BRUNO CAMPAGNOLO DE PAULA

SISTEMA INTELIGENTE PARA JOGOS DE ESTRATÉGIA BASEADOS EM TURNOS: UMA ABORDAGEM UTILIZANDO PLANEJAMENTO BASEADO EM CASOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

CURITIBA

MAIO / 2007

BRUNO CAMPAGNOLO DE PAULA

SISTEMA INTELIGENTE PARA JOGOS DE ESTRATÉGIA BASEADOS EM TURNOS: UMA ABORDAGEM UTILIZANDO PLANEJAMENTO BASEADO EM CASOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Alessandro Lameiras Koerich
Co-orientador: Prof. Dr. Fabrício Enembreck

CURITIBA

MAIO / 2007

De Paula, Bruno Campagnolo
D419s Sistema inteligente para jogos de estratégia baseados em turnos : uma
2007 abordagem utilizando planejamento baseado em casos / Bruno Campagnolo De Paula ; orientador, Alessandro Lameiras Koerich ; co-orientador, Fabrício Enembreck. – 2007.
130 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2007
Inclui bibliografia

1. Jogos por computador. 2. Raciocínio. 3. Planejamento. 4. Inteligência artificial. I. Koerich, Alessandro Lameiras. II. Enembreck, Fabrício. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD 20. ed. – 004.16

A minha amada esposa

Daniele,

minha querida avó

Flávia

e ao meu saudoso avô

Sebastião.

Agradecimentos

Meu primeiro agradecimento é para o Professor Doutor Alessandro Koerich. Sem dúvida, além de um exemplo como Professor, sua paciência, profissionalismo, e, principalmente, estímulos nos momentos difíceis foram essenciais para o desenvolvimento deste trabalho. Também agradeço a meu co-orientador Professor Doutor Fabrício Enenbreck pelas idéias, críticas e pela coerência em todos os momentos.

Registro meus agradecimentos, também, algumas pessoas e empresas que apoiaram direta ou indiretamente minha vida acadêmica e profissional nos últimos anos: da PUCPR, os Professores Marcos Shmeil, Edson José Pacheco, Marco Eleutério, Henri Eberspächer e Andreia Malucelli; da *Continuum Entertainment*, Alexandre Vrubel e demais companheiros; do CEPIMC, os Professores João Batista da Rosa, Daniel Eggert, Gezelda de Moraes , Diogo Domanski e Débora Puppin. Não posso furtar um agradecimento especial a meus alunos e ao pessoal do TECPAR por participarem de testes com o protótipo de meu sistema.

A minha mãe Luci por dedicado toda sua vida a mim e por ter me ensinado a importância do estudo. A minha avó Flávia pelas orações e pelo carinho. Ao meu avô Sebastião pelo exemplo de vida e trabalho. A meu pai Dagoberto pela ajuda nos momentos que precisei. Fica também minha gratidão a Cícero e Cecília pela presença alegre e constante. Aos meus queridos amigos, Rafael Cavichioli, Juliana Dallagnol, Rafael Bardal, Adriane Granatto e Luiz Cláudio Guarita, cuja amizade foi essencial.

Devo irrestrito reconhecimento a minha amada esposa Daniele Bagatoli. Seu amor, sua inteligência, carinho e motivação são imprescindíveis para minha vida.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

ARG	*Alternate Reality Games* – Jogos de Realidade Alternativa
BDI	*Befief-Desire-Intention* – Crença-Desejo-Intenção
CAT	*Case-Based Tactician*
FPS	*first-person shooter* – jogo de tiro em primeira pessoa
GDC	*Game Developers Conference*
GOAP	*Goal Oriented Action Planning* - Planejamento de Ações Orientado a Objetivo
GPL	*General Public Licence*
GPS	*General Problem Solver*
HTN	*Hierarquical Task Network Plannning*
IA	Inteligência Artificial
JA	Jogador Automático
MMOG	*Multiplayer Massive Online Game* - jogos *online* massivos *multiplayer*
NPC	*Non player character* – personagem controlado pelo computador
PBC	Planejamento Baseado em Casos
PDDL	*Planning Definition Language* - Linguagem de Definição de Planos
RBC	Raciocínio Baseado em Casos
RTS	*Real Time Strategy* – Jogos de Estratégia em Tempo Real
SMS	*Short Message System*

RESUMO

Desde meados da década de 50, jogos têm sido utilizados para examinar e testar conceitos em diversas áreas da Inteligência Artificial (*IA*). As primeiras pesquisas foram baseadas em jogos clássicos como Damas e Xadrez que se caracterizam pela simplicidade das regras e objetivos. Mais tarde, com o desenvolvimento dos primeiros jogos comerciais diversas técnicas foram aplicadas para simular uma inteligência aos elementos da tela, sendo que os jogos de estratégia destacam-se como os pioneiros na utilização de técnicas de *IA* e, um campo de testes para a *IA* para ambientes em tempo real. O desempenho da *IA* dos jogos de estratégia, entretanto, é pobre para os padrões humanos, não acompanhando a evolução apresentada pela *IA* de jogos clássicos.

Apresenta-se, neste trabalho, a criação de um jogador automático para a classe dos jogos de estratégia baseados em impérios (*empire-based strategy games*). Tal jogador automático foi construído através de técnicas de *Planning* (Planejamento), mais especificamente Planejamento Baseado em Casos (*PBC - Case-Based Planning*). O jogo escolhido para testes, chamado ***Promisance***, é um jogo *multiplayer online* no qual os jogadores administram um império. Os planos realizados pelos jogadores humanos foram armazenados em uma base de planos a partir da qual foram gerados jogadores automáticos que imitam as estratégias dos jogadores humanos. Os resultados experimentais mostram que os jogadores automáticos imitam as estratégias dos jogadores humanos de maneira convincente, povoando o mundo do jogo com jogadores automáticos semelhantes ao grupo de jogadores humanos.

Palavras-chave: jogos de computador, *Case-based reasoning*, *case-based Planning*.

1 INTRODUÇÃO

O objetivo deste capítulo é desenvolver uma contextualização inicial para a área de Inteligência Artificial (*IA*) para jogos e definir, dentro desta área, qual o desafio enfrentado por este trabalho. Finaliza-se com as motivações para o trabalho e a proposta realizada de resolução do problema citado. Tal proposta é desenvolvida nos capítulos subseqüentes conforme a organização referenciada no final deste capítulo.

1. 1 APLICAÇÃO DE *IA* EM JOGOS CLÁSSICOS

Desde meados da década de 50, jogos vêm sendo utilizados para examinar e testar conceitos em diversas áreas da Ciência da Computação, principalmente na subárea da *IA*. As primeiras pesquisas foram baseadas em jogos clássicos como Damas [SAM1959] e Xadrez [SHA1950] que se caracterizam pela simplicidade de compreensão das regras e objetivos e por permitirem que um programa acesse o estado do mundo do jogo de maneira completa. Com a percepção completa de cada movimento possível e da conseqüência direta desses movimentos, a criação de um programa jogador de Xadrez pode ser simplificada com a utilização de um algoritmo de busca que escolhe o melhor movimento a ser feito a partir das finitas possibilidades de movimento em cada rodada. Destaca-se, todavia, que no caso do Xadrez, a árvore de busca gerada é de tamanho proibitivo. Considerando-se que cada turno possui em média 35 jogadas possíveis (*branching factor*) e os jogos perduram, em média, 50 turnos para cada jogador, a árvore de busca gerada terá 35^{100} nós [RUS1995]. Eliminando as posições impossíveis sobram 10^{40} diferentes posições possíveis em um jogo de Xadrez.

Justifica-se, portanto, que a pesquisa na área de *IA* para jogos manteve-se em constante evolução já que, para grande parte dos jogos clássicos, é extremamente cara computacionalmente a construção de um jogador automático, quando se emprega um algoritmo de busca. Deve-se destacar que Xadrez e Damas foram os jogos escolhidos como campo de teste da *IA* devido à propriedade de serem bem definidos, ou seja, com todas as variáveis de estado a cada turno sendo observáveis. Nos jogos citados, além de todas as

jogadas serem conhecidas por todos os jogadores envolvidos, não há o fator sorte envolvido. Tais jogos podem ser classificados como **jogos determinísticos de informação completa**.

Outros jogos, porém, como Gamão e Banco Imobiliário inserem o fator sorte através do uso de dados. Dessa forma, o número de nós da árvore de busca cresce em relação ao número de possibilidades que o dado insere. Jogos deste tipo não deixam de ser classificados como **jogos de informação completa**. Entretanto, são classificados como **jogos não-determinísticos**.

Por outro lado, existe uma classe de jogos nos quais os jogadores possuem informações parciais referentes ao estado do jogo, estes são **jogos de informação parcial**. Exemplos de jogos de informação parcial são: *poker*, *bridge*, *pif* e batalha naval. Dentre os jogos citados apenas o jogo de batalha naval não é influenciado pela sorte. Reafirma-se, assim, a dificuldade para aplicar uma abordagem de busca por força bruta em diversos tipos de jogos. Em um jogo de *bridge*, por exemplo, se cada carta possível for considerada um nó da árvore de busca serão gerados $5{,}6x10^{44}$ nós no pior caso e $2{,}3x10^{24}$ nós no caso médio [SMI1998].

Além da busca heurística, outras técnicas de *IA* também foram utilizadas. Em 1959, Samuel aplicou conceitos de aprendizagem de máquina para a geração de um jogador automático de damas [SAM1959]. Os resultados obtidos, inclusive, constituíram as bases para a Aprendizagem de Máquina como ciência, sendo que a maioria das idéias desenvolvidas por Samuel para a aprendizagem continua em uso e com grande aplicação na área de *game playing* [FÜR2000].

Outro exemplo de técnica aplicada foi o trabalho de [NEW1972] na área de solucionadores gerais de problemas (*GPS – General Problem Solver*). Além de provar teoremas em lógica e geometria, o *GPS* também foi aplicado em jogos de xadrez.

Além das técnicas já citadas, Redes Neurais e Agentes destacam-se entre as técnicas mais usuais aplicadas a jogos. Em menor escala, para Algoritmos Genéticos e *Case-Based Reasoning* também existem aplicações em jogos. Na década de 90, segundo Schaffer [SCH2001], destacaram-se as técnicas como Simulação de Montecarlo e Aprendizagem por Diferença Temporal.

Os principais marcos para a *IA* em jogos clássicos são as vitórias [SCH2001] e, até mesmo, derrotas dos jogadores automáticos. Em 1994, *Chinook* torna-se o primeiro programa

de computador a vencer um campeão mundial em um jogo. Mais especificamente, o programa venceu o campeão mundial de Damas, Marion Tinsley, que teve de se retirar devido a problemas de saúde após seis empates. No domínio do Xadrez, tem importância a vitória do programa *Deep Blue* em uma série de jogos de exibição contra o campeão mundial Kasparov. Em outro marco, no ano de 1997, o programa Logistello venceu todos os jogos contra o campeão mundial de *Othello*[1], determinando a superioridade da máquina no jogo.

Independente da técnica utilizada percebe-se que a evolução dos jogadores automáticos foi bastante considerável nos últimos anos. Alguns jogos, inclusive, já não representam desafio algum para o computador, como, por exemplo, *Othello*. Tais jogos são considerados resolvidos, ou seja, todas as suas possibilidades são discerníveis algoritmicamente e não existe possibilidade do computador ser derrotado. A Tabela 1 sumariza os jogos clássicos que já foram completamente resolvidos e os jogos que ainda não são páreo para o raciocínio humano.

Jogo	Tipo do Jogo	Estado atual do jogador automático
Connect Four (Lig 4)	Determinístico Informação Completa	Resolvido
Go-Moku	Determinístico Informação Completa	Resolvido
Qubic	Determinístico Informação Completa	Resolvido
Nine man's Morris (Trilha ou Moinho)	Determinístico Informação Completa	Resolvido
Othello	Determinístico Informação Completa	Joga **melhor** que os seres humanos.
Damas	Determinístico Informação Completa	Joga **melhor** que qualquer ser humano vivo.
Gamão	Jogo com Sorte Informação Completa	Joga **melhor** que todos os seres humanos exceto cerca de 10.
Xadrez	Determinístico Informação Completa	Joga **melhor** que qualquer ser humano exceto cerca de 250.
Go	Determinístico Informação Completa	Joga **pior** que o melhor campeão de nove anos de idade.
Bridge	Determinístico Informação Completa	Joga pior que jogadores humanos medianos.

Tabela 1 - Desempenho de programas de computador em alguns jogos clássicos.[SMI1998].

Observando a Tabela 1 deduz-se que jogos de informação parcial ou com um número proibitivo de possibilidades como *Go* ainda constituem um desafio relevante para a pesquisa no campo da *IA* para jogos clássicos. Quanto a *Go*, cabe um comentário sobre o motivo da

[1] Jogo de tabuleiro de duplas, também conhecido como Reversi. Em um tabuleiro de 8 x 8 peças, dois jogadores se alternam de maneira a colocar uma peça de cada vez com o objetivo de transformar as peças do adversário em peças de seu time.

dificuldade. Além de o jogo ter uma árvore de busca bastante profunda, os jogadores humanos não sabem explicar **o motivo** de uma configuração ser boa ou ruim. A vantagem do ser humano é tão significativa que existe um prêmio de um milhão de dólares para o primeiro programa que vencer um campeão tailandês júnior [THE2002].

Destaca-se, portanto, a necessidade da continuidade das pesquisas na área de geração de jogadores automáticos. O presente trabalho é uma contribuição para esta continuidade. A classe de jogos focalizada, porém, não pertence ao domínio dos jogos clássicos, mas ao universo dos jogos exclusivamente para computador. Assim, torna-se importante uma sucinta contextualização desta área.

1. 2 APLICAÇÃO DE *IA* EM JOGOS DE COMPUTADOR

Desde o desenvolvimento dos primeiros jogos comerciais diversas técnicas foram aplicadas para simular a inteligência humana. Uma das primeiras estratégias perceptíveis pelo público em geral foi a *IA* dos fantasmas do jogo *Pac-man*. O algoritmo utilizado transmitia a sensação que os fantasmas objetivavam a captura do jogador, perseguindo-o. Este comportamento é implementado através de regras. Entretanto, após algumas partidas, era perceptível a previsibilidade de movimento dos fantasmas, pois as regras eram fixas e simples. Pode-se afirmar, portanto, que os primeiros jogos não utilizavam técnicas de *IA* clássica, apenas tentavam transmitir uma **ilusão de inteligência** para o jogador.

Jogos de estratégia destacam-se como os pioneiros na utilização de técnicas de *IA* [TOZ2002]. Técnicas de *pathfinding* (busca de caminhos) foram características neste gênero através da aplicação de algoritmos de busca como o *A** [HAR1968]. A *IA* para jogos de computador focou-se nesta área por vários anos devido ao fato que as restrições quanto à velocidade de processamento e quantidade de memória exigida para a busca sempre foram questões críticas a serem resolvidas. Como o processamento gráfico era mais importante, técnicas de *IA* mais caras não poderiam ser aplicadas.

Aos poucos algumas técnicas mais tradicionais foram utilizadas em gêneros específicos. Por exemplo, jogos de simulação como *The Sims* e *Creatures* utilizam conceitos de vida artificial. *FPSs* (*first-person shooters* – jogos de tiro em primeira pessoa) como *Quake* aplicam máquinas de estado para representar o comportamento dos personagens. Máquinas de

estado do tipo *Fuzzy*, por sua vez, permitem mais profundidade e aleatoriedade no comportamento dos personagens sendo utilizadas, também, em *The Sims*.

Um destaque na aplicação bem sucedida de técnicas de *IA*, no caso, de Aprendizagem de Máquina, é *Black & White*[2]. Tal jogo foi projetado de forma a utilizar a *IA* como parte integrante da jogabilidade. Dessa forma, o jogador controla um agente que possui capacidade de aprender. O jogo merece uma explanação resumida por seu pioneirismo na aplicação prática de conceitos-chave de Aprendizagem de Máquina e Agentes. Embora em um jogo comercial, geralmente, as decisões de *IA* sejam fechadas, o projetista de sua *IA* disponibilizou uma visão pessoal sobre a arquitetura utilizada no jogo em [EVA2001]. Os agentes do mundo do jogo, representados por uma criatura semi-controlável pelo jogador, deveriam preencher dois requisitos: i) parecer pessoas, ou seja, serem psicologicamente plausíveis, maleáveis e empáticas; ii) deveriam ser úteis para o jogador, auxiliando na resolução de desafios no mundo do jogo, sendo que para isso deveriam poder ser treinadas.

O primeiro requisito é preenchido através da aplicação de uma arquitetura de agentes *BDI* [RAO1995] (*Befief-Desire-Intention* – Crença-Desejo-Intenção). Crenças referentes a instâncias de objetos individuais são representadas simbolicamente como uma lista de pares atributo-valor, crenças sobre tipos de objetos são representadas através de árvores de decisão e os desejos são representados através de *perceptrons*. Para fazer com que o agente seja plausível, há uma explanação clara durante o jogo dos motivos do agente possuir certas crenças e desejos, sendo estas absorvidas através da percepção do ambiente por meio de seus sentidos. Quanto à maleabilidade, o agente tem capacidade de aprender através da realimentação do jogador, através de comandos explícitos, observando as atitudes dos outros agentes e refletindo sobre a sua própria experiência. Para permitir estes tipos de aprendizagem a criatura constrói dinamicamente uma árvore de decisão para minimizar a entropia do conhecimento adquirido a partir dos atributos observáveis do mundo. Tal tarefa é feita baseando-se no trabalho de Quinlan com o sistema *ID3* [QUI1986]. Para que o jogador pudesse ter empatia pelas criaturas objetivou-se assegurar que estas tivessem empatia recíproca por quem as controla. Dessa forma, a mente da criatura inclui um modelo simplificado da mente do jogador, construído através das observações do que ele está fazendo. Esse modelo mental foi projetado através de uma representação com estruturas de dados

[2] *God game* no qual o jogador faz o papel de um deus que possui seu avatar na terra e deve controlá-lo de forma a interagir com seus súditos. O avatar aprende conforme as recompensas dadas a ele.

simbólicas e não simbólicas. Outro ponto interessante para a geração da empatia é a associação de objetivos à criatura relacionados diretamente com seu controlador, como, por exemplo, o desejo de ajudar seu mestre e o desejo de ter atenção.

Rememora-se o segundo requisito referente aos agentes do *Black & White*: eles devem ser úteis para o jogador, ou seja, ajudá-lo em algumas tarefas. Embora pareça ser conflitante com o primeiro requisito, por implicar no tolhimento da autonomia do agente, a solução encontrada foi criar um agente completamente autônomo de início e que evolui, através do treinamento, para um agente que só faz as atividades ordenadas pelo jogador. Tal modificação no agente dá ao jogador sentimento de satisfação por ter conseguido treinar a sua criatura.

Black & White é o exemplo mais significativo, mas outros jogos também utilizaram técnicas de *IA* provenientes da academia. A Tabela no Anexo 1 sumariza exemplos de técnicas de *IA* para jogos e iniciativas comerciais que as aplicaram. Assim, cada vez mais a *IA* está integrada ao ciclo de desenvolvimento dos jogos e, conforme será explanado na seção referente à motivação para o projeto (Seção 1. 4), a *IA* para jogos e a sua integração com a *IA* acadêmica estão sendo consideradas com mais seriedade tanto pelos desenvolvedores quanto pela academia. Steve Rabin, editor da série *AI Game Programming Wisdom* [RABIN2004] dá uma ordem de grandeza do tamanho da área de *IA* para jogos: de 2002 ao início de 2004 cerca de 150 artigos envolvendo técnicas de *IA* para jogos foram escritos por membros da indústria de jogos. O autor também comenta que *IAs* que aprendam e se adaptem serão requeridas para a maioria dos gêneros de jogos. Ressalta-se, porém, que ainda existem desafios a serem vencidos quanto à aplicação de técnicas de Aprendizagem de Máquina no contexto da *IA* comercial (Seção 1. 4).

Independente da técnica utilizada, a *IA* para jogos de computador tem uma diferença importante quanto ao objetivo em relação à *IA* para jogos clássicos. Enquanto esta se preocupa em superar o ser humano, aquela se foca não na otimização do desempenho de um *NPC (non player character* – personagem controlado pelo computador), mas na diversão do jogador e sua experiência geral. Desafios persistem quanto ao desempenho dos *NPCs (non player characters* – personagens controlados pelo computador). Tais desafios, principalmente quanto aos jogos de estratégia, são o foco deste trabalho e serão discutidos nas próximas seções.

1. 3 DESAFIO

Um dos gêneros de jogo de maior sucesso é o jogo de estratégia, principalmente, o sub-gênero da estratégia em tempo real (*RTS – Real Time Strategy*). O desempenho da *IA* dos *RTSs*, entretanto, é pobre para os padrões humanos, não acompanhando a evolução apresentada na *IA* de jogos clássicos. Tal atraso se deve a uma série de motivos, segundo Buro [BURO2003]:

- O mundo do jogo apresenta muitos objetos, informação parcial e ações tanto no nível micro quanto macro;
- O mercado limita os recursos destinados à CPU;
- Há pouca competição entre os pesquisadores para desenvolver *IAs* que competem entre si devido à falta de padronização na definição da arquitetura da *IA* entre diferentes jogos. E, mesmo as iniciativas existentes não são focadas em jogos de estratégia de múltiplos objetos;

Devido ao primeiro fator citado, a *IA* dos jogos de estratégia utiliza, geralmente, máquinas de estado para representar seu comportamento. Se, para um jogo de informação parcial como *Bridge* é muito caro computacionalmente utilizar um método de busca devido ao tamanho da árvore gerada, em um jogo de estratégia as possibilidades elevam mais ainda a complexidade. A aplicação de máquinas de estado, porém, implica em previsibilidade para o sistema, pois as máquinas de estado caracterizam-se por serem determinísticas.

O último problema a ser contextualizado é relacionado à avaliação da criação da *IA*. Conforme o que já foi dito na seção anterior, as técnicas de *IA* para jogos de computador diferenciam-se por objetivar a maximização da diversão. Como medir o quanto um jogador está se divertindo? Poder-se-ia convencionar que o jogador se diverte quando se sente desafiado e se estiver jogando com um adversário humano a sua altura. Assim, para saber se um jogador está se divertindo com um jogador automático seria interessante a aplicação do Teste de Turing [TUR1950] e conseqüente verificação se o jogador crê que seu adversário seja humano. Uma crítica persiste: o objetivo da *IA* não pode ser apenas imitar o ser humano, pois jogar com uma pessoa pode também não ser divertido dependendo da pessoa que é imitada.

O desafio do trabalho que se apresenta reside na implementação e teste de um sistema inteligente como jogador automático de jogos de estratégia para resolver os problemas supracitados. Tal agente imita os estilos de jogo dos jogadores utilizando técnicas de Planejamento Baseado em Casos (*PBC – Case-Based Planning*). E, tal imitação deve inserir aspectos interessantes para o jogador, surpreendendo-o através de comportamentos inesperados, e, até mesmo, gerando comportamentos inéditos.

1. 4 MOTIVAÇÕES NO CONTEXTO ACADÊMICO E COMERCIAL

A presente seção apresenta a motivação para o projeto destacando seus aspectos inéditos e importantes. O foco desta motivação está tanto no ineditismo no aspecto acadêmico, relacionado à pouca quantidade de publicações nas áreas em que o projeto se encaixa, quanto na parca utilização das técnicas citadas em jogos comerciais.

Embora a pesquisa referente a jogos clássicos remonte à década de 50, apenas nos últimos anos a academia voltou-se aos jogos de computador. O motivo deste interesse resulta do crescimento da complexidade e do realismo dos jogos de computador atuais. Assim sendo, o pesquisador pode trabalhar com outros campos da *IA*, os quais enfatizam aspectos diferentes e mais complexos que os tratados pela pesquisa com jogos clássicos.

A lista de razões pelas quais os pesquisadores da academia devem mais seriedade à indústria de jogos é extensa e foi trabalhada por Laird e van Lent em um dos primeiros artigos que prega a interação entre academia e indústria [LAIRD2000]:

- Os desenvolvedores de jogos reconhecem a necessidade de *IA* para controlar comportamentos de personagens humanos;
- A indústria de jogos é altamente competitiva e a *IA* pode ser vendida até mesmo como um fator impactante no *marketing* de um jogo. Por exemplo, o jogo *Full Spectrum Warrior*, desenvolvido originalmente para o treinamento do exército americano, destaca-se pelas suas técnicas de *IA* adaptativa. Tal fator é citado em apresentações sobre o jogo entre desenvolvedores [MIL2005], em críticas sobre o jogo e, também na documentação ou caixa que acompanha o jogo;
- Desenvolvedores de jogos são tecnologicamente antenados com as últimas inovações;

- A indústria de jogos é grande e comparável à indústria de cinema em lucro;
- O processamento gráfico, com o advento das placas de vídeo *3D*, foi movido do processador central para um processador exclusivo. Dessa forma, mais processamento pode ser destinado a *IA*. Tal mudança dá mais liberdade ao pesquisador para justificar a implementação de técnicas de *IA* mais caras computacionalmente.

Por outro lado, a indústria também passou a respeitar mais as técnicas provenientes da academia. Paul Tozour do estúdio Ion Storm discute os fatores que fizeram com que o desenvolvimento da *IA*, uma atividade relegada aos últimos meses do desenvolvimento de um jogo, passasse a ter papel decisivo com programadores destinados a *IA* desde o primeiro dia de um projeto [TOZ2002]. Primeiramente, as restrições de hardware tornaram-se menos proibitivas. O aumento da memória dos consoles e o processamento dos gráficos em um processador em separado tornaram possível destinar uma proporção maior do processamento à *IA*. Por fim, a *IA* está deixando de ser considerada pela indústria como uma subárea da "magia", ou seja, era comum, anteriormente, a separação completa da *IA* das outras fases do desenvolvimento de um jogo sem a sua compreensão quanto aos desafios e entendimento. Por exemplo, era comum a escolha de uma linguagem de *scripting* para implementar a inteligência de um jogo considerando apenas o conhecimento da equipe de desenvolvimento e sem considerar quais eram os desafios a serem resolvidos por essa linguagem e a sua adequabilidade à resolução desses desafios.

Persistem, entretanto, alguns problemas quanto à aplicação de técnicas de Aprendizagem de Máquina no mundo da *IA* comercial principalmente em um ambiente em tempo real. Os principais problemas são também citados por Tozour [TOZ2002]:

- Os sistemas de aprendizagem de máquina podem aprender as lições erradas a partir de jogadores que joguem errado;
- É difícil fazer o ajuste de parâmetros referentes à diversão e a jogabilidade. As funções de *fitness* não devem se preocupar apenas com a competência do jogador automático, mas também com a diversão de quem está jogando;
- Algumas técnicas são difíceis de modificar, testar e fazer a depuração, como, por exemplo, Redes Neurais;

- Os personagens em alguns jogos são efêmeros e é difícil (e, talvez, desnecessário) representar uma *IA* que seja observável. Muitos jogos apresentam personagens com uma vida curta ou mesmo fora do campo de observação imediato do jogador.

Especificamente sobre a proposta corrente, alguns outros tópicos devem ser considerados como as motivações mais relevantes tanto no contexto comercial quanto acadêmico.

1.4.1 Ausência de pesquisa relacionada à *IA* de jogos baseados em império

Pouca literatura existe na área de jogos de estratégia baseados em império (*empire-based games*). O tamanho proibitivo no espaço de busca faz com que as técnicas clássicas de busca que já se mostraram bem sucedidas em jogos como Xadrez e Damas não sejam aplicáveis.

Alguns dos métodos usuais para o *game playing* de jogos baseados em império são citados por Freeburg [FRE2002]: sistemas baseados em regras, máquinas de estado finitas, sistemas multi-agentes e *goal-directed reasoning*. A técnica mais comum é a utilização de regras fixas criadas no próprio código do jogo. Embora tais *scripts* que implementam a inteligência do sistema possam ser construídos de forma complexa, imitando diversos aspectos do raciocínio humano, uma vez que o jogador aprenda o seu funcionamento, o desafio é prejudicado, pois limita o jogador automático à utilização de estratégias projetadas pelo programador. Além disso, o jogador automático terá uma tendência a repetir comportamentos conforme receber os mesmos estímulos do jogador, ou seja, não **há aprendizado, mas memorização**.

Outro ponto relevante é que no caso de sistemas baseados em regras, a inteligência do jogador automático dificilmente será boa o suficiente apenas utilizando as mesmas informações disponíveis para o jogador humano. Dessa forma, os jogos comerciais utilizam-se com freqüência de estratégias que podem ser consideradas como *cheating* (trapaça), tendo acesso a informações sobre o estado do mundo que um jogador humano não perceberia. A percepção do *cheating* da *IA* por parte do jogador frustra e é alvo de críticas em muitos jogos. Um exemplo típico de *cheating* é quando, em um jogo de estratégia baseado em mapa, o jogador automático consegue visualizar a totalidade do mapa.

Consideradas estas limitações relacionadas à principal técnica utilizada nos jogos comerciais, é desejável que se desenvolvam novas técnicas que sejam adaptativas e que evitem a necessidade de *cheating* por parte da *IA*. Mock e Freeburg [MOC2002], por exemplo, utilizaram, em uma versão *Lite* (simplificada) do jogo *Empire*, uma estratégia de busca heurística hierárquica para um jogador inteligente que não usa regras codificadas. Shapiro [SHA2002], por sua vez, desenvolveu um jogador automático para o jogo *Diplomacy* utilizando aprendizagem por diferença temporal.

As referências citadas acima são as poucas que utilizam técnicas de *IA* adaptativas em jogos de estratégia baseados em império. Justifica-se, portanto que uma das principais motivações para o projeto é a necessidade de se aplicar técnicas de *IA* mais eficientes em jogos de estratégia baseados em império.

1.4.2 Necessidade de pesquisa relacionada à *IA* de jogos de computador utilizando técnicas de Planejamento

As ações referentes à implementação de agentes que antecipem e imaginem o que devem fazer são apontadas como novos desafios para a *IA* baseada em personagens para jogos [ISL2002]. A imaginação (ou, mais formalmente, o planejamento) já é foco de algoritmos para, dado um objetivo e o estado atual, executar um conjunto de ações. Entretanto, tais algoritmos não são adaptados às restrições em tempo real que são típicas de um jogo.

Além disso, mais especificamente quando se considera o planejamento efetuado por um personagem deve-se destacar que personagens diferentes podem ter comportamentos diferentes, ou seja, dependendo do estado mental, os planos podem ser pessimistas ou otimistas, por exemplo. Considerar o estado mental de quem planeja foge do escopo de um planejador tradicional e motivam o presente trabalho a buscar alternativas para a criação de agentes que possuam comportamentos diferentes.

Destaca-se que a utilização de Planejamento em *IA* para jogos é um campo atual e promissor, tendo algumas propostas discutidas na *GDC* (*Game Developers Conference*) do ano de 2005 [MIL2005]. Uma delas é aprofundada no Capítulo 2 [ORK2004].

1.4.3 Oportunidade de pesquisa sobre algoritmos para jogos *multiplayer*

O desenvolvimento de algoritmos para jogos com mais de um jogador é um desafio devido ao número de possibilidades. Encaixam-se neste estilo, por exemplo, alguns jogos de

cartas. Quando o jogo possui um número indeterminado de jogadores a complexidade dos algorimos aumenta. Destaca-se que além dos algoritmos *multiplayer* serem uma pesquisa nova [STU2002] há pouca pesquisa referente a jogos com dezenas de jogadores.

1.4.4 Oportunidade de pesquisa na área de algoritmos para jogos *multiplayer* no contexto comercial

Os jogos *online* massivos *multiplayer* (*MMOG – Multiplayer Massive Online* Game) permitem que milhares de jogadores interajam em um ambiente único. O desenvolvimento da *IA* para estes jogos é uma motivação a ser considerada por ocuparem uma fatia cada vez maior do mercado de games. O desafio principal nesta classe de jogo está principalmente em relação à inteligência destinada aos personagens que interagem diretamente com o jogador humano. Os ambientes *multiplayer* se baseiam na interação e sociabilização entre os jogadores. Uma comunidade interessante e que atraia diversos tipos de pessoas deve primar pela diversidade que pode ser estimulada, através do enriquecimento do ambiente com jogadores automáticos semelhantes ao jogador humano.

Conforme contextualizado no Anexo 3, destaca-se que a quantidade de jogos massivos lançados ou em desenvolvimento faz com que os jogadores tenham mais opções de mundos a considerar. Assim, a tendência é a geração de comunidades menores para cada jogo e, logo, com uma densidade demográfica menor. Cria-se, desta forma, um conjunto de nichos de jogadores espalhados em diversos mundos. Pode-se questionar, de fato, se é relevante a preocupação com o povoamento de um mundo *online* frente às notícias de milhões de jogadores interagindo nos principais jogos. Entretanto, cada vez mais, conforme a distribuição e divulgação dos jogos *online* ficam mais baratas e eficientes, fica explícito o fenômeno da Cauda Longa [AND2006]. Este fenômeno econômico é típico da economia na Internet, sendo caracterizado quando a receita total de uma grande quantidade de produtos de nicho com baixos volumes de vendas, é igual à receita total dos poucos grandes sucessos. É relevante, portanto, este povoamento de jogos massivos com jogadores diversos e que maximizem a diversão mesmo com poucos jogadores humanos.

1.4.5 Necessidade de jogos assíncronos para atingir os jogadores casuais

Os principais jogos *online* massivos atuais enfatizam a interação síncrona entre os seus participantes. Tais jogos, através da utilização de um avatar em um mundo virtual, exigem,

para que a evolução de um jogador seja efetiva, sua dedicação por centenas de horas. Enfatiza-se, no estilo de jogabilidade dos jogos atuais, a saída do jogador do mundo real em favor do mundo virtual.

Pode-se questionar que esse nível de dedicação exigida afugenta os jogadores casuais dos jogos massivos. Jogos assíncronos, por sua vez, surgem como uma maneira de conectar pessoas em espaços *multiplayer* com maior flexibilidade e em conformidade com a vida diária [BOG2004]. As interrupções entre os turnos dos jogadores são partes inerentes ao jogo assíncrono. Assim, os jogadores não precisam estar conectados de maneira síncrona por diversas horas para se divertir com a sua comunidade, podendo escolher, com liberdade, qual o melhor momento de jogar e integrar o jogo com sua vida diária.

O modelo assíncrono configura-se interessante para jogos de celular, por exemplo. Justifica-se tal observação referenciando o sucesso da troca de mensagens *SMS* (*Short Message System*) como uma das aplicações de maior sucesso nas redes celulares devido a sua natureza rápida e de fácil integração ao dia-a-dia.

Outro tipo de jogo com um aspecto assíncrono bastante forte é a classe dos *ARGs* (*Alternate Reality Games* – Jogos de Realidade Alternativa), na qual situações de jogo são combinadas com a realidade, interagindo com mídias do mundo real como *sites*, telefonemas e vídeos. Nestes jogos, semelhantes a gincanas virtuais, geralmente seguem-se pistas com o objetivo de descobrir novas pistas, caracterizando-se por exigir e encorajar a interação entre os jogadores. *ARGs* são uma forma de *MMOGs* nos quais aproveita-se a mídia *online* como uma forma conveniente e barata de distribuição [MAR2006].

Jogos assíncronos massivos oferecem uma maneira factível de inserir os jogadores casuais no mundo dos jogos massivos *online*. A aplicação de técnicas de *IA* tem aplicabilidade na geração de jogadores automáticos interessantes para povoar os mundos *online* assíncronos. Destaca-se que diferentemente dos jogos síncronos, os quais exigem uma resposta em tempo real, abre-se, em um mundo assíncrono persistente, a possibilidade de aplicação de técnicas *offline* de *IA*.

1.4.6 Jogos de estratégia são um campo de testes para *IA* em tempo real

Uma grande variedade de problemas fundamentais da pesquisa com *IA* em tempo real podem ser estudados através dos jogos de estratégia [BURO2003]:

- Planejamento em tempo real das ações do adversário;
- Tomada de decisão mediante a incerteza;
- Modelagem do oponente através de Aprendizagem de Máquina;
- Raciocínio espacial e temporal;
- Gerenciamento de recursos;
- Colaboração;
- *Pathfinding*.

Após exposição das motivações do projeto, passa-se à descrição sucinta do trabalho e o que será apresentado nos próximos capítulos.

1. 5 TRABALHO DESENVOLVIDO

Efetuou-se a criação de um jogador automático para a classe dos jogos de estratégia baseados em impérios (*empire-based strategy games*) e um conjunto de ferramentas para a criação de jogadores automáticos neste tipo de jogo. O jogo utilizado para testar a arquitetura chama-se ***Promisance***. ***Promisance*** é um jogo *multiplayer* que pode ser jogado via *Web* através de uma interface *HTML* e um servidor *PHP*.

O objetivo do jogo é gerenciar um império de maneira a maximizar o tamanho e o dinheiro deste império. O jogador pode, dentre outras ações possíveis: explorar o ambiente, construir mercados, treinar tropas e atacar outros impérios. O ataque, se bem sucedido, implica no ganho de território. Cada ação desempenhada no jogo gasta um número de turnos que são limitados. A cada período de tempo o jogador ganha mais um pacote de turnos, o qual deve gastar, sendo penalizado se acumular turnos além de um limite.

Considerando que o jogador possui cerca de vinte ações possíveis (descritas na seção 3. 4 e, com mais detalhes, no Anexo 2) e o jogo permite infinitos adversários, é impossível a aplicação uma técnica de busca tradicional para a criação de um jogador automático que jogue ***Promisance***, encaixando-se no cumprimento das motivações descritas nas seções anteriores. Rememora-se também a taxonomia aplicável ao ***Promisance***: é um jogo de informação parcial, pois um jogador não tem informações completas sobre o estado do adversário. Quanto

ao aspecto do determinismo das ações, o jogo é **determinístico**, ou seja, não é considerável a influência da sorte.

Assim, o principal produto do projeto desenvolvido é um sistema inteligente que possui jogadores automáticos construídos através de técnicas de Planejamento, mais especificamente *PBC*. Justifica-se em capítulo subseqüente o porquê da aplicabilidade desta técnica no projeto conforme suas características.

Devido à característica *multiplayer* do jogo é possível armazenar as ações e estados de todos os jogadores que interagirem com o sistema. Dessa forma, propõe-se a criação de módulos de aprendizagem que imitem os planos dos jogadores e abstraiam suas características comportamentais em relação à execução das ações no jogo, imitando, portanto, os estilos de jogo. Propõe-se, também a criação de camadas que corrijem e antecipem erros nestes planos, aprendendo também regras não explicitadas sobre o funcionamento do jogo.

1. 6 CONTRIBUIÇÕES RELEVANTES

As principais contribuições deste projeto são:

- Discussão e aplicação de técnicas de *IA*, mais especificamente *PBC*, em jogos de estratégia;
- Discussão e aplicação de técnicas de *IA*, mais especificamente *PBC*, em jogos *multiplayer online*;
- Metodologia que possibilita a utilização dos planos gerados pelos usuários dos jogos *online* para criar jogadores automáticos que imitem o estilo de jogo da comunidade;
- Através da aplicação da correção dos planos e da antecipação de erros nos planos, o sistema é capaz de criar planos inéditos e explicar falha em planos anteriormente aplicados;
- Criação de uma ferramenta que facilite a construção de jogadores automáticos que utilizam *PBC*;
- Proposição de uma arquitetura de sistema que seja genérica e adaptável a outros jogos *online* de estratégia.

1. 7 ORGANIZAÇÃO DO DOCUMENTO

Este trabalho tem por objetivo, portanto, apresentar um jogador automático inteligente para jogos de estratégia baseados em turnos. A inteligência do jogador automático será uma aplicação de técnicas de *PBC*, conforme já citado anteriormente.

Divide-se o documento em cinco partes principais:

- *Introdução*: na qual se delimita a área do conhecimento na qual o problema se classifica dentro do contexto da *IA*. Também possui seção referente à motivação, ou seja, comentários referentes à importância e contribuições do trabalho;
- *Revisão Bibliográfica*: discussão das principais iniciativas utilizadas para o embasamento da proposta de resolução do problema descrito;
- *Metodologia*: descrição das técnicas e teorias que foram aplicadas para resolver o problema e quais serão os procedimentos de validação aplicados;
- *Resultados*: resultados obtidos com o teste do jogador automático tanto em um ambiente real quanto em um ambiente simulado;
- *Conclusões e Trabalhos Futuros*: Fechamento deste trabalho, reafirmando suas contribuições relevantes e comentando sobre os resultados mais importantes obtidos.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo contextualiza as principais áreas em que esta pesquisa se insere. Considerando que o problema já foi definido na Introdução como pertencente à área de *IA* para jogos de entretenimento, faz-se necessária a focalização da classe de jogo que será tratada: os jogos *online*. Faz-se necessário, portanto, uma contextualização com a história dos jogos *online* citando os principais marcos e o panorama atual.

Tal discussão é sucinta, pois foge do escopo este documento, o qual se concentra em *IA*. Dessa forma, o tópico que discute a genealogia dos jogos *online* aparece no Anexo 3. Assim, esse capítulo se concentra na discussão das duas subáreas da *IA* envolvidas no projeto: Planejamento e Raciocínio Baseado em Casos. Além dos principais fundamentos de cada área, algumas aplicações em jogos também são discutidas.

2. 1 PLANEJAMENTO

Um dos grandes desafios da *IA* é a geração de um solucionador geral de problemas. Tal programa aceitaria descrições gerais de problemas e chegaria automaticamente a uma solução. Duas são as motivações para a criação deste tipo de programa segundo Geffner [GEF2002]. Primeiramente, facilitaria a compreensão dos processos de pensamento humano, pois o ser humano é um solucionador geral de problemas por natureza. Por outro lado, no aspecto técnico, a modelagem de programas em um alto nível é, geralmente, mais simples que codificar sua solução de forma procedural.

As técnicas de Planejamento vêm de encontro a essa necessidade de resolver problemas em um alto nível. Uma técnica baseada em Planejamento se preocupa com a síntese automática de ações ordenadas para resolver um dado problema. Tais ações são geradas a partir de três partes que descrevem a situação [RUS1995]: um estado inicial, um estado final desejado (objetivo) e operadores (ações), sendo os últimos responsáveis por executar as transformações de um estado a outro.

É importante a diferenciação de uma estratégia de Planejamento em relação às técnicas de resolução de problemas baseadas em busca. Três idéias-chave diferenciam as técnicas

[RUS1995]. Primeiramente, no domínio do Planejamento, a representação de estados, ações e objetivos é aberta. Geralmente, os estados e objetivos são representados através de uma linguagem formal como predicados da lógica de primeira ordem. As ações podem ser representadas por pré-condições e efeitos que permitem a associação entre os estados e as ações possíveis para aquele estado. Em um problema de busca, tal conexão não é explicitada, provocando a expansão de uma árvore de busca mesmo que a ação a ser tomada não tenha relação com o estado.

Outro ponto-chave é a liberdade do sistema planejador em adicionar ações ao plano sempre que forem necessárias e em qualquer ordem. Dessa forma, não há uma conexão entre a ordem de execução do processo de Planejamento e a escolha da ordem de execução das ações.

Finalmente, a maior parte dos objetos é independente das outras partes em problemas de Planejamento. Essa característica faz com que se possa dividir o problema para resolvê-lo. O plano gerado pode, portanto, ser subdividido em subplanos com poucas interações entre si e gerados de maneiras diferentes. Problemas com grande interação entre os subplanos (*puzzles*, por exemplo) são mais adaptáveis a estratégias de busca do que a utilização de Planejamento.

Existem diversas representações possíveis para os estados e operadores as quais permitem a representação dos problemas de Planejamento. A representação clássica é caracterizada pelo uso da linguagem STRIPS [FIK1971]. Os estados, tanto o inicial quanto o objetivo, são representados por conjunções de predicados aplicados a símbolos, possivelmente negados. Os operadores ou ações consistem em uma descrição da ação, de pré-condições e de efeitos. Pré-condições são conjunções de literais positivos que representam uma condição que deve ser verdadeira antes que a ação possa ser executada. Os efeitos, por sua vez, podem ser conjunções de literais positivos ou negativos e descrevem o que acontece após a execução da ação.

Os *planners* clássicos consideram as transições entre os estados determinísticas e o conhecimento completo do estado inicial. Em caso contrário, é necessário trabalhar com uma representação que considere a incerteza.

Várias outras taxonomias podem ser aplicáveis aos problemas de Planejamento. Faz-se necessário, portanto, o corte e a focalização na sub-área específica de *IA* para jogos. Assim,

diversas aplicações de Planejamento em jogos foram desenvolvidas tanto no contexto acadêmico quanto comercial. As próximas seções destacam algumas destas iniciativas.

2. 2 PLANEJAMENTO APLICADO A JOGOS

Conforme já comentado na introdução deste trabalho, a pesquisa sobre *game playing* fixou-se, por muito tempo, em resolver os problemas usando técnicas de busca e algoritmos como o MINIMAX. Entretanto, essa abordagem difere da maneira como o ser humano pensa. A mente humana não busca muitas jogadas a frente e, além disso, a busca é feita de forma lenta. Por outro lado, a mente humana aprende e cria estratégias facilmente e é guiada pela intuição e por aspectos emotivos. Outro ponto a ser considerado é o poder do cérebro humano em reconhecer padrões e gerar planos a partir dos padrões reconhecidos.

Assim, a ação de reconhecer os padrões (o estado do mundo) e escolher um plano a seguir é uma atividade de planejamento. Como a ação de planejamento também pode ser feita de maneira computacional, existem diversas iniciativas que aplicam técnicas de Planejamento a jogos clássicos e a jogos de entretenimento. A próxima seção trata das aplicações clássicas, relacionadas aos jogos com regras bem definidas como Xadrez e *Bridge*. Em seguida, algumas aplicações de técnicas de Planejamento em jogos comerciais são citadas.

2.2.1 Técnicas de planejamento em jogos clássicos

O jogo que obteve maior sucesso na aplicação de métodos de Planejamento foi o jogo de *bridge* [GHA2004]. Seu estudo principal foi o trabalho de Nau, Smith e Throop para o desenvolvimento do software *Bridge Baron* [SMI1998]. *Bridge* é um jogo de cartas no qual quatro jogadores competem entre si sem conhecer as cartas do adversário. Desta forma, é um jogo de informação parcial. A possibilidade de distribuição inicial das cartas é muito grande. Caso fosse aplicada uma técnica de busca, no caso médio 10^{24} nós de uma árvore de busca deveriam ser percorridos e no pior caso 10^{44} nós seriam necessários [SMI1998].

Além disso, diferentemente do Xadrez, no qual cada partida dura horas, cada jogada de *Bridge* tem um tempo de duração de poucos minutos. Para reduzir o tamanho do espaço de buscas, Nau utilizou uma técnica de Planejamento Hierárquico (*HTN Planning – Hierarquical Task Network Plannning*). Nesta abordagem os planos são criados por decomposição de tarefas (*task decomposition*). Neste processo, o sistema decompõe as tarefas em subtarefas

cada vez menores até que sejam encontradas tarefas primitivas que possam ser executadas diretamente. Tais tarefas são denominadas métodos.

A abordagem supracitada aplica decomposição *HTN* para gerar uma árvore de busca na qual cada movimento corresponde a uma diferente **estratégia** e não a uma diferente carta. Assim, a base de conhecimentos do sistema conhece diferentes estratégias, as quais são coleções de métodos. Caso um movimento não se encaixe em uma estratégia plausível ele não é considerado na busca. As árvores de busca geradas com a aplicação desta técnica foram reduzidas para 26.000 nós no caso médio e 300.000 nós no pior caso. Destaca-se que o programa comercial *Bridge Baron* implementa tal abordagem sendo o vencedor do campeonato mundial de *Bridge* por computador em 1997 [GHA2004].

As razões para o sucesso da aplicação de técnicas de Planejamento em *Bridge* exemplificam os tipos de jogos nos quais o Planejamento melhor se encaixa segundo Tella [TEL1997]. O jogo deve possuir um número pequeno de estratégias disponíveis as quais podem ser programadas facilmente. O número de interações dos planos no domínio do jogo também deve ser controlável e a cada jogada o número de esquemas possíveis do conjunto de estratégias deve, também, ser pequeno.

Em jogos como Go e Xadrez o conhecimento estratégico não é suficiente e, por isso, apenas a utilização do Planejamento não é suficiente para a construção de um jogador automático. Em Xadrez, por exemplo, a quantidade de estratégias possível é da ordem de dezenas de milhares e, quando uma estratégia é escolhida, devem-se considerar dezenas de possibilidades de resposta para a estratégia. Um cenário mais difícil ainda se configura em relação aos jogos comerciais.

2.2.2 Aplicações de planejamento em jogos comerciais

A necessidade de utilização de Planejamento em jogos comerciais se baseia em um desafio prático. O desenvolvimento de um jogo comercial torna-se cada vez mais complexo. O jogador também espera mais do jogo. Uma das expectativas mais difíceis de ser implementada é a jogabilidade não-linear. Para que os jogos sejam mais abertos é essencial a implementação de *NPCs*, mais inteligentes para interagir com ambientes cada vez mais dinâmicos nos quais devem aprender rapidamente como agir e atuar de forma crível. Mais que o meio acadêmico, as próprias desenvolvedoras percebem que máquinas de estado e/ou regras

fixas não estão mais sendo suficientes para modelar personagens que atuam nos jogos da próxima geração [ORK2004]. A aplicação de técnicas de Planejamento pode ser um caminho interessante para lidar com esses desafios.

Ávila [AVI2004] baseia-se na técnica proposta por Smith, Nau e Throop [SMI1998] para codificar estratégias que um ou mais *NPCs* deveriam executar em um ambiente altamente dinâmico. O ambiente escolhido foi o jogo de tiro em primeira pessoa *Unreal Tournment*. Dessa forma, as tarefas de alto nível foram modeladas de forma a indicarem objetivos complexos que um agente ou time devem ter. Um exemplo de tarefa de alto nível é *Domination*(X) a qual pode ser representada na Figura 1. Tais tarefas de alto nível, denominadas métodos, são representados por pré-condições, subtarefas e ordenações. Observa-se que as pré-condições no método *Domination*(X) (responsável por dominar uma região) são relacionadas a um conjunto de agentes e não ao comportamento individual de um agente. No caso deste método, para sua execução é necessário, dentre outras pré-condições, a garantia de um certo número de indivíduos em cada time (pré-condição *numberPlayersTeam*) e a divisão em três grupos (pré-condição *Divide3Groups*), dentre outras. Como sub-tarefas a serem realizadas, tem-se a patrulha (*PatrolLocations*) e cobertura (*CoverLocations*) do terreno. No caso deste método não há nenhum tipo de relação de ordem temporal.

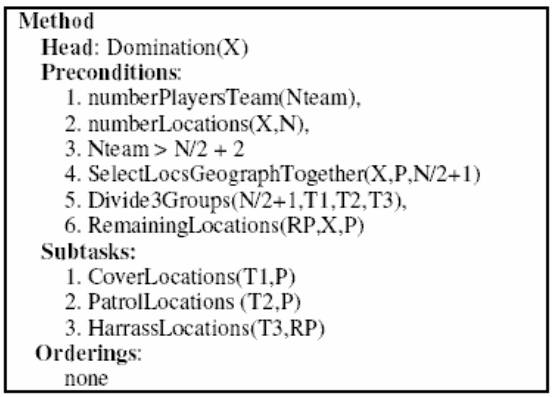

Figura 1 - Exemplo de método para decomposição da tarefa Domination [AVI2004].

As tarefas de baixo nível, ou seja, decompostas a partir das de alto nível, variam desde objetivos intermediários como capturar certa localização até ações concretas como atacar um inimigo próximo. Deve-se destacar que o uso de *HTN Planning* é justificado pelo fato de que além de ser mais expressivo que o STRIPS *Planning* [FIK1971], o planejamento hierárquico é

natural em domínios como o de planejamento militar. Ações militares são feitas em níveis, havendo uma distinção entre os níveis operacional, estratégico e tático. Cada nível é controlado por um componente diferente da hierarquia militar. Por exemplo, enquanto o sargento comanda as tropas em terra, os generais planejam as ações no nível estratégico. Ou seja, enquanto o nível estratégico se preocupa com uma visão mais a longo prazo da batalha, o nível tático traduz as decisões estratégicas em ações efetivas de médio prazo. O nível operacional é eminentemente técnico e direcionado a processos específicos. Dada esta descrição, é importante observar que se guarda uma certa semelhança entre o planejamento de uma batalha e o planejamento dentro de organizações empresariais.

Em uma outra aplicação de destaque no que se refere à implementação de planejamento em tempo real em jogos, a *Monolith* utiliza em seus jogos mais recentes uma arquitetura modular para ajudar os *game designers* (projetistas de jogos) a caracterizar o comportamento dos *NPCs*. Orkin descreve a arquitetura *Goal Oriented Action Planning* (GOAP – Planejamento de Ações Orientado a Objetivo) em [ORK2004]. Em tal arquitetura, modelada utilizando a linguagem *Planning Domain Definition Language* (*PDDL* – Linguagem de Definição de Planos), definida em [MCD1998], o *NPC* possui em um nível macro um conjunto de objetivos a serem satisfeitos. O *NPC* tenta satisfazer esses objetivos baseando-se em uma priorização referente à situação corrente. Como saída, o planejador gera uma seqüência de ações para satisfazer o objetivo. *PDDL* é modular, representando a *IA* do *NPC* em objetivos e ações (com pré-condições e efeitos). Os objetivos e as ações são desacoplados, não existindo mapeamento explícito entre objetivos e suas ações e ações e outras ações. Tal característica facilita o compartilhamento de comportamentos entre *NPCs* diferentes.

Mesmo com a pouca aplicação atual, Planejamento é uma técnica com grande potencial de contribuição para a indústria de jogos. Conforme citado por Orkin [ORK2004], os projetos estão cada vez maiores e seria uma economia interessante o compartilhamento de comportamentos entre *NPCs* e, até mesmo a reutilização de comportamentos entre projetos diferentes. A Figura 2 representa uma situação comum na qual *NPCs* de um mesmo jogo têm alguns comportamentos semelhantes e outros diferentes. O mesmo ocorre entre *NPCs* presentes em gêneros de jogos diferentes. Por exemplo, na Figura 2, os objetivos de um agente são representados por quadrados e as ações possíveis por círculos, tanto um Soldado (*Soldier*) quanto um Alienígena (*Alien*) apresentam o comportamento de Ataque (*Attack*) e os

objetivos de Perseguição (*Chase*) e Eliminação de Ameaça (*Eliminate Threat*). Caso fosse utilizada a estratégia de implementação de uma máquina de estado ou regras para cada comportamento, conforme normalmente é feito nos jogos comerciais, além da codificação de tais máquinas ser responsabilidade do programador, podem-se criar relações artificiais de herança ou repetição de código. Por exemplo, mesmo o soldado e o alienígena possuindo comportamentos semelhantes há diversas diferenças entre eles, difíceis de serem representadas de maneira realista em uma abordagem orientada a objetos devido às preocupações com o momento e com os meios que um *NPC* realiza uma ação.

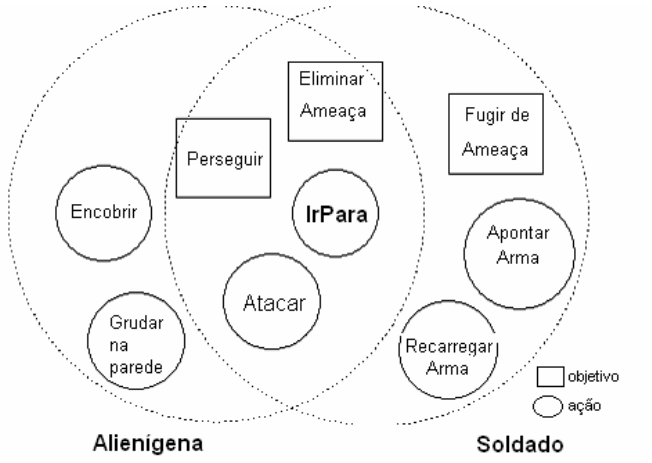

Figura 2 - Comportamentos compartilhados entre diferentes *NPCs* [ORK2004].

A utilização de uma técnica de Planejamento permite a definição em alto nível dos comportamentos de forma separada do código. Enquanto os programadores preocupam-se em implementar as ações e objetivos de forma atômica, os projetistas apenas especificam **quais** são as ações, pré-condições, efeitos e objetivos de cada *NPC*, mas sem se preocupar com **quando** e **como** um *NPC* realiza a ação.

2. 3 RACIOCÍNIO BASEADO EM CASOS

Raciocínio Baseado em Casos *(RBC - Case-based reasoning*) é um método de resolução de problemas no qual problemas são resolvidos relembrando-se situações anteriores similares e reutilizando-se tais situações para chegar a solução de um novo problema [RIE1989]. Constitui uma técnica particularmente apropriada quando existe um histórico bem documentado de casos resolvidos. Tornou-se uma metodologia bastante enfatizada a partir do final da década de 80 [HAM1989] e início dos anos 90 [KOL1993]. *RBC* se baseia no

processo de raciocínio para resolução de problemas. Tal processo é muito semelhante ao processo mental dos advogados para tratar um caso: constroem argumentos a partir de casos precedentes (jurisprudências) [KOL1993].

RBC contrasta diretamente com os sistemas baseados em regras. Nestes, a experiência a partir do conhecimento do especialista permite a geração de regras para se chegar a uma solução. O *RBC*, por sua vez, enfatiza o uso de uma memória, inspirada nos trabalhos de Schank e Abelson sobre Memória Dinâmica [SCH1982], representada por uma base de conhecimentos indexada na qual os problemas e soluções estão armazenados. Tal experiência é armazenada sob a forma de casos pertencentes a uma biblioteca de casos. Essa biblioteca pode ser alimentada por novos casos conforme novos problemas forem resolvidos, ou seja, o aprendizado em *RBC* pode ser incremental e *online*.

O caso é um episódio vivido que permite uma representação simplificada da realidade do conteúdo da experiência e reorganizado para facilitar a melhoria da performance conforme novos casos são acumulados. Quanto maior a experiência, mais recheado de casos é o sistema, mas, não necessariamente, mais competente. *RBC* tem sido utilizado em uma série de tarefas de resolução de problemas de foco analítico ou sintético. Assim sendo, problemas típicos de classificação (seleção de produtos, detecção de fraudes e até mesmo, determinação de sentenças de crimes [BAI1986]) e/ou que demandam a criação de uma nova solução (planejamento da manufatura, gestão de redes, planejamento de batalhas [GOO1989]) são aplicações comuns de *RBC*.

A representação de um caso contém o problema e a solução e depende dos requisitos do domínio e da tarefa. Casos podem conter em sua representação elementos como vetores de características, pares ordenados atributo-valor ou predicados da lógica de primeira ordem. Para acelerar o processo de recuperação dos casos, deve existir um processo de indexação cujo objetivo é dar ao sistema conhecimento sobre como armazenar e comparar casos levando em conta o propósito do que o sistema faz. Por exemplo, para um mecânico e para um cliente de locadora, a descrição de um automóvel é bem diferente. Assim, a indexação deve ser feita a partir de atributos ou características também diferentes.

O cálculo da similaridade é um conceito essencial ao *RBC* e permite que se compare quando dois casos são parecidos. Por exemplo, para uma representação de casos como pares atributo-valor, a similaridade poderia ser calculada como a soma da similaridade entre cada

atributo do caso aplicando-se um peso diferente para cada atributo. O cálculo da similaridade é utilizado em diversas etapas do ciclo de funcionamento do *RBC* e, primeiramente, na recuperação do caso.

A Figura 3, a partir da proposta de Aamondt e Plaza [AA1994], sumariza o funcionamento desse ciclo.

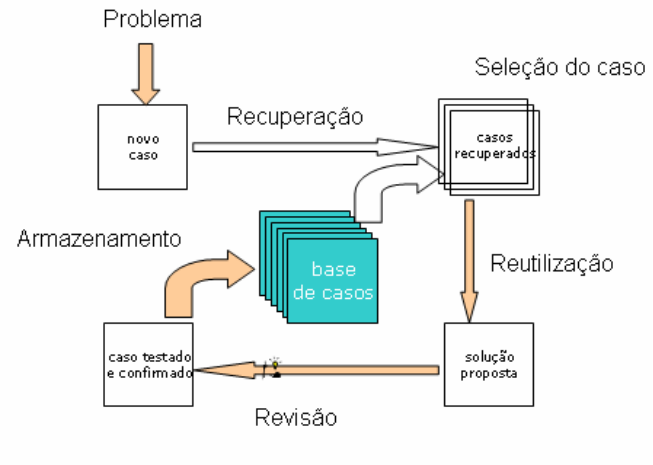

Figura 3 - Ciclo do *RBC*.

Seguindo a figura acima, o Ciclo de Funcionamento do *RBC* pode ser resumido em cinco passos, segundo descrição de [KER2003]:

1. *Recuperação*: Dada uma nova observação, recupera-se um conjunto de casos na base de casos a partir das características relevantes e indexadas da observação.
2. *Seleção do caso*: Dentre estes itens recuperados, é selecionado o mais similar segundo algum critério de similaridade. Tal caso servirá como base para interpretação da observação.
3. *Reutilização*: O caso é adaptado a partir da diferença entre a antiga observação e a nova.
4. *Revisão*: O novo caso é aplicado e sua aplicabilidade é avaliada.
5. *Armazenamento*: Se bem sucedido, o caso é armazenado na biblioteca e indexado.

Merece alguns comentários a etapa de adaptação do caso. Diversas formas de adaptação são discerníveis segundo a visão geral dada por Wilke [WIL1998]. A adaptação pode ser nula, por exemplo, em um problema de classificação cuja saída é apenas uma classe na qual o problema está associado. A adaptação pode ser feita de maneira estrutural (*transformational adaptation*), com a utilização de regras ou fórmulas para preencher ou alterar características do caso que está sendo adaptado. A adaptação pode ser derivacional (*generative adaptation*), reutilizando no caso corrente as regras ou fórmulas que produziram a solução no caso recuperado. Além destas principais formas de adaptação, também se distinguem a adaptação composicional e a adaptação hierárquica. Na primeira, novos componentes de solução adaptados de vários casos anteriores são combinados para produzir uma nova solução composta. Na adaptação hierárquica existem vários níveis de abstração e a adaptação é realizada de cima para baixo.

A aplicação de *RBC* tem como vantagens relevantes a solução de problemas em domínios parcialmente compreendidos e a possibilidade da experiência mostrar o que deu errado no passado, apontando para as partes importantes de um problema. Certas desvantagens também se destacam [KER2003]. Primeiramente, nem sempre é simples a aquisição dos casos. Muitos sistemas limitam-se a utilizar um especialista humano para a geração da base de casos e não aplicar nenhuma regra de adaptação. A modelagem do caso também deve ser feita com cuidado: o acúmulo de semelhanças "irrelevantes" faz com que certos casos sejam escolhidos em detrimento dos outros, ou seja, como ter certeza que as propriedades A e B serão determinantes na recuperação de um caso que contém vinte atributos?

Após essa descrição sucinta dos principais elementos do *RBC*, faz-se necessária a focalização do presente trabalho em uma aplicação de *RBC* na área de planejamento de ações em jogos de estratégia, com o objetivo de criar jogadores automáticos. A escolha de *RBC* para a resolução de tal problema vai de encontro a característica de que muitas vezes o domínio da aplicação é parcialmente compreendido, ou seja, em um jogo de estratégia nem todas as regras e implicações das regras são conhecidas pelo jogador. Além disso, *RBC* apresenta a possibilidade do sistema aprender de forma incremental e *online*, sendo uma aplicação conveniente para um ambiente em tempo-real. Na próxima seção, a aplicação de *RBC* em Planejamento é desenvolvida referenciando trabalhos nos quais ela foi utilizada.

2. 4 PLANEJAMENTO BASEADO EM CASOS

PBC consiste no reuso de planos bem sucedidos com o objetivo de resolver novos problemas de planejamento [SPA2001]. *PBC* utiliza uma diferente abordagem em relação ao Planejamento tradicional. Ao invés de um problema de busca de uma seqüência de ações que transforma um estado inicial em um estado final, *PBC* é baseado na adaptação de casos para resolver novos problemas [COX2002]. Tal necessidade de modificação dos planos é justificável devido à grande quantidade de planos possíveis [BER1998]. *PBC* é também mais uma tentativa de reduzir o custo computacional de um problema de Planejamento devido a seu espaço de busca proibitivo.

Um dos primeiros *case-based planners* foi desenvolvido por Hammond [HAM1989] para auxiliar na construção de receitas de comida chinesa. O sistema, denominado CHEF, recupera receitas de pratos e as adapta para um novo problema utilizando o conhecimento específico do domínio. Assim, dado um conjunto de objetivos e a situação corrente, o sistema age em três passos:

- Buscar na base de planos um plano que resolva um problema similar à situação corrente;
- Adaptar a solução para que encaixe nas novas circunstâncias;
- Armazenar a nova solução.

Como o conhecimento do plano pode ser incompleto um plano falho pode ser gerado. Para corrigir a falha, o planejador deve explicar como a falha ocorreu e usar a explicação para encontrar um conjunto de estratégias de reparo na memória, escolhendo uma para executar o reparo [COX2002]. A estratégia de recuperação dos planos considera, também, a antecipação de falhas, evitando situações que gerem falhas no futuro.

Os passos característicos do *PBC* podem ser uma especialização dos já citados para o *RBC* por Aamodt e Plaza [AA1994]. Em [BER1998] explica tais passos, os quais seguem a maioria das abordagens para sistemas que usam *PBC*. As seções seguintes descrevem estes passos segundo o autor.

2.4.1 Recuperação e organização da base de casos

Um caso em *PBC* é caracterizado por um problema e seu plano de resolução. Dessa forma, o planejador busca na memória soluções bem sucedidas referentes a problemas similares. Como vários candidatos podem ser encontrados, deve-se utilizar um critério para ordená-los. Tal ordenação é construída através de uma medida de similaridade. Como geralmente a adaptação do plano é necessária, a medida de similaridade deve levar em conta o esforço necessário para reutilizar a solução na resolução de um novo problema [SMY1993].

A quantidade de trabalho desta fase depende da filosofia do planejador. Duas estratégias são possíveis: *antecipate-and-avoid* e *create-and-debug*. A primeira, introduzida por Hammond [HAM1989], antecipa os problemas possíveis, procurando planos que os evitem. A segunda dá ênfase à adaptação do problema com o objetivo de tratar as falhas após elas aparecerem no processo de planejamento.

Ainda quanto ao esforço necessário para computar a similaridade, quanto maior a quantidade de casos visitados durante a recuperação, mais cara computacionalmente fica a recuperação de um caso. Entretanto, quanto maior a quantidade de casos visitados, menor o esforço necessário para a adaptação do caso. Dessa forma, existe um ponto ótimo, no qual o esforço total é mínimo. Essa situação é representada na Figura 4. A linha 1 representa o esforço de reutilização do caso, o qual, quanto maior o número de casos, mais possibilidades existem, e, portanto, mais fácil de realizar adaptações. A crescente linha 2 representa a situação de crescimento da dificuldade em se recuperar um caso conforme aumenta o número de casos a ser pesquisado. A linha 3, por fim, representa a soma entre esses dois esforços e deixa claro o ponto de esforço mínimo.

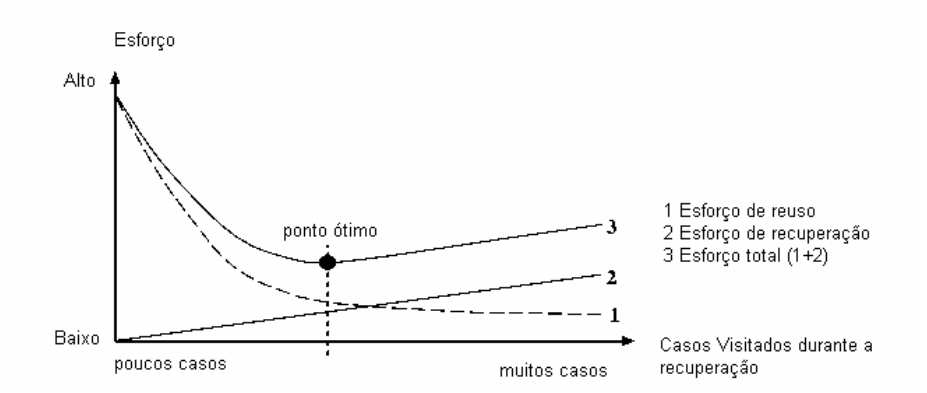

Figura 4 - Relação entre esforço para recuperação e para reuso [BER1998].

2.4.2 Adaptação e reutilização de soluções anteriores

Conforme já citado anteriormente na Seção 2. 3, são duas as principais abordagens em *RBC* para a adaptação: estrutural e derivacional. A primeira foi utilizada em planejadores pioneiros e limita-se a modificar a solução baseada na diferença entre o caso recuperado e a descrição do problema. A aplicação da segunda implica em repetir as decisões que foram feitas durante o processo de resolução do problema recuperado.

2.4.3 Revisão e avaliação da solução

A validação e avaliação da solução podem ser feitas no mundo real ou em uma simplificação deste mundo. Quando não se tem conhecimento exato do domínio, caso a validação seja feita apenas no ambiente simulado, não se pode ter certeza de sua aplicabilidade.

2.4.4 Armazenamento de novos casos

Após a execução do plano, se este for bem sucedido, é um provável candidato a ser incluído na biblioteca de planos. Entretanto, não será esta a ação a ser tomada se o caso for muito similar a algum já existente. Assim sendo, os casos novos ou que sofreram adaptação são os casos que têm maior probabilidade de serem armazenados, pois agregam um valor maior à base de casos. No caso do plano falhar, a base de planos deve ser alterada de alguma forma, para que a partir do entendimento do porquê da falha o problema possa ser prevenido.

2. 5 PBC E RBC APLICADO A JOGOS COMERCIAIS

Embora não se tenha encontrado jogos comerciais que utilizam as técnicas de *PBC*, algumas iniciativas acadêmicas testaram a alternativa em ambientes simplificados. Ambas têm influência direta sobre o trabalho proposto. As próximas seções tratam destas iniciativas.

2.5.1 RBC aplicado em *SimCity*

Em uma das primeiras aplicações de *RBC* na *IA* de um jogo comercial de destaque, Fasciano [FAS1996] criou um jogador automático para o jogo de simulação de gerenciamento de cidades *SimCity*[3]. SimCity é uma simulação em tempo real que permite que o jogador

[3] Um dos primeiros jogos de simulação que primou pelo realismo na representação das cidades. Website do jogo: http://simcity.ea.com/

assuma o papel de um prefeito. Com tais poderes, o jogador pode influenciar em diversos aspectos sociais e econômicos, por exemplo, localização de indústrias, infra-estrutura dos transportes e resolução dos efeitos de desastres naturais.

Grande parte do desafio para o jogador humano, e conseqüentemente, para o jogador automático, é prever quais serão os efeitos das construções nas cidades. Não apenas o número de construções influencia no decorrer da simulação, mas, principalmente, a localização das construções no terreno e como cada tipo de construção é ligada entre si. Como a cidade é grande e complexa, não é possível prever o efeito imediato e exato de uma ação.

O autor construiu, portanto, um planejador baseado em casos chamado MAYOR. Este agente trata os seguintes problemas, típicos de um ambiente complexo e dinâmico:

- Conciliação de tarefas de resposta imediata e tarefas de melhorias da cidade;
- Gerenciamento de interrupções, emergências e oportunidades;
- Priorização de atividades que sejam mais importantes;
- Resolução de problemas sem ter um conhecimento completo do mundo.

O último ponto merece um esclarecimento: assim como um jogador humano, o agente não possui a priori uma noção da **influência** exata de cada ação. O entendimento do mundo vai ficando mais acurado conforme aumenta a experiência do jogador.

A arquitetura de MAYOR consiste em um conjunto de módulos que monitoram e trabalham independentemente para alcançar certas condições ao mundo. Tais módulos geram tarefas que são fornecidas a um escalonador de tarefas centralizado. O agendador prioriza as tarefas e seleciona uma para execução.

A utilização de planejamento baseado em casos no sistema MAYOR aproveita uma das características mais marcantes desta técnica segundo Hammond [HAM1989] que é trabalhar com domínios de entendimento incompleto. O agente, portanto, se baseia na reutilização de um conjunto de planos que provaram ser bem sucedidos no passado. Entretanto, tais planos não serão sempre bem sucedidos, e, neste caso o sistema analisa os motivos da falha e como corrigir o plano para o domínio do SimCity.

A biblioteca de planos possuía originalmente trinta planos construídos por um especialista humano. A Figura 5 mostra o exemplo de um caso típico, indexado pelas

demandas comerciais, residenciais e industriais de uma região. O caso é encontrado, portanto, através dessas características índice. Também fazem parte do caso as expectativas após a execução do plano.

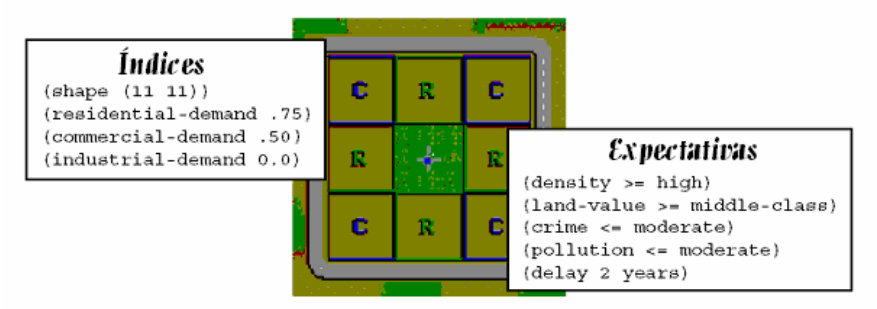

Figura 5 - Caso exemplo do sistema MAYOR [FAS1996].

A correção do plano, no caso de falhas, segue três passos. Primeiramente, quando os planos não atendem às expectativas, MAYOR tenta determinar quais fatores causaram a falha, ou seja, contribuíram para que as expectativas não fossem atendidas. Uma rede de dependência, baseada no trabalho de Riesbeck de modelagem do conhecimento na economia [RIE1983], é utilizada para gerar possíveis explicações para a falha a partir das dependências entre os parâmetros que agem no mundo. Por exemplo, quando aumenta o tráfego, conseqüentemente a poluição aumenta. Dessa forma, se a aplicação do caso da Figura 6 falhasse na expectativa relacionada à poluição (*pollution* <= *moderate*), então, talvez um dos fatores que estivesse influenciando fosse o tráfego. Uma representação gráfica da rede de dependência utilizada é ilustrada na Figura 6. Novamente, o conhecimento contido na rede foi gerado por um especialista humano.

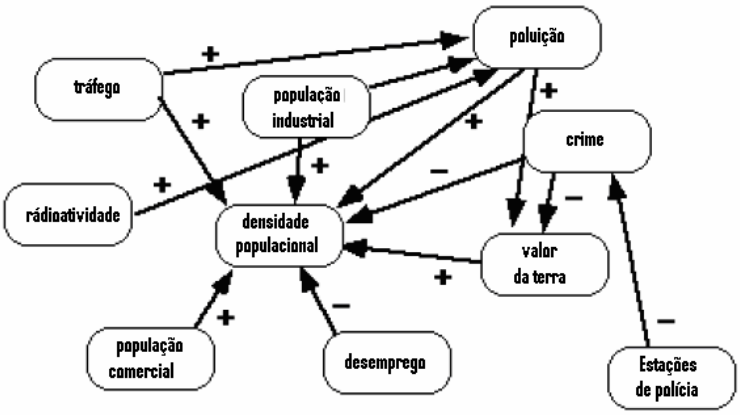

Figura 6 – Exemplo de uma rede de dependência [FAS1996].

O próximo passo é gerar uma estratégia de reparo baseada no fator com maior contribuição para a falha. Cada fator é associado a um módulo de reparo o qual é convocado e aplica uma estratégia de reparação para a vizinhança do local onde o plano foi aplicado anteriormente e falhou.

O último passo referente à correção do plano age na prevenção de falhas e alimentação da base de casos. O reparador determina se o fator com maior contribuição para falha descoberto no passo anterior existia em algum nível antes do plano ser executado. Dessa forma, antes que o plano seja executado da próxima vez, tal fator é verificado e caso esteja com um valor problemático a estratégia de reparo é disparada antes do plano ser feito.

2.5.2 Reconhecimento de Planos em *Space Invaders*

A *IA* nos jogos atuais deve suportar o desenvolvimento de personagens mais críveis. Dois dos novos desafios para a pesquisa da *IA* de personagens para jogos, segundo Isla e Blumberg são a antecipação e a imaginação [ISL2002]. Parecer inteligente não consiste apenas na habilidade de ser previsível (agir de forma inteligente para uma certa situação), mas também na habilidade de fazer previsões e agir em **antecipação** aos eventos previstos. Além disso, ao confrontar-se com um problema, um personagem inteligente deve analisar múltiplos cenários antes de tomar uma decisão, e, segundo algum parâmetro, escolher qual o plano a executar para resolvê-lo. O personagem deve ser capaz de **imaginar e planejar** quais ações devem ser tomadas/executadas.

Fagan e Cunninghan [FAG2003] remetem aos problemas acima, principalmente à antecipação, para um sistema que prevê as próximas ações de um jogador no jogo clássico *Space Invaders*. Seu objetivo principal é dar suporte a *NPCs* que tenham um modelo do comportamento do jogador para se antecipar e se adaptar às ações deste. O sistema, chamado COMETS, usa reconhecimento de planos baseado em casos [KER2003] para implementar o poder do *NPC* em observar o comportamento do jogador e identificar planos recorrentes. Os planos observados são armazenados em uma biblioteca de planos e, em tempo real, o comportamento atual do jogador é comparado com a os casos na biblioteca para identificar qual é o plano que está sendo seguido. Deve-se comentar que o trabalho de [KER2003] limitou-se ao problema de reconhecer o plano, ficando para o futuro o suporte a um comportamento adaptativo por parte dos *NPCs*.

Uma das principais vantagens na utilização de reconhecimento de planos baseado em casos está na construção da biblioteca de planos. Ao invés de ser construída por um especialista humano, são capturados os planos dos jogadores conforme vão jogando. Isto implica na captura, também, das características de jogo e estilo dos jogadores.

Os planos são compostos de estados e ações que proporcionam transições entre os estados. Nos planejadores tradicionais consideram-se também as pré-condições para que uma certa ação seja executada. Entretanto, como o processo de [KER2003] se focou na observação dos planos ao invés do gerenciamento do processo de planejamento, não se consideraram as pré-condições.

Sobre a representação dos estados em Kerkez [KER2003], um estado é representado por um conjunto de predicados de primeira ordem. Além disso, para facilitar a indexação é utilizada uma representação que divide os estados em abstratos e concretos. Fagan e Cunninghan [FAG2003], por sua vez, utilizam uma simplificação dos estados para o plano do jogador na qual três estados são possíveis, conforme ilustra a Figura 7:

- *Safe (a)*: quando a nave está em segurança atrás de um obstáculo;
- *Unsafe (b)*: a nave está em "terreno aberto", mas não está sendo ameaçado por tiros;
- *Very unsafe (c)*: a nave está em "terreno aberto" e ameaçada por um tiro.

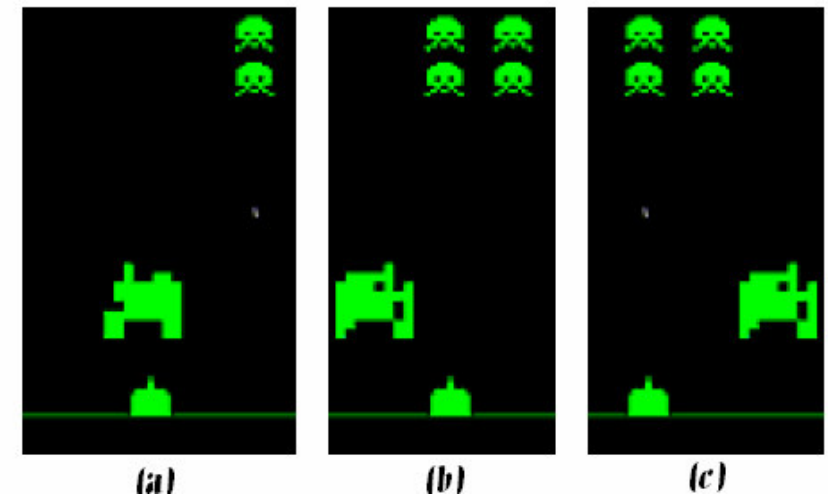

Figura 7 – Estados possíveis no jogo Space Invaders .

As ações que servem de ligação entre os estados são seis: *fire, hide, emerge, dodge, suicide e exogeneous,* respectivamente: atirar, esconder-se, emergir, rodear, suicidar-se e evento do jogador ficar sob o fogo inimigo. A biblioteca de planos foi construída dinamicamente após o jogador interagir com o ambiente por três seções. Cada plano é dividido em pequenos subplanos caracterizados por um valor de suporte, ou seja, pelo número de vezes que o subplano aparece na base. Os planos com alto valor de suporte são incluídos na base. O reconhecimento do plano do jogador é feito através do *matching* de três passos consecutivos existentes em um subplano na base de planos.

2.5.3 *RBC* em *RTSs*

Cheng e Thawonmas [CHE2004] discutem a aplicabilidade de técnicas de reconhecimento de planos baseado em casos em *RTSs*. Os autores objetivam não a criação de jogadores automáticos, mas sim a utilização da técnica para auxiliar os jogadores humanos em tomadas de decisão. E, devido ao tamanho proibitivo do conjunto de estados, sugerem que se limite o escopo do ambiente a ser analisado, dividindo em níveis estratégicos, táticos e operacionais a consideração do caso. Outra sugestão relevante refere-se à necessidade de estudo detalhado de considerações temporais e cadeias de eventos.

2.5.4 O ambiente Wargus

Aha, Molineaux e Ponsen [AHA2005] introduzem o primeiro sistema baseado em casos projetado para vencer oponentes aleatórios em *RTSs*. Tal sistema trabalha no nível tático, sendo aplicado no jogo Wargus, um clone aberto de Warcraft II. Com um mapa de 128 por 128 e um espaço de decisão por jogada de tamanho aproximado de 1.5×10^3, é essencial a diminuição deste espaço de decisão para a criação de um jogador automático. Uma possibilidade é colocar mais conhecimento sobre o domínio, o que exige um especialista.

A abordagem escolhida, *Case-Based Tactician* (CAT), trabalha com um formato de caso composto por quatro partes:

- *BuildingState*: Valor inteiro que indica o estado do império do jogador (por exemplo iniciando uma construção, completando uma construção);

- *Description*: Conjunto de características referentes à situação atual (e.g. número de construções do oponente, número de construções do jogador, número de unidades de combate).
- *Tactic*: Seqüência de ações de contra-estratégia referentes ao *BuildingState*.
- *Performance*: Reflete a utilidade de escolher uma certa tática em relação ao *BuildingState*.

Destaca-se que após cem jogos de teste, o jogador automático consegue vencer cerca de 80% dos jogos. Deve-se rememorar, porém, que este resultado embora indique uma excelente performance, foge do objetivo do presente trabalho, que é a criação de um adversário que seja desafiador, mas possível de vencer. De [AHA2005], como trabalho aproveita-se em parte a estratégia de representação dos casos.

2.5.5 Outras iniciativas de destaque

Aha, Molineaux e Ponsen [AHA2005] apresentam a tabela a seguir quanto à pesquisa relacionada a RBC em games.

Gênero	Exemplos	Descrição	Abordagem	Tarefa a ser Otimizada
Esportes de time	RoboCup Soccer [WEN2001]	Planejamento em tempo real multiagente	Várias	Passe, seleção dos membros de um time
Ação individual	Bilestoad [GOO1989], Space Invaders [FAG2003]	Planejamento em tempo real individual	Visualização de projéteis, Reconhecimento de planos	Inflingir danos, reconhecimento de planos
God games em tempo real	SimCity [FAS1996]	Gerenciamento de cidade em tempo real	Adaptação de planos	Planejamento
Estratégia em turnos	Freeciv [ULA2004]	Gerenciamento de civilização baseado em turnos	Recuperação de falhas em planos baseado em reflexão	Defender uma cidade
Estratégia em tempo real	Wargus [CHE2004], [AHA2005]	Gerenciamento limitado de cidades em tempo real	Casos hierárquicos, seleção de planos	Gerenciamento de sub-tarefas, vencer

Tabela 2- Pesquisa relacionada a *RBC* em games [AHA2005].

Na Tabela 2, além das já citadas [FAS1996][CHE2004], é interessante destacar que a maior parte das iniciativas trata de problemas referentes a tempo real e envolvem a geração de estratégias para resolução de problemas de Planejamento.

2. 6 CONSIDERAÇÕES FINAIS

Tratou-se, neste capítulo, acerca das técnicas e teorias que serão aplicadas para a criação do jogador automático. Destaca-se, primeiramente, que quanto maior a experiência de

um Sistema de Casos, maior a competência do sistema. Além disso, um sistema baseado em *RBC* é bastante adequado para ambiente multiusuário como um jogo *online*.

Quanto à área de Planejamento, é essencial justificar o porquê da inaplicabilidade de abordagens tradicionais e a necessidade de uma abordagem baseada em casos. Primeiramente, o presente trabalho tem como objetivo a sua aplicação em um contexto genérico, incluindo descrições parciais e imprecisas de ambientes. As abordagens tradicionais exigem a definição de pré-condições para a execução e uma ação e, portanto, o conhecimento completo das regras de comportamento.

Novamente, um jogo *online* foi escolhido como aplicação, pois é possível a captura de uma quantidade considerável de planos dos jogadores diminuindo neste caso a dificuldade de reuso e também a necessidade de adaptação. Mesmo assim, a adaptação do caso será necessária, pois permitirá a extração de conhecimento acerca do sistema. O próximo capítulo descreve a metodologia de desenvolvimento mostrando quais aplicações das técnicas descritas foram efetivamente empregadas no projeto e de que maneira.

3 METODOLOGIA

Esta seção descreve a criação da arquitetura para construção de jogadores automáticos para a classe dos jogos de estratégia baseados em impérios. Tal arquitetura foi construída através de técnicas de *RBC*, mais especificamente *PBC*.

O objetivo da arquitetura é povoar o mundo do jogo com jogadores automáticos interessantes. Tal arquitetura é composta por três camadas. A primeira camada é responsável pela recuperação dos planos mais semelhantes ao estado atual do jogador automático e pela aplicação do plano no ambiente do jogo. Esta base de planos é construída a partir da captura das ações realizadas por jogadores humanos. Caso haja falha no plano, a segunda camada, denominada camada reparadora, permite a correção do plano e a anotação do possível motivo do erro. A terceira camada, por sua vez, é responsável por prevenir possíveis erros antes de ocorrerem, a partir da informação de erro preenchida pela segunda camada.

O ambiente de aplicação desta arquitetura são jogos *online multiplayer* devido à necessidade de captura de planos. Nesta seção, após a descrição detalhada de cada módulo do sistema, são justificadas as restrições, adaptações e cortes feitos no jogo original para a geração do jogador automático e detalhes da técnica de implementação do jogador automático de ***Promisance***. Por fim, o mecanismo que será utilizado para avaliação do sistema é apresentado.

3. 1 DESCRIÇÃO DO JOGADOR AUTOMÁTICO

O objetivo do jogador automático em um jogo de estratégia não é apenas maximizar sua performance. Deve-se destacar que o objetivo da *IA* para jogos de entretenimento não é a geração de um jogador que vença sempre. O que faz um jogo divertido não necessariamente corresponde a fazer os personagens com mais inteligência. O jogador humano deve ter a possibilidade de vencer de alguma maneira. A diversão pode ser maximizada, portanto, fazendo algumas falhas do *NPC* serem intencionais, mas plausíveis.

Liden [LID2004] trata desta questão da "estupidez artificial" para a classe dos jogos de tiro. Para os jogos de estratégia, alguns dos truques citados pelo autor poderiam ser

aplicáveis. Por exemplo, o *NPC* poderia atacar um império do jogador até que este estivesse quase morto. Quando o império fica num limite aceitável, a efetividade do ataque é diminuída. Assim, o jogador tem a oportunidade de revidar e, com o sucesso do revide um sentimento de "desafio vencido" é estimulado. Esta e outras técnicas poderiam ser implementadas para estimular a diversão. Entretanto, deve-se destacar que caso o jogador percebesse o truque, ao invés de divertir-se, ficaria decepcionado por ter sido enganado e a sua imersão no jogo seria prejudicada. Além disso, enquanto em um jogo de tiro, a vida útil de um *NPC* é curta, um império de um jogo de estratégia tem suas minúcias analisadas por um tempo considerável pelo jogador humano.

A arquitetura proposta para a classe dos jogos de estratégia baseados em Império não vai deixar de inserir erros e imperfeições. Entretanto, tais erros e imperfeições não serão codificados no jogador automático. O jogador automático imitará os comportamentos e estilos de jogo dos outros jogadores a partir de uma base de planos criada através da captura das ações dos jogadores.

Deve-se destacar que isso não garante a diversão do jogador. Dessa forma, além da imitação dos estilos da comunidade, o jogador automático também criará planos inéditos aprendendo a antecipar as falhas e evitá-las. Esse conhecimento, além de útil para a própria comunidade de jogadores automáticos, também pode ser utilizado, futuramente, para auxiliar jogadores humanos.

3.1.1 Geração da base de planos

A geração inicial da base de planos não é feita de maneira aleatória e tampouco sob a responsabilidade de um especialista humano. Assim sendo, inicialmente, existem apenas jogadores humanos cadastrados no sistema. Todos os jogadores são monitorados e seus planos armazenados. Deve-se destacar que um plano é formado, segundo a arquitetura proposta por Kerzez [KER2003], por uma seqüência de pares estado-ação. Quanto às variáveis de estado, elas podem ser de dois tipos: algumas delas são usadas no processo de recuperação do caso e outras não, servindo apenas como quantificadores do estado e, possivelmente, sendo usadas durante a antecipação de falhas.

Quanto às variáveis de ação, podem ser fixas, caracterizando e diferenciando cada uma das ações, ou quantificadores de ação, ou seja, parâmetros da ação que podem ser utilizados

no processo de adaptação. Destaca-se a geração de uma ferramenta que facilita a captura dos pares estado-ação. Tal ferramenta, com uma de suas telas ilustrada na Figura 8, permite que a partir da captura dos planos dadas as ações realizadas pelos jogadores, se escolha para cada variável capturada quais são as variáveis de estado, quantificadores de estado, variáveis fixas de ação e quantificadores de ação. Além disso, a ferramenta também permite a associação de filtros de entrada e de saída para os valores das variáveis. Um último passo explicitado na Figura 8 é que se optou pelo armazenamento destas informações e das meta-informações relacionadas num formato de banco de dados relacional.

Associada a esta ferramenta de geração de planos, todo o acesso à base e ao jogo em si foi encapsulado em dois *web services*. O primeiro é responsável por servir ao jogador automático de interface de acesso ao jogo permitindo a execução das ações pelo jogador automático. Outro *web service* disponibiliza consultas à base de planos para verificação, por parte do jogador automático, das ações mais adequadas em relação ao seu estado.

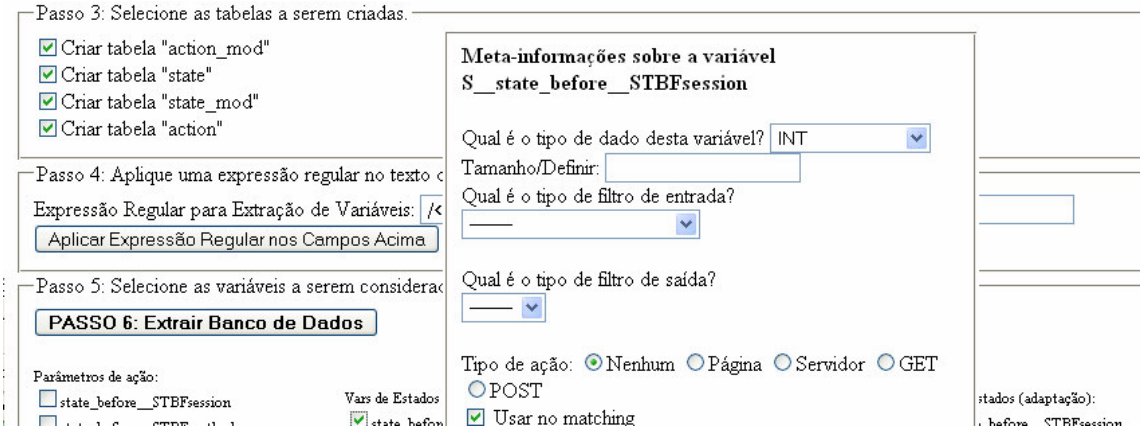

Figura 8 – Ferramenta de Apoio à Geração dos Planos.

O tamanho máximo de cada plano é o correspondente a uma seqüência de jogo do jogador, ou seja, cada plano é formado por todos os pares ação-estado desde o *login* até o *logout* do jogador em uma seção. Destaca-se que, dependendo do jogo o tamanho dos planos gerados pode ser um problema. Por exemplo, se cada jogador tiver até 300 turnos acumulados para gastar e existissem ações cujo custo é de apenas um turno, o tamanho máximo de um plano seria de trezentos pares.

Observa-se, porém, que um plano pode conter diversos componentes distintos. Além disso, alguma ação executada pelo jogador tem chance de não pertencer ao plano que está

sendo executado. Dessa forma, o processo de reconhecimento de planos vai operar em subplanos de tamanho menor. Tais subplanos serão armazenados em uma base de subplanos temporária. Os subplanos com maior suporte, ou seja, com mais ocorrências e com mais poder de maximização da posição do jogador no ranking, serão transferidos para a base de casos definitiva. Esse processo é ilustrado na Figura 9.

Um parâmetro deverá ser alvo de testes: o tamanho dos subplanos, ou seja, a quantidade de pares estado-ação a serem armazenados. Fagan e Cunningan [FAG2003] utilizam subplanos com tamanho quatro. Os planos de um jogo de estratégia, porém, são mais complexos e devem envolver mais ações. Uma janela maior, porém, implica em dificultar o *matching* de um subplano que seja semelhante a outro. Uma janela menor é desejável se o ambiente do jogo contiver poucos pares estado-ação e objetivar-se que o *matching* seja facilitado.

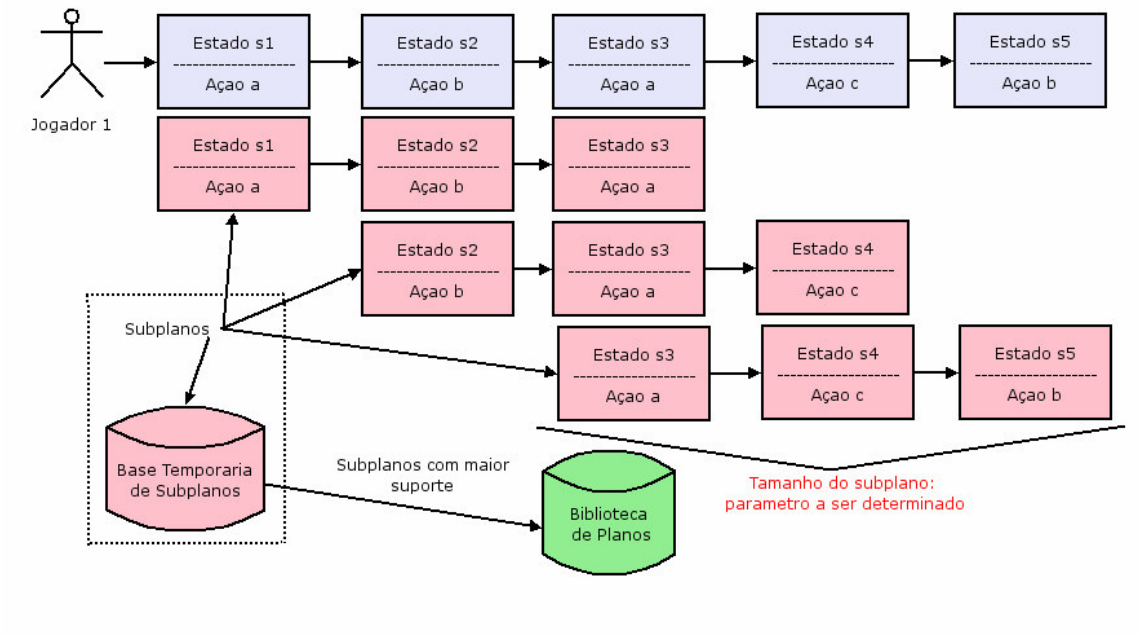

Figura 9 – Criação da Biblioteca de Planos.

Na Figura 9, o plano formado por cinco pares, representado pelo par (s1/a) até (s5/b) é, segundo um tamanho de janela de subplano igual a três pares estado-ação, gerador de três subplanos que são colocados em uma base temporária de subplanos. Os subplanos com maior suporte (os que mais ocorrem) são copiados para a Biblioteca de Planos. Observa-se que a necessidade deste passo se dá apenas quando existe a possibilidade de alta repetição de

subplanos, por exemplo, em ambientes nos quais se apresentam um número limitado de estados. Outra justificativa possível para a execução deste passo, rememora o exposto quanto ao esforço para a recuperação versus esforço para reuso (Figura 4). Se o esforço para recuperação for baixo, não se justifica a necessidade de uma base temporária.

Outro ponto a ser considerado é o método de comparação dos subplanos. Um subplano provavelmente deve ter pequenas diferenças em relação a outro. Diferenças inclusive em relação à ordem de execução das ações. Propõe-se, para facilitar o *matching* entre subplanos semelhantes, a utilização de uma representação abstrata dos planos [KER2003]. Tal representação aponta para os subplanos concretos com aspecto semelhante. Um subplano abstrato seria representado pelas ações que estão associadas a ele e pelo número de vezes que cada ação aparece no subplano. Cada subplano concreto associado ao subplano abstrato incrementa um contador de suporte. Assim, os subplanos com maior suporte são os subplanos com maior freqüência. A necessidade deste plano abstrato justifica-se caso o mundo do jogo possua como característica a repetição de subplanos. Um subplano abstrato ou estratégia pode ser considerado como pertencente ao contexto do planejamento tático-estratégico. Ou seja, se for comum a repetição de estratégias e a ordem de execução das tarefas não é tão relevante, justifica-se o uso de uma camada de planos abstratos. Entretanto, se a ordem de execução das ações é muito importante, a geração de subplanos abstratos esconde características do plano e gera uma semelhança falsa entre dois planos diferentes. Não houve preocupação com a escolha entre dois planos com a mesma representação abstrata.

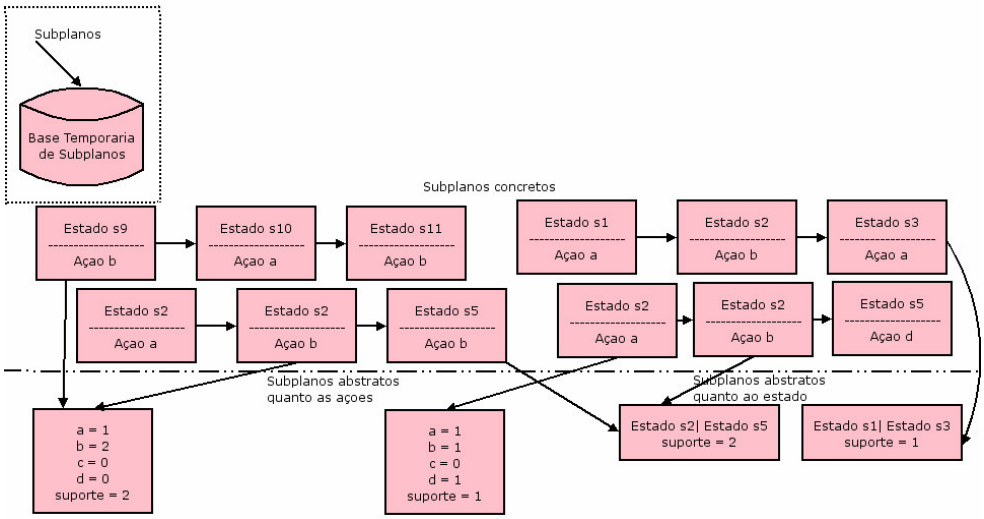

Figura 10 – Separação em subplanos concretos e abstratos.

Na Figura 10, por exemplo, estão sendo armazenados os subplanos abstratos quanto às ações. Por exemplo, a seqüência de s9/b=>s10/b=>s11/b e s2/a=>s2/b=>s5/b, são indexadas como um mesmo subplano abstrato pois os dois subplanos possuem a mesma distribuição de ações.

Subplanos com ações diferentes, porém, podem ter objetivos semelhantes. Por exemplo, o dinheiro de um império pode ser aumentado tanto pela venda de itens no mercado quanto por um ataque bem sucedido a um inimigo. Outra indexação faz-se necessária: o agrupamento de subplanos semelhantes em relação ao estado inicial e estado final do subplano. Tal indexação, além de incrementar o contador de suporte do subplano abstrato, será essencial no processo de recuperação do caso. Deve-se recordar que através da diferença entre o estado inicial e o estado final obtém-se qual é o objetivo do plano. Voltando à Figura 10, a seqüência (s2/a) até (s5/d) é indexada no mesmo subplano que a seqüência (s2/a) até (s5/b), pois possuem o mesmo estado inicial e final. Já que o objetivo final é o mesmo, a ordenação do plano mais eficiente pode ser feito através do menor plano ou o mais barato computacionalmente segundo alguma métrica do domínio.

Os subplanos com maior suporte são candidatos para a inclusão na base de planos (Figura 11). Outro parâmetro a ser configurado, portanto, é qual o valor de suporte que justifica a aceitação do subplano. De tal base de planos provêem a inteligência do jogador automático.

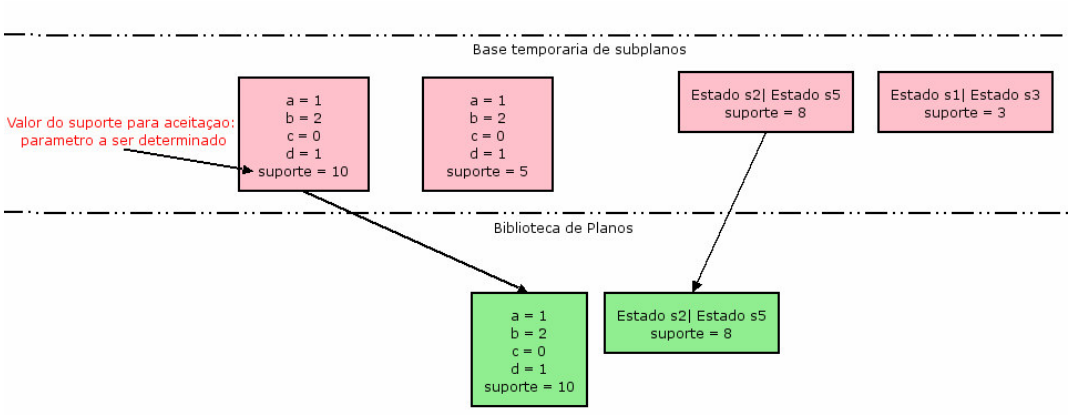

Figura 11 – Criação da Base Definitiva de Planos.

Outro critério possível de ser utilizado para o corte na base de planos é o sucesso do plano. Por exemplo, planos pertencentes a jogadores automáticos cuja performance em

relação aos outros piorou em um período de tempo poderiam ser excluídos da base de planos. Este corte é discutível, pois o objetivo do sistema é garantir a credibilidade dos jogadores automáticos. Planos mal sucedidos também podem ser interessantes de serem explicitados e trazem mais humanidade ao comportamento dos jogadores automáticos. Destaca-se que a base de planos é compartilhada por todos os jogadores automáticos. Dessa forma, os jogadores automáticos podem ter acesso a todos os planos da base de planos tanto gerados pelos jogadores humanos quanto pelos automáticos.

3.1.2 Processo de recuperação dos subplanos

Inicialmente o sistema não possui nenhum jogador automático. Decorrido alguns dias de jogo, conforme os jogadores forem interagindo com o ambiente, os planos vão sendo incluídos na base de planos e chega o momento de utilizar estes planos através da geração de uma população de jogadores automáticos que execute os planos que foram acrescentados à base de planos até o momento.

A *IA* destes jogadores automáticos possui três camadas distintas para decidir a ação executada pelo jogador automático. Tal arquitetura guarda semelhança com a arquitetura de *subsumption* de Brooks [BRO1991] na qual uma camada inibe e tem precedência sobre a outra.

A primeira camada, denominada Camada Planificadora, tem precedência sobre a segunda, a qual é responsável por corrigir erros. A terceira camada, denominada Camada Antecipadora, por sua vez, é ativada sempre após a escolha do plano que será seguido. Tal camada, conforme descrito na próxima seção, previne os possíveis erros que estiverem relacionados ao plano escolhido. No início, os planos não possuem seqüências de correção associadas a eles.

A Camada Planificadora faz uma busca em todos os estados componentes dos planos da base de planos usando algum critério de similaridade. Para o estado mais similar encontrado, a ação associada a este estado é executada. O próximo passo do jogador automático é seguir a próxima ação do plano. Todavia, antes de executar a próxima ação, o jogador automático verifica se a ação gerou o estado previsto ou semelhante. No caso de falha, o processo de busca de um estado semelhante pode ser executado novamente, se isso

for possível. Uma nova falha implica na inexistência de um estado em um plano que se adapte a situação. Neste caso, a segunda camada da *IA* (Camada Reparadora) fica responsável por decidir o comportamento do jogador automático.

Tal camada é baseada no módulo *Neighborhood Repair* contido no trabalho de Fasciano [FAS1996] com o jogo *SimCity*. O jogador automático procura verificar as diferenças entre o estado atual e o estado que era esperado com a execução do plano. Para os parâmetros com diferenças significativas é aplicada uma ou mais estratégias de reparo. Por exemplo, a ação selecionada pela Camada Planificadora era um ataque a um império de tamanho X, e raça Y, um pouco maior em relação ao império do jogador automático. Tal ação deveria conquistar as terras desse império. Entretanto, o império escolhido para o ataque foi um império semelhante ao atacado pelo plano (tamanho X e raça Y), mas as tropas do jogador automático desertaram. Nas regras do jogo, essa situação é representada por uma diferença máxima de duas vezes o máximo do número de setores do império para um ataque ser bem aceito pelas tropas. Assim, a segunda camada vai perceber-se o motivo do ataque ter sido mal sucedido está em alguma diferença entre o império do plano que foi bem sucedido e o outro e vai realizar alguma ação para ficar mais semelhante a esse império. Neste caso, o jogador automático perceberia que a diferença entre o número de setores é um fator de diferença e realizaria uma ação cujo objetivo fosse aumentar o número de setores. Só então, o jogador automático tentaria novamente executar o plano. Se o plano for bem sucedido é provável que a ação de correção tenha sido eficiente. Neste momento a rotina de prevenção realizada é associada tanto ao plano realizado quanto à ação. Dessa forma, da próxima vez que o plano for realizado, previne-se o problema.

Esta estratégia só é válida em jogos com aspecto determinístico ou quase determinístico e que seja possível o acesso a maior parte das variáveis representativas do estado. De outra forma, em um jogo cujo resultado das ações fosse dependente da sorte não seria possível a certeza que a alteração em um parâmetro teria sido a responsável por corrigir um problema. A Figura 12 exemplifica o processo descrito. Além disso, a aplicação da correção pela camada reparadora também só é possível no caso de jogos cujas ações não impliquem em uma mudança significativa no estado do jogo quando uma ação errada for realizada. Se for este o caso, ganha prioridade na execução a Camada Antecipadora e a Camada de Reparo pode ser usada para, ao invés de aplicar a correção, apenas anotar hipóteses que explicam o fracasso do plano. Na Figura 12, isso representaria a eliminação do

passo 5 e, no passo 6, o armazenamento não de um plano de correção, mas sim de hipóteses de erro que seriam testadas não mais pela Camada de Reparo mas sim pela antecipadora.

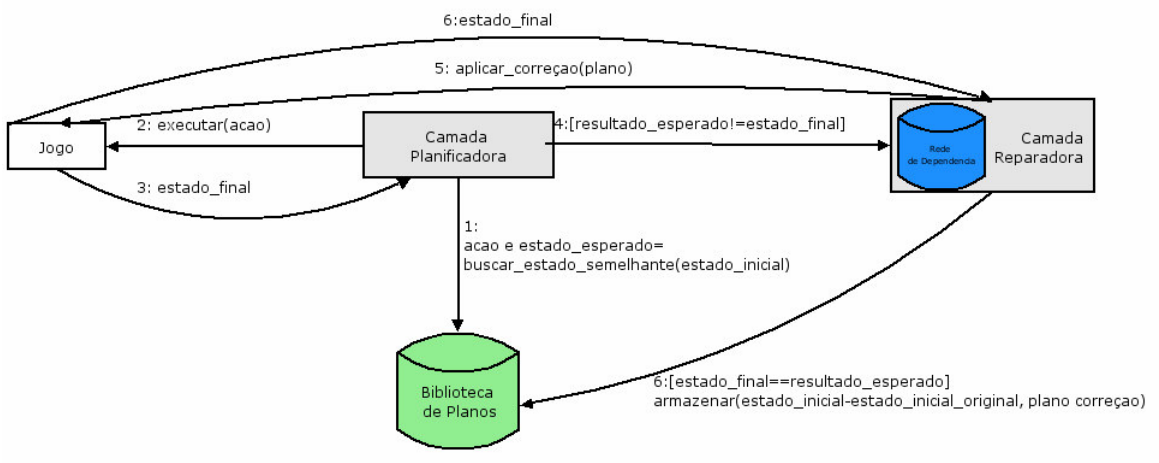

Figura 12 – Processo de Recuperação e Reparo dos Casos.

Quanto às possíveis estratégias de correção, o principal desafio de sua codificação está na interdependência entre os parâmetros. Por exemplo, uma estratégia de correção que aumente um valor de parâmetro também pode aumentar outro. Dessa forma, além do aumento de um dos parâmetros é necessária alguma ação para manter o equilíbrio nos outros. Seria interessante passar a responsabilidade a um especialista que determinasse estas interdependências a priori e explicitamente associasse ações corretivas aos parâmetros. Seguindo o modelo apresentado por Fasciano [FAS1996], pode ser utilizada uma rede de dependências para modelar as dependências entre os parâmetros. Outra alternativa seria a prospecção, na própria base de planos, de planos que provoquem a correção dos parâmetros diferentes em relação ao estado objetivo.

3.1.3 Prevenção de falhas e realimentação da base de casos

O algoritmo de prevenção de falhas introduz a preocupação em antecipar as falhas nos planos antes de sua ocorrência. Caso uma rotina de correção seja bem sucedida e um problema tenha sido detectado, ela pode ser chamada antes da execução do plano para garantir o seu sucesso.

Um problema é detectado pela Camada Reparadora através dos valores das diferenças entre as variáveis de estado entre o estado desejado e o estado atual. Cada diferença será candidata a ser combatida por uma estratégia de reparo.

As estratégias de reparo bem sucedidas provocam a associação da condição para que a estratégia seja chamada ao plano cujo reparo foi necessário. Destaca-se que as ações realizadas para o reparo do plano e o plano associado, se bem sucedido, também podem ser acrescentados à biblioteca de planos. Dessa forma, planos novos não são gerados apenas pela intervenção humana, mas também pela aprendizagem dos jogadores automáticos, os quais anotam na base de planos as possíveis pré-condições para um plano ser executado.

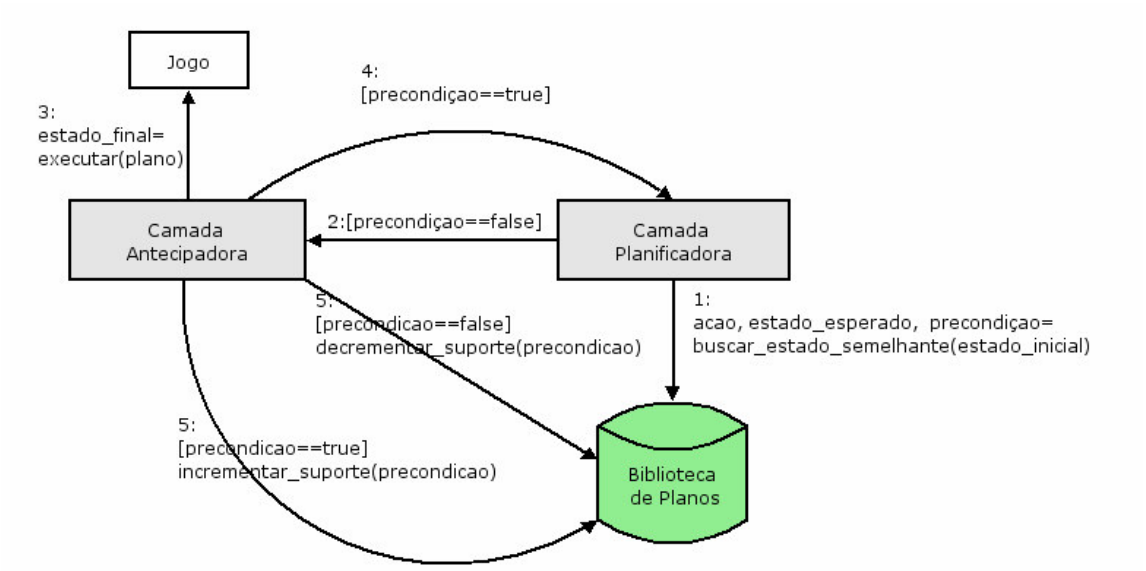

Figura 13 – Processo de Antecipação de Erros.

Na Figura 13, o processo de antecipação de erros é caracterizado pela busca de pré-condições descobertas pela camada de reparo devido a uma correção bem sucedida ou uma hipótese para correção da falha no passo 1. Caso a pré-condição seja falsa, executa-se a estratégia de reparo associada a ela (passo 3). Sendo verdadeira, retorna-se a Camada Planificadora ao comando, executando a ação desejada antes da execução da correção, conforme a Figura 12. Dependendo se houver falha ou sucesso na execução da ação de reparo é realizada a anotação em relação ao plano se aquela ação de reparo foi bem sucedida ou não. Tal anotação aumenta ou diminui o suporte à pré-condição associada à ação de reparo. Define-se o suporte como a quantidade de vezes que a ação de reparo é bem sucedida menos a quantidade de falhas. Além dessa informação poder ser utilizada para ordenação na

recuperação de ações de reparo, ela também é importante para povoar e validar a rede de dependência entre os parâmetros. Ou seja, pode-se inferir **quais ações** conseguem modificar **quais parâmetros** do estado do jogador automático.

A arquitetura proposta foi testada em um jogo *online* baseado em turnos com interface web chamado ***Promisance***. A próxima seção descreve este jogo e quais adaptações foram necessárias para encaixá-lo à infra-estrutura descrita.

3. 2 O JOGO *PROMISANCE*

Promisance[4] é um jogo *multiplayer* de estratégia em turnos que permite o gerenciamento de um império persistente. Sua interface é essencialmente textual, implicando na possibilidade de ser jogado em qualquer navegador web.

Promisance foi lançado em 1999 e é um jogo aberto cujo código está sob a licença *GPL* [5](*General Public Licence*). Seu mecanismo de funcionamento por turnos é semelhante ao dos *door games*, sendo esta classe de jogos sua principal inspiração. Os principais jogos que precederam o ***Promisance*** quanto a mecânica e temática foram *TradeWars*, *Solar Realms Elite e Falcon's Eye*.

Após se cadastrar no sistema o jogador recebe um império. Esse império é caracterizado, principalmente, pelos valores de:

- *Money (Dinheiro)*: dinheiro que o império possui. Dinheiro permite que sejam feitas construções.
- *Land (Terra)*: número de setores que o império possui. Cada construção ocupa um setor.
- *Mana (Magia)*: Energia utilizada pelos magos para fazer magias.
- *Grains (Grãos)*: Quantidade de comida disponível para a população.
- *Health (Felicidade)*: Saúde da população em geral (civis, tropas e magos). Influencia na capacidade de defesa e ataque.

Cada novo império recebe inicialmente uma quantidade padrão para cada uma das variáveis acima. Independente do momento que o império for criado, todos os novos

[4] O código-fonte do jogo está disponível em: http://sourceforge.net/projects/promisance/

jogadores recebem impérios iguais. No decorrer do jogo, porém, conforme as escolhas do jogador, tais impérios se tornarão diferentes. Além disso, durante a criação do império também pode ser feita a escolha de sua raça, o nome do império e a era temporal que ele está alocado. A raça reflete em multiplicadores para a evolução do império. Por exemplo, um império com raça Troll tem 24% a mais de poder ofensivo que um império humano com o mesmo número de tropas, mas, por outro lado, tem 10% a menos de poder defensivo que o mesmo império. A era reflete as construções possíveis de se construir e as tropas que podem ser treinadas.

O objetivo principal do jogo é fazer com que o valor do império cresça. Esse valor, denominado *networth* é calculado a partir do tamanho do império e de quanto dinheiro o império possui. Como o valor do *networth* é público, o jogo apresenta um *ranking* que permite a comparação do valor de *networth* entre todos. Considera-se, portanto, que os impérios vencedores são aqueles que estiverem nas primeiras colocações do *ranking* ao final de um *round* de jogo. Os *rounds* geralmente tem uma duração de dois a quatro meses.

Quanto ao gerenciamento do império, diversas ações estão disponíveis. Tais ações estão relacionadas, principalmente, à construção da infra-estrutura da civilização, à exploração e prospecção de novas terras e a geração de tropas. Para executar uma ação, um certo número de turnos é gasto. Inicialmente, os jogadores têm disponíveis 400 turnos. A cada 10 minutos, dois turnos são ganhos pelo jogador. Um total de 800 turnos, no máximo, pode ser acumulado para cada império.

As tropas são essenciais ao aspecto *multiplayer* do ambiente, pois os impérios podem interagir de três maneiras diferentes:

- *Ataque direto*: o ataque a um império inimigo além de diminuir as tropas do adversário, conquista parte de seu território. Caso seja mal-sucedido, o ataque diminui o número de tropas e a moral da população.
- *Ataques mágicos*: tal classe de ataques é dependente do número de magos que o império possui e da quantidade de mana disponível. Ataques mágicos são poderosos, mas são mais caros ao império.

[5] Licença de código que libera as modificações no código contanto que elas também sejam liberadas ao público.

- *Criação de Alianças*: quando dois ou mais impérios se aliam, é proibido declarar guerra entre os membros da aliança. Além disso, todos devem ajudar com o envio de tropas quando um dos membros é atacado. Um império que faz parte de uma aliança tem mais chance de evoluir.

Um império é destruído quando perde todas as suas terras. É proibido a um mesmo jogador possuir mais de um império. Essa restrição se dá pela vantagem associada às alianças em relação aos impérios que não fazem parte de alianças.

3. 3 ADAPTAÇÕES FEITAS AO JOGO

O mundo de ***Promisance*** é desafiador e complexo para o jogador iniciante. Dada a vantagem dos impérios participantes de uma aliança, o novato tende a coligar-se logo de início. Para entrar em uma aliança há uma fase de negociação que foge ao ambiente do jogo, ou seja, geralmente a entrada em uma aliança é combinada em fórum, por email ou qualquer outra forma de comunicação fora do ambiente do jogo.

Foge do escopo deste trabalho o desenvolvimento de um jogador automático apto a participar de uma negociação neste sentido. Desta forma, **todos** os tipos de comunicação entre os jogadores serão impossibilitados dentro da interface do jogo para que tanto o jogador automático quanto os jogadores humanos tenham as mesmas ações possíveis. Além disso, para dificultar a comunicação entre os jogadores humanos em um fórum externo, o jogador humano não terá acesso ao nome e identificador dos impérios inimigos ao acessar o *ranking*. Por exemplo, o jogador A visualiza para os impérios B, C e D, os nomes X, Y e Z. O jogador E, por sua vez, enxerga os nomes Z, X e Y para os impérios citados, respectivamente. A Figura 14 mostra a nova interface gráfica implementada para o jogo e visão que um império tem de seus adversários.

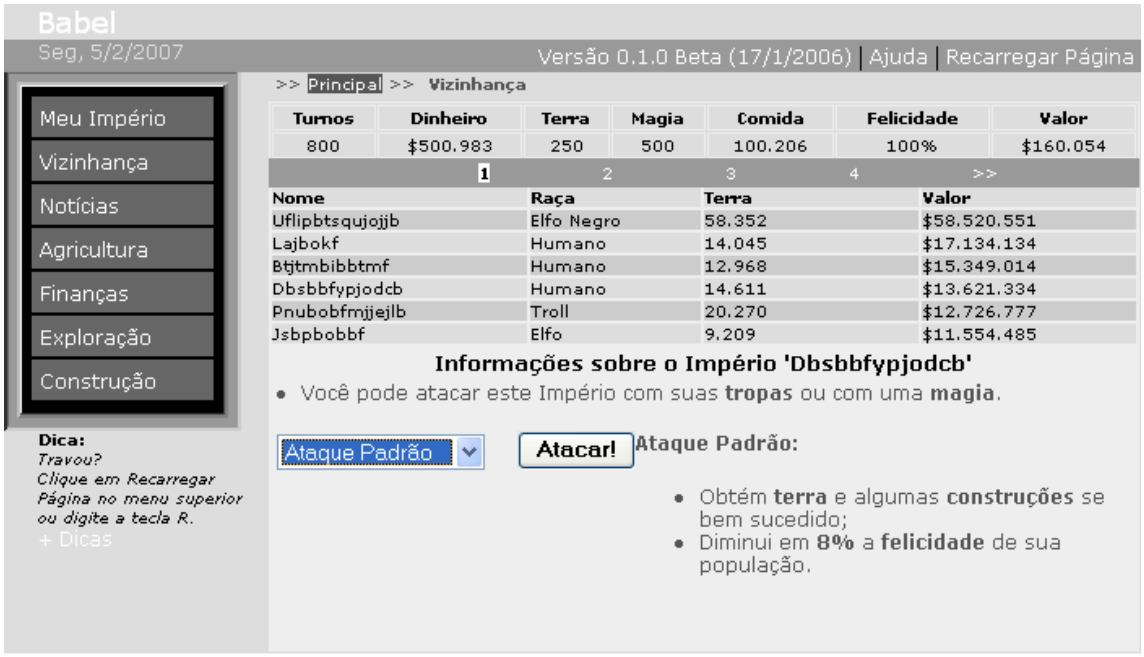

Figura 14 – Interface gráfica criada para o jogo Babel Promisance.

Toda a funcionalidade referente à formação de alianças será, portanto, extirpada, já que a comunicação entre os impérios não será mais fator chave na jogabilidade. Salienta-se, porém, que seria um trabalho futuro relevante à implementação de um jogador automático apto a negociação e participação em alianças.

Outro módulo excluído do sistema é a loteria. O jogador poderia comprar até três bilhetes e, se sorteado, ganharia o prêmio referente a uma porcentagem de todo o lucro obtido com os bilhetes. Justifica-se a retirada deste módulo por ele ser dependente apenas da sorte.

No Anexo 2 estão descritos mais detalhes sobre o funcionamento básico do ***Promisance***, descrevendo cada tela do jogo e as ações que um jogador, inclusive o jogador automático, deverão considerar.

3. 4 REPRESENTAÇÃO DO ESTADO E AÇÕES

Os planos são formados por pares estado-ação. O estado, por sua vez, é formado por uma série de dados numéricos referentes ao status do império. As variáveis de estado que serão utilizadas no processo de recuperação do caso são: dinheiro, valor, raça, terra, magia, comida e felicidade. A medida de similaridade utilizada para a recuperação dos planos foi a distância Euclidiana entre o estado do jogador automático e os estados na base de planos.

Nem todas as características são utilizadas para o processo de indexação dos casos, tendo, portanto, um peso zero durante o cálculo da similaridade. Tais características serão utilizadas, se necessário, pela Camada Reparadora e pela Camada Antecipadora no caso de falhas no plano e/ou antecipação de erros. As características usadas por essas duas camadas são: população; produção estimada; consumo estimado; porcentagem de ações ofensivas; porcentagem de ações defensivas; quantidade de áreas comerciais, residenciais, militares, industriais, escolares, agrícolas, defesas nas fronteiras, terra não utilizada, magos, soldados; catapultas, aviões, navios; despesas esperadas; renda *per capita*; pontos ofensivos e pontos defensivos. Tais características são valores inteiros contínuos armazenados na base de dados relacional. As estratégias de reparo são utilizadas tentando aumentar ou diminuir algum desses parâmetros antes da execução de um plano.

A Tabela 3 sumariza as ações disponíveis para o jogador humano e para os jogadores automáticos. Rememora-se que a descrição mais completa destas ações está disponível no Anexo 2.

Ação	Parâmetros
Gerar Dinheiro	Número de turnos gastos na especulação.
Explorar	Número de turnos gastos na exploração.
Plantar	Número de turnos gastos na agricultura.
Construir	Número de áreas comerciais, residenciais, militares, industriais, escolares, agrícolas e defesas nas fronteiras.
Gerenciar Indústria	Porcentagem de soldados, catapultas, aviões e navios.
Gerenciar Taxas	Mudança na taxa de impostos.
Atacar	Alvo a ser atacado.
Ataque Surpresa	Alvo a ser atacado.
Ataque Terrestre	Alvo a ser atacado.
Ataque Aéreo	Alvo a ser atacado.
Ataque Marítmo	Alvo a ser atacado.
Bola de Fogo	Alvo a ser atacado.
Tufão	Alvo a ser atacado.
Tempestade	Alvo a ser atacado.
Terremoto	Alvo a ser atacado.
Assalto	Alvo a ser atacado.
Login	Email e Senha.

Tabela 3- Ações que os jogadores humanos e automáticos podem executar.

3. 5 AVALIAÇÃO DO SISTEMA

Três métricas principais para verificação do sucesso do sistema foram utilizadas. A primeira, de aspecto mais subjetivo é um teste realizado periodicamente com alguns jogadores, para os quais será relacionada uma lista de impérios. Cada jogador deverá discernir, nesta lista, quais são os jogadores automáticos e quais são os jogadores humanos. Tal teste permite a descoberta e discussão se podemos atribuir aos jogadores automáticos a noção de inteligência no domínio do jogo. Tem-se, portanto, semelhança de objetivos com o Teste de Turing, o qual consiste em instruir um operador, fechado em uma sala, a descobrir quem responde suas perguntas, introduzidas através do teclado: um outro homem ou uma máquina. O projeto proposto, portanto, não objetiva a geração de um jogador automático muito melhor que o humano. O foco principal é a criação de jogadores automáticos com os quais os jogadores humanos possam se identificar.

A abordagem baseada em casos permite a imitação dos estilos dos jogadores humanos, logo, a identificação com eles. Por exemplo, em um jogo *online* de estratégia se o império é atacado, muitas vezes o jogador humano tem vontade de retribuir o ataque. Existem situações

nas quais claramente essa revanche pode ser mal sucedida. O jogador humano mais passional, dessa forma, não avalia os riscos e, mesmo que perca parte de suas tropas, deseja o revide de alguma maneira. Essa estratégia, mesmo que não tão bem sucedida, se tiver alto valor de suporte poderá entrar na base de planos. Ou seja, os jogadores automáticos poderão ter um estilo de jogo agressivo como um jogador humano. Justifica-se, portanto, a aplicabilidade do teste proposto para verificar se os jogadores automáticos estão imitando de maneira crível o estilo humano.

Entretanto, existe a possibilidade do estilo dos jogadores humanos imitados não ser divertido ou interessante para alguns jogadores. Como a diversão é um fator subjetivo, propõe-se um segundo teste associado ao Teste de Turing: além de questionar ao jogador quem é um jogador automático, também se questionará uma nota para a experiência que o jogador teve ao interagir com o império adversário.

Quanto à validação do sucesso das técnicas de planejamento e da infra-estrutura de jogadores automáticos, será necessário verificar o posicionamento e distribuição do jogador automático no *ranking* após a execução de um plano. O aprendizado de novos planos ocorre devido ao trabalho das camadas Reparadora e Antecipadora. Para mensurar a qualidade e ineditismo destes planos, será necessária a análise por um especialista humano. Não se vê outra métrica possível para a medição da criatividade do jogador automático. Outra alternativa é verificar numericamente a performance do plano gerado, ou seja, o aumento no *networth* que a execução do plano proporciona ignorando possíveis influências dos outros impérios no momento.

Para verificar se o sistema está efetivamente aprendendo, deve-se verificar a criação de estratégias de reparo bem sucedidas ou até mesmo, novos jogadores automáticos podem ser acrescentados ao sistema periodicamente e sua performance deve ser comparada com os anteriores ou com os jogadores humanos. Tal periodicidade de criação, também deve ser alvo de ajustes, mas está estimada em uma ou duas semanas.

Os jogadores automáticos novos devem ter acesso a toda a base de planos até o momento. Novos planos inseridos, contudo, não serão visualizados pelos jogadores automáticos recém-criados. Dessa forma, será possível a comparação entre jogadores automáticos criados em momentos diferentes do *round*. Se a biblioteca de planos estiver sendo útil para a aprendizagem do jogador automático, a velocidade de evolução dos novos

jogadores automáticos deve ser maior do que a dos jogadores automáticos antigos. Define-se a velocidade de evolução de um jogador automático através da quantidade de *networth* que o jogador automático cresce para uma quantidade fixa de turnos ou ações realizadas.

3. 6 CONSIDERAÇÕES FINAIS

Esse capítulo descreveu os pormenores da implementação do jogador automático construído para interagir com jogos de estratégia baseados em turnos. Descreveu-se, também, sua aplicação ao jogo Promisance. No próximo capítulo são descritos os resultados obtidos.

4 RESULTADOS

Esta seção apresenta alguns resultados obtidos através da aplicação da arquitetura proposta em um jogo *online multiplayer* real com um grupo de teste. O primeiro teste, realizado em Agosto de 2006 é descrito inicialmente. Através deste teste foi possível compreender melhor do funcionamento do jogo alvo e, além disso, foram obtidas interessantes conclusões referentes ao comportamento do grupo.

O segundo teste, realizado no final de Janeiro e início de Fevereiro de 2007, nos permitiu verificar a adequabilidade dos jogadores automáticos à resolução do problema e a credibilidade dos jogadores automáticos em relação ao comportamento humano. Por fim, um terceiro teste envolveu uma pequena quantidade de jogadores automáticos e uma quantidade maior de jogadores humanos. Neste caso, observou-se que os jogadores automáticos não se destacaram no ambiente sendo indistinguíveis dos humanos.

4. 1 RESULTADOS DO PRIMEIRO TESTE

No teste realizado, com um grupo limitado a sete jogadores humanos, focalizou-se o processo de captura dos planos. Foram obtidos 5.133 pares estado-ação em um período de jogo de três semanas. O tamanho da janela de subplanos foi o tamanho da própria seção de jogo, ou seja, todos os pares entre um *login* e um *logout* aos jogadores humanos. Com esse tamanho de janela de subplanos não houve a necessidade de uma base temporária, conforme descrito na seção 3.1.1.

A indexação do estado, objetivando a recuperação do plano, foi feita utilizando-se as seguintes características: dinheiro, *networth* (valor do império), quantidade de terra, quantidade de magia, quantidade de comida e felicidade geral do povo. Com estas características pode-se extrair 2.357 estados diferentes pelos quais os jogadores passaram. Esse número já considera a normalização dos valores das variáveis de estados aplicando um filtro nestas características através da retirada dos dois caracteres menos significativos. Quanto às ações associadas ao estado, foram capturadas dezesseis ações diferentes dentre ações de gerenciamento e tipos de ataque. Cada ação, exceto a alteração na taxa de impostos e

a alteração nas proporções de produção de cada tipo de armamento, gasta um certo número de turnos. A Tabela 4, a seguir, sumariza as ações que foram mais executadas pelo grupo.

# de execuções	Ação
938	Ataque padrão a outro império
793	Construção
729	Geração de dinheiro
474	Exploração do terreno
247	Agricultura

Tabela 4- Ações mais executadas pelo grupo.

A informação sobre os planos mais executados permite detectar a necessidade de separar em planos abstratos e planos concretos. Verificou-se que com o tamanho de janela utilizado não havia necessidade de efetuar tal separação. Os planos gerados pelos jogadores humanos são bastante diferentes entre si e nenhum plano caracterizou-se por ter o mesmo número de ações ou o mesmo estado final e estado inicial.

Com estes dados observa-se que o grupo de jogadores, embora impedido de comunicar-se diretamente, priorizou a execução de uma ação de ataque, a qual reflete a necessidade de interação entre os jogadores.

Quanto aos estados mais observáveis, encontramos o estado que se caracteriza por ter valor zero na quantidade de dinheiro. Tal estado ocorreu para mais de um jogador humano ou jogador automático e aparentou, inicialmente, ser uma falha do jogo. Posteriormente, verificou-se que este estado (e outros cujo valor de dinheiro chega a zero) seria gerado pela inexperiência dos jogadores que gastavam demais seus recursos sem se preocupar com a geração de novos recursos. Por fim, também foi possível a utilização destes recursos como estratégia de jogo, ou seja, houve casos em que **deliberadamente** o jogador não tentou eliminar a falta de dinheiro.

#	Dinheiro	Networth	Terra	Mana	Comida
55	0	12 milhões	9 mil	1,6 milhão	91 milhões
54	960 milhões	19 milhões	11 mil	7 milhões	130 milhões
38	270 mil	200 mil	330	500	47 mil
33	100 mil	150 mil	250	500	10 mil
31	580 milhões	20 milhões	1,2 mil	7,7 milhões	140 milhões

Tabela 5 - Estados mais observados e características do estado.

Este primeiro teste, foi relevante no sentido de entender melhor como um grupo de jogadores interage permitindo prever o comportamento que seria esperado pelos jogadores automáticos. As tendências do grupo podem ser utilizadas pelos jogadores automáticos tornando bastante comum a ocorrência desses estados e das ações citadas em jogadores automáticos. Além disso, esse teste permitiu a delimitação do tamanho da janela de subplanos como sendo o próprio tamanho do plano (seção completa entre um *login* e um *logout* no jogo) e a verificação de que não era necessária a separação em planos abstratos e concretos. Tal separação só seria necessária, portanto, quando existisse uma repetição de estados significativa.

É importante destacar que quanto à seleção do plano a ser escolhido, no teste realizado com o *Promisance* optou-se pela escolha do plano que maximizasse o *networth*, que é o valor do império, calculado através da composição entre o dinheiro e a quantidade de terras. Neste primeiro teste não houve a interação entre os jogadores automáticos e os jogadores humanos.

Entretanto, a título de ilustração, foi executado um jogador automático em um ambiente simulado, ou seja, sem a evolução característica de um ambiente com jogadores humanos, com o mesmo conjunto de jogadores humanos. O jogador automático utilizou os 5.133 pares estado-ação do grupo de jogadores do primeiro teste. Tal jogador foi executado durante quatro mil turnos, ou seja, uma quantidade maior do que a quantidade de turnos gasta por um jogador humano individualmente durante o teste sem os jogadores automáticos. A Figura 15 demonstra a evolução do jogador automático durante estes turnos. É interessante observar a existência de um patamar no gráfico de dinheiro (Figura 16) que explicita a utilização do estado 55, ou similar, citado na Tabela 5. Observa-se que o jogador automático segue a tendência de comportamento do Jogador 1. Isso se deve ao fato que ao chegar num estado com dinheiro nulo, o jogador automático procurou executar planos de quem: (1) fosse semelhante a ele e (2) tivesse uma performance com aumento do *networth*. É interessante destacar que o Jogador 1 usou a falta de dinheiro como estratégia e o jogador automático seguiu essa estratégia, pois é possível aumentar o *networth* mesmo sem dinheiro. Clarificando-se: se outro jogador humano também caísse nessa situação, mas fosse perdendo em *networth*, o jogador automático preferiria sempre a estratégia vencedora, seguindo o estilo de jogo do Jogador 1.

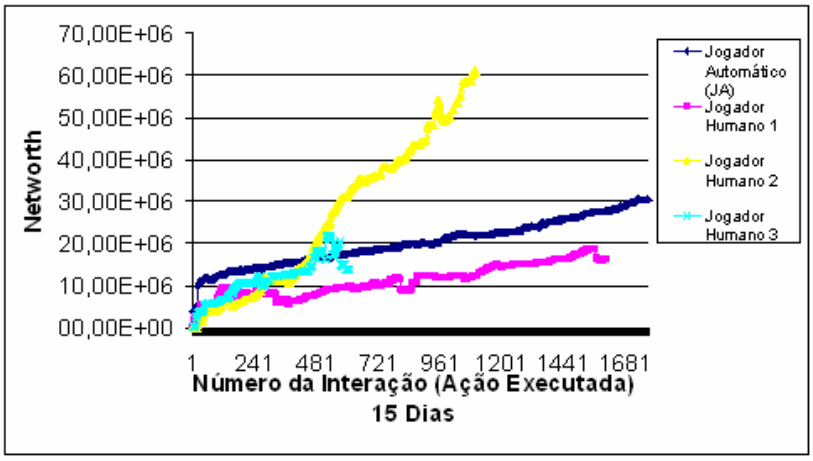

Figura 15 – Evolução do *networth* comparando o jogador automático simulado com os melhores jogadores humanos.

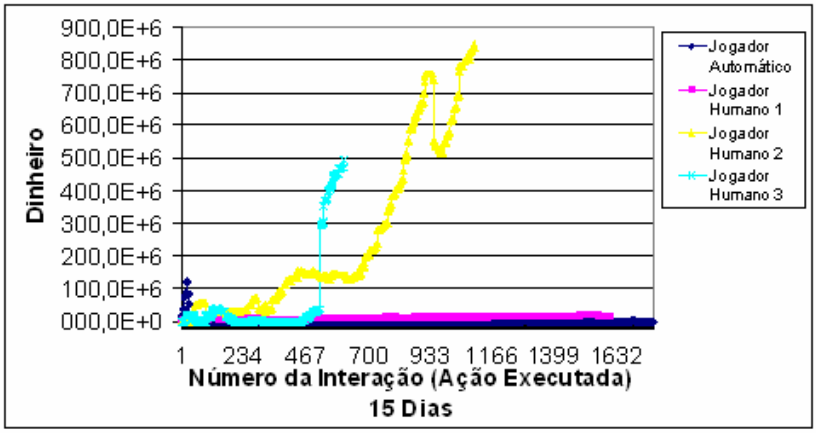

Figura 16 – Evolução do Dinheiro comparando o jogador automático simulado com os melhores jogadores humanos.

4. 2 RESULTADOS DOS TESTES DE JANEIRO DE 2007

O segundo teste realizado se limitou a um conjunto de dez jogadores humanos. Neste teste, realizado durante o período de dez dias compreendido entre 27 de janeiro a 5 de fevereiro, o perfil de usuários é semelhante ao do perfil anterior. Este segundo teste destaca-se por ter sido realizado com os jogadores automáticos inseridos no ambiente. Destaca-se que os jogadores humanos não tinham conhecimento da existência de jogadores automáticos ou da investigação científica que estava sendo realizada. O sistema contou com doze jogadores automáticos com raças diferentes, porém, executados durante os últimos quatro dias de teste. Os jogadores automáticos eram de raças distintas: quatro humanos, dois elfos, dois anões,

dois trolls, e dois elfos negros. Cada raça possui características diferentes. Essa distribuição foi escolhida para repetir a distribuição dos jogadores no ambiente do jogo.

No fim deste teste, os jogadores foram informados a cerca da existência de jogadores automáticos e questionados sobre **quem** eram os jogadores automáticos no ambiente. Para evitar possíveis deduções a partir do nome dos jogadores automáticos, os nomes foram criptografados de maneira diferente para cada jogador. Assim, mesmo conversando com outro jogador fica dificultada a descoberta de quem é um jogador automático.

Primeiramente, deve ser feita uma análise das características do ambiente e uma comparação com o ambiente anterior. Quanto ao número de pares estado-ação, obteve-se cerca de 3427 pares no total. Cada par identifica uma interação de um jogador com o sistema, ou seja, a realização de uma ação, e é usado na construção da base de casos. O número de estados únicos obtidos foi de 2135 estados diferentes. Assim como no teste anterior, todos os pares estado-ação foram transportados para a tabela de planos definitiva não apenas os planos com maior suporte. Da mesma forma, como a janela para a realização da recuperação dos casos é de tamanho igual à seção de jogo, também não existiu a indexação em estados abstratos, muito menos a base temporária de planos. Quanto à representação do estado, as variáveis utilizadas para o *matching* foram, conforme citado anteriormente: dinheiro, *networth* (valor do império), quantidade de terra, quantidade de magia, quantidade de comida e felicidade geral da população. As demais variáveis foram utilizadas no processo de antecipação de erros.

Semelhante ao teste anterior, a informação sobre os estados e ações mais recorrentes pode ajudar a caracterizar o comportamento do grupo. Quanto às ações prediletas dos jogadores, diferentemente do outro grupo, a produção de dinheiro foi à tarefa mais executada conforme a Tabela 6. O contato com os jogadores permitiu descobrir que isso se deve ao fato desta ação ser interpretada como solução para a diminuição no dinheiro do império.

# de execuções	Ação
517	Produção de dinheiro
467	Ataque padrão a outro império
407	Construção
378	Exploração do Terreno
179	Ataque terrestre com catapultas

Tabela 6- Ações mais executadas pelo grupo.

Quanto aos estados mais observáveis é interessante constatar que o estado com valor nulo não aparece entre os mais freqüentes.

# de observações	Dinheiro	Networth	Terra	Mana	Comida
116	501 mil	160 mil	300	Mil	100 mil
33	100 mil	148 mil	300	Mil	10 mil
19	6,4 milhões	6,1 milhões	8,3 mil	47 mil	19,4 milhões
18	346 mil	6,2 milhões	8,3 mil	16 mil	17,5 milhões
15	3,5 milhões	7,6 milhões	5,5 mil	1,4 milhões	70 milhões

Tabela 7 - Estados mais observados e características do estado.

Das métricas citadas no capítulo anterior (seção 3. 5), primeiro trata-se da evolução dos jogadores automáticos e sua velocidade de evolução. Dois parâmetros refletem diretamente no desempenho do jogador automático: o valor do império (*networth*) e a quantidade de terras. Também se analisa a evolução do dinheiro do Império, pois é a pré-condição mais importante para a execução das ações ser bem sucedida.

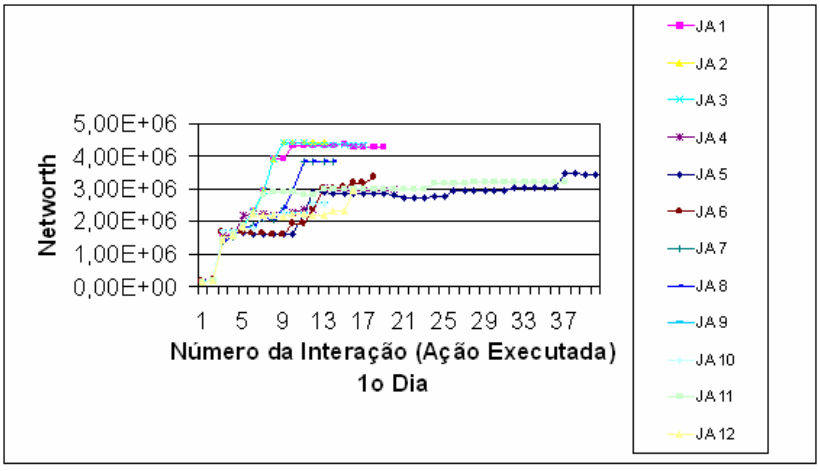

Figura 17 – Evolução do *networth* no primeiro dia para os jogadores automáticos.

Quanto à execução dos jogadores automáticos, a bases de casos de todos é a mesma. Mesmo o jogo sendo determinístico a evolução dos jogadores automáticos não é idêntica pois: a raça dos jogadores automáticos difere e a interação com os outros jogadores automáticos implica em mudanças no estado do jogador. Foram criados um número próximo de jogadores automáticos em relação aos jogadores humanos: 12 automáticos para 10 humanos.

No primeiro dia de execução dos jogadores automáticos, as primeiras ações executadas levaram inicialmente a um estado similar até meados da quinta ação. A partir deste ponto, os jogadores automáticos, por serem pertencentes a raças diferentes, responderam às ações de maneira diferente. Percebe-se, portanto, na Figura 17, o agrupamento de três jogadores automáticos da raça humana (jogador automático 1 ao 3). Quanto ao dinheiro e terra (Figura 18 e 19) fica dentro do previsto a distribuição, principalmente para os jogadores automáticos 5 e 11. Os gráficos de ambas as figuras são semelhantes pois pode existir uma relação proporcional entre Terra e Dinheiro. No caso dos Elfos (Jogador automático 5), há uma bonificação na exploração e na renda per capita, impulsionando a quantidade de terras obtida.

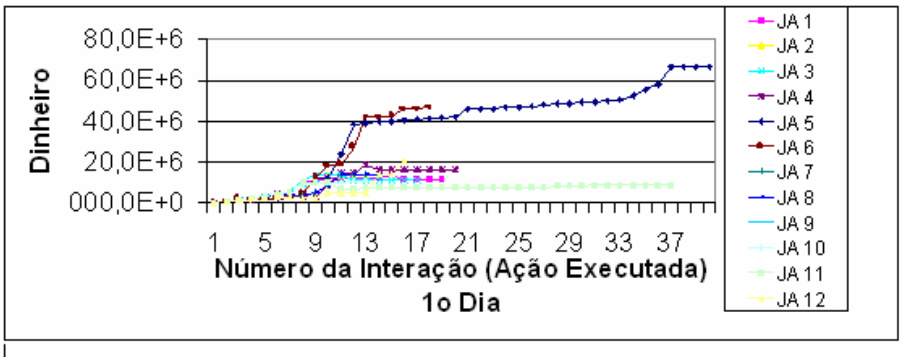

Figura 18 – Evolução do Dinheiro no primeiro dia.

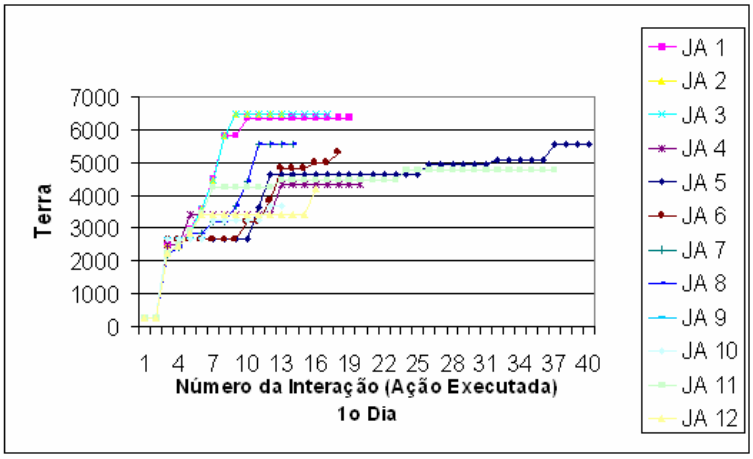

Figura 19 – Evolução das terras no primeiro dia.

Sob o prisma de quatro rodadas completas (correspondendo a quatro dias), pode-se fazer uma comparação entre resultado da execução dos jogadores automáticos e o resultado das interações dos jogadores humanos. Nas Figuras 20 e 21 observa-se um agrupamento mais coeso no caso dos jogadores automáticos e uma tendência mais uniforme de crescimento. Quanto ao desempenho, o melhor jogador automático, em quatro dias, alcançou um *networth* de quase 14 milhões, enquanto dois jogadores humanos ultrapassaram a barreira dos 15 milhões. É interessante, portanto, destacar, que o comportamento dos jogadores automáticos assumiu um valor médio em relação ao grupo. Nem se destacou demais, nem se encontrou entre os piores. O povoamento do ranking, inclusive, projetou os jogadores automáticos às posições medianas do ranking, nunca chegando a ameaçar a liderança.

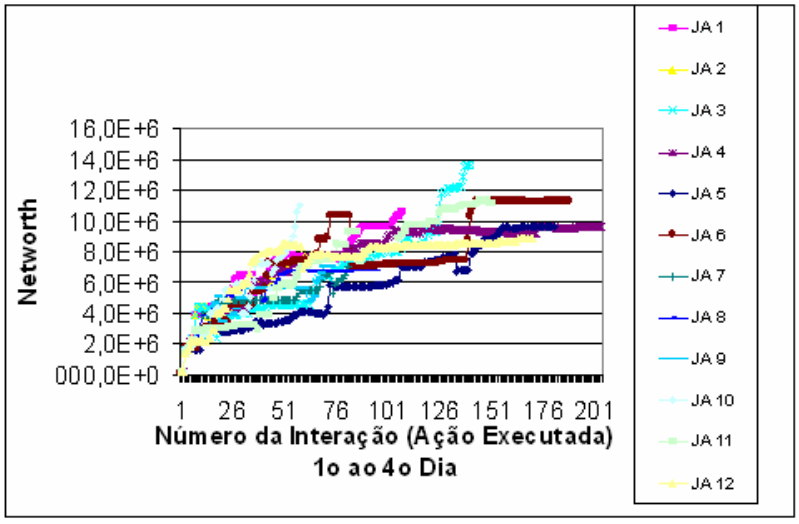

Figura 20 – Evolução do *networth* do primeiro ao quarto dia para os jogadores automáticos.

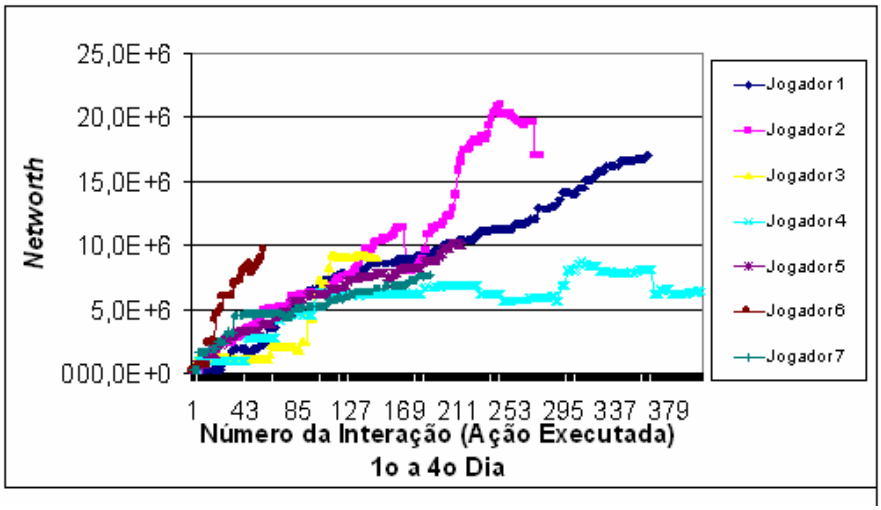

Figura 21 – Evolução do *networth* do primeiro ao quarto dia para os jogadores humanos.

Também foi implementada uma simulação quanto à evolução do jogador automático além do tempo previsto, ou seja, sem o suporte de uma base de casos com quantidade de planos coerente com a quantidade de iterações executadas. Nesse caso, ilustrado nas Figuras 22 e 23, é interessante constatar que o *networth* do jogador automático cresceu rapidamente, por sempre procurar a melhor estratégia, mas ficou estagnado em um patamar fixo, que, embora fosse ainda com um *networth* maior que a maioria dos outros jogadores automáticos, ainda assim, parece esperar pela chegada dos outros, ou seja, permanece estável em um patamar constante pois não tinha mais como evoluir devido a inexistência de planos de jogadores humanos que tivessem passado por aquela situação.

Esse comportamento é análogo ao comportamento típico em jogos de carro para Fliperama, nos quais o jogador automático vai mais devagar de maneira imperceptível para dar chance ao jogador humano de alcançá-lo. É, portanto, uma técnica de "estupidez artificial" [LID2004] que surgiu de maneira emergente. Provavelmente, conforme os outros jogadores humanos fossem alcançando o estado do jogador automático existiriam estados os quais o jogador automático copiaria para melhorar seu desempenho.

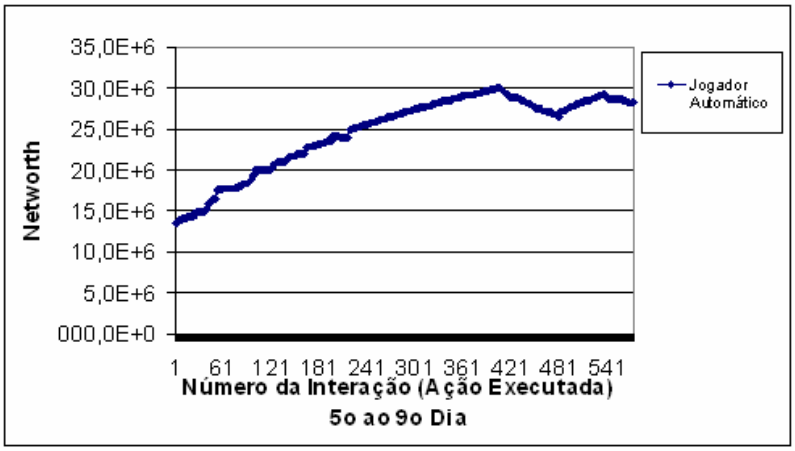

Figura 22 – Evolução do *networth* para um jogador automático simulado.

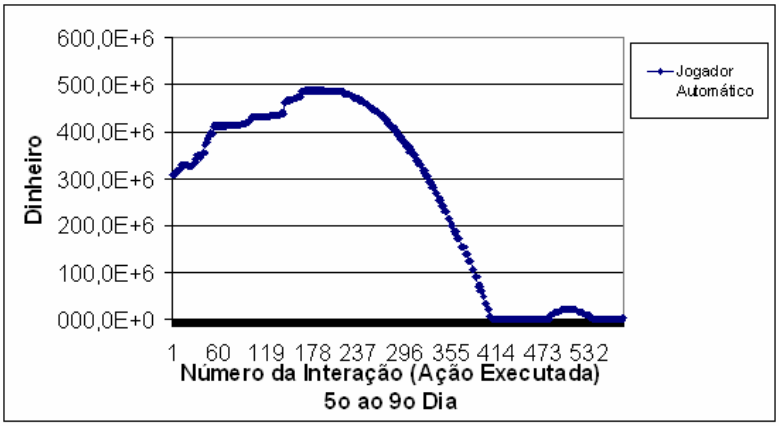

Figura 23 – Evolução do dinheiro para um jogador automático simulado.

Ainda dentro do escopo de comportamentos emergentes ou inesperados ao jogador automático e em relação aos planos de reparo gerados pelos jogadores automáticos também há situações nas quais o jogador automático implementa ações que parecem ser falhas, mas, na verdade, deixam o jogador automático mais interessante. Por exemplo, um erro ocorreu quando o jogador automático tentou executar uma ação de ataque em relação a um império mais fraco. A Camada Reparadora anotou as diferenças em relação ao império que havia atacado o império de maneira bem sucedida. Dentre essas diferenças, destaca-se uma estratégia que consiste em diminuir o número de magos. Tal estratégia enfraquece as forças do império do jogador automático, mas, pelo menos, permite que ele esteja apto a atacar um império não muito forte.

Outro exemplo de estratégia emergente que foi descoberta pelo jogador automático foi relacionado a pré-requisitos para a realização de magias. A realização de uma magia implica na necessidade de uma quantidade mínima de poder mágico. Era esperada a utilização de uma estratégia de reparo que aumentasse a quantidade de poder mágico, por exemplo, a construção de escolas de magia. Entretanto, as ações que foram escolhidas pelo jogador automático eram ações que sempre aumentavam o lucro esperado pelo império (diferença entre o que é gasto e o que é arrecadado), como, por exemplo, a construção de áreas comerciais. Percebeu-se que o aumento do lucro, proporcionava um aumento na população e no número de magos o que, conseqüentemente, gerava um aumento no poder mágico do império.

Os resultados descritos limitam-se a apresentar uma análise quantitativa da evolução do jogador automático. É essencial, também, apresentar uma análise qualitativa através do questionamento aos jogadores sobre quem são os jogadores automáticos. De maneira similar ao Teste de Turing, os jogadores humanos foram estimulados a responderem quais impérios são jogadores automáticos no fim do período de testes. Dos dez jogadores ativos, quatro responderam à pesquisa.

O primeiro jogador constatou a existência de seis jogadores automáticos. Destes, foram classificados corretamente apenas dois jogadores automáticos. Verificando o *ranking*, esse jogador supôs que os piores jogadores seriam os jogadores automáticos.

O segundo jogador, após tentar atacar diversos impérios no primeiro dia fechou-se em um regime pacifista, ou seja, não atacou nenhum outro império e investiu em defesas. Dessa forma, não soube responder quais seriam os jogadores automáticos por não haver maneira de diferenciá-los. O perfil deste jogador também é característico em muitos jogos *online*: algumas pessoas preferem não interagir com o seu grupo de jogo. Mesmo assim, não se pode retirar para estes jogadores a importância de estarem inseridos em um ambiente bem povoado: o jogador pode usar o *ranking* para poder comparar sua performance com a de outros jogadores e sentir-se estimulado e desafiado a continuar a jogar e vencer seus adversários. E, o *ranking* manteve-se coerente, doutra forma o jogador poderia perceber mudanças bruscas e irreais caso o jogador automático não fosse crível.

O terceiro jogador supôs que três outros jogadores humanos fossem jogadores automáticos. Sua justificativa para a escolha de cada jogador foram os ataques mal sucedidos que recebeu e com os horários de jogo singulares de um dos jogadores automáticos. Um ponto

pode ser retirado desta explicação: primeiramente, a *IA*, nos jogos tradicionais, ainda é associada a um comportamento diferente ao de um grupo de jogadores. Nos testes implementados, como os jogadores automáticos imitaram o comportamento médio do grupo, as falhas também ficaram na média do esperado. Ou seja, o número de sucessos e de falhas nas interações entre jogadores automáticos e humanos deve ser compatível.

O quarto jogador escolheu dezesseis candidatos a jogador automático na lista de vinte e duas opções (doze automáticos e dez humanos) disponibilizada. Destes dezesseis, onze eram realmente os jogadores automáticos. Assim, o jogador também não conseguiu distinguir corretamente humanos e jogadores automáticos, pois acabou escolhendo a maioria dos jogadores presentes no sistema, deixando de fora apenas os mais bem sucedidos no *ranking*. Novamente, o jogador humano tem a expectativa que os jogadores automáticos tenham sempre uma performance pior que a dos jogadores humanos.

4. 3 RESULTADO DOS TESTES DE FEVEREIRO DE 2007

No último conjunto de testes realizado seis jogadores automáticos interagiram com vinte e cinco jogadores humanos, gerando um conjunto de 11.000 pares-estado ação capturados em um total de 700 planos. Além da já esperada evolução semelhante dos jogadores autoimáticos, conforme se pode visualizar na Figura 24, também foi feito teste no sentido de comparar a performance de um jogador automático usando antecipação de erros e um jogador automático que não a utilizou. Neste teste, mostrado na Figura 25, observou-se que o jogador automático utilizando estratégias de correção faz menos iterações e obtém um resultado semelhante de *networth*.

Quanto ao Teste de Turing realizado, oito jogadores humanos responderam à pesquisa, indicando cinco possíveis jogadores automáticos. Novamente, foi feita a suposição que os piores e os melhores jogadores no *ranking* eram automáticos e em média a taxa de acerto foi de 25%.

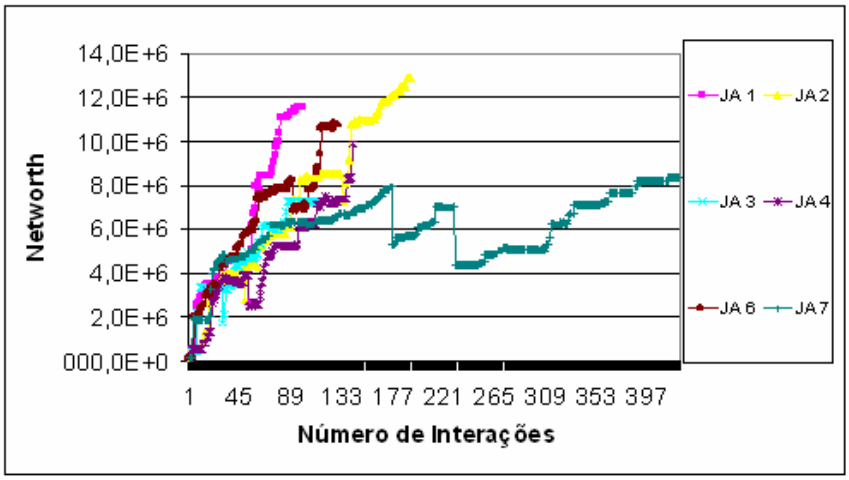

Figura 24 – Evolução do networth dos jogadores automáticos no terceiro round.

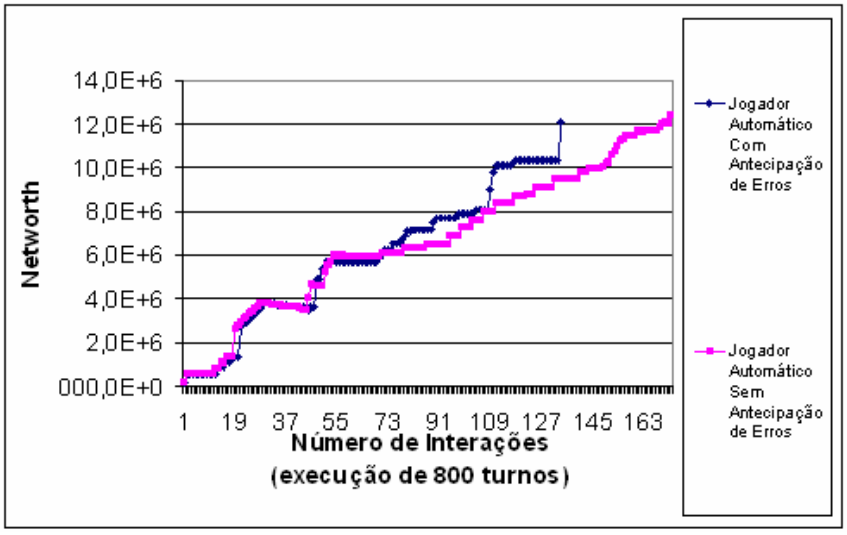

Figura 25 – Comparação entre jogador com e sem antecipação de erros.

4. 4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os resultados obtidos. Destaca-se a adaptação do jogador automático frente ao ambiente, comportando-se conforme o perfil dos jogadores. Tal adaptação implementa um balanceamento emergente dentro do jogo. Ou seja, a dificuldade se adaptou ao grupo de maneira não esperada e não previsível. Observa-se, também, o espalhamento uniforme dos jogadores automáticos em relação ao ranking. Rankings das rodadas 2 e 3 estão disponíveis em anexo.

Também é encorajador o resultado obtido no “Teste de Turing”. Não foi possível aos jogadores, dentro sistema, distinguir quem era um jogador automático e quem era um jogador humano. Na próxima seção, faz-se a revisão do exposto e o fechamento com possíveis propostas de continuidade deste trabalho.

5 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho apresentou uma série de contribuições relevantes não apenas para a área de *IA* para Jogos de Entretenimento. Em primeiro lugar, discutiu-se e aplicou-se com sucesso técnicas de *PBC* em um jogo de estratégia baseado em *web*. Assim sendo, tão importante quanto o teste da adequabilidade da técnica em um jogo, também se pode afirmar que um sistema de *PBC* teve sucesso em um ambiente multiusuário. Dessa forma, outros tipos de *sites* poderiam ser beneficiados com a aplicação de *PBC* no modelo de tratamento e representação dos casos trabalhado, o qual revela uma preocupação em seguir o modelo de comportamento do usuário. Por exemplo, pode-se estender o modelo para permitir o povoamento de mundos virtuais como *sites* de relacionamento e *ARGs*, por exemplo.

Outra contribuição foi a implementação do preenchimento da base de estratégias de reparo por parte da Camada Reparadora. Além de permitir por parte do jogador automático jogador a possibilidade de antecipar erros e evitá-los através da chamada de planos que reparem o erro, o conhecimento armazenado por essa camada também permitiu descobrir relações implícitas entre diferentes parâmetros. Tais relações já podem servir de auxílio para um especialista gerar uma rede de dependências semelhante à da Figura 6, explicitando o conhecimento que está oculto. O jogador automático pode, portanto, ter um papel de facilitador na descoberta de relações dentro do sistema.

Além do especialista, mesmo o jogador comum poderia se beneficiar desse conhecimento. Podem-se vislumbrar três abordagens possíveis: i) um agente assistente pode servir de conselheiro e sugerir ao jogador qual a melhor ação a ser tomada conforme o estado do império, ii) utilizando a informação de erros, o assistente poderia avisar, antes do jogador realizar uma ação, que possivelmente ela seria mal sucedida e sugerir uma estratégia de reparo, iii) por fim, o assistente poderia apenas responder ao jogador como fazer para aumentar ou diminuir um certo parâmetro de seu império a partir das estratégias de reparo bem sucedidas. Esse agente conselheiro fica como uma possível diretiva futura. Entretanto, a utilização de um assistente neste estilo pode abrir o questionamento por parte dos outros jogadores que o uso de um assistente é uma forma de trapaça.

Quanto ao comportamento apresentado pelos jogadores automáticos obtiveram-se resultados encorajadores. O jogador automático segue a tendência do grupo gerador da base de casos, ou seja, apresenta um crescimento com velocidade coerente com a velocidade do grupo. Dessa maneira, cumpre-se a expectativa do jogador em interagir com um adversário semelhante a ele mesmo. É interessante destacar o teste realizado em um ambiente simulado no qual o jogador automático cresce até um patamar e, posteriormente tem seu desempenho piorado devido a inexistência de planos que possuam estados semelhantes ao estado atual do jogador automático.

Quanto ao "Teste de Turing", ou seja, o questionamento subjetivo aos jogadores humanos sobre quem seriam os jogadores automáticos, observou-se que não foi possível ser feita essa diferenciação. O jogador automático desenvolvido, portanto, foi bem sucedido em apresentar um comportamento crível em relação ao esperado para um jogador humano, ficando indiscernível dentro do universo dos jogadores.

Para estimular a interação entre os jogadores automáticos e humanos, como trabalho futuro, seria possível e desejável a implementação de um jogador automático com habilidade em negociação e participação de alianças de jogadores automáticos com seres humanos. Tal jogador automático, destaca-se, não necessariamente envolve técnicas de processamento de linguagem natural. Poder-se-ia, por exemplo, criar uma interface de comunicação neutra responsável por limitar a interação entre os jogadores humanos e automáticos.

A arquitetura desenvolvida é genérica o suficiente permitindo sua aplicação em outro jogo semelhante apenas com redefinições das ações e variáveis de estado, a captura dos planos e um serviço que faça uma interface de execução com o jogo. Alternativas de jogos *online* para futuros testes não faltam, conforme verifica-se no Anexo 3.

É importante destacar a importância do presente trabalho principalmente para os jogos com menos jogadores. Embora os grandes jogos *online* sejam responsáveis pela maior parte dos jogadores, cada vez mais é comum a criação de nichos de jogadores com interesses por jogos diferentes. Seguindo a tendência atual na *web* [AND2006], a contribuição econômica desses nichos poderá ser relevante no mercado de jogos. Deve-se salientar, porém, que o requisito para um jogo ser bem sucedido na aplicação do Planejamento é a pouca integração entre subplanos. Jogos que envolvam *puzzles*, por exemplo, não são uma boa escolha.

Um outro trabalho futuro de possível valor seria a comparação da técnica baseada em *RBC* com alguma abordagem hierárquica semelhante à aplicada no *Bridge Baron* [SMI1998]. O ***Promisance***, e os jogos de sua família permitem a especificação de estratégias. Tais estratégias poderiam ser utilizadas para alimentar um planejador hierárquico.

Por fim, também parece interessante a utilização de reconhecimento de planos não apenas para copiar os estilos do jogador, mas também para manter um modelo mental dos adversários. Essa implementação envolveria apenas a modificação da representação do estado, armazenando também informações sobre o adversário. Embora o jogador automático não possa ter acesso a todas as variáveis referentes ao jogador, ele pode manter uma representação simplificada do usuário. Tal representação pode auxiliar na tomada das decisões de ataque e defesa.

LUDOGRAFIA

ADVENTURE. Will Crowther. 1974.

AIRFIGHT. Kevin Gorey. 1974.

AIR WARRIOR. GEnie. Kesmai.1986.

AVATAR. 1978.

BARREN REALMS ELITE. 1990.

BLACK & WHITE. Eletronic Arts Games. Lionhead Studios. 2001.

CITY OF HEROES. NCSoft. Cryptic Studios. 2004.

CLUB CARIBE. QuantumLink. LucasFilm. 1988.

CREATURES. Millenium Interactive. Mindscape Group. 1996.

DECWAR.

DIPLOMACY.

DND. Daniel Lawrence; Gary Whisenhunt; Ray Wood . 1974.

DOOM. Activision. iD Software. 1993.

DOGFIGHT. 1973.

DRAGON'S GATE. 1991.

EVE ONLINE. CCP Games. CCP Games. 2003.

FALCON'S EYE.

EMPIRE. John Daleske; Silas Warner. 1974.

EVERQUEST. Sony Online Entertainment. Sony Online Entertainment. 1999.

EVERQUEST II. Sony Online Entertainment. Sony Online Entertainment. 2004.

FULL SPECTRUM WARRIOR. THC Inc. Pandemic Studios. 2004.

GEMSTONE II. 1988.

HABITAT. QuantumLink. LucasFilm. 1985.

HUNT THE WUMPUS. Gregory Yob. 1972.

ISLANDS OF KESMAI. Compuserve. Kesmai. 1984.

LEGEND OF RED DRAGON. Seth Able. 1989.

LINEAGE. NCSoft. NCSoft. 2003.

MATRIX ONLINE. Sega. Monolith Productions. 2005.

MAZE WAR. Steve Colley. 1973.

MEGA WARS I.

MEGA WARS III.

MERIDIAN 59. 3DO Company. 3DO Studios. 1996.

MINES OF MORIA. Chuck Miller. 1974.

MUD. Richard Bartle, Roy Trubshaw. 1978.

NEVERWINTER NIGHTS. QuantumLink. Stormfront Studios. 1991.

OUBLIETTE. Jim Schwaiger. 1977.

PAC-MAN. Midway. Namco. 1979.

QUAKE. Activision. Id Software. 1996.

RABBIT JACK'S CASINO. QuantumLink. Rob Fulop. 1986.

RAGNAROK ONLINE. Level Up Games. Gravity Corp. 2002.

RIM WORLDS WAR. GEnie. Jessica Mulligan. 1986.

SIM CITY. Eletronic Arts. Maxis. 1989.

SOLAR REALMS ELITE. Amit Patel. 1990.

SPACE INVADERS. Midway. Taito. 1978.

SPACEWAR!. Steve Russell; Dan Edwards; Alan Kotok; Peter Sampson; Martin Graetz. 1962.

SPASIM. Jim Bowery. 1974.

STAR TRADER. Dave Kaufman. 1974.

STAR TREK. Mike Mayfield. 1972.

STAR WARS GALAXIES. LucasArts. Sony Online Entertainment. 2003.

STELLAR WARRIOR. GEnie. Kesmai. 1985.

TENNIS FOR TWO. Willy Higinbothan. 1958.

TRADE WARS. Chris Sherrick. 1984.

THE SIMS. Eletronic Arts Games. Maxis. 2000.

THE SIMS ONLINE. Eletronic Arts Games. Maxis. 2002.

Unreal Tournment

UNREAL TOURNMENT. GT Interactive. Epic Games. 1999.

ULTIMA ONLINE. Eletronic Arts. Origin Systems. 1997.

USURPER. 1993.

UTOPIA. Swirve. Disponível em: http://games.swirve.com/utopia/

WARCRAFT. Blizzard Entertainment. 1994.

WARCRAFT II. Blizzard Entertainment. 1995.

WORLD OF WARCRAFT. Vivendi Universal. Blizzard Entertainment. 2004.

ZORK. Tim Anderson, Marc Blank, Bruce Daniels, Dave Lebling. 1977.

REFERÊNCIAS BIBLIOGRÁFICAS

[AA1994] AAMODT, A.; PLAZA, E. *Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches.* AICom - Artificial Intelligence Communications, IOS Press,Vol. 7, N.º 1, 1994, p. 39-59

[AND2006] ANDERSON, C. *A Cauda Longa: Do mercado de massa para o mercado de nicho*. Rio de Janeiro, Elsevier, 2006.

[AHA2005] AHA, D.W., MOLINEAUX, M.; PONSEN, M. *Learning to win: Case-based plan selection in a real-time strategy game*. In: ICCBR - Proceedings of the Sixth International Conference on Case-Based Reasoning, 2005, p. 5-20, Chicago, IL: Springer.

[AHL1977] AHL, D.H. *What to do after you hit RETURN: A computer games book from People's Computer Company.* Hayden Book Company. 1977.

[AVI2004] AVILA, H. M.; FISHER, T. *Strategic Planning for Unreal Tournment Bots*. Workshop on Challenges in Game AI, 2004. disponível em: http://www.cse.lehigh.edu/%7Emunoz/Publications/AAAIWS04Munozh.PDF. Acesso em: 26/03/2005.

[BAI1986] BAIN, W. M. *Case-Based Reasoning: A Computer Model of Subjective Assessment*. Ph.D. Thesis, Yale University, New Haven, CT, 1986.

[BER1998] BERGMANN, R.; AVILA, H. M.; VELOSO, M. *et al. Case-Based Reasoning applied to Planning Tasks*. In: LENZ, M., BARTSCH-SPOERL, B., Burkhard, H. D.; WESS, S. Case-Based Reasonig Technology from Foundations to Applications, Berlim, Springer, 1998. p. 166-199.

[BIG1998] BIGUS, J. P.; BIGUS, J**.** *Constructing intelligent agents with Java: a programmer's guide to smarter applications*. New York: Wiley, 1998.

[BOG2004] BOGOST, IAN. *Asynchronous Multiplay: Futures for Casual Multiplayer Experience*. Paper presented at the Other Players Conference. Center for Computer Games Research, IT University of Copenhagen, Denmark, 2004. Disponível em: http://itu.dk/op/proceedings.htm. Acesso em: 15/5/2005.

[BOW2001] BOWERY, J. *Spasim (1974): The First First-Person-Shooter 3D Multiplayer Networked Game*. Disponível em: http://www.geocities.com/jim_bowery/spasim.html . Acesso em: 16/4/2005.

[BRO1991] BROOKS, R.A. *How to build complete creatures rather than isolated cognitive simulators.* In: VANLEHN, K. Architectures for Intelligence: The Twenty-second Carnegie Mellon Symposium on Cognition, New Jersey, Hillsdale, Lawrence Erlbaum Assosiates, 1991, p. 225-239.

[BURO2003] BURO, M.; FURTACK, T. *RTS Games as a Test-Bed for Real-Time AI Research*. In: CHEN, K. *et al*. Proceedings of the 7th Joint Conference on Information Science, Alberta, Edmonton, 2003, p. 481–484.

[BUR1995] BURKA, L. P. *The MUDLine*. Disponível em: http://www.linnaean.org/~lpb/muddex/mudline.html. Acesso em: 16/3/2005.

[CHE2004] CHENG, D. C.; THAWONMAS, R. *Case-Based Plan Recognition For Real Time Strategy Games*. In: Proceedings of the Fifth Game-On International Conference, Reading, UK, 2004, p. 36-40.

[COS2000] COSTIKYAN, G. *The Future of Online Games*. In: Creative Good. 2000. Disponível em: http://www.costik.com/ogrfinal.zip . Acesso em: 12/1/2005.

[COX2002] COX, M. T.; MUÑOZ-AVILA, H.; BERGMANN, R. *Planning in Case-Based Reasoning: Commentary*. 2002.

[EHR2002] EHRENREICH, R. Legend of the Red Dragon Case History. School Report, 2002. Disponível em: http://www.bbsdocumentary.com/library/PROGRAMS/DOORS/LORD/ehrenreich.pdf. Acesso em: 9/1/2005.

[EVA2001] EVANS, R. *The Future of AI in Games: A personal view*. Game Developer Magazine, Agosto de 2001. Disponível em: http://www.gameai.com/blackandwhite.html. Acesso em: 15/3/2005.

[FAG2003] FAGAN, M.; CUNNINGHAM, P. *Case-Based Plan Recognition in Computer Games*. In: ICCBR, 2003. Disponível em: http://www.cs.tcd.ie/publications/tech-reports/reports.03/TCD-CS-2003-01.pdf . Acesso em: 14/3/2005.

[FAS1996] FASCIANO, M. J. *Real Time Case Based Reasoning in a Complex World*. Technical Report TR-9605, University of Chicago. 1996. Disponível em: http://www.cs.uchicago.edu/files/tr_authentic/TR-96-05.ps. Acesso em: 15/3/2005.

[FIK1971] FIKES, R. E., NILSSON, N. J. *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving*. Artificial Intelligence, 2(3-4), 1971.

[FRE2002] FREEBURG, N. *A Heuristic Search Algorithm for Empire-Based Games.* CS 470 Project, 2002.

[FÜR2000] FÜRNKRANZ, J. *Machine Learning in Games: a Survey*. Nova Science Publishers, 2000, p. 11-59.

[GEF2002] GEFFNER, H. *Perspectives on Artificial Intelligence Planning*. In: Eighteenth National Conference on Artificial Intelligence, 2002, p. 1013-1023. Disponível em: http://www.tecn.upf.es/%7Ehgeffner/html/reports/perspectives.ps. Acesso em: 15/6/2006.

[GHA2004] GHALLAB, M. NAU, D. TRAVERSO, P. *Automated Planning: Theory and Practice*. Morgan Kauffmann Publishers, 2004.

[GOO1989] GOODMAN, M. *CBR in Battle Planning*. In: Proceedings of the Second Workshop on Case-Based Reasoning, Pensacola Beach, FL, US. 1989. p. 264-269.

[GOL2001] GOLDBERG, M. *The History of Computer Gaming*, 2001. Disponível em: http://www.classicgaming.com/features/articles/computergaminghistory/. Acesso em: 16/3/2005.

[HAM1989] HAMMOND, K. J. *Case-Based Planning: Viewing Planning as a Memory Task.* San Diego: Academic Press, 1989.

[HAM1994] HAMMOND, K. J. *Case-Based Planning: A Framework for Planning from Experience*, In: Cognitive Science. Vol. 14. 1994. p. 385-443.

[HAR1968] HART, P. E.; NILSSON, N. J.; RAPHAEL, B. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*. IEEE Transactions on Systems Science and Cybernetics SSC4. 1968, Vol 2., p 100-107.

[HUN2005] HUNTER, W. *The Dot Eaters - Player4 Stage1 - Classic Video Game History*. 2005. Disponível em: http://www.emuunlim.com/doteaters/ . Acesso em: 11/1/2005.

[ISL2002] ISLA , D.; BLUMBERG, B. *New Challenges for Character-Based AI for Games*. AAAI Spring Symposium on AI and Interactive Entertainment, Palo Alto, CA, 2002, p. 41-45.

[KEN2003] KENT, S. L. *Alternate Reality: The History of Massively Multiplayer Games*. 2003. Disponível em: http://archive.gamespy.com/amdmmog/week1/index.shtml. Acesso em: 15/3/2005.

[KER2003] KERKEZ, B. *Incremental Case-Based Plan Recognition With Incomplete Plan Libraries.* 2003. Disponível em:
http://personal.ashland.edu/~bkerkez/Research/public/BKerkez_PhD_Dissertation_WSU_2003.pdf
Acesso em: 16/3/2005.

[KLA2005] KLASTRUP, L. *Towards a Poetics of Virtual Worlds - Ph.D.thesis by Lisbeth Klastrup*. 2005. Disponível em: http://www.klastrup.dk/thesis.htm. Acesso em: 15/3/2005.

[KOL1993] KOLODNER, J. Case-Based Reasoning. São Mateo: Kaufmann Publishers, Inc., 1993.

[KOS2002] KOSTER, R. *Raph's Page: Online World Timeline*. 2002. Disponível em: http://www.legendmud.org/raph/gaming/mudtimeline.html. Acesso em: 9/1/2005.

[LAIRD2000] LAIRD, J. E.; VAN LENT, M. *Human-Level AI's Killer Application: Computer Game AI*. Proceedings of AAAI 2000, Austin, TX, Agosto de 2000. p. 1171-1178.

[LEB1979] LEBLING, P. D.; BLANK, M. S.; ANDERSON, T. A. *Zork: A Computarized Fantasy Simulation Game*. IEEE Computer. 12(4). 1979. p. 51-59.

[LID2004] LIDEN, L. *Artificial Stupidity: The Art of Intentional Mistakes.* In: *AI Game Programming Wisdom II*. Charles River Media. Rockland. 2004.

[MCD1998] MCDERMOTT, D., *et al*. *PDDL – The Planning Domain Definition Language*. 1998.

[MAR2001] MARKOWITZ, M. *Space Games of Fame - About the Games*. 2001. Disponível em: http://www3.sympatico.ca/maury/games/space. Acesso em: 12/1/2005.

[MAR2006] MARTIN, A. *Alternate Reality Games White Paper : The IGDA Alternate Reality Games SIG*. 2006. Disponível em: http://igda.org/arg/resources/IGDA-AlternateRealityGames-Whitepaper-2006.pdf. Acesso em: 15/1/2007.

[MIL2005] MILLER, A. *AI Roundup*. 2005. Disponível em: http://www.gamedev.net/columns/events/coverage/feature.asp?feature_id=77. Acesso em: 22/3/2005.

[MOC2002] MOCK, K. *Hierarchical Heuristic Search Techniques for Empire-Based Games*. In: International Conference on Artificial Intelligence, Las Vegas, NV, 2002.

[MOR1990] MORNINGSTAR, C.; FARMER, F. R. *The lessons of Lucasfilm's habitat*. In BENEDIKT, M. Cyberspace: First steps Cambridge, MA: MIT Press, 1990, p. 273-302.

[MUL1999] MULLIGAN, J. *GateCentral: Ultima: Timeline1*. 1999. Disponível em: http://www.gatecentral.com/shared_docs/Timeline1.html. Acesso em: 11/1/2005.

[NEW1972] NEWELL, A.; SIMON H. A. *Human Problem Solving*. Englewood Cliffs. Prentice Hall, 1972.

[ORK2004] ORKIN, J. *Symbolic Representation of Game World State: Toward Real-Time Planning in Games*. AAAI Fall Symposium, Perth, Australia, 2004, p. 41-46.

[PAT2005] PATEL, A. *Solar Realms Elite 0.995: Documentation*. Disponível em: http://www-cs-students.stanford.edu/~amitp/Articles/SRE-Documentation.html. Acesso em: 19/3/2005.

[PRI2003] PRITCHETT, J. *History of Trade Wars Variations*. Disponível em: http://www.eisonline.com/twhistory/. Acesso em: 9/2/2005.

[QUI1986] QUINLAN, J. *Induction of decision trees*. Machine Learning, 1986, Vol. 1 p. 81–106.

[RAO1995] RAO A. S.; GEORGEFF, M. *BDI Agents: from theory to practice*. In: Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA, 1995, p. 312–319.

[RABIN2004] RABIN, S. *AI Game Programming Wisdom II*. Charles River Media. Rockland, 2004.

[RIE1983] RIESBECK, C. K. *Knowledge Reorganization and Reasoning Style*. Technical Report. YALEU/DCS/RR 270. Yale University. 1983.

[RIE1989] RIESBECK, C. K.; SCHANK, R. C. *Inside Case-Based Reasoning*. Lawrence Erlbaum, 1989.

[RUS1995] RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach.* Prentice Hall. 1995.

[SAM1959] A. L. Samuel. *Some Studies in Machine Learning using the Game of Checkers*. IBM Journal of Research and Development, 1959, 3(3), p. 210-229.

[SCH1982] SCHANK, R. C. *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press, 1982.

[SCH2001] SCHAEFFER, J. *A Gamut of Games*. AI Magazine, 2001, p. 30-46.

[SHA1950] SHANNON, C. E., *Programming a computer for playing chess*, Philosophical Magazine Vol. 41, 1950, 256-275

[SHA2002] SHAPIRO, A.; FUCHS, G.; LEVINSON, R. *Learning Game Strategy Using Pattern-Weights and Self-Play*. In: International Conference on Computers and Games, Edmond, Canada, 2002, p. 42-60.

[SMI1998] SMITH, S. J. J.; NAU, D. S.; THROOP, T. *Computer Bridge: A Big Win for AI Planning*. AI Magazine, 1998, p. 93-105.

[SMY1993] SMYTH, B.; K. M. T. *Retrieving Adaptable Cases: The role of Adaptation Knoledge in Case Retrievals*. In Lecture Notes in Articial Intelligence, 837. Springer Verlag, 1993, p. 209-20.

[SPA2001] SPALAZZI, L. A. *Survey on Case-Based Planning*. Artificial Intelligence Review, 2001, Vol. 16, Num. 1, p.3-36.

[STU2002] STURTEVANT, N. *A Comparison of Algorithms for Multi-player Games*. In: Procedings of the 3o Conference on Computers and Games, 2002, p. 108-122.

[TEL1997] TELLA, J. *Seminar on Knowledge Engineering: Planning in Games.* Seminar on Knowledge Engineering - Helsinki University of Technology, 1997.

[THE2002] THE GUARDIAN. Guardian Unlimited | *Online* | Do not pass Go. 2002. disponível em: http://www.guardian.co.uk/*online*/story/0,3605,817484,00.html. Acesso em: 1/4/2005.

[THO2004] THOMPSON, G. *The aMazing history of Maze – It's a small world after all.* DigiBarn Computer Museum/Computer History Museum. Disponível em: http://www.digibarn.com/history/04-VCF7-MazeWar/index.html . Acesso em: 15/3/2005.

[TOZ2002] TOZOUR, P. *The Evolution of Game AI*. AI Programming Wisdom, Charles River Media, 2002, p. 3-15.

[TUR1950] TURING, A. M. *Computing machinery and intelligence.* In: Mind: A Quarterly Review of Psychology and Philosophy, 1950.

[ULA2004] ULAM, P. GOEL, A. JONES, J. *Reflection in action: Model-based self-adaptation in game playing agents.* In D. Fu & J. Orkin (Eds.) Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop (TR WS-04-04). San Jose, CA: AAAI Press., 2004.

[WEN2001] WENDLER, J. KAMINKA, GA. & VELOSO M.. *Automatically improving team cooperation by applying coordination models*. In B. Bell & E. Santos (Eds.) Intent Inference for Collaborative Tasks: Papers from the AAAI Fall Symposium (TR FS-01-05). Falmouth, MA: AAAI Press. 2001.

[WIL1998] WILKE, W. B. *Techniques and Knowledge Used for Adaptation during Case-Based Problem Solving*. In: Proceedings of 11th International Conference On Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1998, p. 497-506.

[WOO1994] WOOLLEY, D. R. Plato: *The Emergence of Online Community*. 1994. Disponível em: http://www.thinkofit.com/plato/dwplato.htm. Acesso em:15/03/2005.

[WOO2005] WOODCOCK, S. *MMOPG Chart*. 2005. Disponível em: http://www.mmorpgchart.com/ . Acesso em: 16/01/2005.

ANEXO 1 – TÉCNICAS DE *IA* EM JOGOS COMERCIAIS

A tabela sumariza as principais técnicas de *IA* aplicadas em jogos comerciais e exemplos de jogos nas quais elas foram aplicadas.

Os tópicos da primeira coluna foram extraídos dos dois primeiros capítulos de *AI Game Programming Wisdom 2* [RABIN2004] e refletem as técnicas de *IA* sob a ótica do desenvolvedor de jogos.

Técnica	Principais gêneros de jogo aplicáveis	Exemplos de Jogos (e desenvolvedores) que utilizaram cada técnica	Comentário
Pathfinding com A*	*RTS* (*Real time strategy* - Jogos de estratégia em tempo real)	Praticamente todos os *RTSs* desde Warcraft (Blizzard).	Busca de caminhos através de busca heurística.
Command Hierarchy (hierarquia de comandos)	*RTS*	Série Close Combat (Atom Games)	Estratégia para lidar com a IA em diferentes níveis: do general ao soldado raso modelados através de uma hierarquia militar.
Dead Reckoning	*FPS* (*First-person shooter*); jogos de esporte; jogos *online*	Quake III (id Games)	Método que objetiva a projeção do movimento futuro de cada agente a partir de sua velocidade, posição e aceleração. Em jogos online é utilizado para combater a latência. Em jogos de esporte para prever a posição do jogador de outro time.
Emergent Behaviour (comportamento emergente)	Jogos de corrida; Jogos de simulação de vida artificial	Downforce (Smartdog)	Comportamento que não é explicitamente programado mas emerge devido a interação de comportamentos mais simples. *Flocking* é o exemplo clássico de comportamento emergente.
Flocking	RTS FPS	Half-life (Valve); Unreal (Atari)	Técnica inserida em *Emergent Behaviour* e responsável pelo movimento de grupos de criaturas através da utilização de regras simples.
Formations (formações)	RTS	Empire Earth (Stainless Stell Studios)	Técnica que objetiva mimetizar as formações militares. Diferencia-se de flocking pois cada unidade é guiada a partir de um objetivo específico a partir de sua hierarquia e posição.
Influence Mapping (mapeamento de influência)	RTS; Jogos de simulação	Age of Empires (Emsemble Studios) Sim City (Maxis);	Mapeamento de influência é um método de visualização da distribuição do poder em um espaço a partir de uma representação bidimensional que reflete a influência de cada unidade na célula que está contida e em seus vizinhos.
Level-of-detail AI (LOD-AI)	RPG; RTS; Jogos de simulação	Republic – The Revolution (Elixir Studios)	Variação na freqüência de atualização das computações realizadas pela IA conforme a proximidade do jogador. Por exemplo, se o personagem não está sendo visto pelo jogador, a sua IA pode ser atualizada apenas quando o resultado for perceptível.
Manager Task Assignment (Atribuição de tarefas por um gerenciador)	Jogos de esporte	*Microsoft Baseball 3D* (Microsoft)	O gerenciamento de tarefas considera, para a resolução de um problema, um conjunto de tarefas e um grupo de agentes. Um gerenciador centralizado prioriza as tarefas e as distribui ao melhor candidato para resolver cada tarefa.
Obstacle Avoidance (desvio de	FPS; RTS		Enquanto algoritmos de *pathfinding* utilizando A* encaixam em ambientes estáticos, nos jogos os ambientes, muitas

obstáculos)			vezes são dinâmicos e o personagem deve desviar de obstáculos no caminho.
Scripting	Todos os gêneros	*Black & White (Lionheart); Half-Life II (Valve); Monkey Island 4 (LucasArts).*	Técnica que corresponde à especificação dos dados ou da lógica do jogo fora do código-fonte principal do jogo. Cada jogo pode ter um nível de *scripting* diferente e quanto mais "scriptado" maior é o poder do *designer* no código do jogo, abrindo a possibilidade, até mesmo, de criação de IAs extensíveis pelo jogador. Linguagens comuns utilizadas para a geração de *scripts* são Lua e Python.
FSM - Finite-State machines (máquinas de estado)	Todos os gêneros	*Half-Life (Valve)*	Conjunto finito de estados e transições, com apenas um estado ativo de cada vez. Cada estado representa, geralmente, um comportamento. É um dos padrões de projeto mais populares na indústria de jogos.
Stack-based State Machine também conhecido como *push-down automata*	Todos os gêneros principalmente FPS e RTS	*Thief: The Dark Project (Looking Glass Studios)*	Extensão à *FSM* que permite o retorno a um estado pelo qual a máquina já passou. Útil para tratar interrupções momentâneas causadas por eventos externos à máquina.
Subsumption Architecture	FPS; RTS Jogos cujo movimento de uma unidade não pode conflitar com os objetivos de alto-nível.	*Halo (Bungie)* *Halo 2 (Bungie)*	Nesta arquitetura há a separação do comportamento de um agente em diversas camadas concorrentes. A camada mais baixa, e com mais prioridade, cuida dos comportamentos mais elementares e as mais altas vão sendo responsáveis por comportamentos cada vez mais complexos, como, por exemplo, estratégias de ataque.
Terrain Analysis	RTS FPS	*Empire Earth(*Stainless Stell Studios) *Age Of Empires (Emsemble Studios)* *C&C Renegade(Westwood Studios)*	Conjunto de métodos que objetivam a análise de terrenos dentro de um jogo para encontrar locais com elementos estratégicos como recursos e emboscadas. Esta abordagem contrasta com a determinação destes elementos de forma fixa pelo editor do mapa.

ANEXO 2 – DESCRIÇÃO DO *PROMISANCE*

Descrição em formato livre do jogo e a partir do manual original contendo as principais telas e ações que podem ser feitas pelo usuário. As telas apresentadas são do jogo reimplementado, já com as adaptações e simplificação da interface do jogo original.

O jogo original está disponível em: http://sourceforge.net/projects/promisance/

Babel ***Promisance*** é um Jogo de Gerenciamento de Impérios baseado em turnos. Ao se cadastrar, o jogador recebe um Império *online* persistente e o objetivo será fazer com que o império cresça e se destaque no ranking.

- Para isso, o jogador pode:
- Explorar novas terras;
- Construir obras para estimular o comércio, a emigração ou, até mesmo a agricultura;
- Atacar outros impérios da vizinhança com suas tropas;
- Investir em magos, que também podem atacar outros impérios;
- E outras atividades.

A tela inicial do jogo é apresentada abaixo. As opções disponíveis são a criação de um novo império ou entrar no jogo para os jogadores já cadastrados. Os demais *links* estão desabilitados.

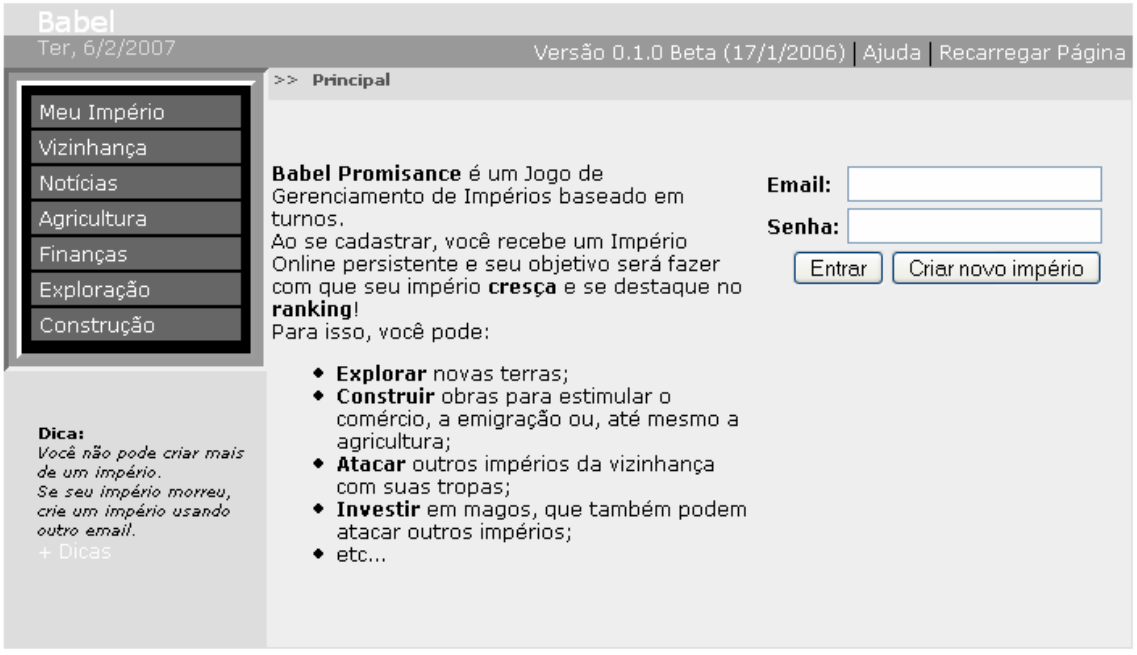

Ilustração A1 - Tela Inicial do Jogo.

Após o *login*, ao jogador são apresentadas as principais características de seu império. Além disso, já é possível acessar qualquer uma das funcionalidades laterais. Ou os botões centrais.

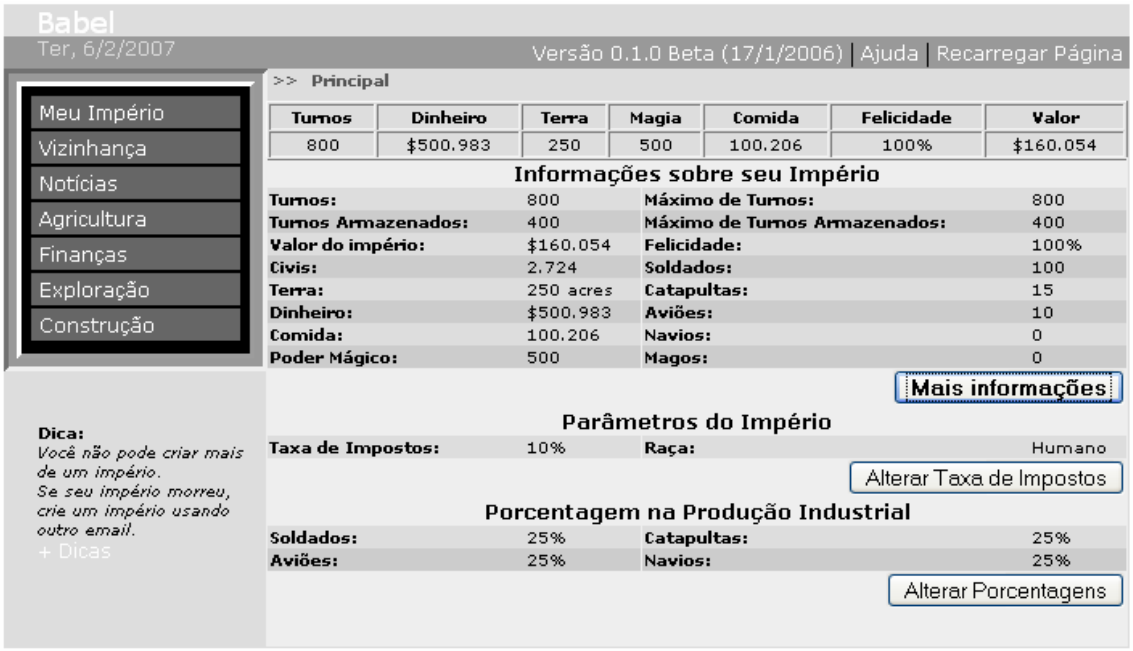

Ilustração A2 - Tela após o login.

Na barra superior, são apresentadas as principais informações sobre o império. Tais informações são utilizadas no processo de recuperação dos planos como as características indexadas:

- *Turnos*: Número de turnos que você possui para utilizar durante o jogo. Cada ação que você realiza consome um certo número de turnos. De tempos em tempos o império recebe mais turnos;
- *Dinheiro*: Quantidade de dinheiro que o império possui. Cada construção ou unidade possui um preço para ser obtida ou para ser mantida. Se o valor de dinheiro ficar com valor negativo, o império está em crise e nenhuma ação pode ser feita até conseguir torná-lo positivo;
- *Terra*: Acres que cada império possui. Cada acre permite que se faça uma construção. Acres são obtidos através de exploração ou atacando outros impérios.

- *Magia*: Poder mágico que o império possui. Quanto mais magos, maior é o poder mágico do império. Magia é utilizada para realizar feitiços de ataque. Cada feitiço tem um custo variável, o qual depende da evolução do império.
- *Comida*: Sacas de comida produzidas nas fazendas e armazenadas em seus armazéns. A cada turno sua população consome e produz comida. Se a quantidade de comida consumida for maior que a produção, as pessoas começam a morrer ou fugir de seu império.
- *Felicidade*: Felicidade geral das pessoas e das tropas. Quanto maior a felicidade, mais a população vai produzir. Se a felicidade ficar muito baixa, as tropas se recusarão a atacar. Impostos muito altos também diminuem a felicidade da população.
- *Valor*: Valor do império em relação aos outros. O objetivo do jogo é aumentar o valor. Assim, quanto maior o império, maior seu valor. O cálculo do valor é feito através do tamanho do império e do dinheiro.

As demais características do império não são usadas na recuperação dos planos, mas algumas delas são usadas no processo de antecipação de erros:

- *Civis*: Quantidade de pessoas que vivem no império. Pessoas são necessárias para produzir dinheiro e pagar impostos. Pessoas necessitam de comida para sobreviver;
- *Soldados*: Unidade especializada em ataques terrestres. Unidade militar mais básica e barata;
- *Catapultas*: Unidade especializada em ataques terrestres mais poderosos que os soldados;
- *Aviões*: Unidade militar especializada em ataques aéreos;
- Navios: Unidade militar especializada em ataques marítmos;
- *Magos*: Magos produzem poder mágico para se realizar feitiços. Em feitiços ofensivos, quanto maior o número de magos, maior a chance do feitiço ser bem-sucedido. Para produzir magos, o império deve possuir áreas escolares. Quando os feitiços falham, uma parcela dos magos morre;

- *Taxa de impostos*: A porcentagem da população multiplicada pela renda per capita que vai para os cofres públicos a cada turno. Para cada 2% acima de 10%, a felicidade máxima da população é cortada em 1%.
- *Raça*: Raça da população do império. Cada raça possui multiplicadores diferentes para o resultado da execução das ações. As raças disponíveis são: humanos, elfos, anões, *trolls*, gnomos, *gremlins*, *orcs*, elfos negros e *goblins*.

Clicando no botão Mais Informações, outra tela apresenta as demais características do império, sendo que algumas delas também entram no processo de antecipação de erros:

- *Turnos Usados*: Total de turnos já utilizados no gerenciamento de seu império desde o cadastramento do império;
- *Produção Estimada*: Tanto fazendas quanto setores não utilizados contribuem na produção de comida. A produção agrícola estimada é uma previsão do valor no próximo turno;
- *Consumo Estimado*: O consumo da comida é feito pelos militares, magos e civis do império. Este valor é o consumo estimado para o próximo turno;
- *Lucro*: Diferença entre a comida produzida e a comida consumida. Quando este valor é negativo, a população está passando fome;
- *Ataques*: Número de ataques realizados pelas tropas;
- *Ataques bem sucedidos*: Número de ataques realizados por suas tropas que conseguiram ser bem sucedido. Um ataque é bem sucedido quando captura terra do império adversário;
- *Defesas*: Número de ataques sofridos por um império;
- *Defesas bem sucedidas*: Número de ataques sofridos por seu império que foram repelidos;
- *Áreas comerciais*: Áreas comerciais estimulam o crescimento da economia do império. Além de aumentarem a renda *per capita* da população também contribuem na geração de dinheiro;

- *Áreas residenciais*: Nas áreas residenciais a população é estimulada a morar. Embora as pessoas possam se instalar nos setores vazios do império, nas áreas residenciais mais pessoas podem morar;
- *Áreas militares*: Áreas militares reduzem os custos de armazenamento e manutenção das tropas. Entretanto, só fazem efeito se o império conter, pelo menos, 30% das áreas como áreas militares;
- *Áreas industriais*: A indústria é completamente destinada à produção de tropas. A porcentagem de cada tropa produzida, a cada turno, pode ser alterada;
- *Áreas escolares*: As escolas são destinadas ao treinamento dos magos do império. Outra utilidade das escolas é servir de morada e produzir certa quantidade de mana;
- *Fazendas*: Nas fazendas os grãos que produzem a comida do império são cultivados. Quanto mais fazendas, mais comida é produzida a cada turno;
- *Defesa Fronteiriça*: Cada unidade de defesa fronteiriça aumenta em 500 pontos defensivos a defesa geral do império;
- *Terra não utilizada*: Cada construção feita em seu império deve ser alocada em um acre de terra livre. Acres são obtidos através de exploração ou de ataques. Acres livres também servem de morada para sua população, embora de maneira menos efetiva que as áreas residenciais;
- *Receita esperada*: Quantidade estimada de dinheiro que vai para os cofres públicos no próximo turno. Depende da população, da renda per capita, da taxa de impostos e da felicidade;
- *Gastos esperados*: Quantidade estimada de dinheiro que sai dos cofres públicos no próximo turno. Depende dos custos para manter as tropas e da quantidade de terra que um império tem para administrar. Para baixar esse valor, sem diminuir o número de tropas, o império deve investir em áreas militares;
- *Lucro esperado*: Quantidade estimada de dinheiro que sobra ou falta a cada turno. Se o valor for negativo, o que faltou é retirado do valor total de dinheiro do império;

- *Renda per capita*: Representa a quantidade de dinheiro que cada civil gera por turno. A taxa de impostos é uma porcentagem aplicada sobre este valor;
- *Pontos ofensivos*: Indica o valor ofensivo total do império, ou seja, é uma estimativa do poder máximo do conjunto das tropas;
- *Pontos defensivos*: Indica o valor defensivo total do império, ou seja, é uma estimativa da defesa máxima do conjunto das tropas.

Clicando no botão Alterar Taxa de Impostos, outra tela apresenta a possibilidade de se alterar a taxa de impostos do império. A taxa de impostos é a porcentagem da renda per capita que vai para os cofres públicos. Também influencia na emigração e imigração do império. Assim, se os impostos estiverem muito altos sua população vai fugir. Além disso, para cada 2 pontos percentuais acima de 10%, a felicidade da população diminui em 1 ponto percentual.

Clicando no botão Alterar indústria, outra tela apresenta a possibilidade de se alterar as porcentagens de produção industrial. A produção industrial indica o quanto, a cada turno, cada unidade é produzida em relação ao potencial total de produção. Cada unidade tem um potencial ofensivo e defensivo diferente. Por exemplo, um navio tem potencial ofensivo cerca de 7 vezes a mais que um soldado.

Na Figura a seguir é mostrada a tela de Vizinhança. Nesta tela o usuário pode interagir com os impérios controlados por outras pessoas ou por jogadores automáticos. Ao clicar sobre o nome de um império, uma caixa de texto permite a escolha de 10 possíveis ações dentre ataques com tropas e ataques mágicos:

- *Ataque padrão*: Obtém terra e algumas construções se bem sucedido Diminui em 8% a felicidade da população;
- *Ataque surpresa*: Diminui em 18% a felicidade da população, mas garante um bônus ofensivo de 25%;
- *Ataque terrestre*: Idêntico ao ataque padrão. Ataca com catapultas e o adversário também defende apenas com catapultas.
- *Ataque aéreo*: Idêntico ao ataque padrão. Ataca com aviões e o adversário também defende apenas com aviões.

- *Ataque marítmo*: Idêntico ao ataque padrão. Ataca com navios e o adversário também defende apenas com navios.
- *Bola de Fogo*: Destrói de 1% a 3% das tropas inimigas;
- *Tufão*: Destrói 10% da comida do império inimigo. Destrói 12% do dinheiro do império inimigo;
- *Tempestade*: Acaba com 3% do poder mágico do império inimigo;
- *Terremoto*: Inutiliza 3% das construções adversárias. O adversário deve ter pelo menos 15 construções de cada para ser atingido;
- Assalto: Inutiliza 3% das construções adversárias. O adversário deve ter pelo menos 15 construções de cada para ser atingido;

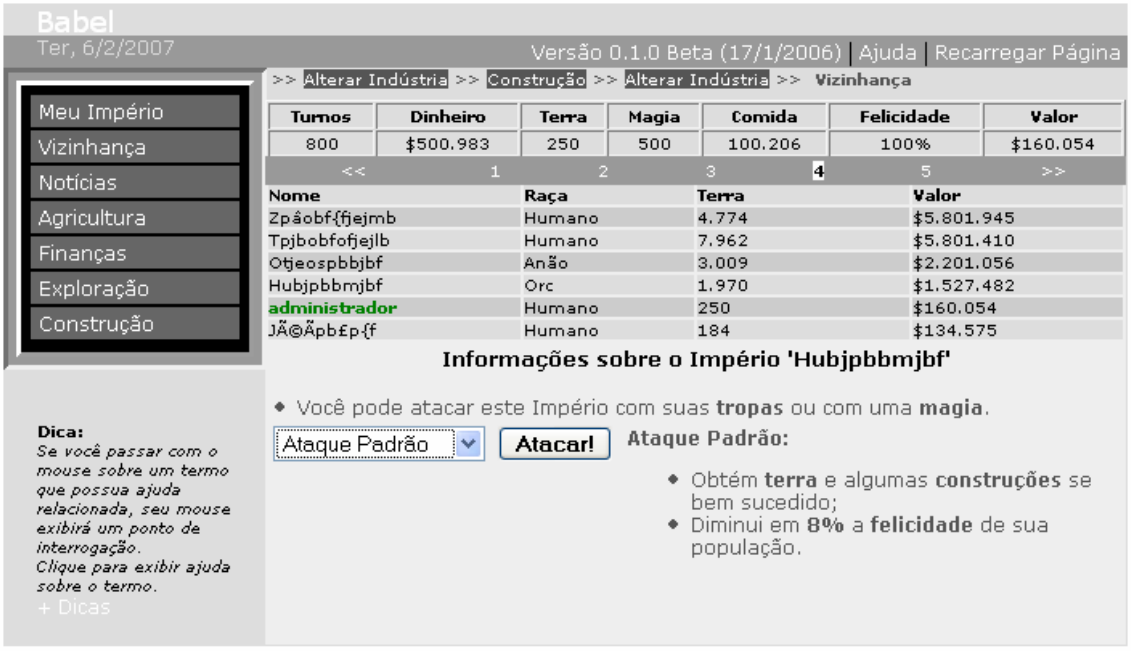

Ilustração A3 - Vizinhança.

Para impedir que os impérios descubram quem são os jogadores automáticos e quem são os seres humanos, o nome dos impérios é criptografado. É importante destacar que cada jogador tem uma visão diferente do seu conjunto de adversários, ou seja, o nome é criptografado de uma maneira diferente.

Todos os ataques realizados são exibidos na tela de Notícias, que permite que se responda também ao ataque.

As funcionalidades de Agricultura, Finanças e Exploração são ações semelhantes. Quanto à primeira, durante os turnos gastos na produção agrícola, a população vai produzir 25% a mais de do que o normal. Na ação de Finanças, embora as despesas não sejam afetadas, o império produz 25% a mais de receita. E, a exploração é a única maneira de obter setores livres para o império sem ser através de um ataque. Cada setor livre pode ser usado para uma construção. A tela de construção, por fim, permite que se construam áreas comerciais, residenciais, militares, industriais, escolares, agrícolas e defesas nas fronteiras. Cada construção tem um preço e, conforme já citado, ocupa um setor livre. Agricultura, Finanças, Exploração e Construção são ações que se caracterizam por um gasto variável de turnos. Ao jogador é apresentada a evolução do império a cada turno, conforme mostrado abaixo.

Ilustração A4 – Navegação pelos turnos.

ANEXO 3 – GENEALOGIA E HISTÓRIA DOS JOGOS *ONLINE*

Discussão e contextualização dos principais jogos e marcos referentes à história dos jogos *online*. Tal seção é importante para reafirmar a importância dos jogos *online* tanto em relação ao mercado quanto em relação à academia.

Devido à classificação do ***Promisance*** como um jogo *online multiplayer* e a possível e desejável aplicabilidade da metodologia proposta em capítulo posterior em problemas e jogos semelhantes, faz-se necessário o enquadramento do jogo em relação aos jogos *online* e, principalmente, jogos *online* de gerenciamento de império. Tal enquadramento abre a possibilidade de trabalhos futuros que apliquem as técnicas de *IA* propostas em outros jogos e, principalmente, contextualiza a origem do jogo em questão.

Colocada como anexo, para não destoar do foco da dissertação, a qual se concentra no aspecto da disciplina da *IA* e, também, por basear-se, principalmente, em referências *online*.

Genealogia dos jogos *online*

Um mundo virtual é uma representação *online* persistente que contém a possibilidade de comunicação síncrona entre usuários e entre o usuário e o mundo com uma infra-estrutura espacial projetada para ser um universo navegável [KLA2005]. Diversos mundos virtuais são jogos *online*. Tais ambientes são caracterizados pelo objetivo primário de entreter os usuários que compartilham o mundo.

Nesta seção, não apenas jogos *online multiplayer* em um sentido restrito são contextualizados. As suas influências e heranças nos jogos em geral também merecem comentários com o objetivo de expandir a aplicabilidade das técnicas de *IA* que estão sendo propostas em domínios diferentes, como, por exemplo, em jogos de estratégia *single-player*.

As principais referências utilizadas para construir essa seção foram as referências *online*: [MUL1999], [KOS2002], [HUN2005], [WOO2005], [MAR2001] e [GOL2001]. Além destas, as principais fontes são referenciadas durante o texto para cada "era" dos jogos *online*. Convencionou-se a separação em seis fases distintas, separadas em um tópico cada:

- Pré-história dos jogos *online*;
- A década de 70 e os primeiros jogos *online*;
- A pré-história da indústria de jogos *online* e sua era de ouro;
- Jogos *online* em BBS;
- A explosão dos jogos para computador;
- A era moderna dos jogos *online*: os jogos massivos *multiplayer*.

É interessante observar que cada geração parece ignorar os eventos da anterior, mal convencionando os seus marcos. Por exemplo, os jogos para *BBS*, em algumas fontes, nem são citados em relação à linha do tempo dos jogos *online*.

Em outro exemplo característico, muitos acreditam que o primeiro jogo *online* foi *Doom* em 1993 por causa do grande sucesso e influência na indústria de jogos. Isto posto, é surpreendente a existência de conceitos que são reinventados desde a década de 70. No caso citado, os jogos de tiro de 1ª pessoa foram criados em 1973. O objetivo desta seção é, portanto, estabelecer os jogos *online* como um tipo de jogo e conceito recorrente e estabelecido no universo lúdico.

Pré-história dos jogos *online*

Convenciona-se como marco inicial dos jogos eletrônicos o jogo desenvolvido pelo físico Willy Higinbotham em 1958 para distrair os visitantes do *Brookhaven National Laboratories*. Na época, era comum a abertura para visitação ao público e o demonstrativo desenvolvido pelo físico fazia sucesso.

O referido jogo, um precursor do *Pong* chamado "*Tennis for Two*", era executado em um osciloscópio de tela de 5 polegadas conectado a duas caixas com um dial e um botão. Cada dial permitia que o jogador controlasse a trajetória de uma bola e o botão rebatia a bola para o outro lado da tela. Embora não houvesse escore, relata-se que os visitantes ficavam horas na fila para jogar a demonstração.

Apesar do sucesso entre os visitantes, o jogo não foi registrado por ser considerado por seu criador como uma idéia óbvia e que não valia a pena sua comercialização. Além disso, "Tennis for Two" era só um demonstrativo. Dessa forma, o papel de primeiro jogo eletrônico geralmente é atribuído para o jogo "*SpaceWar!*" de 1962. De fato, é o primeiro software desenvolvido para um computador de uso geral, no caso um mainframe PDP-1 utilizado pelos pesquisadores do MIT.

O jogo era uma competição entre duas naves atraídas por um sol central que tinham que controlar seu combustível e possuíam armas para destruir o adversário. Os controles, para dois jogadores, também incluíam um botão de pânico que fazia uma rotina de fuga, a qual levava a nave ao hiperespaço e fazia com que a nave sumisse por alguns instantes e fosse rematerializada em um local aleatório. Como programa demonstrativo, atendeu com eficiência aos critérios de demonstrar os recursos do computador testando-os ao limite, além de consistir em um *framework* consistente para testes já que em cada execução do jogo era diferente o comportamento da simulação.

Deve-se destacar que além de serem os primeiro jogos eletrônicos, os jogos citados, embora não tenham nenhum aspecto *online*, como por exemplo, uma rede de computadores, já possuíam características *multiplayer*. Tanto que o início da era *online* se dá com uma versão de SpaceWar! para o sistema *online* PLATO.

A década de 70 e os primeiros jogos *online*

Koster [KOS2002]cita como o primeiro jogo realmente em rede uma versão de SpaceWar! para o serviço PLATO em 1969. Tal jogo, escrito por Rick Blomme, é considerado como o marco inicial da era dos mundos *online* segundo a Linha do Tempo dos jogos *online* compilada por Jessica Mulligan [MUL1999]. A mesma autora também contextualiza que de 1970 a 1977 uma série de outros jogos *multiplayer* surgiram no mesmo sistema.

Dois fatores explicam o sucesso do PLATO (*Programmed Logic for Automatic Teaching Operations*), originalmente um sistema de tempo compartilhado dedicado a experimentos na área de educação, como plataforma para jogos: poder gráfico superior a qualquer plataforma da época e a uniformidade do hardware, ou seja, como nos consoles atuais, os terminais tinham o mesmo poder computacional. Dessa forma, os jogos tinham a mesmo desempenho em todos os pontos da rede, embora a velocidade de conexão pudesse variar. Outros fatores para o sucesso da plataforma em jogos são citados por Woolley [WOO1994], além dos acima:

- Áreas de memória compartilhadas;
- Processamento centralizado de cada tecla pressionada;
- Resposta rápida das teclas;
- Possibilidade de abortar a saída do *display*.

Alguns jogos de destaque da época populares ou surgidos na plataforma PLATO são citados abaixo. Deve-se destacar que as datas e funcionalidades dos jogos são polêmicas, e, muitas vezes imprecisas. Além disso, por seu caráter *online* e completamente centralizado, novas funcionalidades eram agregadas e retiradas dos jogos no decorrer dos anos. Outro ponto importante a ser considerado é o fato dos jogos poderem ser apagados do sistema. Assim, outra pessoa poderia refazer o jogo anterior a partir do que lembrava e, é claro, adaptá-lo e melhorá-lo. Os jogos sempre se renovavam, e tal dinamismo era o que mantinha o interesse dos jogadores.

Sem seguir uma ordem de importância ou cronológica, destaca-se:

- *Dogfight* - 1973: simulador de vôo no qual dois jogadores controlam aviões que tentam se destruir entre si. Era um jogo simples, mas que já exibia o problema hoje conhecido como *lag*, ou seja, o jogador com maior velocidade de conexão com o servidor tinha vantagem em relação ao outro.

- *Empire* – 1974: jogo de gerenciamento de impérios espaciais com ambientação baseada no universo do seriado *Star Trek*. *Empire* suportava até 32 jogadores divididos em 4 times (Federação, Romulanos, Klingons e Orions). A primeira versão foi escrita por John Daleske e Silas Warner. *Empire* é um dos jogos que mais influenciou os jogos de estratégia posteriores, sendo o primeiro jogo *online* de estratégia com times competindo entre si. Predecessor dos jogos de estratégia em tempo real.

- *Spasim* – 1974: Jogo referenciado como primeiro jogo de tiro 3D em primeira pessoa [BOW2001]. Semelhante a *Empire*, era um jogo com 32 jogadores desenvolvido por Jim Bowery. Cada jogador era representado para o outro por uma espaçonave em *wireframe* cuja visualização era atualizada a cada segundo. Existe uma controvérsia relacionada a um outro jogo, chamado *Maze* sobre qual dos dois jogos é o pioneiro no universo dos jogos 3D. Destaca-se que *Spasim* é melhor documentado que *Maze* e, além disso, este último, utilizava uma perspectiva fixa.

- *Airfight* – 1974 / 1975: Inspirado no código do Spasim, o programador Silas Warner criou o primeiro simulador de vôo *3D*, chamado *Airace*. Inspirado em tal código, o programador Kevin Gorey criou uma versão *3D* do jogo *Dogfight*. Era um simulador de vôo em tempo real com visão 3D do horizonte, do aeroporto e do inimigo.

- *DND* - 1974: Não era um jogo *multiplayer*, mas foi um dos primeiros jogos de masmorra gráficos a serem executados no sistema PLATO. Provavelmente derivado de um jogo de 1972 de outro sistema. DND serviu de inspiração para todos os subseqüentes. Foi o primeiro jogo a ter início, meio e fim e a apresentar lojas para comprar itens e um chefe protegendo o objetivo final.

- *Mines of Moria* – 1974: Primeiro jogo de exploração de masmorras, no qual múltiplos jogadores exploravam 248 diferentes labirintos em bandos ou

individualmente. Cada cena era composta por uma visão detalhada incluindo a sala, monstros, portas e corredores, além da força e poderes do personagem.

- *Oubliette* – 1977: Um jogo de *RPG* (Role Playing Game – jogo de interpretação de papéis) com bastante influência do *Dungeons & Dragons*. Foi desenvolvida a possibilidade de criar o seu personagem e salvá-lo para utilização posterior. O jogador, assim, desenvolvia os atributos, classes e raças de seu personagem. O ambiente 3D em primeira pessoa desenhado por linhas (*wireframe*) também se destaca como contribuição deste jogo.

- Avatar – 1978: jogo de masmorras também baseado no recém lançado universo do *RPG* de mesa *Dungeons and Dragons*. Possivelmente o jogo mais popular na história do sistema [WOO1994]. Foi feito com o objetivo de suplantar *Oubliette*. O tamanho e complexidade do jogo demandaram um desenvolvimento de três a quatro anos para ser completado. Permitia a comunicação entre os jogadores, troca de itens e a atribuição de *quests* essenciais para a troca de nível. O jogo ainda tem muitos fãs e pode ser acessado em servidores dedicados.

Embora o sistema PLATO, pelos motivos já citados, seja o berço da maior parte dos jogos *online* da década de 70, alguns outros sistemas tiveram iniciativas de destaque. Cita-se, portanto, outros jogos *online* da década de 70 com importância em outros sistemas:

- *Maze* (também conhecido como *Maze War*) – 1974*:* Também é referenciado como o primeiro jogo de tiro 3D em primeira pessoa [THO2004]. Introduz o conceito de jogadores *online* usando avatares e perseguindo uns aos outros em um labirinto.

- *DECWAR* (também conhecido como DECWARS) - 1978: Sofisticado jogo de estratégia em tempo real destinado para de um a dezoito jogadores desenvolvido para a plataforma DEC-10 na *University of Texas at Austin*. Semelhante ao *Empire* do sistema PLATO. O primeiro jogo espacial multiplayer de gerenciamento de império (*MegaWars III*) é derivado das idéias expostas em DECWARS [MAR2001].

- *Empire* – 1972 / 1978: Precursor dos jogos de guerra entre civilizações. Também é um jogo em tempo real no qual de 4 a 100 jogadores interpretam líderes de uma nação e devem gerenciar recursos e população civil e militar. O jogo e suas variantes persistem até hoje com torneios e servidores disponíveis pela Internet. É diferente do *Empire* da plataforma PLATO e foi escrito originalmente para uma plataforma *HP* e depois adaptado para o DEC-10 em 1978.

- *MUD (Multiple User Dimension, Multiple User Dungeon, ou Multiple User Dialogue)* – 1978*:* Richard Bartle e Roy Trubshaw começam o desenvolvimento do *MUD* em 1978. *MUDs* são ambientes virtuais nos quais os usuários controlam um personagem virtual. O jogador pode explorar áreas, conversar com outros personagens, combater monstros, obter itens, etc. *MUDs* são jogos textuais bastante dependentes da interpretação e imaginação dos participantes assim como em um *RPG (Role-playing game*) de mesa. Tais ambientes, com temática medieval influenciada pela ambientação criada por Tolkien, são a principal influência para os jogos *online* mais visitados na atualidade como *World of Warcraft*, *Everquest* e outros.

Alguns jogos da época, mesmo não sendo *online*, merecem também destaque por conter elementos que foram agregados por jogos *online* posteriores:

- *Star Trek* – 1972: Jogo de estratégia por turnos escrito em *BASIC* por Mike Mayfield no ano de 1972. Um dos primeiros jogos espaciais a serem criados. Sua fama e contribuição estão em ser um dos primeiros jogos textuais. Tal fato implicou na pouca necessidade de recursos computacionais necessários para jogá-lo [MAR2001]. Devido a sua simplicidade, portanto, *Star Trek* foi um dos jogos mais adaptados para computadores pessoais e *mainframes* da época. *Empire* (do sistema *PLATO*) se baseia em *Star Trek*.

- *Hunt the Wumpus* – 1972: Escrito por Gregory Yob em 1972, *Hunt the Wumpus* é um jogo de esconde-esconde em um labirinto com o formato de um dodecaedro. Pode ser considerado um precursor dos jogos do gênero *Adventure* (Aventura, jogos com histórias interativas). Sua inovação está em provar que um mundo virtual poderia ser em qualquer formato e tamanho diferente do

tradicional *grid* com duas dimensões. Além disso, demonstrou também como o formato do mundo do jogo tem influência direta na jogabilidade. Outra contribuição relevante é a introdução à perspectiva do jogador em incorporar um personagem. Em todos os jogos anteriores, o jogador **controlava** algum objeto, no *Hunt the Wumpus*, entretanto, o jogador passou a ser tratado como **você**, ou seja, o jogador **é** alguém no mundo do jogo.

- *Adventure* (também conhecido como *Advent* ou *Colossal Cave*) – 1974: Primeiro jogo do gênero de *Adventure*. Desenvolvido por Will Crowther era um simulador de exploração de cavernas expandido por Don Woods, em 1976, em uma aventura medieval com elfos e *trolls*. O sucesso deste tipo de jogo provocou o surgimento de diversas variantes e jogos semelhantes. O mais famoso é *Zork* [LEB1979]. Uma das variantes do jogo chamada *DUNGEN* tem influência direta sobre o desenvolvimento do *MUD* [BUR1995]. Destaca-se que o *MUD* não deixa de ser uma versão *multiplayer* de um jogo de *Adventure*.
- *Star Trader*- 1974: Jogo de nave baseado em setores no qual a nave do jogador movimenta-se entre esses setores vendendo e comprando combustível, comida e equipamento. A principal contribuição desse jogo é o fato de ter servido de base para as dinâmicas de *trading* (negociação) em jogos de estratégia como *Trade Wars* e posteriores [PRI2003].

Uma observação é pertinente em relação à ambientação da maioria dos jogos da época. O sucesso dos universos das franquias *Star Trek*, *Star Wars* e dos livros de Tolkien era grande no meio acadêmico. A influência de Tolkien se dá, principalmente, através do lançamento do primeiro *RPG* (*Role Playing Game* – jogo de interpretação de papéis), *Dungeons & Dragons* em 1973. Dessa forma, justifica-se o porquê da maioria dos jogos da época e subseqüentes ter temática relacionada à fantasia medieval ou ficção científica.

O cenário descrito nesta seção com servidores *mainframe* e jogos em terminal com temática espacial ou medieval foi restrito, portanto, ao meio acadêmico, que, efetivamente, tinha acesso ao uso de redes de computadores. A indústria de jogos, e, conseqüentemente, o consumidor comum não teve acesso a jogos *online* até a popularização do *modem*. A década de 80 foi o início desta popularização.

A pré-história da indústria de jogos *online* e sua era de ouro

Em 1979, a *Compuserve* tornou-se o primeiro serviço a oferecer *email* para usuários de computadores pessoais. Alguns dos primeiros jogos *online* comerciais também floresceram neste ambiente: *MegaWars I, MegaWars III* e *Islands of Kesmai*. O primeiro, lançado em 1983, era uma conversão de *DECWARS* para a plataforma da Compuserve removendo qualquer referência ao universo de *Star Trek* para evitar problemas de *copyright* [MAR2001]. Seu diferente sucessor na Compuserve, *MegaWars III*, segundo a linha do tempo de Koster [KOS2002], é derivado de outro jogo chamado "S", de 1979, ambos criados pela Kesmai, empresa fundada por Kelton Flinn e John Taylor. *MegaWars III* guarda a honra de ter uma das vidas mais longas para um jogo *online* pago. Ele foi descontinuado apenas em 1998 [MAR2001]. Lançado em 1984, a Kesmai desenvolveu, também na Compuserve, *Islands of Kesmai*. O potencial comercial dos jogos *online* já começa a se configurar: para jogá-lo, o jogador deveria pagar doze dólares a hora. *Islands of Kesmai* foi um *RPG online* que permaneceu por treze anos [MUL1999].

Com o sucesso da Compuserve na área de jogos *online*, outros serviços surgiram. O principal competidor da Compuserve era o serviço GEnie (*GE Network for Information Exchance*) lançado em 1985. *Stellar Warrior*, uma versão simplificada de *Mega Wars III*, foi um dos primeiros jogos a serem lançados para o serviço. O preço para acessar os serviços da GEnie era de seis dólares a hora. Com o tempo, o preço de acesso diminuiu para três dólares. Insiste-se, novamente, no potencial comercial já existente: Mulligan cita em [KOS2002] que no ano de 1991, o usuário médio de jogos *online* do serviço GEnie gastava uma média de 156 dólares, ou seja, 32 horas de jogatina mensal.

Outros jogos relevantes do serviço GEnie foram *Rim Worlds War* e *Air Warrior*. *Rim Worlds War* é o primeiro PBEM (Play by email – jogo por email) em um servidor comercial [MUL1999]. *Air Warrior*, lançado em 1986, é o primeiro jogo massivo *multiplayer* baseado nos recursos gráficos. Era um simulador de combate aéreo único, até então, por ter sido portado para múltiplas plataformas (Macintosh, Amiga, Atari ST, e IBM PC) e os clientes de cada plataforma conseguiam competir entre si em um mesmo mundo virtual [KOS2002]. O sucesso desse jogo só foi suplantado pelo jogo de *RPG Gemstone II*, lançado em 1988. Outro *RPG* de importância foi *Dragon's Gate*, surgido em 1991.

Outro competidor importante a entrar no mercado em Novembro de 1985 foi o serviço *QuantumLink*. O destaque desta plataforma era ter uma interface gráfica diferenciada construída especialmente para os usuários de computadores *Commodore 64/128*. Os usuários dos demais serviços estavam acostumados com uma interface textual. Assim, os jogos da *QuantumLink* aproveitaram desde o princípio o potencial gráfico possível ao Commodore. Um comentário merece ser feito à evolução da *QuantumLink*. Após alguns anos seu nome foi mudado para *America Online*, empresa que foi o embrião para a atual AOL-Time-Warner.

Um dos jogos de maior impacto da *QuantumLink* foi *Habitat,* lançado em 1985 [MOR1990]. Desenvolvido pela LucasFilm, *Habitat* foi uma das primeiras tentativas de criar um mundo virtual *multiplayer* nos moldes dos atuais *MMORPGs (Massive Multiplayer Online RPGs)*. Sua contribuição mais marcante foi a introdução ao uso do termo "*avatar*" como a representação gráfica do usuário no mundo digital. Habitat era um mundo virtual no qual os personagens podiam se comunicar, jogar, casar com outros, fundar religiões, etc. A lição ensinada por Habitat foi que o *cyberespaço* é definido mais pelas interações entre os atores envolvidos do que pela tecnologia implementada.

Outros destaques da *QuantumLink* foram *Rabbit Jack's Casino* (1985 / 1986), o primeiro jogo *multiplayer* gráfico do sistemas; *AppleLink* (1988), era uma rede de jogos para os computadores *Apple II*; *Club Caribe* (1988), ambiente derivado do Habitat; *Neverwinter Nights*, lançado no início da década de 90 foi o jogo primeiro jogo *online* a licenciar a ambientação de *Dungeons & Dragons*.

Iniciava-se, com as iniciativas citadas, a era de ouro dos serviços *online* proprietários pagos. A indústria *online* em 1993 tinha um número estimado de 3 a 10 milhões de casas conectadas e um faturamento de 10 a 15 milhões de dólares num universo de cerca de 15 jogos *online* espalhados pelos cinco principais serviços. Por outro lado, a Internet (na época ainda no formato de DARPA-net) ainda era limitada ao meio acadêmico. Os jogos mais populares nesse ambiente eram os derivados do *MUD* de Richard Bartle.

Por fim, a era de ouro dos provedores de rede fechada durou até os serviços *online* começarem a ser suplantados pelo acesso à *Internet* em 1993 e os jogos em computadores pessoais da linha IBM-PC terem um crescimento com o sucesso com o lançamento do jogo *Doom*. Em seção subseqüente, tal era será focalizada. Antes, todavia, faz-se necessária a

abertura de um parêntese para a exploração de um universo paralelo ao descrito, mas não menos importante: os jogos para *BBS* (*Bulletin Board System*) ou *door games*.

A geração representada pelos jogos supracitados teve grande influência na indústria atual, sendo chave para criação dos principais conceitos que hoje são aplicados nos jogos *online*. Destaca-se o modelo de negócios aplicado: o interesse dos serviços *online* proprietários pagos era a maximização das horas de acesso, pois quanto mais horas o usuário jogasse, mais ele pagava.

Na mesma época que os jogos descritos nesta seção, existiam jogos nos quais não havia vantagem ao jogador que passar mais tempo conectado já que todos têm a mesma quantidade de turnos para gastar diariamente. Tal modelo de jogo, essencialmente assíncrono, é derivado diretamente dos *door games*, os jogos disponíveis em *BBSs* a partir da metade da década de 80.

Jogos *online* em BBS

A partir da segunda metade da década de 70, o computador pessoal começou a se popularizar entre os jovens hobbystas americanos. Começaram a surgir os primeiros computadores com poder de processamento e preço consideráveis ao uso doméstico. Nada mais natural do que as principais aplicações para um microcomputador serem jogos, principalmente na linguagem BASIC.

Em 1977, Dennis Hayes desenvolve o modem para computadores pessoais e começa a vender aos hobbystas. Tal dispositivo popularizou-se entre os donos de computadores pessoais que podiam utilizá-lo, inicialmente, para troca de arquivos ou para se conectar com as redes disponíveis nas Universidades. Em 1978, Ward Christensen e Randy Suess lançaram o primeiro *CBBS* (*Computer Bulletin Board System*). Quem possuía um *modem* poderia conectar-se ao microcomputador da *CBBS*, chamada *RCPM* (Remote CP/M – sistema operacional da época), e operá-lo à distância. Foi a primeira iniciativa civil com objetivo de criar uma comunidade virtual, ou seja, independente de uma Universidade ou empresa. Depois de certo tempo, os criadores começaram a cobrar pelo acesso aos arquivos que estavam disponibilizando. Em 1980, a CBBS tinha cerca de 11000 usuários. O pioneirismo da iniciativa merece destaque também pelo fato dos criadores não conhecerem, na época que o sistema foi lançado, PLATO, a ARPAnet, ou qualquer rede comercial.

Vários *CBBSs* (ou, simplesmente *BBSs*) surgiram, com as funcionalidades de sistemas de mensagens, discussões e, principalmente, troca de arquivos entre os usuários e os operadores do sistema. O ano de 1981 destaca-se pelo lançamento da plataforma que divulgou, realmente, a computação ao alcance de todos: o *IBM-PC*. Logo aparecem BBSs compatíveis com o sistema.

Tom Jennings lança em 1984, o software *FidoNet*. Tal programa permitia o transporte automático da mensagens de uma *BBS* a outra que tivesse o *software* instalado. Em um horário pré-determinado, o servidor de uma *BBS* telefonava para o servidor de outra e estes trocavam mensagens. O sistema teve sucesso não apenas por permitir o intercâmbio de informações entre usuários de BBSs diferentes, mas também por rodar em computadores IBM-PC. No fim de 1984 a primeira conexão internacional foi feita pela FidoNet. Em 1985 já existiam 600 nós da rede principalmente nos Estados Unidos e Canadá.Com o software de Jennings, qualquer um poderia montar ser um *sysop* (*System Operator* – operador do sistema) e ter sua *BBS* em casa. As *BBSs* tornaram-se, portanto, bastante populares entre as pessoas fora do meio acadêmico ou militar.

Outro elemento responsável pela popularização das BBSs, no contexto da mídia, foi o filme *War Games* (“Jogos de Guerra”). Na história, um adolescente hacker conecta-se ao computador do departamento de defesa americano e quase deflagra uma guerra mundial.

Já que a conexão com as *BBSs* era telefônica, e, geralmente as pessoas possuíam apenas uma linha disponível, os primeiros jogos disponíveis não tinham aspecto *multiplayer*, além de um possível *ranking* no qual as pessoas poderiam comparar sua performance. Não havia, portanto, real competição entre os jogadores, já que, provavelmente eles não estariam conectados ao mesmo tempo como em um sistema *online* de rede fechada. E, mesmo os jogos para mais de um jogador previam que apenas uma pessoa estivesse conectada ao jogo ao mesmo tempo, sendo, por esta característica, assíncronos. Tais jogos, com apenas um jogador conectado em uma fatia de tempo, podem ser classificados como *multiplayer, single user* (MPSU).

Os jogos para *BBS* também eram chamados de *Door Games*. Outra característica interessante é que os usuários recebiam, geralmente, um número limitado de ações disponíveis durante uma seção. Essa limitação garantia que os usuários pudessem jogar pelo menos uma vez por dia. Os jogos nasciam nas próprias *BBS* e, quando faziam sucesso, seu

criador disponibilizava para as outras *BBS* gratuitamente ou mediante uma taxa. É interessante observar que o modelo de jogos *shareware* foi aplicável aos *Door Games*, ou seja, os *sysops* poderiam instalar gratuitamente em seus sistemas uma versão limitada de um jogo e caso ele fizesse sucesso entre sua comunidade, a versão sem restrições seria comprada.

Em 1984, Chris Sherrick escreve *Trade Wars* para sua *BBS*. Tal jogo tornou-se um dos *door games* mais influentes e um dos primeiros a permitir a interação entre jogadores. *Trade Wars* é um jogo de impérios espaciais baseado em turnos. Não há combate em tempo real. Cada jogador recebe um número limitado de turnos por dia. Quando os turnos acabam, outro jogador recebe turnos e pode jogar. Embora acredite-se na influência de *DECWAR* e *MegaWars* como inspiração para *Trade Wars*, o criador do jogo comenta que não conhecia tais jogos na época [PRI2003]. Assim, os conceitos de *Trade Wars* são originários de *Star Trader*, do *boardgame Risk* e do jogo *Hunt the Wumpus*. Como o código do *Trade Wars* era aberto o que provocou o surgimento de diversas versões que evoluíram o conceito inicial, implementando, por exemplo, a possibilidade dos jogadores interagirem com o sistema de forma concorrente.

Com certeza um dos *door games* mais populares foi *LORD* (*Legend of the Red Dragon*), criado por Seth Able em 1989 [EHR2002]. *LORD* é um *RPG* com temática medieval com um mundo bastante completo. Embora o jogador tivesse um conjunto limitado de ações disponíveis, como a maior parte dos *door games*, LORD era um jogo textual, o apelo imersivo era grande pois o mundo do jogo era bem construído e possuía uma série de ocorrências aleatórias que faziam com que cada seção de jogo fosse diferente da outra. Uma série de histórias secundárias também mantinha o interesse dos jogadores estimulando a sua imaginação devido à riqueza do mundo *online* apresentado. Na época, as BBSs comerciais começaram a crescer. Tais *BBSs* poderiam disponibilizar dezenas de linhas para seus clientes. Os jogadores também tinham capacidade de se comunicar com outros, fazendo alianças, lutando entre si ou apenas conversando em uma interface parecida em formato de *chat*. LORD foi o primeiro jogo a dar importância a atividades cotidianas de interação virtual entre os jogadores humanos. No ambiente do jogo, por exemplo, o jogador poderia conversar em uma taverna, fazer apostas e, até mesmo, casar.

O sucesso de LORD influenciou o surgimento de outros jogos e trouxe um público diferente para o mundo das BBSs: as pessoas que acessavam as BBSs com o objetivo de

jogar. A expansão da base de usuários das BBSs a partir do fim da década de 90 só é interrompida com a popularização da Internet.

Do fim da década de 80 até o início da segunda metade de década de 90 foi o auge das BBSs, principalmente da plataforma IBM-PC. Alguns jogos de destaque:

- *Solar Realms Elite (SRE)* [PAT2005] - 1990: O objetivo deste jogo de gerenciamento de impérios é o controle de um sistema estelar com inicialmente seis planetas. O jogo enfatizava a colonização de novos planetas, a troca de itens com outros impérios e a conquista através do poder bélico das tropas. Tem como inspiração o *Space Empire Elite*, jogo das BBSs de plataforma Atari ST.
- *Barren Realms Elite* (BRE) – 1990: Jogo semelhante ao SRE, no qual 25 jogadores interagem. É possível o envolvimento de até 255 *BBSs* em um jogo só.
- *Falcon's Eye* – Tem como característica a construção de império no período medieval e é do mesmo autor do *BRE*. Influenciou na inspiração dos jogos por turnos textuais modernos como *Utopia*, que também é do mesmo autor.
- *Usurper*- 1993: *RPG online* com temática medieval. Permite que os jogadores formem times. A característica inovadora deste jogo é a presença de *NPCs* com as mesmas características do jogador e com uma *IA* simples. Dessa forma, o jogo, em uma BBS começava menos vazio para os primeiros jogadores.

Uma funcionalidade interessante de alguns destes jogos como, por exemplo, no *BRE* é a possibilidade de jogos *inter-BBS*. Utilizando uma técnica de envio de mensagens análoga à de redes do tipo FidoNet, era possível organizar campeonatos, nos quais os impérios de uma *BBS* competiam com os de outra.

A Internet provocou a perda de interesse pelas das pessoas pelas *BBSs*. Aos poucos os sistemas foram fechando e os *door games* foram esquecidos. Era difícil para os sistemas com interface essencialmente textual competir com a rica interface gráfica do navegador Web. A principal contribuição dos *door games* foi a popularização dos jogos *online* fora do meio acadêmico e/ou das grandes empresas de jogos.

Outro ponto relevante a se destacar é a possível descentralização do jogo em diversos servidores. Assim, enquanto nos serviços de rede fechada havia um servidor que abrigava o mundo do jogo, nos jogos de *BBS* surgiram jogos com suporte a múltiplos servidores descentralizados, existindo, como já citado, até mesmo competição entre os servidores de *BBSs* diferentes.

A explosão dos jogos para computador

O lançamento do jogo de tiro em primeira pessoa *Doom*, da idSoftware, no fim de dezembro de 1993, pode ser considerado o ponto-chave para a popularização dos jogos para *PC*. E, um dos destaques de *Doom* era a possibilidade de jogar com até 4 adversários via *modem* ou rede. *Warcraft*, jogo de estratégia em tempo real de combate entre impérios, também se destacou neste momento de transição.

A indústria de jogos para computador teve um crescimento extraordinário a partir de 1994 com o surgimento de outros títulos do gênero *FPS* e *RTS* (Real-Time Strategy – estratégia em tempo real). Essenciamente eram clones de *Doom* e *Warcraft*, muitos destes implementando a jogatina *online* através do protocolo *TCP/IP*, utilizado na Internet. *Quake* (1996), também da idSoftware e também um *FPS*, foi um dos primeiros a ter suporte a criação de servidores do jogo via Internet. Após o lançamento, os servidores de *Quake* se espalham pelo mundo. Em algumas noites, mais de 80000 pessoas simultâneas jogam espalhadas pela Internet [MUL1999].

Comparando com o contexto explanado na seção referente a serviços *online* de rede fechada, a nova estrutura popularizou os jogos que permitiam que qualquer provedor com um nó da Internet, ou, até mesmo, jogadores individuais, instalassem um servidor de um jogo. Assim, além dos jogadores não estarem mais limitados a usarem os jogos disponíveis na rede proprietária que acessavam, as pessoas poderiam interagir pela Internet com indivíduos de qualquer parte do mundo com acesso à rede mundial. Por outro lado, a maioria dos jogos *online* da geração de *Doom* e *Warcraft* explorava a jogatina entre grupos pequenos até no máximo 16 pessoas. Jogos com centenas de pessoas simultâneas como *Air Warrior* e *Habitat*, nos quais havia o compartilhamento de **um mesmo mundo *online***, demandavam uma estrutura não adaptada à Internet naquele momento.

Até 1996, enquanto os jogos *online* pela Internet se multiplicavam, os provedores de rede fechada foram se tornando provedores Internet e/ou fechando suas portas. Em 1997 apenas a *AOL*, após comprar a Compuserve no ano anterior, permanecia ativa. Entretanto, seu modelo de negócio foi alterado: ao invés de cobrar pela quantidade de horas de acesso, seguindo o modelo dos provedores Internet, a *AOL* começou a cobrar um valor mensal fixo. Essa mudança quebrou, definitivamente, o modelo de negócios dos provedores de jogos. No modelo anterior eles recebiam uma fração do valor referente a cada hora que o jogador ficasse conectado. Dessa forma, os jogos foram construídos para encorajar às pessoas a ficarem conectadas o maior tempo possível. No modelo novo, maior tempo de conexão a um preço fixo era extremamente prejudicial à *AOL.* A natural conseqüência foi a saída da maioria dos provedores de jogos da *AOL* ou sua adaptação através do pagamento de um valor mensal extra referente ao acesso ao jogo [COS2000]. Essa mudança justifica o porquê da receita de jogos *online* ter diminuído entre 1996 e 1997 quando era esperada uma expansão.

Os jogos *online* de todos os serviços *online* de modelo fechado, portanto, foram desaparecendo por não estarem adequados ao modelo de pagamentos mensais. Alguma iniciativa tinha que ocupar aquele espaço. O surgimento dos jogos massivos *multiplayer,* em 1996, marcou o início de uma nova geração. Tal geração, iniciada pelos jogos *Meridian 59* e *Ultima Online*, além de atender a necessidade de mundos *online* persistentes com muitas pessoas interagindo, suplantou todas as escalas das gerações anteriores em tamanho e complexidade.

A era moderna dos jogos *online*: os jogos massivos *multiplayer*

Embora todos os elementos caracterizadores de um jogo *online* massivo *multiplayer* já existissem desde a década de 80 (por exemplo, em *Habitat*), os jogos estavam limitados a redes fechadas e atendiam ao modelo de negócio de pagamento pelo tempo. O primeiro jogo massivo *multiplayer* acessível pela Internet com um mundo persistente representado de forma gráfica foi *Meridian 59*. Lançado em 1996, existem controvérsias se tal jogo realmente pode ser considerado como um jogo massivo [KEN2003]. A definição de massivo não é clara, mas *Meridian 59* permitia apenas 250 pessoas em cada servidor, valor muito distante do comum para os jogos posteriores. Entretanto, certamente foi o primeiro jogo *online* gráfico baseado em *Internet* com modelo de mensalidade. O jogo, de ambientação medieval, funcionava como

um *MUD* gráfico com visão em primeira pessoa 2.5 D, ou seja, uma "falsa sensação" de 3 dimensões como no jogo *Doom*. Não se obteve grande sucesso comercial: o jogo limitou-se a 12000 pessoas.

Ultima Online foi o sucesso que realmente iniciou a era moderna dos jogos *online*. Segundos os dados de pesquisa realizada por Woodcock [WOO2005], sobre a quantidade de assinantes em jogos *online* massivos, em pouco mais de um ano alcançou a marca de 100000 jogadores pagantes. Seu pico de pagantes foi no primeiro semestre de 2003, quando havia 250.000 assinantes. Oito anos após seu lançamento manteve uma base de mais de 150000 usuários.

Ultima é um *RPG online* medieval com visão *top-down* em terceira pessoa. Tal visão, embora tenha sido utilizada pelo jogo *Diablo*, anteriormente, foi uma das principais inovações e foi aplicada na maioria dos jogos subseqüentes. Com Ultima as portas foram abertas para o lançamento de diversos outros jogos *online* massivos. A sigla cunhada para classificá-los, a partir de *Ultima*, é *MMOG*.

Outro jogo *online* surgido em 1999 foi *Everquest*. Diferente do mundo totalmente medieval de *Ultima*, *Everquest* é um mundo fantástico repleto de magia e raças diferentes. *Everquest* mantém uma média de 400.000 assinantes. Outro mundo medieval pioneiro foi *Asheron's Call*, criado pela *Microsoft* em 1999. Teve seu pico de assinantes em janeiro de 2002 com cerca de 120.000 assinantes. Em sua pesquisa, Woodcock [WOO2005] considera a era de ouro dos jogos *online* o período até abril de 2001 quando os jogos citados tiveram grande crescimento e inspiraram outras iniciativas. De maio de 2001 a abril de 2002 um período de transição indicou a saturação do mercado dos *MMORPGs* de fantasia. Nesse período o número de assinantes diminuiu e se espalhou em diversos sistemas. Muitos jogos foram cancelados, conduzindo a uma fase caracterizada pela competição.

O mercado continua a crescer com mais pessoas interessadas em jogar *online*. Estima-se que de Outubro de 2004 a Janeiro de 2004, nos Estados Unidos, cerca de 1 milhão de pessoas começaram a assinar algum jogo. Entretanto, o processo de canibalização dos assinantes é característico da atualidade. Dessa forma, os novos jogos roubam a atenção e os assinantes dos anteriores em um processo de renovação contínua. Como a infra-estrutura para manter um *MMOG* é cara, a perda de assinantes implica em prejuízo. Nesse cenário, é difícil acreditar que os jogos *online* da geração iniciada por *Ultima Online* tenham duração tão longa

quanto os da década de 80. Os ambientes estão mais descartáveis e o jogador tem inúmeras opções de mundos para compartilhar. Só na Coréia, por exemplo, existem centenas de jogos massivos *online*.

Outro ponto a se destacar é a temática comum à maioria dos jogos de sucesso. Embora existam ambientes como *Matrix Online* (mundo da franquia *Matrix*), *Eve Online* (ficção científica), *City Of Heroes* (cidade com super-heróis e vilões), *The Sims Online* (extensão *online* ao jogo *The Sims*) e *Star Wars Galaxies* (universo da franquia *Star Wars*), mais de 80% dos *MMOGs* são *RPGs* de fantasia. *World of Warcraft (WoW)* e *Everquest II*, lançados no segundo semestre de 2004, despontam como os principais responsáveis pelo crescimento do mercado e canibalização dos assinantes dos outros jogos. Ambos também contribuíram com a manutenção da proporção destinada ao gênero dos *RPGs* de fantasia. *WoW*, inclusive, é o MMORPG com crescimento mais rápido na base de usuários: no primeiro dia, 200,000 jogadores criaram contas no sistema, em dois meses mais de 700,000 unidades do jogo já haviam sido vendidas nos *EUA*, Austrália e Nova Zelândia. Além disso, em poucos meses o jogo também quebrou o recorde de número de usuários concorrentes nos *EUA*, com mais de 200.000 pessoas simultâneas jogando. O jogo também foi disponibilizado em fevereiro de 2005 para o mercado europeu e também alcançou recordes expressivos com 180000 usuários concorrentes, distribuídos em 80 servidores. Seu sucesso em todas as regiões é justificado pelo seu apelo tanto para o jogador casual quanto para o *harcore*.

Até agora, apenas o mercado americano de jogos *online* foi contextualizado com destaque. Deve-se discernir, porém, que o maior mercado para jogos *online* do mundo é a Coréia do Sul. Alguns fatores políticos e culturais explicam tal situação [KEN2003]:

- *Banimento dos produtos japoneses*: devido a situação gerada após a segunda guerra mundial, haviam leis banindo a importação de produtos japoneses. Dessa forma, era impossível ao coreano ter acesso aos principais consoles de vídeo-game, já que estes eram fabricados por empresas japonesas.
- *Densidade populacional*: a Coréia é um dos países mais densos populacionalmente. Um computador além de caro, também ocupa um espaço não desejável na residência. Isso justifica a enorme quantidade de centros com computadores disponíveis à população (chamados de *PC-Baangs*), um modelo muito semelhante às *lan-houses* brasileiras. Tais locais são centros de lazer e

de interação social entre os coreanos, ou seja, estimulam a criação de comunidades.

- *Incentivo do governo coreano*: o governo coreano investiu em serviços de banda larga e telefonia. Muitos jogos tem acesso estimulado pelas operadoras de telefonia.
- *Modelo facilitado de tarifação*: os americanos estavam acostumados a pagar utilizando cartão de crédito. Na Coréia um jogo *online* pode ser tarifado através de desconto direto na conta corrente ou conta telefônica. Além disso, o coreano está acostumado a pagar diretamente pelo conteúdo, realizando micro-pagamentos, por exemplo, se o jogador deseja jogar 10 partidas de um certo jogo ele paga por estas 10 partidas e não uma mensalidade.

Neste cenário, os jogos *online* coreanos são jogos que estimulam a interação em grupos. Além disso, são jogos ideais para jogar em *lan-houses* na presença de amigos. O primeiro jogo coreano de destaque é *Lineage*. Lançado pela *NCSoft* em 1997, chega a ter mais de 3.000.000 de assinantes em meados de 2003. Recorda-se, porém: o modelo coreano não se baseia em assinaturas. Pelo menos 50% dos jogadores vão aos *PC-Baangs* e pagam pelo número de horas jogam, efetivamente, o jogo.

Os jogos de peso no mercado coreano também são essencialmente com temática de fantasia medieval seguindo a tendência iniciada no mercado americano. Dessa forma, os coreanos também têm a possibilidade de exportar seus jogos. Além disso, por se focarem mais no aspecto social do jogo, tem possibilidade de aceitação em países diferentes dos focalizados pelos americanos. *Ragnarok Online*, por exemplo, já tem servidores nas Filipinas, Japão, Taiwan e, foi o primeiro *MMORPG* importado a ser disponibilizado para o mercado brasileiro. Com 250.000 em 2005 e mais de um milhão de jogadores cadastrados é o jogo *online* mais bem sucedido no Brasil. Deve-se destacar, porém, que o primeiro *MMORPG* brasileiro é *Erinia* lançado em 2004 por uma empresa originalmente curitibana.

Uma das principais contribuições desta geração é que o modelo de negócio. O foco passou a ser no serviço e não no produto. Ou seja, a manutenção da comunidade tem papel chave. Técnicas de *IA* podem ser facilitadores para tal manutenção. A *IA* não serve para substituir o jogador, mas sim para melhorar a experiência.

ANEXO 4 – RANKING DAS RODADAS

Este anexo apresenta o ranking obtido para as rodadas nas quais os jogadores humanos e automáticos interagiram.

Resultado da Segunda Rodada (Jan/Fev 2007)

Colocação	Tipo	Networth
1	Humano	72591982
2	Humano	19297506
3	Humano	18655559
4	Jogador Automático 6	15483631
5	Humano	15411470
6	Jogador Automático 10	15142742
7	Jogador Automático 5	13495338
8	Jogador Automático 11	13028227
9	Jogador Automático 3	12443893
10	Jogador Automático 12	12379579
11	Jogador Automático 4	12078996
12	Jogador Automático 1	11764931
13	Humano	11382933
14	Jogador Automático 2	10403998
15	Humano	10389575
16	Jogador Automático 9	7969393
17	Jogador Automático 8	6749882
18	Humano	6735699
19	Humano	6587604
20	Jogador Automático 7	6568127

Resultados da Terceira Rodada (Fev/2007)

Colocação	Tipo	Networth
1	Humano	135072907
2	Humano	25354862
3	Humano	19060877
4	Humano	16831230
5	Humano	12424323
6	Humano	11749395
7	Humano	11192230
8	Humano	10926308
9	Jogador Automático 1	10835310
10	Humano	10059101
11	Humano	9969258
12	Humano	9887671
13	Jogador Automático 2	9853419
14	Humano	9679473
15	Jogador Automático 6	8979769
16	Humano	8386124
17	Jogador Automático 4	7955111
18	Jogador Automático 3	7256574
19	Jogador Automático	6350844
20	Humano	6296287
21	Humano	6038672

22	**Humano**	**5859799**
23	**Jogador Automático 7**	**5714609**
24	**Humano**	**4305934**
25	**Humano**	**4209886**
26	**Jogador Automático 5**	**4047388**
27	**Humano**	**3806810**
28	**Humano**	**3547342**
29	**Humano**	**3382718**
30	**Humano**	**3066850**
31	**Humano**	**2761314**