the heart”
— JEAN BAPTISTE MASSIEU

Aos meus pais e irmãos, de forma especial, pela presença fundamental e diferencial em cada momento de minha vida.

À minha mestra, orientadora e amiga Patrícia Tedesco pela oportunidade de trabalharmos juntos e de contar com o seu apoio e dedicação em cada passo da construção desse trabalho.

A todos os docentes do Centro de Informática que apesar de não colaborarem diretamente na construção desse trabalho, influenciaram sobremaneira em minha formação acadêmica. Dentre os quais destaco Silvio Melo, Geber Ramalho, Francisco de Assis e Aluízio Araújo, em ordem cronológica, pela importância e impacto que tiveram em minha vida acadêmica e pela admiração e amizade que nutro por eles.

A todos os meus amigos, dos mais antigos aos mais recentes, pela companhia e momentos vividos no período da graduação. Em especial a Aércio Filho, Afonso Ferreira, André Lima, Ives José, Leonardo Costa, Marco Túlio Albuquerque e Vilmar Nepomuceno pela presença sempre constante.

Aos alunos Renan Weber, José Carlos, Livar Cunha e Rafael Nóbrega pelo apoio e participação direta no desenvolvimento desse trabalho e implementação do protótipo.

E finalmente ao Centro de Informática e Universidade Federal de Pernambuco por terem me proporcionado um maravilhoso ambiente de estudo e pesquisa.

RESUMO

O desenvolvimento de jogos é uma atividade complexa e multidisciplinar a qual envolve conhecimentos profundos em diversas áreas da computação. E para tratar esse problema, a indústria de jogos utiliza padrões de projeto e frameworks para encapsular essa complexidade. É essa prática que permite o desenvolvimento com qualidade e eficiência de projetos profissionais.

Atualmente já são utilizados frameworks para encapsular a complexidade de diversas áreas de conhecimento comuns ao desenvolvimento de jogos, dentre elas, a área de computação gráfica, de áudio, de física. No entanto, a área de Inteligência Artificial não acompanhou essa evolução e encontra-se defasada nesse aspecto.

O objetivo desse trabalho é prover uma pesquisa das técnicas mais utilizadas em jogos divididas por gêneros. Além disso, é projetado um motor de Inteligência Artificial para jogos, nomeado REvolution AI Engine, com o propósito de encapsular as técnicas pesquisadas.

Palavras-chave: Inteligência Artificial, Entretenimento Digital, Engenharia de Software, Padrões de Projeto, Frameworks, Motores.

ABSTRACT

The game development is a complex and multidisciplinary activity which includes deep knowledges from several areas of computer science. And to treat this problem, the game industry uses design patterns and frameworks to encapsulate this complexity. It's these practices that enable a development of professional projects with quality and efficiency.

Nowadays industries are already using frameworks to encapsulate the complexity of many knowledge areas correlated to the game development, between them, computer graphics, audio and physics. However, the Artificial Intelligence area does not follow this evolution and now is dephased in this aspect.

The objective of this work is to provide a research of the most used techniques in the game area divided by genre. Moreover, a game artificial intelligence engine is developed, nominated REvolution AI Engine, with the intention of encapsulate the research's techniques.

SUMÁRIO

INTRODUÇÃO **1**

1.1 OBJETIVOS 3
1.2 IA VERSUS IA EM JOGOS 3
1.3 VISÃO GERAL 4

GÊNEROS DE JOGOS **5**

2.1 GÊNEROS DE JOGOS 6
2.1.1 Role Playing Games (RPGs) *6*
2.1.2 Adventure *8*
2.1.3 Estratégia *11*
2.1.4 First-Person Shooter / Third-Person Shooter (FTPS) *14*
2.1.5 Plataforma *16*
2.1.6 Esportes *18*
2.1.7 Simulação *20*
2.1.8 Corrida *24*
2.1.9 Luta *26*
2.2 ANÁLISE COMPARATIVA 27
2.3 OUTRAS TÉCNICAS 28
2.3.1 Técnicas *28*
2.4 CONCLUSÕES 30

MOTORES DE IA **31**

3.1 DISCUSSÃO 31
3.1.1 Unreal Engine 3 ..... *31*
3.1.2 Source Engine *34*
3.1.3 Reality Engine *34*
3.1.4 Torque Game Engine *35*
3.1.5 CryEngine *36*
3.2 ANÁLISE COMPARATIVA 37
3.3 CONCLUSÕES 38

REVOLUTION AI ENGINE **39**

4.1 ELICITAÇÃO DE REQUISITOS 39
4.3 PROJETO DA ARQUITETURA 40
4.3.1 Modelagem *40*
4.3.2 Arquitetura *43*
4.4 IMPLEMENTAÇÃO 44
4.5 CONCLUSÕES 44

CONCLUSÕES E TRABALHOS FUTUROS **45**

5.1 CONTRIBUIÇÕES 45
5.2 DIFICULDADES ENCONTRADAS 45
5.3 TRABALHOS FUTUROS 46
5.4 CONSIDERAÇÕES FINAIS 46

REFERÊNCIAS **47**

APÊNDICE A **55**

A.1 SISTEMA DE TEMPO REAL 55

A.1.1 Algoritmos *56*

APÊNDICE B **58**

B.1 PROJETO DISTRIBUÍDO DA IA 58

APÊNDICE C **60**

C.1 ARQUITETURA EM DETALHES 60

C.1.1 Módulo Sistema Multiagentes *60*

C.1.2 Módulo Agente *61*

C.1.3 Módulo Comportamento *62*

C.1.4 Módulo Sistema de Tempo Real *63*

C.1.5 Módulo Avançado *64*

LISTA DE FIGURAS

FIGURA 1 WORLD OF WARCRAFT 7
FIGURA 2 MONKEY ISLAND 9
FIGURA 3 COMMANDOS 9
FIGURA 4 AGE OF EMPIRES III 11
FIGURA 5 COUNTER STRIKE 14
FIGURA 6 SUPER MARIO WORLD 16
FIGURA 7 SUPER MARIO 64 17
FIGURA 8 FIFA SOCCER 2004 19
FIGURA 9 FLIGHT SIMULATOR: A CENTURY OF FLIGHT 21
FIGURA 10 THE SIMS 2 21
FIGURA 11 GRAN TURISMO 24
FIGURA 12 STREET FIGHTER 2 26
FIGURA 13 FERRAMENTA UNREAL KISMET 33
FIGURA 14 ARQUITETURA DO AGENTE 42
FIGURA 15 ARQUITETURA DO MOTOR 43
FIGURA 16 ABORDAGENS DE SISTEMA DE TEMPO REAL 56
FIGURA 17 DIAGRAMA DE CLASSES DA ENGINE 61
FIGURA 18 DIAGRAMA DE CLASSES DO AGENTE 62
FIGURA 19 DIAGRAMA DE CLASSE DOS COMPORTAMENTOS 63
FIGURA 20 DIAGRAMA DE CLASSES DO SISTEMA DE TEMPO REAL 64
FIGURA 21 DIAGRAMA DE CLASSES DA REDE NEURAL E DO ALGORITMO GENÉTICO 65

LISTA DE TABELAS

TABELA 1 LINHA DO TEMPO DA INTELIGÊNCIA ARITFICIAL PARA JOGOS 1
TABELA 2 COMPARATIVO DE TÉCNICAS VERSUS GÊNEROS 27
TABELA 3 LISTA DE TÉCNICAS 30
TABELA 4 COMPARATIVO DOS MOTORES SEGUNDO CRITÉRIO ABRANGÊNCIA 37
TABELA 5 PORCENTAGEM DE ABRANGÊNCIA 38
TABELA 6 LISTA DE REQUISITOS 39

CAPÍTULO 1

INTRODUÇÃO

“A journey of a thousand miles must begin with a single step”
— LAO-TZU

A Inteligência Artificial (IA) é aplicada em jogos desde os anos 70 durante o surgimento dos primeiros jogos caracterizados pela simplicidade e falta de recursos. A origem humilde da área, a IA para jogos, influenciou e ainda influencia a percepção do público. Ainda nos dias de hoje, os fantasmas Inky, Pinky, Blinky e Clyde do jogo Pac-Man assombram essa área. E até muito recentemente a indústria de jogos pouco havia realizado para mudar essa percepção.

Entretanto uma revolução vem acontecendo. Os últimos anos testemunharam a IA para jogos muita mais rica que no passado e contribuindo para a construção de jogos com maior nível de entretenimento [Schwab 2004]. Devido à evolução dos hardwares, mais especificamente das placas de vídeo, a qualidade gráfica dos jogos elevou-se de maneira surpreendente tornando a IA um fator crítico para o sucesso de um jogo, decidindo quais jogos se tornam bestsellers e determinando o destino de vários game studios. A IA para jogos vem se transformando de um mero coadjuvante à estrela de primeira grandeza dessa indústria.

A tabela abaixo apresenta a evolução da área.

Tabela 1 Linha do Tempo da Inteligência Aritficial para Jogos

Sem IA	Space War! - Primeiro jogo para computador escrito para o minicomputador PDP 1. Requer 2 jogadores.	1962
	Pong – Versão eletrônica do jogo 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, controles simples a dificuldade crescente no decorrer do tempo.	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 – Um dos primeiros jogos de luta, um contra um, com o computador como adversário.	1984
FSM's	Herzog Zwei - O primeiro RTS a surgir e o mundo experimenta pela primeira vez uma péssima implementação do algoritmo de pathfinding.	1990
	Doom – Início oficial da era do FPS.	1993
Várias Técnicas	BattleCruiser: 3000AD – Primeiro uso de redes neurais em um jogo comercial.	1996
	Deep Blue derrota o atual campeão mundial de xadrez Gary Kasparov.	1997
	Half-Life – A inteligência artificial para jogos encontra-se em seu auge. O jogo faz grande uso de linguagens de script.	1998
	Black & White – Utiliza criaturas que usam aprendizado por reforço e por observação.	2001

Há décadas atrás, a IA era concebida e produzida nos últimos instantes da atividade de desenvolvimento dos jogos [Rabin 2002]. Ou seja, nos últimos dois ou três meses do projeto a equipe criava a IA específica para o jogo e em paralelo desenvolvia várias outras atividades de alta prioridade (principalmente as atividades relacionadas à fase de testes e ao gerenciamento do projeto de configuração).

Nos últimos anos, a indústria de jogos atravessou um processo de maturação para solucionar esses problemas. Associado à preocupação crescente com a área de IA outro fator contribuiu sobremaneira para a profissionalização e fortalecimento da indústria: o mercado. O mercado mundial de jogos vem crescendo a uma taxa superior do que jamais esperado [DFC, PWHC, ELSPA].

A *DFC Intelligence* [DFC], uma firma de consultoria e pesquisa de mercado focada na área de entretenimento, prevê que as vendas de jogos para computadores, *consoles* e dispositivos móveis, incluindo *hardware* e *software*, devem crescer de U$ 23.2 bilhões, em 2003, para U$ 31.6 bilhões, em 2009. A previsão é que o mercado de jogos *online* atinja U$ 9.8 bilhões em vendas ao final dos próximos cinco anos, resultando num rendimento total da indústria de U$ 41.4 bilhões em 2009.

O amadurecimento da indústria de jogos contou com a inserção de técnicas e processos advindos das áreas de gerência de projetos e engenharia de software que implicaram na criação de grupos especializados nas empresas com responsabilidades sobre

determinadas áreas, tais como, computação gráfica, criação (arte e modelagem) e, mais recentemente, Inteligência Artificial.

A criação desses grupos e posteriormente de empresas especializadas nessas áreas contribuíram para a produção de ferramentas e frameworks. A área de computação gráfica, por exemplo, conta com diversos frameworks e motores gráficos [Genesys, Irrlicht, Ogre3D, Crystal] para inserir maior dinâmica, qualidade e robustez aos jogos.

A área de IA ainda não despertou totalmente para essa necessidade de criação de frameworks e ferramentas utilitárias que contribuem para não tão somente garantir robustez e qualidade como também facilitar as atividades relacionadas a testes e *tuning*.

1.1 OBJETIVOS

O desenvolvimento de jogos consiste em uma atividade árdua e não-trivial, pois jogos são complexos sistemas de tempo-real multimídia e interativos envolvendo profundo conhecimento de diversas áreas da computação e ciências humanas. Como forma de diminuir, controlar e distribuir a complexidade existente na produção de jogos, a indústria faz uso de frameworks e padrões de projeto [Reality] com os quais encapsula a complexidade em bibliotecas e modelos de arquitetura.

O presente trabalho tem por objetivo preencher uma lacuna no estudo da Inteligência Artificial para jogos. E para consolidar os estudos na área pretendo modelar, arquitetura e implementar um motor de Inteligência Artificial para jogos.

O motor visa também auxiliar no desenvolvimento de jogos na empresa Manifesto Game Studio[1], e em especial na criação de seu primeiro produto.

1.2 IA VERSUS IA EM JOGOS

A definição de inteligência é bastante nebulosa. A visão do dicionário é que inteligência significa a capacidade de adquirir e aplicar conhecimento, no entanto essa definição é muito abrangente. Interpretada literalmente pode-se concluir que um termostato é inteligente. O termostato adquire conhecimento da temperatura do ambiente e aplica o que aprendeu acionando o condicionador de ar. O dicionário sugere ainda que a inteligência demonstra a faculdade de pensar e raciocinar. Apesar dessa nova definição soar menos abrangente (o termostato foi deixado para trás), ela adiciona dois novos conceitos ainda mais nebulosos.

De fato, a verdadeira definição da inteligência é um antigo debate e não faz parte do escopo desse trabalho. E desenvolver jogos bons e com qualidade não requer o conhecimento desse conceito. A discussão travada aqui visa demonstrar as diferenças

[1] A Manifesto Game Studio é uma empresa de desenvolvimento de jogos para computadores atualmente incubada no Recife Beat.

existentes no conceito tradicional do que é inteligência e inteligência artificial para o que é a inteligência em jogos.

A definição acadêmica da inteligência artificial dada por [Russel e Norvig 200] prega que a IA é a criação de programas de computadores que emulam as ações e pensamentos humanos. Já a IA para jogos é o código dentro de um jogo que controla de forma computacional os oponentes para que esses aparentem tomar decisões inteligentes quando o jogo apresenta múltiplas escolhas para uma determinada situação. A palavra aparentar utilizada na frase é o ponto chave para a distinção entre a IA e a IA em jogos. A IA em jogos não se interessa pela resposta que o sistema inteligente gera nem como o sistema funciona internamente. O interesse está em como o sistema atua, e não como ele pensa.

E essa diferença surge devido aos jogadores não estarem interessados em saber se o jogo utiliza uma rede neural, uma máquina de inferência baseada em regras lógicas ou uma máquina de estados finitos. Estão interessados em como o jogo aparente ser inteligente.

1.3 VISÃO GERAL

No capítulo 2 deste documento apresentaremos uma visão geral sobre a área de entretenimento digital e os principais gêneros de jogos existentes. Ainda nesse capítulo é apresentada uma análise das principais técnicas associadas a cada um dos gêneros.

No capítulo 3 apresentaremos os principais motores de Inteligência Artificial para jogos existentes no mercado e suas principais características. Ao final do capítulo é realizada uma comparação entre os motores existentes.

No capítulo 4 será apresentada a REvolution AI Engine com uma descrição detalhada sobre os seus requisitos, modelagem e arquitetura. E finalmente no capítulo 5 serão apresentadas nossas conclusões acerca do trabalho desenvolvido, bem como suas contribuições e possibilidades de trabalhos futuros relacionados.

CAPÍTULO 2

GÊNEROS DE JOGOS

“We do not stop playing because we grow old.
We grow old because we stop playing”
— AUTOR DESCONHECIDO

A indústria de entretenimento digital, assim como as demais indústrias de software, para diminuir, controlar e distribuir a complexidade existente na produção de jogos utiliza *frameworks* e padrões de projeto para encapsular a complexidade em bibliotecas e modelos de arquitetura [Gamma et al. 1995].

Dentro da área de entretenimento digital, a área de processamento gráfico respondeu pelos primeiros passos nessa direção com a criação de motores gráficos para auxiliar o tratamento da complexidade relativa a essa área. Em seguida, outras áreas começaram a seguir o mesmo caminho de sucesso. Frameworks e motores foram também criados para tratar os recursos de áudio (trilhas e efeitos sonoros e mais recentemente som tridimensional). O mesmo aconteceu para a área de redes de computadores para a criação de jogos multiplayer e para a área de física de jogos (tratamento de colisão, cinemática reversa, etc) [TGE ,Unreal, Reality, Source].

Seguindo um caminho contrário, a área de Inteligência Artificial não acompanhou os passos seguidos pela outras áreas e manteve-se durante bastante tempo esquecida pela indústria. A conseqüência natural é que não existem motores específicos para essa área.

O atraso nessa área está relacionado à maior complexidade em analisar e definir um conjunto de técnicas úteis para um determinar gênero ou grupo de gêneros. Isso ocorre principalmente devido a um mesmo gênero de jogo, Esporte, por exemplo, utilizar técnicas diferentes e alcançar resultados semelhantes. Para a troca de mensagens e informações entre os elementos e personagens dentro de um jogo de Esporte, por exemplo, pode-se utilizar um sistema de mensagens ou ainda permitir que os elementos acessem uns aos outros para identificar mudanças no ambiente.

O propósito desse capítulo é estudar os gêneros de jogos com o intuito de identificar técnicas e abordagens mais comuns para propor um motor mais genérico para atender as necessidades de todos os gêneros de jogos.

2.1 GÊNEROS DE JOGOS

O estudo e a observação dos gêneros de jogos existentes são de grande importância, não somente para identificar as técnicas mais utilizadas, como também para mapear as técnicas por gênero. A partir dessa observação podem-se definir os critérios necessários para iniciar um estudo sobre os motores de Inteligência Artificial.

Abaixo segue uma lista descrevendo os gêneros e respectivas abordagens utilizadas.

2.1.1 Role Playing Games (RPGs)

Role-Playing Games, ou em uma interpretação literal, jogos de interpretação de papéis, geralmente posicionam o jogador em um ambiente de fantasia ou ficção científica e conduzem o gameplay[2] através de um roteiro predeterminado. A maioria desses jogos prevê o jogador exercendo o papel de um tipo específico de aventureiro especializado em certo conjunto de habilidades (tais como combater ou arremessar feitiços). Essa variedade de aventureiros controlados pelo jogador é denominada classe e os jogadores podem normalmente controlar um ou mais desses personagens.

O gênero RPG está entre os primeiros disponíveis para PC's. Os primeiros RPG's são baseados em textos ou com arte baseada em caracteres ASCII (como Rogue [Rogue] e NetHack [NetHack]). Posteriormente surgiram os RPG's gráficos (como Dungeons & Dragons [Dungeons&Dragons] e mais recentemente Star Wars [Star Wars]) e todo o resto transformaram-se em história.

A principal característica desse gênero é a imersão, ou seja, fazer o jogador identificar-se com o personagem principal a ponto de ser capaz de despender grande quantidade de tempo no desenvolvimento desse personagem e, eventualmente, finalizar o jogo

Jogos e séries de sucesso no gênero são RuneQuest [RuneQuest], Magic: The Gathering, Vampire: The Masquerade [Vampire], Lord of the Rings [Lord], WarCraft: The Role Playing Game [Warcraft] e World of Warcraft [Wow].

[2] Gameplay consiste na jogabilidade ou forma e condições de interação com o jogo.

Figura 1 World of Warcraft

TÉCNICAS

Os RPG's são em sua maioria jogos extensos e com uma vasta variedade de escolhas que permitem ao jogador trilhar caminhos distintos e adquirir diferentes experiências. Devido a isso, jogadores investem grande quantidade de tempo nos jogos (pesquisas indicam que nos jogos para PC jogadores demandam um mínimo de 40 horas por título), o que torna mais visível e óbvio qualquer comportamento repetitivo [Schwab 2004].

A seguir as técnicas mais comuns ao gênero RPG.

Sistema de Script

É técnica a qual utiliza uma linguagem de programação simplificada – geralmente ferramentas de script são construídas para facilitar a edição, seja de forma gráfica ou textual – para construir elementos de IA, lógica e comportamentos [Russell e Norvig 2003].

A maioria dos RPG's utiliza esse recurso em demasia, devido à maioria desses jogos possuir uma linha de pensamento, ou história, bastante específico. Scripts são utilizados porque a maioria dos RPG's é linear ou com bifurcações lineares, ou seja, não há liberdade de escolha para o jogador. O jogador deve seguir por determinados caminhos e escolhas pré-definidas. E dessa forma, o gênero funciona bem com a técnica de script.

Scripts são utilizados para uma grande variedade de construções de jogos incluindo diálogos, flags para eventos no jogo, inimigos específicos ou comportamento de NPC's, e muitos outros.

Máquina de Estados Finitos

Máquinas de Estados Finitos, ou Finite State Machine (FSM), é uma estrutura de dados composta por 3 elementos: estados da máquina, condições de entrada e funções de transição que servem como linhas de conexão entre os estados [Schwab 2004].

As FSM's permitem ao desenvolvedor dividir o jogo em estados explícitos em cada qual personagens irá comportar-se diferentemente e gerenciar esses comportamentos com blocos de códigos específicos. Por exemplo, inimigos controlados por IA dentro de um RPG podem possuir os estados Atacar, Fazer Cobertura, Recuar e Obter Recursos. O estado corrente do inimigo dependerá de indicadores tais como saúde, nível de ataque e defesa, dentre outros. Assim, esta técnica se mostra bastante útil tanto para os RPG's quanto para outros gêneros de jogos.

Sistema de Mensagens

Sistema de Mensagens consiste em um conceito simples. Ao invés de uma determinada entidade A checar a entidade B por mudanças particulares a cada instante de tempo, A é informado sobre mudanças através de uma mensagem enviada por B para A somente quando a mudança ocorre. Isso significa que nenhuma entidade desperdiça tempo e processamento realizando checagens para determinar se mudanças aconteceram [Schwab 2004].

Com tantos elementos em um mundo RPG, a necessidade de comunicação entre entidades é enorme. Assim, um sistema de mensagens é bastante útil neste gênero, pois a informação pode ser passada entre os membros de um grupo de maneira eficiente, facilitando o diálogo entre o grupo.

2.1.2 Adventure

O Adventure coloca o jogador no papel do protagonista da história, e normalmente requer do jogador a habilidade para a resolução de vários quebra-cabeças utilizando diferentes artefatos.

Os primeiros Adventures eram textuais. e os jogadores usavam o teclado para realizar ações tais como "pegar chave" ou "ir para leste" e o computador descrevia o resultado da ação, o que estava acontecendo.

Com os gráficos tornando-se mais comuns, os Adventures baseados em textos foram suplantados, e as descrições textuais trocadas por ilustrações. Esses Adventures gráficos ainda utilizavam somente comandos textuais.

O crescimento do uso do mouse conduziu ao gênero "Point-and-Click" dos jogos Adventure, nos quais o jogador não mais utiliza comandos textuais. Por exemplo, o jogador pode clicar no ícone "mão" e então selecionar um determinado elemento na tela a ser pego.

Os exemplos de maior sucesso são Maniac Mansion: Day of the Tentacle [DoTT], King's Quest [KingsQuest] e Monkey Island [MonkeyIsland].

Figura 2 Monkey Island

Figura 3 Commandos

TÉCNICAS

A seguir as técnicas mais comuns ao gênero Adventure.

Máquina de Estados Finitos

Muitos elementos de exploração e resolução de quebra-cabeças, típicos do gênero Adventure, combinam bem com os sistemas de IA baseado em FSM's. A existência de guardas com um determinado estado de alerta (sim ou não), ou ainda com uma enumeração de estados (como Neutro, Irritado, Alerta, Insano e Furioso) exemplificam o uso da técnica. Dessa forma, as FMS's provêm a melhor solução ao jogo.

Sistema de Script

Alguns Adventures utilizam muitos diálogos e seqüências que mostram como resolver um determinado quebra-cabeça em algum outro lugar ou nível do jogo. O sistema de script permite aos programadores e designers adicionar essas características aos jogos. Essa técnica é muito utilizada para jogos com um roteiro ou história linear, como é o caso dos Adventures.

Sistema de Mensagens

Nos Adventures é senso comum a conversão de determinadas ações do jogador - tais como empurrar uma alavanca, movimentar três pedras utilizando um certo padrão ou acender luzes de um quarto escuro - em eventos e mensagens a serem compartilhadas e endereçadas a outros lugares ou objetos do jogo. Ao empurrar uma alavanca pode-se abrir uma porta em um outro lugar do jogo e ao acender as luzes de um ambiente deve-se iluminar o mesmo para permitir a visualização por parte do jogador.

O uso de sistemas de mensagens facilita a comunicação entre os elementos do jogo e elimina a necessidade da verificação constante por flags relacionadas a eventos. No caso de um inimigo controlado por IA com o propósito de entrar em um castelo trancado, Esse deve comunicar a sua intenção de penetrar no castelo através de uma solicitação para a abertura dos portões. Essa comunicação pode ser realizada através de um sistema de mensagens ou o inimigo pode ter acesso aos controles do portão e modificar uma flag de controle de acesso ao castelo. Essa última opção necessita que elementos do jogo (inimigo e portão no exemplo) possuam referências mútuas e acessem uns aos outros e é uma prática não recomendada.

Lógica Nebulosa (Fuzzy)

O tratamento de dados sensoriais em jogos é de natureza complexa. Os oponentes controlados por IA necessitam de decisões fuzzy para lidar com suas percepções sobre o estado do mundo. Por exemplo, considere um dado guarda complacente com a movimentação do jogador. O jogador pode passar despercebido caso não ultrapasse "muito" as fronteiras permitidas, ou seja, não penetre tanto o campo de visão do guarda. Esse conceito de o quão se pode ou não penetrar no campo de visão do guarda é um conceito tratado pela Lógica Nebulosa.

2.1.3 Estratégia

O gênero Estratégia foca no planejamento cuidadoso e na habilidade de gerenciar recursos de maneira a alcançar a vitória. Os jogos de estratégia podem ser baseados em turnos ou em tempo real, mas existem alguns jogos que realizam a mistura desses dois estilos de jogos (como X-COM [XCOM]). Os jogos baseados em turnos são aqueles nos quais é alocado um tempo para cada jogador montar a sua estratégia, como por exemplo, o xadrez. Nos jogos em tempo real, como o próprio nome sugere, são jogos nos quais a estratégia deve ser montada e executada a cada instante, em tempo real. O gênero tem se popularizado desde os anos oitenta.

Os jogos baseados em turnos foram originalmente à forma mais comum de jogos de estratégia devido à restrição dos computadores de tratar a interação em tempo real. A vasta maioria dos jogos de estratégia baseados em turnos pode ser chamada de “jogos de estratégias de guerra” devido ao foco no combate militar (ex: jogo de estratégia de guerra sob a perspectiva de um General). Para os jogos em tempo real o foco é geralmente em táticas de batalhas militares (por exemplo, reconhecimento e defesa dos flancos).

Exemplos de jogos baseados em turnos incluem Sid Meier’s Civilization [Civ3], Heroes of Might and Magic [HMM] e a Fire Emblem [FireEmblem]. E de jogos em tempo real são: Warcraft [War3], Starcraft [Starcraft], Command and Conquer [Command&Conquer], Total Annihilation e Age of Empires [Age].

Figura 4 Age of Empires III

TÉCNICAS

Os jogos deste gênero são geral os que requerem computação mais intensiva, simplesmente pelo fato de que envolve numerosas unidades a serem coordenadas. E ainda é necessário compartilhar os recursos de tempo de CPU com outras funções como detecção de colisão e rotinas gráficas. Esse gênero enfatiza a utilização da estratégia em contraposição a ação rápida e ao uso de rápidos reflexos que não são, normalmente, necessários para o sucesso nesse gênero [Schwab 2004]. A seguir as técnicas mais utilizadas no gênero Estratégia.

Sistema de Mensagens

Os jogos de estratégia envolvem, sobretudo a comunicação entre as unidades e elementos do jogo. A comunicação é fundamental para delegar responsabilidades e atividades a outras unidades, descobrir as unidades que fornecem determinados serviços, obter informações gerais sobre estado do mundo, entre várias outras funções.

A comunicação ocorre tanto no sentido horizontal, entre elementos de um mesmo nível. Essa comunicação corresponde, em um jogo de estratégia de guerra, a comunicação realizada entre soldados e entre generais, mas não entre ambos. E a comunicação pode ocorre no sentido vertical, entre elementos de níveis diferentes. Nesse caso a comunicação envolveria soldados e generais, continuando o exemplo anterior.

Com a quantidade de unidades dentro de um jogo de estratégia podendo chegar a níveis extremos, o tratamento das mudanças de estados do jogo acarretam em um alto custo computacional. O uso de sistemas de mensagens torna a comunicação de eventos e flags fácil e rápida entre as unidades do jogo.

Máquina de Estados Finitos

Pode ser bastante útil em várias atividades de inteligência artificial que fazem parte do gênero de Estratégia. Unidades de inteligência individuais (inimigos podem ser controlados e terem os seus comportamentos definidos por FSM's), sistemas dentro de um nível estratégico (a construção de uma cidade pode ser uma FSM simples que define a ordem de construção), e vários outros elementos podem ser beneficiados pela FSM.

Máquina de Estados Fuzzy

O nível estratégico dos jogos é um dos problemas do gênero que não funcionam bem com as soluções baseadas em máquinas de estado regulares. A preponderância de informações pouco precisas sobre os oponentes e o mundo, combinado com o número de micro decisões que necessitam serem tomadas, fazem com que um oponente de IA possa realizar várias ações possíveis, e sendo todas as decisões vitoriosas. Por exemplo, o oponente pode não saber a quantidade de tropas inimigas, nem conhecer as suas reservas de ouro, mas pode acreditar ainda na possibilidade de vitória.

Dessa forma, o sistema mais adequado é a máquina de estados fuzzy (FuSM), que possui a estrutura e comportamento das máquinas de estado, mas considera a imprecisão inerente às decisões desse gênero.

Inteligência Artificial Hierárquica

Os jogos de Estratégia possuem múltiplos requisitos, e por vezes conflitantes. Por exemplo, imagine a seguinte situação: um jogador necessita movimentar um exército do ponto A para o ponto B, mas no caminho, uma emboscada acontece e os membros do exercito estão sendo atacados. As unidades atacadas devem parar e contra-atacar? O exército inteiro deve parar a movimentação e certificar-se de que o problema está resolvido? Ou o exército deve ignorar a ameaça e continuar marchando? A resposta é determinada pelo tratamento dado pela inteligência artificial, seja esse tratamento individual ou visando o grupo (tático[3] ou estratégico[4]). A inteligência artificial hierárquica provê o cumprimento de tarefas de alto nível, mas também parecendo inteligente no nível de unidade.

Planejamento

Os jogos de Estratégia envolvem tática e planejamento, e para completar atividades de alto nível vários pré-requisitos devem ser adicionados ao plano corrente. Para a atividade de alto nível, atacar o inimigo por vias aéreas, existem vários pré-requisitos a serem adicionados ao plano corrente. O primeiro passo do plano consiste em adquirir fundações tecnológicas na hierarquia de construções (por exemplo, pode ser necessário construir a infantaria anteriormente à construção da força aérea). E como segundo passo, determinar os recursos necessários para concluir o plano, que em caso de deficiência pode disparar um plano secundário de adquirir mais recursos [Schwab 2004] [Russel e Norvig 2003].

Data-Driven Systems

Essa técnica consiste na criação de editores para parametrizar a tomada de decisão. Ferramentas são desenvolvidas para tornar a atividade de testes e de refinamento da aplicação mais simples e intuitivos.

Muitos dos mais populares jogos de Estratégia estão transferindo grandes porções da tomada de decisões de inteligência artificial para uma forma que não código, uma forma mais flexível, criando um simples ambiente com parâmetros ou definição de regras. Isso permite aos designers trabalhar nos jogos de maneira mais fácil por garantir o acesso ao jogo e ao tuning da IA e os consumidores podem também brincar com os parâmetros [Schwab 2004].

[3] Tática corresponde aos métodos utilizados para ganhar um conflito de pequena escala, executar uma otimização, etc.

[4] Estratégia é um plano de ação de longo prazo designado para alcançar uma determinada meta. A principal diferença existente entre estratégia e tática é que tática corresponde aos meios usados para ganhar um objetivo enquanto estratégia é o plano total.

2.1.4 First-Person Shooter / Third-Person Shooter (FTPS)

Esse gênero enfatiza tiros e combate de um ponto de vista específico. No caso dos FPS's, coloca o jogador atrás de um revólver, ou outra arma, segurando-o com a mão. Essa perspectiva visa a imersão e identificação do jogador com o personagem principal, a sensação de "estar lá". Em sua maioria requer bons reflexos e velocidade por parte do jogador.

No caso dos TPS's emprega uma perspectiva em terceira pessoa para o jogador. Essa perspectiva é normalmente atrás do personagem do jogo, mas às vezes de uma perspectiva isométrica[5].

Alguns Shooters apresentam uma perspectiva padrão, primeira ou terceira pessoa, mas permitem ao jogador escolher entre as duas. Outros trocam entre as duas perspectivas em determinados pontos no jogo.

Para tornar-se um jogo efetivo, um FTPS necessita ser ao mesmo tempo rápido e tridimensional, o que eliminava a maioria dos hardwares dos consumidores no início dos anos noventa. Doom revolucionou o gênero utilizando técnicas inovadoras para tornar o jogo rápido o bastante para executar na maioria das máquinas.

Atualmente os mais populares jogos do gênero são Counter Strike [CS], Halo [Halo] e Unreal Tournament [UnrealTournament].

Figura 5 Counter Strike

[5] Perspectiva Isométrica é um método de representação visual de objetos tridimensionais em duas dimensões nos quais os ângulos entre as projeções dos eixos x, y e z são todos idênticos e iguais a 120º.

TÉCNICAS

Máquina de Estados Finitos

A principal técnica de programação de IA para jogos aparece novamente aqui nesse gênero. Devido ao tempo de vida da maioria dos inimigos nesses jogos ser bastante pequeno, não é necessário um planejamento real. A inteligência Artificial nesse caso envolve um número mínimo de estados, geralmente são eles: Atacar, Recuar, Explorar e Procurar Itens. O resto da inteligência advém do sistema de navegação, modelos de movimentação dos personagens e outras rotinas de suporte.

Máquina de Estados Fuzzy

FuSM's são utilizadas nesse gênero devido especialmente ao número de variáveis fuzzy ser pequeno, o que diminui a complexidade do sistema e evita problemas de cálculos combinatório prejudiciais aos sistemas fuzzy [Schwab 2004].

Os dados de entrada de um estado para um determinado oponente em um FTPS não são necessariamente regulares (não fuzzy). Um oponente controlado por Inteligência Artificial pode estar com 23% de saúde, mas possuir uma ótima arma e então perseguir o jogador sem ser notado. Mesmo o oponente estando bastante machucado, ele deveria atacar o jogador? A resposta é provavelmente sim. Pois mesmo o jogador estando danificado, as chances dele obter sucesso tendo uma arma de qualidade e perseguindo o jogador sem ser notado são grandes. No entanto o oponente somente obtém essa solução quando está considerando um sistema de várias entradas fuzzy.

Sistema de Mensagens

Em jogos com estilos mais voltados para batalha em campo aberto, o gameplay pode ser descrito como um modelo físico do mundo com tratamento dos comandos de entrada (significando que o gameplay envolve tratar os comandos de entrada dos jogadores utilizando a parte do código que trata da física do jogo para movimentar os personagens e realizar a colisão entre os mísseis com os jogadores). Devido a isso o uso de sistemas de mensagens é apropriado. E mais, a maioria desses jogos é multiplayer utilizando um modelo cliente servidor e ainda nesses casos a utilização de sistemas de mensagens auxiliar na comunicação entre servidor e clientes.

Sistema de Script

Alguns jogos mais modernos utilizam uma linguagem de script de alto nível Todos os elementos do ambiente, tais como, inimigos, diálogos, interação do jogador com o mundo e "cut scenes[6]" estão associados, totalmente ou parcialmente, ao sistema de script. A utilização de sistemas de script auxilia em geral a atividade de contar a história. Nos demais jogos, o uso de scripts auxilia na inserção de "cut scenes" e movimentações de câmera.

[6] Cut Scenes são seqüências de um jogo as quais o jogador não possui controle. São normalmente utilizadas para introduzir uma história ou prover informações, atmosfera, diálogos e pistas.

2.1.5 Plataforma

Jogos de plataforma são assim chamados por compreenderem jogos nos quais o fator gravidade restringe a viagem do jogador através de superfícies horizontais que são referenciadas como plataformas.

Este gênero é um dos primeiros para computador e tornou-se muito popular. No entanto, como o crescimento da popularidade dos jogos com gráficos tridimensionais o gênero foi perdendo espaço.

Tradicionalmente, jogos de plataforma são bidimensionais, e apresentam-se através de uma visão lateral do mundo (ver Figura 6). Com o advento dos gráficos tridimensionais removeu-se a restrição de visão e criaram-se mundos com plataformas totalmente tridimensionais (ver Figura 7). Entretanto, isso acarretou a perda da simplicidade de controle e gameplay dos jogos bidimensionais. Apesar de tudo, jogos de plataforma 3D não são incomuns (por exemplo, Super Mario 64 [Mario64] e Donkey Kong Jungle Beat [DKJB]). Séries de jogos de plataformas notáveis incluem Donkey Kong [DKJB], Super Mario Bros. [MarioBros], Sonic the Hedgehog [Sonic], Spyro the Dragon [Spyro] e Crash bandicoot [Crash].

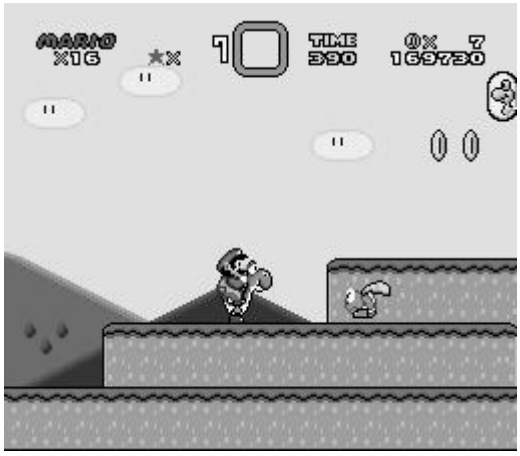

Figura 6 Super Mario World

Figura 7 Super Mario 64

TÉCNICAS

A seguir as técnicas utilizadas no gênero Plataforma.

Máquina de Estados Finitos

As máquinas de estados finitos também são úteis nesse gênero. Os jogos de plataforma apresentam inimigos simples e diretos com geralmente poucos comportamentos exibidos. Por questões de simplicidade, a maioria dos inimigos segue comportamentos padrões, ou seja, movimentam-se em uma direção até encontrar um obstáculo e então invertem a sua direção e deslocam-se até o obstáculo seguinte para repetir novamente o comportamento. Exceto no caso de chefes em que o comportamento é mais complexo e geralmente baseado em diferentes e inúmeros estados ou scripts [Schwab 2004].

Sistema de Mensagens

O estilo quebra-cabeças de grande parte dos jogos de plataforma propicia a utilização de mensagens de eventos para notificar inimigos e elementos do ambiente sobre mudanças no estado do jogo. Por exemplo, ao pular sobre um determinado botão, esse deve informar ao portão localizado no final da fase para permanecer aberto durante um certo intervalo de tempo. Essa comunicação pode ser realizada por um Sistema de Mensagens.

Sistema de Script

A utilização de scripts é uma forma natural de desenvolver a IA desses jogos devido à natureza comportamental dos inimigos, sejam inimigos comuns encontrados durante a travessia da fase ou estágio ou chefes inimigos enfrentados ao final de determinadas fases. O comportamento desses elementos tende a seguir um determinado padrão e repetir determinadas ações continuamente.

O uso de script em jogos de plataforma permite um controle fino sobre o fluxo de partes específicas do jogo, como o encontro com o chefe inimigo ao final de uma fase. É comum em jogos de plataforma existir ainda um personagem em jogo com a função exclusiva de auxiliar o jogador com relação aos comandos e movimentos especiais do jogador. Script pode auxiliar na tarefa de adicionar as ações a esse personagem, tais como diálogos e esperar que o jogador pratique os movimentos ensinados.

Data-Driven Systems

A movimentação da câmera para jogos de plataforma tridimensionais normalmente torna-se bastante complexa. Conseqüentemente, caminhos pré-definidos podem ser construídos através de um editor de nível para esses jogos caso nenhuma solução algorítmica seja adequada.

Os Data-Driven Systems podem auxiliar também na escolha da localização dos inimigos dentro do jogo, ou seja, no povoamento dos níveis do jogo. E para isso o sistema pode utilizar informações sobre a movimentação dos diferentes inimigos a serem incorporados ao nível assim como informações sobre os padrões de movimentação mais comuns dos jogadores. Dessa forma o sistema pode prever onde o jogador deve estar a cada instante do jogo e colocar um inimigo nessas posições para dificultar a passagem do jogador e tornar o jogo mais difícil e interessante.

2.1.6 Esportes

Os jogos de esporte emulam a prática dos tradicionais esportes físicos tais como futebol, futebol americano, vôlei, vôlei de praia, boxe, golfe, tênis, boliche, etc. Alguns enfatizam atualmente a prática do esporte do ponto de vista do atleta, o que significa que em um jogo de futebol, por exemplo, o jogador é responsável por movimentar os jogadores, driblar e chutar a bola (tais como FIFA Soccer [FIFA], NBA Live [NBA] e Madden NFL [NFL]). Enquanto outros enfatizam a prática do esporte do ponto de vista do técnico ou treinador e para isso considera a estratégia por trás do esporte (tais como Championship Manager [CM]) nos quais o jogador assume o papel de um técnico ou dirigente do time e deve lidar com atividades relacionadas ao gerenciamento da equipe, tais como, treinamento, negociação de jogadores (compra e venda), entre outras. Outros ainda satirizam o esporte através de um efeito cômico (tal qual em Arch Rivals [ArchRivals]).

Figura 8 FIFA Soccer 2004

TÉCNICAS

A seguir as técnicas mais encontradas nos jogos de Esporte.

Máquina de Estados Finitos e Máquina de Estados Fuzzy

Os gêneros de jogos que se comportam de maneira cíclica são mais simples de adaptar-se a um modelo de Inteligência Artificial baseado em estados do que os jogos com comportamento mais dinâmico. Todos os jogos desse gênero seguem um conjunto de regras (por exemplo, em futebol existem o estado lateral, estado escanteio e estado pênalti), e a estrutura da IA dentro das regras do jogo é baseada em estados. Mas dentro de certos estados, as decisões a serem tomadas tanto pelo técnico quanto pelos jogadores podem não ser claras e nítidas. Nesse caso, decisões *fuzzy* devem ser tomadas em quase todos os jogos de esporte, e FuSM's pode prover essa tomada de decisão nebulosa.

Data-Driven Systems

Com uma grande quantidade de jogadores, dados estatísticos e animações, jogos de esporte contam com no mínimo alguns "data-driven AI". E mais, com a tendência por jogos mais realísticos e ao mesmo tempo online (tempo real), "data-driven systems" facilitará o tuning da IA.

Alguns das técnicas mais usadas são:

- Playbooks

Ao invés de criar todo o comportamento de um determinado personagem, a melhor maneira consiste em criar comportamentos atômicos que os jogadores controlados por IA possam realizar e então possuir um editor ao qual os designers possam encadear esses comportamentos atômicos em comportamentos mais complexos para os personagens e times. Dessa maneira os designers podem experimentar os novos comportamentos e escolher dentre os melhores enquanto os programadores de IA podem concentrar-se na criação de novos comportamentos.

- Animation Picking

Tornando possível definir, seja através de um editor visual ou de alguma ferramenta de scripting, as condições específicas para articular a animação adequada para um determinado comportamento, designers podem rapidamente esclarecer os tipos de animações que fazem sentido para cada ação dentro do jogo e podem mudar ou expandir a lista de animações quando necessário sem mudança de código.

- Player Statistics

Nesse nível, os jogadores necessitam de dados estatísticos que se aproximem e reproduzam a vida real. Estatísticas adicionais devem ser criadas ainda de maneira que a gama de atributos esteja relacionada à forma de simulação do jogo. No caso de um jogo de esporte, os atributos possíveis são força, velocidade, agilidade, habilidade, etc.

Sistema de Mensagens

Com vários jogadores comunicando-se entre si, e em um ambiente dinâmico, faz total sentido utilizar um sistema de mensagem dentro dos jogos de esporte.

2.1.7 Simulação

Jogos de Simulação objetivam simular atividades específicas da vida real da forma mais realística possível. Como por exemplo, pilotar um avião considerando aspectos como física e outras limitações do mundo real.

Alguns requerem um grande desafio de leitura antes de iniciar o jogo, por exemplo Flight Simulator 2004: A Century of Flight [FlightSimulator] requer o conhecimento sobre os mecanismos e instrumentos de uma aeronave para poder realizar decolagens, vôos e atrelagens. Dessa forma antes de iniciar o jogo é necessário ler e reler o manual para aprender esses conceitos. Enquanto outros jogos possuem um simples tutorial, por exemplo, SimCity [SimCity] simula uma cidade e envolve conceitos comuns e cotidianos, assim não necessita de nenhum treinamento. Diferentemente dos demais gêneros, muitas simulações não possuem um conjunto de metas a serem alcançadas com o intuito de vencer o jogo.

Alguns dos jogos desse gênero, como simuladores de vôo, possuem um público limitado, enquanto outros, como The Sims [Sims] possuem um enorme público.

Simuladores de vôo é um subgênero de simulação, assim como simuladores de guerra. Jogos como The Sims, SimCity e SimEarth [SimEarth] são combinações de simulação e estratégia.

Figura 9 Flight Simulator: A Century of Flight

Figura 10 The Sims 2

TÉCNICAS

A seguir as técnicas comuns ao gênero Simulação.

Sistema de Informação Baseado na Localização

São rotinas de ajuda que melhoram o processo de tomada de decisão através da adição de informações ao motor de IA. Essa informação extra é armazenada em um banco de dados centralizado que é associado ao mundo de alguma maneira. Estes dados podem ser vistos como dados de percepção especializados que podem incluir lógica e inteligência de baixo nível.

As formas mais conhecidas desses sistemas são apresentadas abaixo.

- Influence Maps

Os Mapas de Influência (IM) estão se tornando uma das estruturas de uso mais comum dentre as técnicas utilizadas em jogos. A sua estrutura genérica e não limitada os tornam quase tão populares quanto as FSM's em facilidade de implementação e adaptabilidade aos diferentes problemas da IA em jogos. O termo Mapa de Influências refere-se ao uso de um array de dados simples, onde cada elemento representa dados sobre uma posição específica do mundo. Mapas de Influência são normalmente conceituados como uma malha 2D que reveste o mundo. A resolução da malha depende do tamanho mínimo dentro do espaço do jogo ao qual é necessário adicionar informação (compromisso entre tamanho dos dados e precisão da influência) [Schwab 2004].

IM's são estruturas para armazenar dados do mundo para auxiliar o planejamento de alto nível do jogo. Em um jogo de estratégia de guerra pode-se utilizar um IM para armazenar o número de unidades abatidas em cada célula (quadrado da malha). Os dados contidos no IM podem ser extraídos, através de uma TA como veremos a seguir, para afetar o sistema de navegação de maneira que unidades não continuarão a tomar caminhos por áreas com alto índice de mortalidade.

- Smart Terrain

Essa técnica tornou-se popular através do jogo The Sims (de fato, o termo foi criado por Will Wright, o criador de The Sims). Essa técnica coloca lógica e dados comportamentais de como usar várias características do mundo e objetos do jogo dentro dos próprios objetos. No jogo The Sims, personagens do jogo são motivados pelas suas necessidades que podem ser atendidas através da conexão com os vários objetos do jogo. Dessa forma, uma geladeira satisfaz a necessidade de comer assim como uma cama satisfaz a necessidade de dormir. O personagem então navega pelo mundo tentando satisfazer suas necessidades não atingidas em um determinado instante de tempo. E realiza isso escutando mensagens enviadas por objetos inteligentes próximos. Essas mensagens comunicam ao personagem a categoria da necessidade que cada objeto satisfaz, e o personagem está livre para usar qualquer objeto para satisfazer-se [Schwab 2004].

Essa técnica empacota no próprio objeto todas as animações de interação, sons e quaisquer outros dados especiais que o personagem pode necessitar para utilizá-lo. Assim, objetos podem ser adicionados a qualquer momento.

- Terrain Analysis (TA)

A obtenção de vários atributos e dados estatísticos dentro de um Mapa de Influência é somente uma parte do problema. É necessário ainda fazer uso desses dados. A TA compreende uma família de métodos que faz uso das técnicas de IM para prover o sistema de IA com número crescente de informações estratégicas sobre mapas, especialmente mapas gerados randomicamente e que não possuem o benefício de terem sido projetados por designers. TA são métodos melhores descritos como reconhecimento de padrões especializado dentro de um array de dados provido por um IM. Ou seja, procura-se dentro de um array de dados de um IM por informações importantes que podem ser exploradas ou que necessitam ser considerado para a tomada de decisões de nível tático e estratégico [Schwab 2004].

Continuando o exemplo anterior no qual foi considerado um jogo de estratégia de guerra, a TA serviria para extrair os dados de índices de mortalidade do IM para auxiliar o planejamento de alto nível. O IM é somente uma estrutura de dados. A transformação desses dados em informações relevantes é responsabilidade da TA.

Redes Bayesianas

São grafos direcionados e acíclicos cujos nós representam variáveis e arcos representam a dependência de relacionamento entre as variáveis. Redes Bayesianas são comumente utilizadas na modelagem do conhecimento em sistemas de suporte a decisão [Russell e Norvig 2003].

Jogos de Simulação utilizam essa técnica para a tomada de decisões de forma racional mesmo em situações desfavoráveis onde não existem informações suficientes para provar que uma dada ação funcionará [Russell e Norvig 2003]. Em jogos como SimCity, por exemplo, a simulação da satisfação dos habitantes da cidade pode ser perfeitamente realizada através de uma rede Bayesiana. A satisfação pode ser obtida através de uma função que considera várias variáveis, tais como habitação, transporte, saúde, segurança, dentre outros.

Redes Neurais

Rede Neural é um grupo neurônios artificiais interconectados que utilizam modelos matemáticos e computacionais para o processamento. Baseia-se em uma abordagem Conexionista[7] para a computação. Redes Neurais são comumente utilizadas na

[7] O princípio central do Conexionismo é que os fenômenos mentais podem ser descritos por uma rede interconectada de unidades simples.

aproximação de funções, predição de séries temporais, classificação e reconhecimento de padrões.

Essa técnica é utilizada em Jogos de Simulação principalmente para o reconhecimento de padrões. O IA do jogo pode analisar o comportamento e atividades do jogador para extrair determinados padrões na tentativa de identificar a sua estratégia de jogo.

2.1.8 Corrida

Os jogos de corrida são um dos gêneros mais tradicionais. Eles normalmente colocam o jogador no assento do motorista de um veículo de alta performance e requerem que o jogador dispute uma corrida contra outros jogadores, ou às vezes contra o tempo.

Esse gênero emergiu no início dos anos oitenta e se mantêm com muita popularidade ainda nos dias de hoje. Outra característica desse gênero é o fato de revolucionar a área de jogos em termos gráficos e de performance. Um subgênero popular dos jogos de corrida é os jogos de corrida de kart, que simplificam os controles dos veículos e introduzem vários obstáculos às pistas.

Jogos de corrida de grande popularidade são Need for Speed [NeedforSpeed], Gran Turismo [GranTurismo] e Mario Kart [MarioKart].

Figura 11 Gran Turismo

TÉCNICAS

A seguir as técnicas mais encontradas nos jogos de Corrida.

Máquina de Estados Finitos

Jogos de Corrida seguem geralmente uma mesma fórmula. São definidos pelas leis da física e possuem objetivos simples, como alcançar a linha de chegada ou resgatar um determinado pacote sobrevivendo aos ataques dos outros jogadores. Dessa forma, os estados de um jogo de Corrida seguem um fluxo comum (Largada, Corrida, Fora da Pista, Chegada), o que os torna candidatos ideais para FSMs .

Sistema de Script

O gênero de Corrida geralmente segue algum tipo de história, apesar de algumas serem bastante genéricas e não limitadas, e funcionam bem com o paradigma de script. Além disso, alguns dos sistemas de tráfego de trânsito e de pedestres podem ser combinados com um sistema de script, onde vários padrões de movimentação são scripts e interagem com o mapa de ruas da cidade. No caso de um carro repentinamente perder o controle e dirigir-se à calçada, o comportamento dos pedestres deve adaptar-se à nova realidade. Os pedestres devem ter os seus comportamentos alterados, excedendo o script normal, para fugir do atropelamento eminente.

Normalmente o sistema de script é somente uma camada com um sistema reativo integrado para afetar os comportamentos descritos no script de formas específicas.

Sistema de Mensagens

O sistema de tráfego e de pedestres utilizam em sua maioria sistemas de mensagem para comunicação entre os agentes e coordenação dos movimentos da mesma forma e complexidade que acontece na vida real.

Algoritmos Genéticos

Algoritmos Genéticos são fundamentados no princípio da evolução. O princípio da evolução postula que as espécies que habitaram e habitam o nosso planeta não foram criadas independentemente, mas descendem umas das outras, ou seja, estão ligadas por laços evolutivos. A evolução, por sua vez, é impulsionada pelo fenômeno da seleção natural. São esses conceitos que embasam a técnica de Algoritmos Genéticos. Para maiores detalhes ver referências [Mitchell 1997].

Alguns jogos de corrida possuem uma enorme quantidade de carros (por exemplo, Gran Turismo tem mais de 500) e cada um requerendo a adaptação / evolução de suas habilidades relacionadas à performance e manejo no volante. Então algumas companhias têm utilizado técnicas para automatizar a atividade de tuning através de uma simples aplicação de algoritmo genético para modificar os parâmetros de performance dos carros até

a obtenção de resultados ótimos. Esses resultados são então armazenados e utilizados diretamente durante o jogo.

2.1.9 Luta

Jogos de luta enfatizam o combate um-a-um entre dois jogadores, dos quais um pode ser controlado por computador. Esses jogos normalmente focam em artes marciais mais atrativas artisticamente, tais como Karate e Kung-fu, e em outras formas de combate sem armas. Alguns desses jogos podem ainda empregar o uso de armas de mão como espadas.

Esse gênero surgiu nos anos oitenta e tornou-se um fenômeno com o lançamento de Street Fighter 2 [StreetFighter]. O gênero ainda é bastante popular. Outros jogos populares neste gênero são King of Fighters [KingOfFighters], Mortal Kombat [MortalKombat], Street Fighter [StreetFighter] e Virtua Fighter [VirtuaFighter].

Figura 12 Street Fighter 2

TÉCNICAS

A seguir as técnicas comuns ao gênero Luta.

Máquina de Estados Finitos

Jogos de Luta são normalmente baseados em estados com o oponente controlado por IA realizando movimentos, mantendo-se parado ou respondendo à determinada colisão. Uma FSM pode atender à demanda da maioria dos jogos desse gênero por prover uma estrutura complexa o suficiente para adicionar estados e a complexidade desejada sem maiores problemas.

Data-Driven Systems

Devido a enorme quantidade de personagens, movimentos, bloqueios, arremessos e combos, e especialmente dado o nível de tuning e balanço necessário para esse gênero, é importante construir um editor gráfico ao qual o designer possa adicionar comportamentos

na forma de scripts. Normalmente, para cada movimento é criado um script para permitir maior precisão da determinação dos tempos de ataques, defesa, efeitos sonoros e colisão e ainda no tamanho da área de colisão assim como o dano infligido. O sistema de colisão é normalmente muito complexo (mesmo o primeiro Street Fighter 2 possuía muitas áreas de colisão por sprite[8]), com tabelas de dados detalhando as animações a serem executadas se determinadas áreas do inimigo foram atingidas, ou bloqueadas, ou qualquer outra coisa. Tabelas adicionais descrevem a personalidade de cada personagem tais como quão agressivo ou quão defensivo é um personagem.

Sistema de Script

Em adição à noção de que os designers necessitam de controle completo sobre as animações de luta (dessa forma, designers requerem um script para detalhar tudo que necessita acontecer durante cada movimento), elementos da história, entre outros.

Os sistemas de script são úteis ainda para adicionar "cut-scenes" ao jogo, como exemplo, quando uma luta inicia e os personagens entram na arena, ou depois de algum personagem ser declarado vencedor e executar algum tipo de dança. Movimentos complexos podem requerer a utilização de scripts.

2.2 ANÁLISE COMPARATIVA

Após o estudo realizado através da análise dos gêneros de jogos e das técnicas mais utilizadas alcança-se a seguinte tabela de mapeamento entre gêneros e técnicas.

Tabela 2 Comparativo de Técnicas versus Gêneros

	RPG	Adventure	Estratégia	FTPS	Plataforma	Esporte	Simulação	Corrida	Luta
Máquina de Estados Finitos	*	*	*	*	*	*		*	*
Máquina de Estados Fuzzy			*	*		*			
Lógica Nebulosa (Fuzzy)		*							
Sistema de Mensagens	*	*	*	*	*	*		*	
Sistema de Script	*	*		*	*			*	*
Data-Driven Systems			*		*	*			*
Planejamento			*						
Inteligência Artificial Hierárquica			*						
Algoritmos Genéticos								*	

[8] Sprite consiste em uma pequena imagem que pode ser movimentada ao redor da tela.

Sistema de Informações Baseado na Localização							*		
Rede Bayesiana							*		
Rede Neural							*		

2.3 OUTRAS TÉCNICAS

Apesar do estudo ter identificado as técnicas mais comuns dentro de cada gênero de jogo, a lista de técnicas propostas acima não é exclusiva. Ou seja, existem técnicas mais comuns em outros gêneros, mas que são aplicáveis também aos demais. A técnica de Redes Neurais, por exemplo, está associada na pesquisa ao gênero de Simulação, mas pode ser perfeitamente utilizada para o gênero de corrida, para treinar a movimentação dos automóveis. A Rede Neural pode ser treinada a partir de dados obtidos por pessoas utilizando o jogo. Dessa forma a RN é ensinada a pilotar o automóvel de forma semelhante à pessoa que forneceu os dados. Outras técnicas podem ser também extensíveis e aplicáveis entre os gêneros.

Além das técnicas citadas na pesquisa existem outras técnicas comuns e úteis a todos os gêneros e exatamente por isso foram excluídas da pesquisa realizada e serão comentadas separadamente.

2.3.1 Técnicas

As técnicas citadas a seguir não estão relacionadas de forma direta a nenhum dos gêneros apresentados. No entanto são encontrados com freqüência em vários estilos diferentes de jogos.

SISTEMA DE NAVEGAÇÃO

O sistema de navegação é responsável por encontrar e traçar rotas de movimentação dentro do jogo. A movimentação deve respeitar determinadas regras e leis, sejam as leis da física ou outras próprias do jogo. Dessa forma o sistema de navegação tem a função tanto de definir a rota a ser seguida como de desviar de possíveis colisões.

A* (A Estrela)

O A* é um algoritmo para encontrar caminhos entre dois pontos de um mapa. Apesar de existirem vários algoritmos diferentes para o calculo de rotas, o A* encontra o caminho mais curto, se existir, e de forma relativamente rápida.

O A* é direcionado ou dirigido, ou seja, não realiza buscas cegas (como um rato em um labirinto), mas ao contrário disso, determina a melhor direção a ser explorada e algumas

vezes realiza “backtracking” para tentar alternativas diferentes. Essas características tornam o algoritmo diferente dos demais devido a sua eficiência e flexibilidade.

Pathfinding Estático e Dinâmico

O pathfinding é uma técnica muito utilizada no desenvolvimento de jogos e envolve o processo de determinação do melhor caminho a percorrer dados os pontos de origem e destino.

No pathfinding estático o caminho e calculado a partir de rotas pré-definidas (pré-computadas). É comum em aplicações de menor porte e estáticas nas quais é possível definir as rotas em tempo de projeto. Na abordagem dinâmica as rotas são calculadas em tempo de execução do jogo. É comum em aplicações de maior porte e com características dinâmicas (vários objetos em movimentação).

BLACKBOARD

A técnica Blackboard é constituída sobre a metáfora do quadro negro encontrado em salas de aulas de colégios e da sua aplicação como ferramenta de resolução de problemas. A técnica é útil por prover um espaço compartilhado no qual os problemas podem ser quebrados e resolvidos de maneira incremental.

Em IA, a técnica Blackboard consiste em um ambiente no qual os agentes podem colaborar para realizar tarefas complexas que estendem as suas capacidades individuais [Deugo et al.2001].

APRENDIZAGEM

A aprendizagem baseia-se no conceito de adaptação da IA de forma genuína com o intuito de forçar os jogadores a procurar continuamente por novas estratégias para superar a IA, ao contrário de somente refinar técnicas simples.

O comportamento dos inimigos controlados por IA geralmente segue um determinado padrão de comportamento e movimentação. E para combater esses inimigos, os jogadores criam estratégias através de identificação dos pontos vulneráveis do inimigo. Após a identificação dos pontos fracos resta ao jogador refinar a sua estratégia para torná-la mais eficiente. Com a aprendizagem os inimigos aprenderão a estratégia do jogador e se comportarão diferentemente a cada embate.

A aprendizagem e adaptação ocorrem de duas possíveis formas em jogos. A primeira é chamada aprendizagem supervisionada e extrai estatísticas do jogo a serem usadas por uma camada de IA para modificar o comportamento dos agentes. As decisões acerca do tipo de estatísticas a serem extraídas e de sua interpretação em termos de mudanças necessárias aos comportamentos são realizadas pelo designer de IA. A segunda técnica é referenciada como aprendizagem não-supervisionada e aplica algoritmos de aprendizagem aos comportamentos dos agentes e requer mínima contribuição humana para

especificar aspectos relativos à adaptação dos comportamentos dos agentes. A terceira é chamada de aprendizagem por reforço e utiliza funções de utilidade para determinar o quão boa ou ruim foi uma determinada ação realizada pelo agente. O agente é então recompensado ou punido por suas ações e tenta maximizar suas recompensas e minimizar as suas punições.

2.4 CONCLUSÕES

A análise dos diversos gêneros de jogos existentes confirma o uso comum de determinadas técnicas em detrimento de outras. Apesar de as técnicas não serem exclusivas, ou seja, técnicas mais comuns de um gênero poderem ser utilizadas em um outro gênero, observa-se um padrão - os gêneros costumam utilizar as mesmas técnicas.

Tabela 3 Lista de Técnicas

Lista de Técnicas
Máquina de Estados Finitos
Máquina de Estados Fuzzy
Lógica Nebulosa (Fuzzy)
Sistema de Mensagens
Sistema de Script
Data-Driven Systems
Planejamento
Inteligência Artificial Hierárquica
Algoritmos Genéticos
Sistema de Informações Baseado na Localização
Rede Bayesiana
Rede Neural
Sistema de Navegação
Blackboard
Aprendizagem

A descoberta dessa característica, de jogos do mesmo gênero utilizarem técnicas semelhantes, permite à área de Inteligência Artificial para jogos progredir e trilhar o mesmo caminho já percorrido pelas outras áreas (gráfica, áudio, rede, etc). A identificação de padrões foi que possibilitou o desenvolvimento de motores nas demais áreas. E da mesma forma esse trabalho comprova ser possível para a área de IA.

CAPÍTULO 3

MOTORES DE IA

“You must do the thing you think you cannot do.”
— ELEANOR ROOSEVELT

A evolução dos motores e *frameworks* na área de entretenimento digital seguiu os passos da técnica dividir para conquistar. E após o desenvolvimento das áreas em separado o passo seguinte foi reunir todo esse conhecimento para a criação de um motor de jogos.

Os motores de jogos concentram o conhecimento nas áreas correlatas e as unem para formar uma aplicação única. A maioria dessas aplicações atuais possui grande parte dos recursos necessários para o desenvolvimento de jogos integrados e com a facilidade provida pela construção de ferramentas para facilitar tanto os desenvolvedores de jogos quanto os designers. A área de IA, apesar de trilhar um caminho mais árduo, consta em grande parte dos motores de jogos, mas ainda de maneira tímida, com poucos recursos.

3.1 DISCUSSÃO

Dessa forma a análise dos motores de Inteligência Artificial coincide com um estudo dos motores de jogos existentes no mercado. Nessa seção os motores de Inteligência Artificial mais utilizados e famosos serão analisados considerando o estudo realizado e a abrangência dessas às técnicas identificadas.

3.1.1 Unreal Engine 3

A Unreal Engine 3 [Unreal] é um framework completo produzido pela Epic Games [Epic] para desenvolvimento de jogos para a nova geração de consoles (Sony Playstation 3 [Playstation] e Microsoft XBOX 360 [XBOX]) e de PC’s equipados com DirectX [DirectX], abrangendo uma vasta gama de tecnologias, ferramentas de criação de conteúdos e infra-estrutura de suporte requerida para os desenvolvedores. Vários jogos foram lançados utilizando essa engine, tais como, Unreal Tournament 2004 [UnrealTournament], Unreal Championship [UnrealChampionship], as séries Sprinter Cell e Harry Potter [HarryPotter], dentre outros. Assim como existem alguns anunciados para produção, como por exemplo, Men of Valor: Vietnam, 2015’s [MoV] e Star Wars: Republic Commando [SWRC].

A Unreal Engine 3 possui um sistema de IA de vários níveis com suporte a pathfinding, navegação de alto nível, tomadas de decisões individuais e IA baseada em times (sistema multiagentes precário). Abaixo descrevemos estes níveis em maiores detalhes.

- O sistema de pathfinding comporta objetos complexos tais como portas e elevadores permitindo a navegação entre os cenários onde os personagens podem apertar botões, abrir portas e desviar de obstruções temporárias para alcançar o seu destino.
- O sistema de navegação suporta ainda táticas de combate de curto prazo tais como fazer cobertura.
- A IA baseada em times é adequada para jogos do gênero FTPS e Estratégia. Esse framework suporta a coordenação de times, gerenciamento de objetivos e de metas de longo prazo.

A construção dos caminhos utilizados pelos sistemas de navegação e pathfinding é realizada através de uma ferramenta, a UnrealEd, na qual os designers podem visualizar e realizar a edição de forma gráfica. A ferramenta possui ainda recursos para customização e dicas para uma melhor utilização.

A Unreal Engine 3 conta ainda com um sistema de script e uma ferramenta visual para sua edição, a UnrealKismet. A ferramenta permite a artistas e designers controle virtualmente ilimitado sobre informações comportamentais de cada nível do jogo sem escrever uma linha de código.

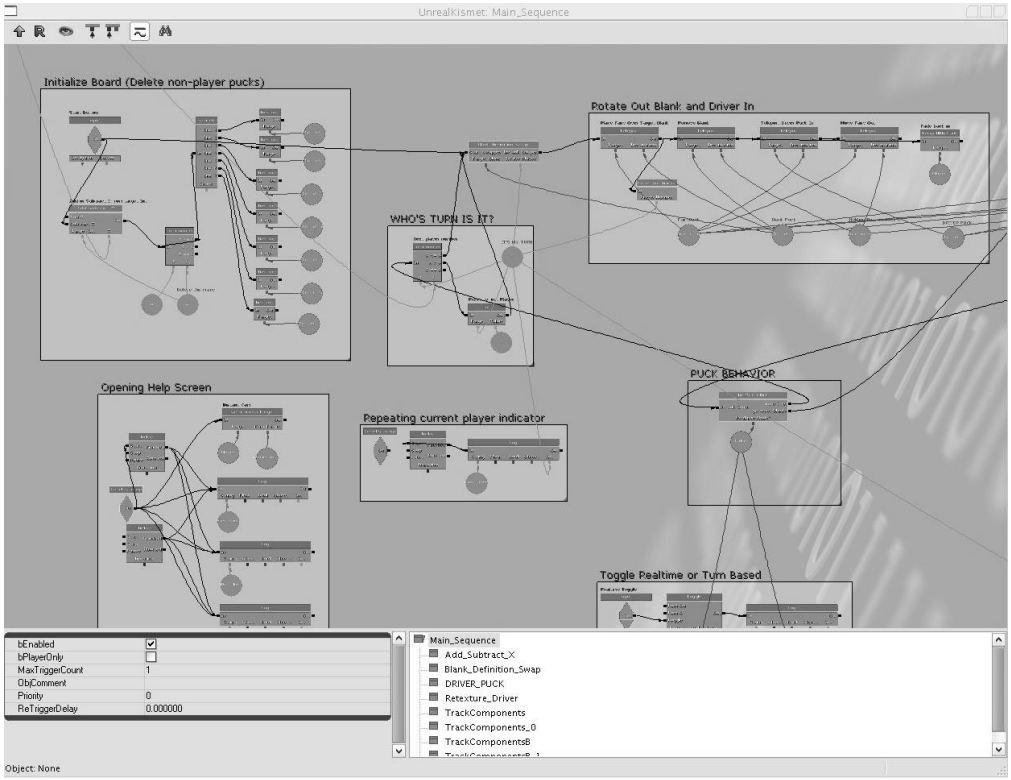

Figura 13 Ferramenta Unreal Kismet

A Unreal Engine 3 é um framework proprietário. Para utilização é necessário comprar a licença de uso e publicação de jogos sobre a sua plataforma. O preço de versão 3 do motor não se encontra disponível publicamente [UELicensing].

No entanto estudando os tipos e valores da versão anterior, Unreal Engine 2, é possível obter uma aproximação. Os tipos de licenças disponíveis são Royalty-Bearing License, Royalty-Free License e Custom License [UELicensing].

O custo de obtenção da Unreal Engine 2 para uma das plataformas apoiadas (PC, XBOX ou Playstation) é de $350.000,00 mais $50.000,00 para cada plataforma adicional sob a licença Royalty-Bearing. Essa licença obriga ainda a empresa a pagar uma taxa de 3% sobre o rendimento de cada jogo vendido, onde o rendimento do jogo é calculado como o preço de atacado subtraído da comissão da empresa manufatora. No caso de MMOG[9]'s, os royalties são cobrados da mesma forma que descrito anteriormente adicionando-se o rendimento sobre assinatura do jogo e propaganda dentro do jogo.

Sob a licença Royalty-Free, o custo para obtenção do motor para uma das plataformas disponíveis é de $750.000,00 mais $100.000,00 por plataforma adicional. Essa licença não possui cobrança sobre jogo vendido.

[9] Massive Multiplayer Online Game são jogos de computador que permitem que centenas e até milhares de jogadores interajam simultaneamente em um mundo de um jogo. Para isso os jogadores devem estar conectados à Internet.

A licença Custom é destinada para o desenvolvimento de produtos não-tradicionais, tais como aplicações voltadas para treinamento. Nesse caso a empresa interessada deve contatar a empresa EpicGames.

3.1.2 Source Engine

Desenvolvida pela ValveSoftware [Valve], a Source Engine [Source] é um framework e conjunto de ferramentas para a produção de jogos envolvendo animação de personagens, física, renderização baseada em shaders e uma avançada IA. A estrutura do motor serviu de base para a construção de vários jogos de sucesso, dentre eles, as séries Half-Life [Half-Life] e Counter-Strike [CS] e ainda Day of Defeat [DayOfDefeat]. A Source Engine juntamente com a Unreal Engine compõem os motores mais utilizados e citados como referência na área.

Os recursos de IA do motor consistem em um sistema de entrada e saída permitindo controle da IA por parte dos designers, um Data-Driven System. O motor conta ainda com um sistema de navegação onde personagens podem correr, voar, pular, agachar, subir escadas e degraus e esconder-se sob a terra. O motor permite tratar eventos relacionados à visão, audição e olfato e ainda adicionar relacionamentos que definem o status amigo ou adversário de outras entidades.

Um outro recurso importante do motor está relacionado ás batalhas dentro de um jogo. O motor permite a formação de uma esquadra ou pelotão com o intuito de trabalhar em conjunto, conhecendo o momento exato de avançar, recuar, mandar “cobrir” um determinado personagem, etc.

A Source Engine é proprietária e está disponibilizada através de uma licença paga. As informações sobre custos e formas de obtenção não são disponíveis publicamente. Para obter maiores informações é necessário entrar em contato e preencher um NDA.

3.1.3 Reality Engine

A Reality Engine [Reality] é um conjunto de ferramentas para o desenvolvimento de jogos desenvolvido pela Artificial Studios [Artificial]. O motor foi desenvolvido sobre a plataforma DirectX 9.0 e é compatível com o DirectX 7 e 8. Vários jogos foram construídos utilizando esse motor, como por exemplo, Armageddon [Armageddon], CellFactor [CellFactor], Metronome [Metronome] e Dreamers [Dreamers].

O motor foi criado para concorrer com os maiores motores do mercado, a Unreal e a Source. A qualidade do motor, principalmente gráfica, é indiscutível. Contando com vários

[10] Non-Disclosure Agreement é uma declaração formal confirmando que a informação confidencial provida será respeitada e não publicada.

recursos avançados não encontrados em outros motores, a Reality Engine despontou para o mercado como um motor com qualidade semelhante às gigantes concorrentes e a um preço mais acessível. Exatamente por esses motivos, a EpicGames, empresa produtora da Unreal Engine, comprou a Reality Engine para descontinuar o projeto e contratou parte da equipe responsável, em especial o arquiteto de software, para trabalhar na Unreal Engine.

A Reality Engine possui um módulo de IA com suporte a algumas técnicas importantes. A primeira técnica é pathfinding, tanto predeterminado quanto dinâmico com desvio de obstáculos. O motor possui também um sistema de tomada de decisões baseadas em máquinas de estados adaptativas. O módulo de IA permite ainda adicionar aos agentes inteligentes a habilidade de responder a estímulos de visão e sonoros.

Todo o módulo responsável pela IA é escrito através do sistema de script do motor. O nome do script utilizado é Reality Script. O propósito disso é facilitar a extensão do motor. Outro ponto corresponde à arquitetura do motor. Para melhorar a performance a IA é executada no servidor e os eventos são replicados para os clientes.

O motor conta com vários editores com recursos visuais avançados para facilitar a edição do ambiente e controlar o comportamento dos personagens.

A Reality Engine é também um motor proprietária. O motor é comercializado sob duas licenças. Uma das licenças compreende a venda da Reality Engine enquanto a outra compreende a da Reality Engine SDK. A diferença essencial consiste na concessão co código fonte. A empresa Artificial Studios não disponibiliza publicamente informações acerca do custo de obtenção, para nenhuma das licenças.

A licença da Reality Engine garante o acesso a todo o código fonte, incluindo exemplos e demos tecnológicos. Essa licença permite o desenvolvimento de um produto comercial ou de inúmeros projetos não-comerciais.

A licença da Reality Engine SDK garante acesso somente ao código binário. A empresa fornece um desconto para empresas ou projetos com foco educacional de 50%.

3.1.4 Torque Game Engine

A Torque Game Engine [TGE], ou TGE, é uma versão modificada de um motor tridimensional para PC's desenvolvida originalmente pela Dynamix para o FPS Tribes 2 [Tribes2] em 2001. Atualmente a TGE está disponível sob licença da GarageGames [GarageGames] para desenvolvedores independentes e profissionais. A TGE é um motor multi-plataforma e segue a mesma categoria que os motores Unreal e Source com recursos gráficos, de áudio, redes, sistema de script e ferramentas auxiliares. Uma das principais diferenças entre a Torque Game Engine e as citadas anteriormente está no foco voltado para o desenvolvedor independente. Vários jogos foram publicados utilizando esse motor,

tais como, Rocket Bowl [RocketBowl], Gold Fever [GoldFever], e WildLife Tycoon: Venture África [WLTVA].

A TGE não contém um sistema ou módulo específico para Inteligência Artificial. O único recurso disponível no motor é o Torque Script – um sistema de script desenvolvido pela GarageGames. O Torque Script é composto por uma linguagem de script semelhante à linguagem C++ e possui modo debug local e remoto. O sistema de script possui ainda uma lista de funções complementares que incluem matemática, manipulação de objetos e arquivos, dentre outras.

A TGE é também uma proprietária. A TGE é distribuída sob duas licenças, a Indie License e a Commercial License. A Licença Indie é voltada para o desenvolvimento de jogos por companhia s de pequeno porte ou indivíduos financiando com recursos próprios o desenvolvimento do jogo. Qualquer uso diferente do mencionado faz parte do escopo da licença Commercial.

A licença Indie é voltada para empresas com lucro anual inferior a $250.000,00. O custo dessa licença é de $100,00 por programador utilizando a TGE ou acessando o código fonte dos jogos. Não existe nenhum custo ou taxa adicional associado às vendas do jogo assim como não é obrigatória a publicação do jogo através da GarageGames.

A licença Commercial é voltada para companhias de qualquer tamanho, entidades governamentais ou indivíduos para criarem e publicarem ou venderem produtos baseados nos executáveis criados a partir do código fonte do motor. O custo da licença Commercial do motor é $495,00, sem taxas adicionais. E a empresa licenciada não necessita de qualquer permissão ou aprovação para publicar, vender e explorar seus produtos.

3.1.5 CryEngine

A CryEngine [CryEngine] é um motor para o desenvolvimento de jogos desenvolvido pela CryTek [CryTek]. Tornou-se conhecido publicamente através do jogo Far Cry [FarCry]. Possui suporte para OpenGL [OpenGL] e DirectX 8/9 e pode ser utilizada para o desenvolvimento de títulos na plataforma de PC's e ainda para os consoles XBOX, PlayStation 2 e GameCube [GameCube]. Os principais jogos lançados utilizando esse motor são: X-Isle: Dinosaur Island e Far Cry.

A CryEngine possui como uma de suas grandes características a possibilidade de criação de mundos enormes e permitir a visualização através do terreno, por parte do jogador, a distâncias próximas a 2km. Isso significa que o jogador pode visualizar objetos localizados no mundo do jogo a uma distância superior à encontrada na maioria dos jogos.

O seu sistema de IA permite a IA baseada em times e a definição de comportamentos através de scripts. O conjunto de ferramentas disponibilizado habilita o usuário a criar inimigos e comportamentos customizados escrever uma linha de código.

A CryEngine é também disponibilizada através de uma licença paga. Não existe nenhum programa para licenciamento do motor de forma não comercial, como por exemplo, para fins educativos. Não existem informações disponíveis sobre forma e custo para obtenção do motor. Para maiores informações é necessário contatar a empresa e preencher um NDA.

3.2 ANÁLISE COMPARATIVA

Essa seção realiza uma comparação entre os motores pesquisados utilizando os critérios de qualidade descritos abaixo.

ABRANGÊNCIA

Nesse critério os motores serão comparados entre si com relação à abordagem das técnicas descritas no capítulo 2.

Tabela 4 Comparativo dos Motores segundo critério Abrangência

	Unreal	**TGE**	**Reality**	**Source**	**CryEngine**
Geral					
Blackboard					
Data-driven System	*	*		*	
Sistema de Script	*		*		
Sistema de Mensagens				*	*
Máquina de Estados					
Máquina de Estados Finitos	*		*	*	*
Máquina de Estados Fuzzy					
Navegação					
A*	*			*	
Pathfinding Estático	*		*	*	
Pathfinding Dinâmico	*		*	*	
Técnicas Avançadas					
Planejamento					
Sistema de Informação Baseado na Localização					
Algoritmo Genético					
Redes Bayesianas					
Redes Neurais					
Aprendizagem					

Supervisionada					
Por Reforço					
Não-Supervisionada					

Considerando a tabela 4 obtemos o percentual de abrangência de cada um dos motores com relação às técnicas.

Tabela 5 Porcentagem de abrangência

	Unreal	**TGE**	**Reality**	**Source**	**CryEngine**
Número de técnicas abordadas	6	1	4	6	2
Total de técnicas	17	17	17	17	17
Porcentagem (%)	**35,3**	**0,05**	**0,23**	**35,3**	**0,12**

Ou seja, a maior porcentagem de abordagem das técnicas é dos motores Unreal e Source. E em média a porcentagem de abordagem é de 14,2%.

CUSTO

A análise do custo deve considerar tanto a característica do motor relacionado à propriedade (proprietário ou livre) quanto o custo de obtenção de uma licença, no caso de um motor proprietário.

No entanto, considerando que todos são proprietários e que não é possível obter os custos de obtenção de todos, a comparação fica limitada e nesse aspecto iguala todas os motores.

3.3 CONCLUSÕES

Na atualidade não existem motores de IA para jogos disponíveis de forma separada das áreas de desenvolvimento de jogos correlatas, tais como gráficos, áudio e física. A IA encontra-se embutida de forma intrínseca às essas outras áreas.

A análise comparativa demonstra que não existe nenhum motor disponível sobre uma licença livre (sem custos). Do ponto de vista da abrangência, os motores também não demonstraram muito avanço, com uma média de abrangência das técnicas de 14,2%.

CAPÍTULO 4

REVOLUTION AI ENGINE

“If you can ‘dream it’ you can do it”
— WALT DISNEY

Nesse capítulo será apresentado o processo de elicitação de requisitos da REvolution AI Engine, bem como o projeto de sua arquitetura. No projeto da arquitetura estão detalhados a modelagem e forma de implementação. Ao final são apresentados os resultados atingidos.

4.1 ELICITAÇÃO DE REQUISITOS

Os requisitos para a definição e construção do motor são fundamentados no estudo contidos nos capítulos 3 e 4. Com base nas técnicas descobertas é realizada a priorização para identificar as mais relevantes e úteis

A utilidade e relevância estão relacionadas principalmente à questão do uso mais freqüente das técnicas. Determinadas técnicas são amplamente difundidas e usadas na construção de jogos e dessa maneira não poderiam ficar de fora da lista de requisitos do motor. As máquinas de estados e o sistema de navegação são algumas das técnicas presentes nessa lista de técnicas essenciais.

Considerando esses aspectos, a tabela 5 apresenta as técnicas e suas prioridades para implementação no motor REvolution AI Engine. A tabela descreve ainda quais requisitos são contemplados pelo motor (elementos em destaque na tabela).

Tabela 6 Lista de Requisitos

Requisito	Prioridade
Máquina de Estados Finitos*	1
Sistema de Navegação*	2
Máquina de Estados Fuzzy*	3
Lógica Nebulosa (Fuzzy)	4
Sistema de Mensagens*	5
Inteligência Artificial Hierárquica*	6
Blackboard*	7
Data-Driven Systems	8
Sistema de Script	9
Planejamento	10

Sistema de Informações Baseado na Localização*	11
Rede Bayesiana	12
Algoritmos Genéticos*	13
Rede Neural*	14
Aprendizagem*	15

A abordagem desses requisitos alcança um total de 70,5% do total de técnicas pesquisas. Esse percentual é maior que o dos motores estudados na seção 3 (ver Tabela 5).

4.3 PROJETO DA ARQUITETURA

O projeto do motor Revolution AI Engine teve que partir do zero, principalmente devido à maioria dos motores existentes [RealityEngine, Torque, Unreal, CryEngine, Source] serem proprietários o que impossibilita a pesquisa e análise de suas arquiteturas.

O motor projetado busca manter uma estrutura modular com responsabilidades bem definidas para cada um dos seus componentes. Outra característica considerada para o desenvolvimento da arquitetura foi extensibilidade. Essa característica é essencial posto que o escopo desse projeto não aborda todas as técnicas possíveis e estudadas nesse trabalho. Dessa forma em uma próxima versão do motor torna-se mais simples a inserção de novos requisitos e técnicas.

A característica multiplayer dos jogos, por exemplo, não é abordada nesse trabalho, no entanto o projeto da arquitetura considera esse ponto de extensão em sua análise permitindo a posterior implementação dessa característica no motor para prover suporte a aplicações em rede.

4.3.1 Modelagem

A primeira regra para os projetos de jogos e de software de forma geral é a regra KISS: Keep It Simple and Straight. E para os que não consideram essa regra em seus projetos de arquitetura a tradução da sigla é Keep It Simple, Stupid.

Para a modelagem é utilizado o método Projeto Distribuído da IA (ver Apêndice B) pelo qual divide-se a inteligência dentro do motor em várias camadas.:

- Camada de Percepção / Eventos

Filtra os dados sensoriais de entrada por relevância dentre outros fatores.

- Camada de Comportamento

Descreve especificamente como realizar determinadas ações.

- Camada de Animação

Determina que animação realizar para melhor adaptar ao estado do jogo.

- Camada de Movimentação

Trata aspectos como pathfinding e colisões.

- Camada de Tomada de Decisão de Curto Prazo

Trata da visão de inteligência de uma entidade inteligente, primeiramente preocupado com a unidade somente.

- Camada de Tomada de Decisão de Longo Prazo

Trata da visão de inteligência de forma geral, como planejamento ou considerações baseadas em times e grupos.

- Camada de Informação Baseada na Localização

Inclui informação transmitida para as entidades através de mapas de influências, smart terrains e técnicas semelhantes.

A modelagem da arquitetura para prover suporte aos requisitos do motor e seguindo o modelo em camadas descrito é apresentada na Figura 14. Nessa figura são identificados vários elementos importantes. O **Ambiente** representa o mundo no qual o agente, elemento ou indivíduo inteligente do jogo está inserido e com o qual se comunica através de **Sensores** (responsáveis por tratar as percepções e eventos do mundo) e **Atuadores** (responsáveis por interpretar as respostas do agente às percepções do ambiente).

A execução interna do **Agente** ocorre da forma apresentada na figura através das camadas e setas indicando a ligação entre as camadas e o sentido do fluxo de informações. A Camada de **Percepção / Eventos** trata as percepções recebidas pelo agente por relevância e alimenta a Camada de **Decisão de Curto Prazo**. Essa camada, por sua vez, trata as informações recebidas e identifica a necessidade de comunicar-se com as Camadas de **Animação**, **Movimentação** e **Comportamento**. Caso a decisão tomada pela camada necessite de navegação, a Camada de Movimentação será acionada. Da mesma forma funciona para as Camadas de Animação e Comportamento.

A Camada de **Decisão de Longo Prazo** se comunica com a Camada de Decisão de Curto Prazo através de uma seta bidirecional. Ou seja, existe passagem de informações em ambos os sentidos. A Camada de Decisão de Curto Prazo alimenta com informações a Camada de Decisão de Longo Prazo e essa por sua vez indica quais as decisões de longo prazo a serem praticadas.

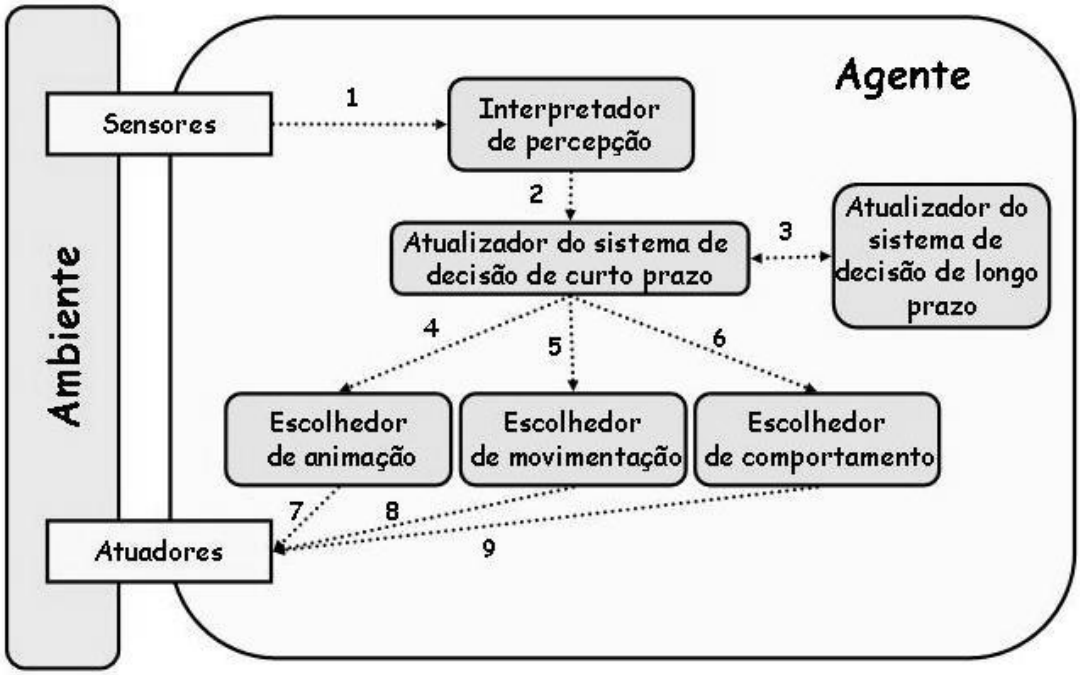

Figura 14 Arquitetura do Agente

A Figura 14 ilustra o motor do ponto de vista do agente e considera o formato de execução e a relação com o ambiente. A observação do motor de um ponto de vista mais abrangente é apresentada na Figura 15.

Os agentes se comunicam com o ambiente da forma apresentada na Figura 14. As diferenças consistem na presença de um Sistema de Informação Baseado na Localização e de um Escalonador.

O **Sistema de Informação Baseado na Localização** provê informação para todos os agentes do motor para servir de base para o planejamento de curto e longo prazo. Como já explicado anteriormente, as principais técnicas desse sistema servem somente para obter dados do ambiente. O tratamento desses dados deve ser realizado através de outras técnicas, como por exemplo, a técnica Terrain Analysis.

O **Escalonador** compreende o sistema de tempo real do motor. Esse sistema é essencial pois a maior parte dos recursos computacionais na área de jogos são alocados para a área gráfica restando pouco recursos para o processamento da IA. Então o tempo destinado para o processamento das atividades relacionadas a IA deve ser aproveitado ao máximo. Para maiores detalhes ver Apêndice A.

A Figura 15 acima destaca ainda uma outra característica presente na arquitetura: o suporte a vários agentes simultâneos. O tratamento dos vários agentes simultâneos associados ainda ao relacionamento e comunicação desses com o ambiente e entre si será realizado através de um **Sistema Multiagentes**.

Dessa forma a figura apresenta as principais características do motor: sistema multiagentes e sistema de tempo real.

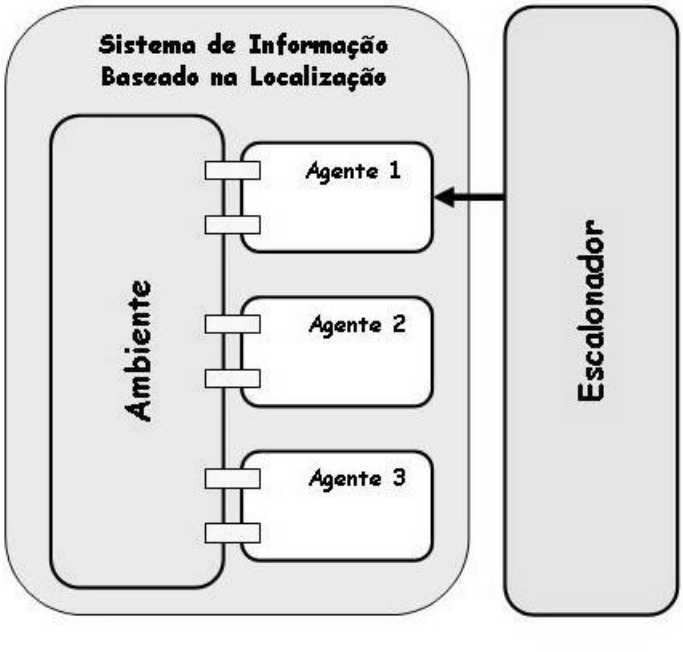

Figura 15 Arquitetura do Motor

4.3.2 Arquitetura

A apresentação da arquitetura está decomposta em vários módulos por motivos didáticos. Os módulos, no entanto, estão conectados por interfaces bem definidas que serão descritas no que se segue. Para maiores detalhes ver Apêndice C.

MÓDULO SISTEMA MULTIAGENTES

Esse módulo representa a base de toda a arquitetura na qual os demais módulos estão apoiados. O Sistema Multiagentes prove suporte para a criação de vários agentes os quais devem pertencer a um container de agentes. O container agrupa agentes com propriedades em comum. O módulo pode conter então vários containeres diferentes os quais estão interconectados por uma estrutura maior: o container principal. Esse container engloba todos os demais containeres.

Esses containeres (container de agentes e container principal) provêem serviços tais como o serviço de páginas amarelas (responsável por identificar os serviços disponíveis e realizados por cada agente) e brancas (responsável por encontrar os agentes e sua localização dentro do sistema).

MÓDULO AGENTE

Esse módulo está inserido dentro do módulo do Sistema Multiagentes e será aqui detalhado. O agente é caracterizado por possuir um identificador único pelo qual será reconhecido dentro do Sistema Multiagentes. A forma de atuação do agente com relação ao ambiente e com os demais agentes depende dos comportamentos que o agente possui.

MÓDULO COMPORTAMENTO

Esse módulo descreve em detalhes os vários tipos de comportamentos passíveis de utilização por parte do agente. O agente pode utilizar um comportamento de uma máquina de estados finitos ou um comportamento repetitivo (cíclico), ou vários outros descritos no Apêndice C.

MÓDULO SISTEMA DE TEMPO REAL

Esse módulo é constituído por um escalonador e por um conjunto de tarefas prontas para serem executadas. O papel do escalonador é definir quais são as tarefas prioritárias dentro de um determinado intervalo de tempo. Essas tarefas estão diretamente relacionadas às ações do agente, ou seja, aos comportamentos.

MÓDULO AVANÇADO

Esse módulo contém os conceitos mais avançados do motor. As técnicas de Redes Neurais e Algoritmos Genéticos estão nesse módulo. Assim como a parte de aprendizagem do motor.

4.4 IMPLEMENTAÇÃO

A implementação do motor utilizou a linguagem de programação C++. Essa linguagem é dominante na indústria de entretenimento digital principalmente devido à sua eficiência. Dessa forma a escolha não pode ser diferente.

Para o desenvolvimento na linguagem selecionada utilizou-se a ferramenta Microsoft Visual C++ [MVC++] e para a construção da arquitetura a ferramenta Jude UML [Jude].

4.5 CONCLUSÕES

Nesse capítulo foi descrito o motor REvolution AI Engine através dos seus requisitos e projeto da arquitetura. Os requisitos são baseados nas técnicas analisadas nos capítulos 2 e 3 e servem como entrada para o processo de projetar a arquitetura.

A arquitetura por sua vez foi desenvolvida com o uso de várias camadas para dividir a complexidade do motor e facilitar a compreensão e extensão.

CAPÍTULO 5

CONCLUSÕES E TRABALHOS FUTUROS

“If you want to know your past, look into your present conditions.
If you want to know your future, look into your present actions.”
— BUDDHIST SAYING

No desenvolvimento desse trabalho tornou-se clara a importância e necessidade da utilização de motores para o desenvolvimento de jogos.

5.1 CONTRIBUIÇÕES

A principal contribuição desse trabalho compreende o estudo, pioneiro no Centro, de técnicas e tecnologias da Inteligência Artificial e suas aplicações no desenvolvimento de jogos.

Os principais resultados obtidos nesse trabalho consistem na pesquisa das técnicas mais comuns na área de entretenimento digital, no estudo aprofundado dessas técnicas e elaboração da arquitetura com o propósito de abordar a maioria delas. Ao final obtém-se a modelagem e arquitetura da REvolution AI Engine.

A divisão da IA em camadas utilizando o método Projeto Distribuído da IA - o que torna a arquitetura mais simples, modular e inteligível - adicionado ao uso de padrões de projeto favorecem o reuso e extensibilidade.

O motor será utilizado na empresa Manifesto Game Studio para o desenvolvimento de jogos e em especial ao seu primeiro produto.

5.2 DIFICULDADES ENCONTRADAS

O estudo dos gêneros de jogos e de suas técnicas realizado com o propósito de identificar padrões e técnicas mais comuns resultou em uma lista enorme com um grande número de técnicas. Isso acontece porque a área de IA para jogos é muito vasta. Essa característica confere muita dificuldade por requerer estudos avançados em todas as técnicas descobertas.

E mais, os jogos são sistemas de tempo real e com várias restrições principalmente no quesito IA. A área gráfica dos jogos consome grande parte dos recursos computacionais de um jogo o que complica o processamento da IA. Esse também é um outro ponto o qual necessitou atenção e dedicação.

5.3 TRABALHOS FUTUROS

Devido ao pioneirismo desse trabalho e aos problemas citados na seção 5.2, o motor Revolution AI Engine não aborda todas as técnicas estudadas nesse trabalho. O primeiro passo para a extensão desse trabalho consiste na modelagem e arquitetura dos demais requisitos. A implementação do motor não está concluída. A finalização da implementação é um outro ponto importante deixado para o futuro.

Outra característica identificada nesse trabalho através do estudo dos motores existentes, no capítulo 3, é a relação intrínseca entre a área de IA dos jogos e as demais áreas de desenvolvimento de jogos (gráfico, física, áudio, etc). Um outro ponto de extensão importante é a união do Revolution AI Engine com essas demais áreas.

A empresa Manifesto Game Studio está desenvolvendo um motor gráfico e em um futuro próximo essa união deve ocorrer.

5.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou as principais contribuições e dificuldades encontradas no decorrer do trabalho. É importante frisar que o principal propósito desse trabalho é dar o primeiro passo dessa longa jornada que é o desenvolvimento de um motor, e principalmente em uma área pouco pesquisada como é a área de motores de IA para jogos.

O trabalho comprova ainda a viabilidade e aplicabilidade de desenvolvimento dessa área pouco explorada até o momento pela indústria de entretenimento digital.

A utilização de técnicas provenientes da área de Engenharia de Software [Vieira et al. 2004] e de Padrões de Projeto [Gamma et al. 1995] auxiliaram no desenvolvimento do projeto e demonstraram ser muito útil também na área de jogos para proporcionar, robustez, reuso, portabilidade e performance.

REFERÊNCIAS

[Farines et al. 2000] Jean-Marie Farines, Joni S. Fraga e Rômulo S. Oliveira. *Sistemas de Tempo Real*. Julho 2000.

[Li e Ravindran 2003] Peng Li e Binoy Ravindran. *Fast, Best-Effort Real-Time Scheduling Algorithms*. IEEE Transactions on Computers, Volume 53, Number 9, pages 1159 - 1175, September 2003.

[Deugo et al. 2001] Dwight Deugo, Michael Weiss e Elizabeth Kendall. *Reusable Patterns for Agent Coordination*. Coordination of Internet Agentes: Models, Technologies, and Applications, Setembro, 2001.

[Schwab 2004] Schwab, Brian. *AI Game Engine Programming*. Charles River Media, Setembro, 2004.

[Russell e Norvig 2003] S. Russell, P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[Gamma et al. 1995] Gamma, Helm, Johnson, and Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[Mitchell 1997] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997

[Rabin 2002] Rabin, Steve. *Ai Game Programming Wisdom*. Vol 1. Charles River Media, 2002.

[Vieira et el. 2004] V. Vieira, S. Costa, R. Valadares. *Melhoria do Processo de Desenvolvimento de Software no Contexto de Jogos*. SBGames, 2004. Curitiba, Brasil.

[FipaACL] Descrição da estrutura FIPA ACL
Site: http://www.fipa.org/specs/fipa00061/
Visitado em 07/2005

[JADE] Java Agent Development Framework
Site: http://jade.tilab.com/
Visitado em 07/2005

[Dungeons&Dragons] Dungeons & Dragons
Site: http://www.wizards.com/default.asp?x=dnd/welcome
Visitado em: 15/08/2005

[Star Wars] Star Wars
Site: http://www.starwars.com/gaming/
Visitado em: 15/08/2005

[RuneQuest] Rune Quest
Site: http://www.runequestrealms.org/
Visitado em: 15/08/2005

[Vampire] Vampire
Site: http://games.activision.com/games/vampire/
Visitado em: 15/08/2005

[Lord] Lord
Site: http://www.eagames.com/official/lordoftherings/
Visitado em: 15/08/2005

[Warcraft] Warcraft
Site: http://www.blizzard.com/war3/
Visitado em: 15/08/2005

[WoW] World of Warcraft
Site: http://www.worldofwarcraft.com/
Visitado em: 15/08/2005

[Unreal] Unreal Engine
Site:
http://www.unrealtechnology.com/html/technology/ue30.shtml
Visitado em: 15/08/2005

[Source] Source Engine
Site: http://www.valvesoftware.com/sourcelicense/
Visitado em: 15/08/2005

[Reality] Reality Engine
Site: http://www.artificialstudios.com/
Visitado em: 15/08/2005

[TGE] Torque Game Engine
Site: http://www.garagegames.com/mg/projects/tge/
Visitado em: 15/08/2005

[DoTT] Wikipedia – Day of the Tentacle
Site: http://en.wikipedia.org/wiki/Day_of_the_Tentacle
Visitado em: 15/08/2005

[KingsQuest] KingsQuest
Site: http://www.vintage-sierra.com/kingsquest.html

Visitado em: 15/08/2005

[MonkeyIsland] Monkey Island
Site: http://en.wikipedia.org/wiki/Monkey_Island
http://www.worldofmi.com/
Visitado em: 15/08/2005

[XCOM] XCOM
Site: http://en.wikipedia.org/wiki/X-COM
Visitado em: 15/08/2005

[Civ3] Civilization 3
Site: http://www.civ3.com/
Visitado em: 15/08/2005

[HMM] Heroes of Might and Magic
Site: http://www.archangelcastle.com/
Visitado em: 15/08/2005

[FireEmblem] Fire Enblem
Site: http://fireemblem.gameboy.com/
Visitado em: 15/08/2005

[War3] Site: http://www.blizzard.com/war3/
Visitado em: 15/08/2005

[Starcraft] Starcraft
Site: http://www.blizzard.com/starcraft/
Visitado em: 15/08/2005

[Command&Conquer] Command & Conquer
Site: http://www.eagames.com/official/cc/generals/us/home.jsp
Visitado em: 15/08/2005

[Age] Age of Empires
Site: http://www.microsoft.com/games/empires/
Visitado em: 15/08/2005

[Halo] Halo
Site: http://www.macsoftgames.com/products/halo/MacSoft-
Visitado em: 15/08/2005Halo.html

[UnrealTournament] Unreal Tournament
Site: http://www.unrealtournament.com/
Visitado em: 15/08/2005

[CS] Counter Strike
Site: http://www.counter-strike.net/

Visitado em: 15/08/2005

[Mario64] Super Mario 64
Site: http://en.wikipedia.org/wiki/Super_Mario_64
Visitado em: 15/08/2005

[DKJB] Donkey Kong Jungle Beat
Site: http://www.donkeykong.com/final/index.html
Visitado em: 15/08/2005

[MarioBros] Super Mario
Site: http://pt.wikipedia.org/wiki/Mario
Visitado em: 15/08/2005

[Sonic] Sonic
Site: http://www.sega.com/sonic/content.php
Visitado em: 15/08/2005

[Spyro] Spyro The Dragon
Site: http://www.spyrothedragon.com/
Visitado em: 15/08/2005

[Crash] Crash Bandicoot
Site: http://www.naughtydog.com/crash/
Visitado em: 15/08/2005

[CM] Championship Manager
Site: http://www.championshipmanager.co.uk/
Visitado em: 15/08/2005

[ArchRivals] Arch Rivals
Site: http://en.wikipedia.org/wiki/Arch_Rivals
Visitado em: 15/08/2005

[FIFA] FIFA
Site: http://www.fifa2005.ea.com/index_flash.jsp?locale=br
Visitado em: 15/08/2005

[NBA] NBA Live 2001
Site: http://nbalive2001.ea.com/
Visitado em: 15/08/2005

[NFL] Madden NFL
Site: http://www.easports.com/games/madden06/home.jsp
Visitado em: 15/08/2005

[FlightSimulator] Flight Simulator
Site: http://www.microsoft.com/games/flightsimulator/

Visitado em: 15/08/2005

[Sims] The Sim
Site: http://thesims.ea.com/
Visitado em: 15/08/2005

[SimCity] SimCity
Site: http://simcity.ea.com/
Visitado em: 15/08/2005

[SimEarth] SimEarth
Site: http://en.wikipedia.org/wiki/Simearth
Visitado em: 15/08/2005

[GranTurismo] Gran Turismo
Site: http://www.granturismoworld.com/
Visitado em: 15/08/2005

[NeedforSpeed] Need for Speed
Site:
http://www.eagames.com/official/nfs/underground2/us/home.jsp
Visitado em: 15/08/2005

[MarioKart] Mario Kart
Site: http://www.mariokart.com/launch/index.html
http://de.wikipedia.org/wiki/Mario_Kart
Visitado em: 15/08/2005

[StreetFighter] Street Fighter
Site: http://en.wikipedia.org/wiki/Street_Fighter
http://de.wikipedia.org/wiki/Mario_Kart
Visitado em: 15/08/2005

[KingOfFighters] King of Fighters
Site: http://en.wikipedia.org/wiki/King_of_Fighters
Visitado em: 15/08/2005

[MortalKombat] Mortal Kombat
Site: http://www.mortalkombatonline.com/
Visitado em: 15/08/2005

[VirtuaFighter] Virtua Fighter
Site: http://en.wikipedia.org/wiki/Virtua_Fighter
Visitado em: 15/08/2005

[Epic] Epic Games
Site: http://www.epicgames.com/

Visitado em: 15/08/2005

[Playstation] Playstation
Site: http://www.playstation.com/
Visitado em: 15/08/2005

[DirectX] DirectX
Site: http://www.microsoft.com/windows/directx/default.aspx
Visitado em: 15/08/2005

[UnrealChampionship] Unreal Champioship
Site: http://www.unrealchampionship.com/
Visitado em: 15/08/2005

[HarryPotter] Harry Potter
Site: http://harrypotter.ea.com/
Visitado em: 15/08/2005

[MoV] Men of Valor
Site: http://www.menofvalorgame.com/
Visitado em: 15/08/2005

[SWRC] Republic Commando
Site: http://www.lucasarts.com/games/swrepubliccommando/
Visitado em: 15/08/2005

[Tribes2] Tribes 2
Site: http://www.planettribes.com/tribes2/
Visitado em: 15/08/2005

[GarageGames] GarageGames
Site: http://www.garagegames.com/
Visitado em: 15/08/2005

[RocketBowl] Rocket Bowl
Site: http://www.garagegames.com/products/58
Visitado em: 15/08/2005

[GoldFever] Gold Fever
Site: http://www.oberongames.com/game.htm?code=110415417
Visitado em: 15/08/2005

[WLTVA] Wild Life Tycoon
Site: http://www.wildlifetycoon.com/
Visitado em: 15/08/2005

[UELicensing] Unreal Technology Licensing
Site: http://www.unrealtechnology.com/html/licensing/terms.shtml

Visitado em: 15/08/2005

[Artificial] Artificial Studios
Site: http://www.artificialstudios.com/
Visitado em: 15/08/2005

[Armageddon] Armageddon
Site: http://www.boanergesstudios.com/
Visitado em: 15/08/2005

[CellFactor] CellFactor
Site: http://www.cellfactorgame.com/
Visitado em: 15/08/2005

[Metronome] Metronome
Site: http://www.tarsier.se/metronome/
Visitado em: 15/08/2005

[Dreamers] Dreamers
Site: http://www.dreamgazers.com/
Visitado em: 15/08/2005

[Valve] Valve Software
Site: http://www.valvesoftware.com/
Visitado em: 15/08/2005

[Half-Life] Half-Life 2
Site: http://half-life2.com/
Visitado em: 15/08/2005

[DayOfDefeat] Day of Defeat
Site: http://www.dayofdefeatmod.com/
Visitado em: 15/08/2005

[Crytek] CryTek
Site: http://crytek.com/
Visitado em: 15/08/2005

[CryEngine] CryEngine
Site: http://www.crytek.com/technology/index.php?sx=cryengine
Visitado em: 15/08/2005

[OpenGL] OpenGL
Site: http://www.opengl.org/
Visitado em: 15/08/2005

[GameCube] GameCube
Site: http://www.nintendo.com/systemsgcn

Visitado em: 15/08/2005

[FarCry] Far Cry
Site: http://www.farcry-thegame.com/
Visitado em: 15/08/2005

[Rogue] Site: http://en.wikipedia.org/wiki/Rogue_(computer_game)
Visitado em: 14/08/2005

[NetHack] Site: http://en.wikipedia.org/wiki/Nethack
Visitado em: 14/08/2005

[DFC] DFC Intelligence.
Site: http://www.dfcint.com/news/prsep222004.html
Visitado em: 15/08/2005

[PWHC] PricewaterhouseCoopers.
Site: http://www.pwc.com/outlook/
Visitado em: 15/08/2005

[ELSPA] ELSPA.
Site: http://www.screendigest.com/publications/reports/games/interactive_leisure_software_2004/press_releases_31_08_2004/view
Visitado em: 15/05/2005

[Genesys] Genesys3D.
Site: http://www.genesis3d.com/
Visitado em: 15/08/2005

[Irrlicht] Irrlicht.
Site: http://irrlicht.sourceforge.net/
Visitado em: 15/05/2005

[Ogre3D] Ogre3D.
Site: http://www.ogre3d.org/
Visitado em: 15/05/2005

[Crystal] Crystal Space 3D.
Site: http://www.crystalspace3d.org/
Visitado em: 15/08/2005

[Jude] Jude UML Modeling Tool.
Site: http://www.esm.jp/jude-web/index.html
Visitado em: 15/08/2005

[MVC++] Microsoft Visual C++.
Site: http://msdn.microsoft.com/visualc/
Visitado em: 15/08/2005

APÊNDICE A

A.1 SISTEMA DE TEMPO REAL

Sistemas de tempo real são aplicações com restrições temporais. Entre os sistemas mais simples estão os controladores inteligentes embutidos em utilidades domésticas, tais como lavadoras de roupa e aparelhos de som. Na outra extremidade estão os sistemas militares de defesa, os sistemas de controle de plantas industriais e o controle de tráfego aéreo e ferroviário [Farines et al 2000].

Para a implementação das restrições dos sistemas de tempo real é necessário o conhecimento da classe de problemas na qual o sistema atuará para escolher a abordagem de escalonamento mais adequada.

Diferentes abordagens de escalonamento são identificadas na literatura, tomando como base o tipo de carga tratado e as etapas no escalonamento. A taxonomia utilizada em [Farines et al. 2000] identifica três grupos principais de abordagens de escalonamento de tempo real: as abordagens com garantia em tempo de projeto, com garantia em tempo de execução e abordagens de melhor esforço.

O grupo garantia em tempo de projeto é formado por abordagens que tem como finalidade a previsibilidade determinista. Os grupos de abordagens de garantia dinâmica e de melhor esforço, ao contrário do grupo anterior, não tratam com uma carga computacional previsível; na verdade, a carga tratada por essas abordagens é dinâmica. A abordagem de garantia dinâmica é própria para aplicações com restrições críticas mas que operam em ambientes não deterministas. Sistemas militares, sistemas de radar e controle aéreo exemplificam alguns desses sistemas que estão sujeitos a cargas dinâmicas e necessitam atender restrições temporais.

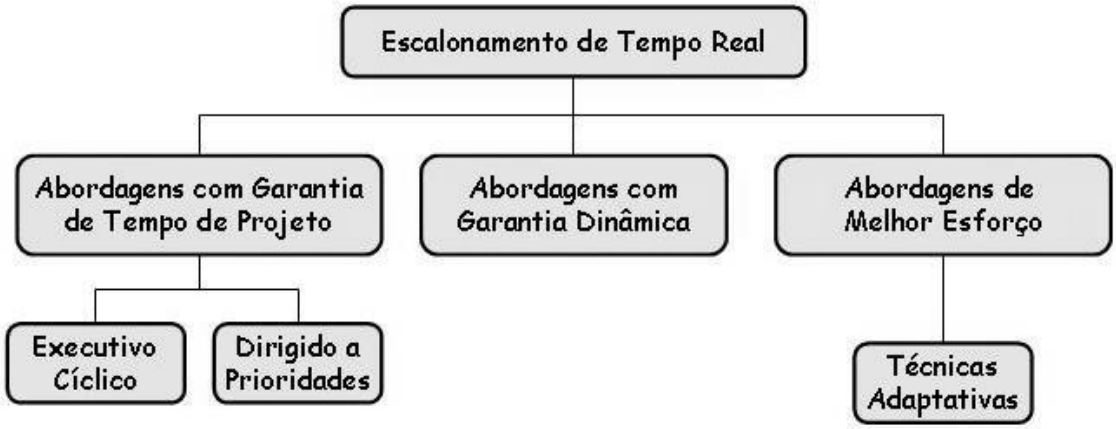

Figura 16 Abordagens de Sistema de Tempo Real

As abordagens de melhor esforço são constituídas por algoritmos que tentam encontrar uma escala realizável em tempo de execução sem realizar testes, ou ainda realizando testes mais fracos. Não existe a garantia de execuções de tarefas atendendo suas restrições temporais. Essas abordagens são adequadas para aplicações não críticas, envolvendo tarefas "soft" onde a perda de deadlines não representa custos além da diminuição do desempenho nessas aplicações. As abordagens de melhor esforço são adequadas para aplicações de tempo real brandas como sistemas multimídias.

O desempenho de esquemas de melhor esforço, na ocorrência de casos médios, é melhor que os baseados em garantia. As hipóteses pessimistas feitas em abordagens com garantia dinâmica podem desnecessariamente descartar tarefas. Nas abordagens de melhor esforço tarefas são abortadas somente em condições de sobrecarga, na ocorrência de falhas temporais ("deadline" pode não ser atendido).

Dentre as abordagens citadas e considerando as características da engine de IA para jogos em construção, a abordagem adequada é a de melhor esforço. Em primeiro lugar por tratar cargas dinâmicas - o que exclui a abordagem com garantia de tempo de projeto – e segundo por possuir um melhor desempenho que a abordagem com garantia dinâmica.

A.1.1 Algoritmos

A implementação da abordagem de melhor esforço selecionada dentre as demais pode ser realizada através de vários algoritmos de escalonamento. Para análise foram considerados quatro algoritmos: DASA, MDASA, LBESA e MLBESA.

Para realizar a seleção do algoritmo a ser utilizada foram consideradas as características de desempenho como fator principal e a complexidade de pior caso [Li e Ravindran 2003]. Após estudar todas esses tecnologias e analisá-los com relação aos fatores considerados conclui-se que os algoritmos MDASA e MLBESA são superiores em

desempenho para a aplicação em questão quando comparados ao DASA e LBESA. E comparando os algoritmos pré-selecionados no quesito complexidade no pior caso o MDASA (complexidade O (n)) supera o MLBESA (complexidade O (n log(n))).

APÊNDICE B

B.1 PROJETO DISTRIBUÍDO DA IA

O Projeto Distribuído da Inteligência Artificial, tradução literal da expressão Distributed AI Design, é um método que visa a simplificação da criação e manutenção da IA através da propagação das tarefas de IA em módulos ou sistemas de camadas interligadas para criar comportamentos inteligentes complexos sem sobrecarregar e complicar o sistema em geral [Schwab 2004].

Para simplificar o método será apresentado, a título de exemplo, uma situação da vida real. Uma determinada pessoa está realizando um deslocamento do ponto A para o ponto B e, de repente, escuta alguém gritar seu nome. A pessoa então muda a sua rota e movimenta-se agora em direção à localização da origem voz. Para simplificar o exemplo as atividades serão decompostas para apresentar exatamente o que ocorreu nesse trecho citado. E assuma que você é o personagem principal desse exemplo.

1. Você estava realizando um deslocamento (executando o comportamento denominado RealizaMovimento). E então você recebe uma entrada na forma de um evento ou mensagem do jogo (seu nome sendo chamado).
2. O seu sistema de tomada de decisão determina que a percepção / evento recebido é importante o suficiente para modificar o seu comportamento em detrimento de um novo comportamento mais aplicável a atual situação.
3. O novo comportamento para o qual você transitou, chamado Investigar, altera a posição do local de destino da sua movimentação para algum ponto próximo ao local de emissão do som. A localização é descoberta por algum algoritmo para adivinhar localizações de sons ou através do uso de cheating.
4. Você executa o seu algoritmo de pathfinding para determinar como alcançar o novo destino. O algoritmo provê o conjunto de pontos aos quais você deve se movimentar em seqüência para chegar lá.
5. O seu código responsável pela movimentação finalmente possui um destino e então calcula que animações executar para obter uma melhor apresentação e aparência.

6. Você inicia o movimento seguindo a lista de ponto do seu caminho, mas aparece uma pessoa em seu caminho. O seu sistema de detecção de obstáculos entra em ação e movimenta você para contornar o obstáculo encontrado. Conduzido agora ao próximo ponto da lista, e usando o sistema para evitar obstáculos para auxiliar você a navegar com objetos dinâmicos à sua volta, você se aproxima do destino.
7. A poucos metros de distância você vê um amigo seu e reconhece que é dele a voz que você escutou.
8. A nova percepção de entrada (a informação de que foi seu amigo que o chamou) modifica o seu estado e comportamento de Investigar para Interrogar e então você diz “O que você deseja?”

A decomposição das atividades listadas acima pode ainda ser maior e descrever o trecho através de atividades de menor nível. A intenção desse exemplo é deixar transparecer o significado do conceito de distribuição e modularidade. O método de distribuição tenta reproduzir a qualidade dos comportamentos do mundo real os quais possuem vários níveis - sistema de percepção, sistema de navegação, sistema de detecção de conflito, etc.

O método de distribuição acarreta na produção de código mais limpo e fácil de manter e é ainda mais fácil de entender e estender. O método espalha a inteligência entre os diferentes sistemas, vários dos quais podem e devem ser utilizados por outros elementos do jogo. O método recomenda a divisão nas seguintes camadas:

1. Camada de Percepção / Eventos
2. Camada de Comportamento
3. Camada de Animação
4. Camada de Movimentação
5. Camada de Tomada de Decisão de Curto Prazo
6. Camada de Tomada de Decisão de Longo Prazo
7. Camada de Informação Baseada na Localização

APÊNDICE C

C.1 ARQUITETURA EM DETALHES

Nesse apêndice é apresentado de forma mais avançada a arquitetura através da descrição de seus módulos. Cada módulo conta com uma descrição e um diagrama de classes. Convém lembrar que a separação em módulos é meramente didática.

C.1.1 Módulo Sistema Multiagentes

O primeiro módulo a ser apresentado é o sistema multiagentes que corresponde à base sobre a qual as demais técnicas e requisitos estão apoiados. O sistema multiagentes é constituído por um container principal (MainContainer) o qual possui vários containeres de agentes (AgentContainer). O container principal contém ainda uma tabela com referência para todos os agentes do sistema (GADT).

O container de agentes contém uma camada de comunicação (ACC) sobre a qual as mensagens são transportadas. O container possui ainda uma tabela com referencia aos agentes locais ao container (LADT) e conhecimento sobre os agentes responsáveis pelos serviços de páginas amarelas e brancas.

O agente possui referência para um agente o qual fornece o conjunto de ferramentas necessárias para a execução e comunicação do agente dentro do sistema. Esse agente (AgentToolkit) fornece uma interface através da qual o agente recupera os agentes responsáveis pelo serviço de páginas amarelas e brancas.

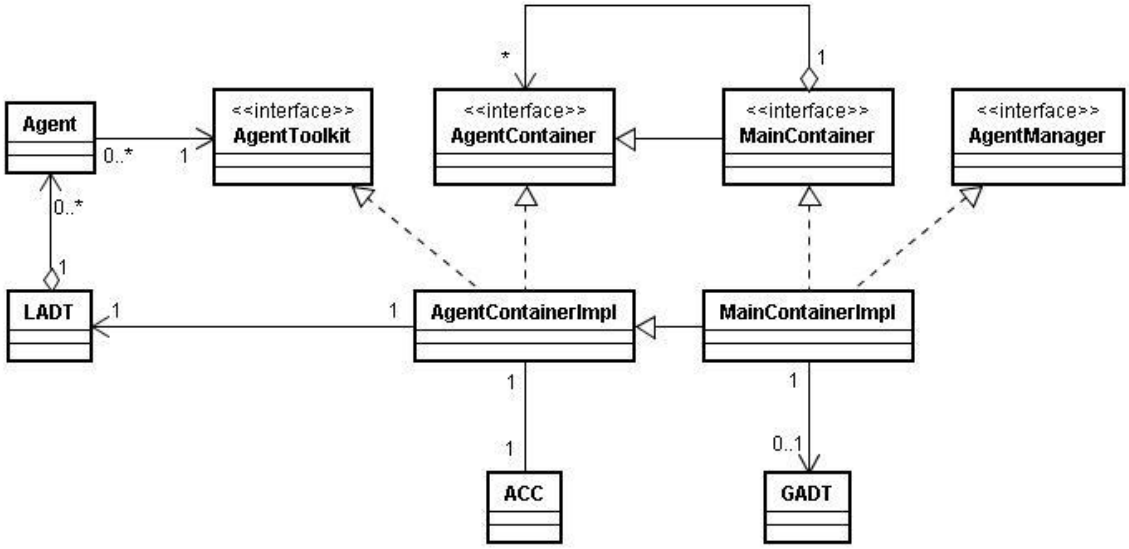

Figura 17 Diagrama de Classes da Engine

C.1.2 Módulo Agente

Na Figura 17 é detalhada a arquitetura do agente. Este possui um identificador único representado pelo AID. A comunicação entre os agentes é realizada através de troca de mensagens. As mensagens recebidas pelo agente são armazenadas em uma fila de mensagens (MessageQueue) para ser tratada pelo agente em uma oportunidade propícia.

A comunicação efetiva entre os agentes depende ainda do uso de um vocabulário comum. A interface Ontology é responsável por desempenhar esse papel.

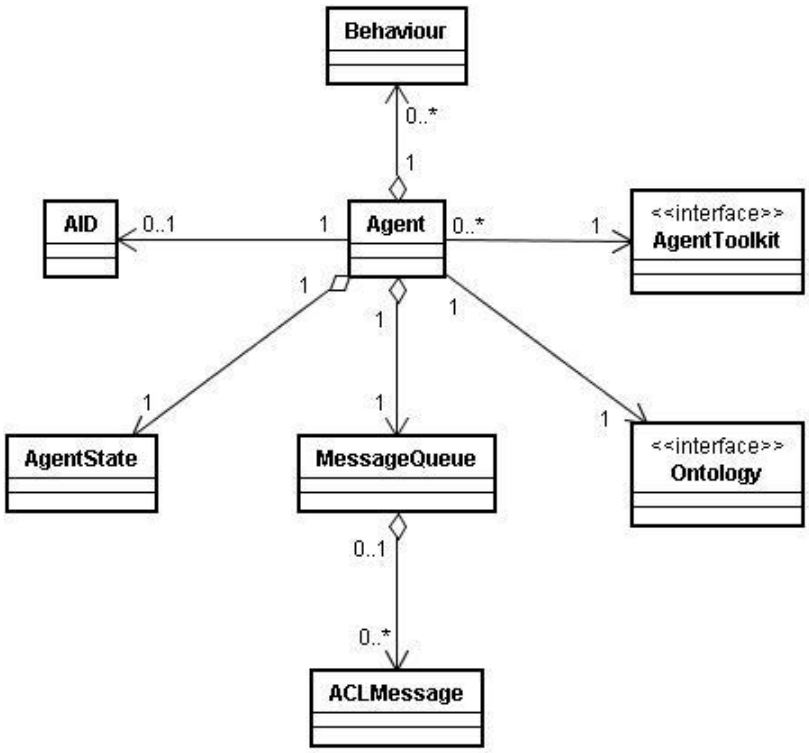

Figura 18 Diagrama de Classes do Agente

C.1.3 Módulo Comportamento

O comportamento apresentado pelo agente é decorrência de sua associação com a classe Behaviour. O agente pode possuir vários comportamentos de diferentes tipos. A figura abaixo exemplifica os tipos existentes.

Os comportamentos são divididos em dois subgrupos principais: simples e compostos. Os comportamentos compostos são formados por um agrupamento de comportamentos simples e seguem o padrão de projeto Composite.

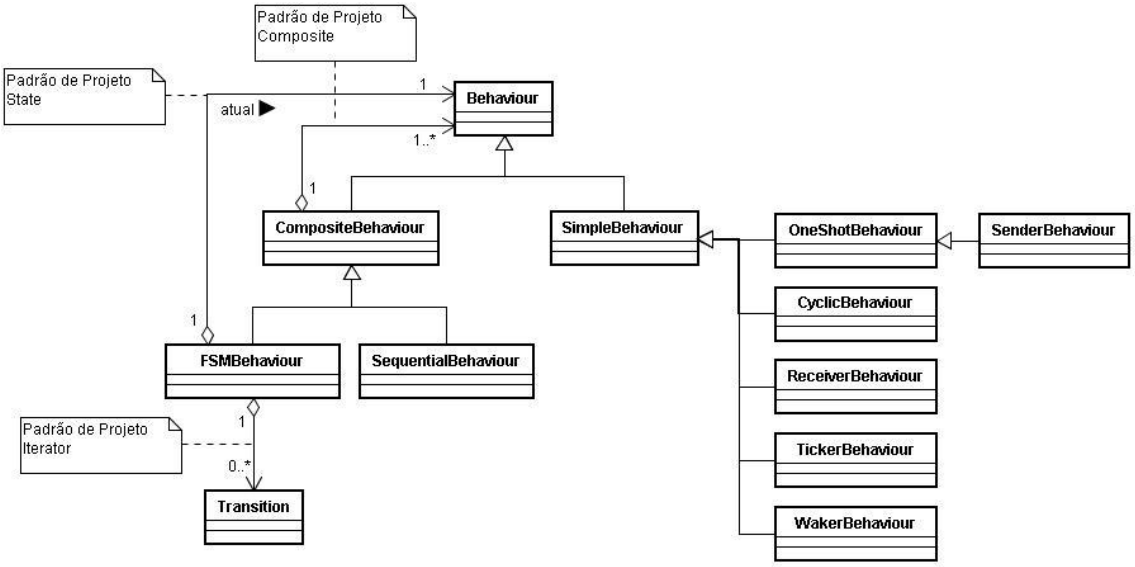

Figura 19 Diagrama de Classe dos Comportamentos

C.1.4 Módulo Sistema de Tempo Real

O sistema de tempo real está representado na figura abaixo. O sistema de tempo real é composto por um escalonador e um conjunto de tarefas as quais somados os seus tempos médios de execução compõem a carga total do sistema. A forma de escalonamento dependerá do algoritmo escolhido (SchedulerAlgorithm). Na figura são apresentados alguns exemplos de algoritmos, no entanto, a engine somente provê suporte para o algoritmo MDASA descrito em detalhes do Apêndice A.

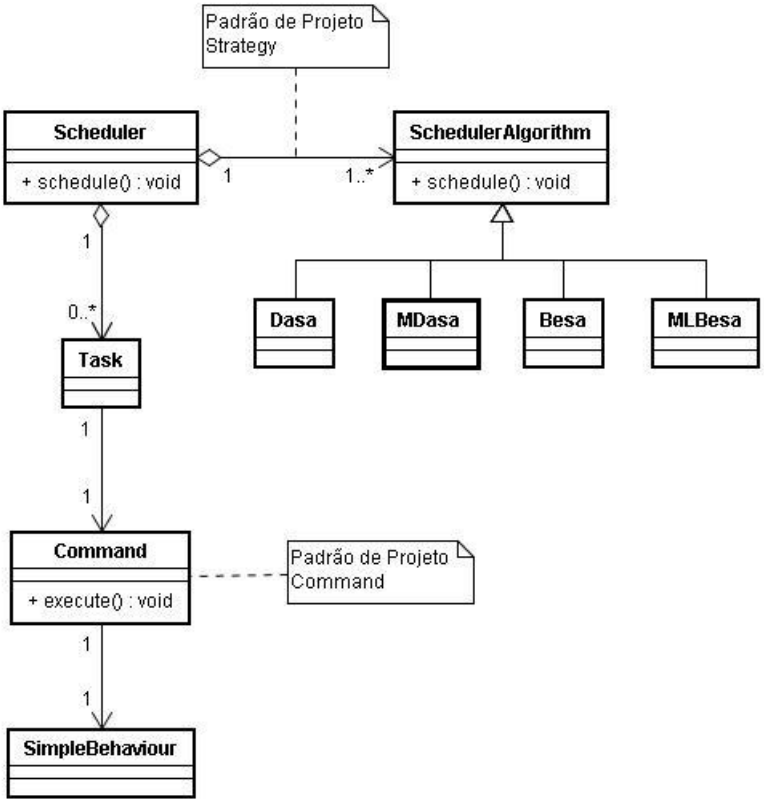

Figura 20 Diagrama de Classes do Sistema de Tempo Real

C.1.5 Módulo Avançado

Nesse módulo se encontram as técnicas mais avançadas do motor. As técnicas de Redes Neurais, Algoritmos Genéticos a ainda a parte de aprendizagem do motor. A Figura 21 apresenta a estrutura da Rede Neural (NeuralNet) a qual é formada por várias camadas (Layer's) de neurônios (Neuron's). Na mesma figura está a estrutuda do Algoritmo Genético o qual é constituído por vários genomas (Genome's).

A aprendizagem está associada à Rede Neural e pode acontecer de supervisionada através da treinamento da rede. Para isso é necessário obter os dados de entrada da rede assim como os dados de saída para indicar como a rede deve aprender. A forma não supervisionada consiste em uma mistura das técnicas de rede neural e algoritmos genéticos e funciona da seguinte forma.

Os genomas do algoritmo genético representarão pesos de várias redes neurais. As várias gerações criadas possuem vários pesos diferenets para a rede. E somente os genomas com os pesos com menor taxa de erro quando simulados na rede neural são passados adiante. O algoritmo genético é então utilizado para minimizar a taxa de erros da rede neural.

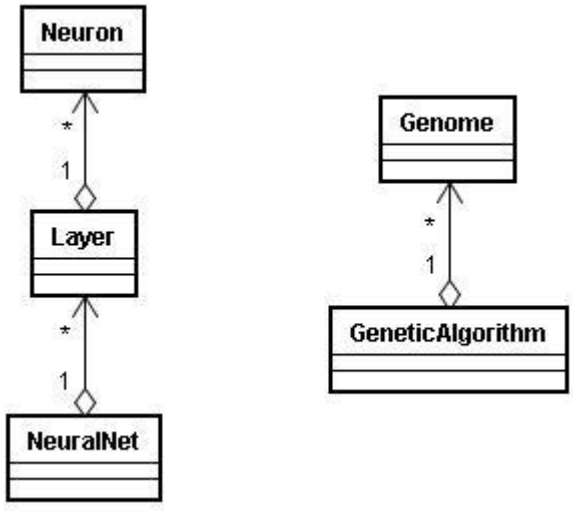

Figura 21 Diagrama de Classes da Rede Neural e do Algoritmo Genético