ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

VIDYA

Um Jogo de Estratégia com Intuição e Inteligência Artificial

Trabalho de Conclusão de Curso

Engenharia da Computação

Marcelo Rodrigo de Souza Pita
Orientador: Prof. Fernando Buarque de Lima Neto, PhD

Recife, 20 de novembro de 2006

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

VIDYA

Um Jogo de Estratégia com Intuição e Inteligência Artificial

Trabalho de Conclusão de Curso

Engenharia da Computação

Este Projeto é apresentado como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Marcelo Rodrigo de Souza Pita
Orientador: Prof. Fernando Buarque de Lima Neto, PhD

Recife, 20 de novembro de 2006

UNIVERSIDADE
DE PERNAMBUCO

Marcelo Rodrigo de Souza Pita

VIDYA

Um Jogo de Estratégia com Intuição e Inteligência Artificial

Resumo

Vidya é um jogo para computador de estratégia e simulação, *single-player*, de gênero *God Game*, que convida o jogador a ser o *Deva* (deus) de um clã de personagens sociais inteligentes e autônomos, chamados *Jivas*, para os quais o jogador deverá fornecer instruções não detalhadas adequadas para a sobrevivência deles.

Diferentemente de outros jogos, nos quais o jogador tem controle direto sobre os personagens, Vidya oferece um modelo de interação jogador-personagem inovador, através do qual o jogador não os controla diretamente, mas fornece somente instruções, ou melhor, intuições, de boas ações, chamadas *vidyas*. O objetivo é ajudar os *Jivas* em seus aprendizados e sobrevivência, contudo, eles têm a liberdade para escolher seguir ou não as sugestões dos *devas*.

Além de um jogo, Vidya pode ser visto como um ambiente onde seres artificiais competem entre si por recursos naturais disponíveis no mundo do jogo, que forma um ecossistema com interessante dinâmica. Além disso, os *Jivas* são seres cujos comportamentos sociais estão baseados em relações de coação. A observação de comportamentos sociais emergentes, dinâmica de populações e equilíbrio de ecossistemas são estudos bastante pertinentes que poderão surgir a partir da análise dos *Jivas*, com possíveis aplicações futuras em problemas reais.

Este trabalho apresenta o jogo Vidya, e tem o foco no desenvolvimento do algoritmo de comportamento inteligente do *Jiva* como indivíduo social, que foi modelado e implementado através de Computação Evolutiva. Esse mesmo algoritmo foi construído de tal forma que poderá ser reusado em outros cenários de jogos, possivelmente em personagens (controlados por computador ou não) que necessitem de algumas das características encontradas nos *Jivas*. Portanto, apresentamos o trabalho principalmente como uma contribuição na área de aplicações de Inteligência Artificial para jogos de computador, podendo no futuro ser usado como uma plataforma para estudos de dinâmica de populações.

Abstract

Vidya is a strategy & simulation computer game, single-player, god-style, that invites the player to be the clan's deva (*i.e.* god) of autonomous intelligent social characters, called *Jivas*, for which the player will not supply detailed suitable instructions for its survival.

Differently for other computer games, in which the player has direct control over characters, Vidya proposes an innovative model of player-character interaction, in which the player does not control characters directly, but only supply instructions, or better saying, intuitions of good actions, called *vidyas.* Although these intuitions help *Jivas* on learning and surviving, they have the freedom to follow them or not.

More than a simple game, Vidya can be seen as an environment where artificial beings compete among themselves for natural resources available in the game World, comprising an ecosystem with interesting dynamics. Moreover, *Jivas* are beings whose social behaviors are based in coaction relationships. Observation of emergent social behaviors, populations dynamics and ecosystems *equilibrium* are relevant studies that can arise from *Jivas*' analysis, with possible future applications in real problems.

This work presents the Vidya game and focus on the development of the *Jiva*'s intelligent behavior algorithm as a social individual. This was modeled and implemented through Evolutionary Computation. This algorithm can also be reused in other game scenarios, possibly in characters (computer-controlled or not) that need some of the features found in the *Jivas*. Hence, this work´s primarily contribution is in applications of Artificial Intelligent algorithms to computer games area, as well in the future, as a test bed for population dynamics.

Sumário

Índice de Figuras

Índice de Equações e Tabelas

Tabela de Símbolos e Siglas

(Dispostos por ordem de aparição no texto)

IA – Inteligência Artificial
PC – *Personal Computer*
IEEE – *Institute of Electrical and Electronics Engineers*
MUD – *Multi User Dungeon*
PDA – *Personal Digital Assistant*
S&A – *Skill-and-Action*
FPS – *First-Person Shooter*
RPG – *Role Playing Game*
MMORPG – *Massively Multiplayer Online Role Playing Game*
UCP – Unidade Central de Processamento
FSM – *Finite-States Machine*
FuSM – *Fuzzy-States Machine*
AG – Algoritmos Genéticos
RNA – Redes Neurais Artificiais
ALife – *Artificial Life*
API – *Application Programming Interface*
OO – *Object-Oriented*
PS – Tamanho da população de indivíduos do AG
F – Número de passos no futuro que a função de aptidão do AG consegue abarcar
CV – Condição vital do *Jiva*
V – Vitalidade do *Jiva*
H – Hidratação do *Jiva*
E – Energia do *Jiva*
AMD – *Advanced Micro Devices*, empresa americana fabricante de circuitos integrados
J2ME – *Java 2 Micro Edition*
MMMOG – *Móbile Massively Multiplayer Online Game*
C.E.S.A.R. – Centro de Estudos e Sistemas Avançados do Recife

Notas do Autor

Os nomes usados no jogo Vidya são retirados do Sânscrito. O Sânscrito é uma linguagem clássica da Índia, uma linguagem litúrgica de religiões como o Budismo, Hinduísmo e Jainismo e uma das 22 linguagens oficiais da Índia. O caráter místico-religioso desta linguagem foi levado em consideração na representação dos nomes em Vidya, já que em Vidya temos elementos como o deus (*deva*), a intuição (*vidya*) e a entidade viva (*Jiva*).

Os textos dentro do jogo para comunicação com o jogador foram escritos em inglês, pela popularização dessa linguagem mundialmente e especialmente pela intenção que se tem em gerar publicações internacionais com o jogo Vidya.

O autor possui experiência no desenvolvimento de jogos para arquiteturas *mobile* com J2ME [1]. Trabalhou no Projeto MMMOG [2], desenvolvido pelo C.E.S.A.R. [3] em parceria com a Meantime [4], na construção de uma plataforma para jogos móveis massivamente multiusuário. Desenvolveu interesse pela aplicação de IA em jogos eletrônicos e *serious games* para simulação de ecossistemas e sociedades, do qual surgiu Vidya como seu primeiro jogo de grande porte agrupando as citadas áreas de pesquisa.

Agradecimentos

Agradeço não ao deus monótono e inoperante que se costuma referenciar, mas ao Deus vivo, descontínuo e perpetuamente operante, o qual conhecemos como Criatividade; essa mesma Criatividade que permite revoluções positivas na Ciência, Artes e Filosofia. Agradeço porque sempre fui beneficiado.

À Cida, minha querida esposa e amiga, sem a qual esta construção não iniciaria, ou ao menos desabaria. A paciência, o estímulo para continuar, o suportar ausências e ocupações diversas minhas e o superar junto momentos difíceis em todos os sentidos são apenas algumas instâncias de suas qualidades expressas. Todo agradecimento aqui será pouco a esta maravilhosa pessoa.

Meus sinceros agradecimentos ao Professor Fernando Buarque, o qual sempre considerarei suas instrutivas e inspiradoras orientações. Além de um exemplo como profissional da Ciência e da Educação, Buarque é uma pessoa excepcionalmente boa, preocupada com o bem-estar de todos e com o desenvolvimento de seu país. Particularmente, é alguém a quem considero como amigo, pelas suas incontáveis demonstrações de benevolência, não somente na Academia. Uma generosa parte do presente trabalho e de minhas futuras conquistas compartilho com ele.

Não poderia deixar de incluir um agradecimento a meus pais, Márcio Pita e Carmem Lúcia Pita, e meu falecido avô, Mário Paulino de Souza, que mesmo que não tenham tido as mesmas oportunidades que eu, me conduziram pelo bom caminho durante minha infância e adolescência. A Gustavo Paiva, pelo auxílio prestado e boa-vontade em contribuir com meus estudos universitários durante os últimos três anos. A Salomão Sampaio, graduando do Curso de Engenharia da Computação da UPE, pela sua contribuição em técnicas de Inteligência Artificial.

Muito obrigado a todos os Professores que fazem o Departamento de Sistemas Computacionais da Escola Politécnica da Universidade de Pernambuco, pela sua dedicação em construir um Curso de alta qualidade, continuamente preocupados com a vivência e aprendizado de seus Alunos dentro (e não raro fora) da Universidade.

Marcelo Pita

Capítulo 1

Introdução

O presente trabalho consiste no desenvolvimento de um jogo para computador chamado Vidya. Mais que um jogo, Vidya é um ambiente virtual onde é possível a observação de uma sociedade de seres virtuais artificialmente inteligentes, chamados *Jivas*, que disputam entre si os recursos naturais disponíveis, e também disputam esses recuros com outros seres que habitam o mundo.

O jogo traz duas perspectivas aparentemente antagônicas quanto aos *Jivas*: uma centrada em suas psicologias individuais e a outra centrada na sociologia de suas relações. Esse aparente antagonismo se mostra presente no jogo através dos conflitos existentes entre as ações individuais de cada *Jiva* e os comportamentos emergentes da sociedade que eles formam. É um antagonismo aparente, pois os próprios comportamentos sociais emergem das ações de cada *Jiva*. A causa do conflito está enraizada na necessidade que os *Jivas* têm de obter recursos naturais (às vezes escassos) para sua sobrevivência.

Diferentemente de outros jogos, em Vidya o jogador não poderá influir diretamente nas ações dos personagens (os *Jivas*), mas sim lhes fornecer a *vidya*, um *input* que cada *Jiva* recebe do jogador e que ele entende como uma percepção interna, ou intuição. O *Jiva* tem "livre-arbítrio" para seguir ou não a *vidya*, e opta por quais decisões tomar baseado em seu aprendizado adquirido no Mundo no decorrer do tempo.

Jogos para computador são, atualmente, a classe de aplicações que recebem maior investimento, possuindo um mercado mundial na faixa das dezenas de bilhões de dólares [1]. Além disso, são *software* que reúnem em si componentes de alta tecnologia, promovendo alta capacitação técnica para os envolvidos em suas implementações.

Com o emprego de técnicas de IA em jogos, tornou-se possível construir estruturas e personagens de jogos com poder de ação e reação bastante elaborados. IA em jogos já é um subárea bastante explorada em Computação, sendo um dos principais fatores a serem analisados no desenvolvimento de qualquer jogo eletrônico [6], [7].

Apesar da vasta aplicação de IA em jogos e de todos os progressos provenientes dessa aplicação, vê-se na maioria dos jogos sistemas agentes especialistas previamente treinados para

realizar uma determinada tarefa. Ainda que essas sejam tarefas inteligentes, são atividades de inteligência estática, cujo grau adquirido e requerido de adaptação às mudanças ambientais não evoluem com o tempo. Ou seja, as fases de treinamento e aplicação são muito bem separadas.

O foco principal deste trabalho está na aplicação de técnicas de IA em jogos, em especial Computação Evolutiva, mostrando ser possível implementar com essa técnica tal ser inteligente, o *Jiva*. *Jivas* são seres cujas inteligências são completamente modeladas, no sentido de que sempre tomam decisões próprias racionalmente. O diferencial aqui está na capacidade que eles têm de aprender com suas decisões, para que tomem decisões mais bem elaboradas futuramente de forma não-supervisionada (*i.e.* autônoma). Em outras palavras, o *Jiva* é um ser evolutivo. O mesmo módulo de inteligência do *Jiva* poderá ser adaptado para futuros jogos, em personagens que precisem desse grau de liberdade e com estas características evolutivas.

Alguns jogos, à parte de seus aspectos lúdicos, serviram como palco para testes de teorias em diversas áreas do conhecimento humano. Jogos desse tipo são conhecidos como *Serious Games* [8]. Vidya poderá servir como um ambiente para observação de comportamentos psicológicos individuais e sociais de seres artificialmente inteligentes, com possíveis aplicações práticas para as áreas de Psicologia, Sociologia e Antropologia. Contudo, esses aspectos foram deixados como extensões futuras deste trabalho.

1.1 Motivações

A principal motivação deste trabalho foi a necessidade crescente, em termos qualitativos, da aplicação de IA em jogos eletrônicos. Com a disseminação de placas aceleradoras gráficas em PCs, os jogos eletrônicos estão se tornando graficamente muito próximos do realismo. Disto surgem duas conseqüências inevitáveis: o processador está mais disponível para execução de IA; para acompanhar a evolução gráfica, os jogos demandam Mundos e objetos com comportamentos mais inteligentes, coerentes com o aspecto visual [9], [10], [11]. Dessa forma, há uma demanda crescente de IA em jogos eletrônicos.

Uma motivação adicional foi a atraente e promissora tentativa de criação de um ambiente simulador propício para a observação de dinâmica de populações, equilíbrio em ecossistemas e observação de comportamento sociais emergentes (*e.g.* ética social).

1.2 Objetivos e Metas

A meta do Projeto que este documento apresenta foi construir um *software*, que é o jogo para computador chamado *Vidya – um jogo de estratégia que inclui intuição e inteligência artificial*. Para conseguir essa meta, os seguintes objetivos específicos foram estabelecidos:

- Modelar e construir o mundo do jogo e seus objetos;

- Desenvolver o motor do jogo, através do qual são captados os *inputs* do jogador pelo teclado e *mouse*; o estado do Mundo é atualizado e objetos são representados graficamente;
- Projetar e desenvolver a IA dos *Jivas*;
- Publicar resultados[1].

1.3 Organização do Documento

1.3.1 Capítulo 2: Conhecimentos Básicos

Neste Capítulo, revisamos as áreas do conhecimento científico úteis para a compreensão de Vidya, incluindo uma Subseção sobre Jogos para Computador e outra explicando as técnicas de IA usadas na modelagem do comportamento inteligente do *Jiva*.

1.3.2 Capítulo 3: Vidya

Neste Capítulo, descrevemos o jogo Vidya. A descrição inclui aspectos de Engenharia de Software, o Mundo do jogo com seus objetos num alto nível, o módulo de decisão inteligente do *Jiva* e um novo modelo de interação jogador-personagem, exibindo o funcionamento do jogo. Uma ênfase maior é dada na aplicação de IA que está centrada na implementação do *Jiva*, principal meta deste trabalho.

1.3.3 Capítulo 4: Experimentos e Resultados

Este Capítulo documenta os experimentos realizados para a validação do jogo Vidya. Os dois primeiros experimentos são para a análise de desempenho computacional e inteligente do algoritmo que modela o comportamento do *Jiva*. O terceiro e último experimento é uma avaliação do jogo feita por usuários, analisando um conjunto de parâmetros.

1.3.4 Capítulo 5: Conclusões e Trabalhos Futuros

Neste Capítulo, revisamos e comentamos as contribuições, os resultados e os impactos do trabalho. Uma lista de futuras contribuições é deixada.

---

[1] Vidya gerou um artigo entitulado "*Vidya: a God Game Based on Intelligent Agents Whose Actions are Devised Through Evoltionary Computation*", submetido ao *IEEE Symposium on Computational Intelligence and Games 2007* (CIG 2007), que é parte do *IEEE Symposia Series in Computational Intelligence 2007* (IEEE SSCI 2007), promovido pela *IEEE Computational Intelligence Society*. Neste momento o artigo está em análise pelo Comitê Técnico da Conferência. A versão completa do artigo submetido está no Anexo C do presente documento.

Capítulo 2

Conhecimentos Básicos

2.1 Jogos

Não há um concenso sobre o que é um jogo. Matemáticos, economistas, políticos, cientistas da computação, *game designers*, jogadores de computador, sociólogos, enfim, todas estas classes de pessoas, interpretam a palavra "jogo" de maneiras diferentes. O conceito de Jogo é muito geral, aplicado em muitas áreas do conhecimento humano, desde simples brincadeiras infantis até complexos projetos governamentais e econômicos. Podemos classificar os conceitos de jogos dentro de dois tipos: conceitos ligados à diversão e conceitos ligados à economia.

Os conceitos ligados à diversão associam jogos a um processo lúdico em que um ou mais participantes (jogadores) fazem parte de um processo competitivo (*e.g.* jogo de xadrez), cooperativo (*e.g.* futebol, entre os participantes de um mesmo time) ou aleatório (*e.g.* jogos de chance, como roleta) para superar um desafio (que pode incluir vencer um jogador oponente) ou resolver um problema. Dentro desse contexto, acrescentamos a definição dada por Wolfgan Krammer [12]:

> "Jogos são objetos compostos por componentes e regras e que têm certos critérios: regras; um objetivo; curso sempre variante; chance; competição; experiência comum; igualdade; liberdade; atividade; imersão no mundo do jogo; e ausência de impacto na realidade."

Os conceitos ligados ao comportamento econômico estão mais associados ao fluxo e gestão de recursos dentro de um sistema (*e.g.* um ecossistema), com possíveis pontos de equilíbrio econômico. A abordagem econômica, que envolve o estudo de tomadas de decisão para, em geral, maximizar os ganhos e minimizar as perdas, é especialmente estudada dentro da Teoria dos Jogos, criada por Neumann e Morgenstern [13], posteriormente ampliada por Nash [14]. Muitas são as áreas em que a Teoria dos Jogos pode ser aplicada, por exemplo: Economia, Política, Gestão empresarial, Dinâmica de ecossistemas e sociedades, etc.

O conceito de jogo, dentro do aspecto lúdico, é independente da mídia em que o mesmo se manifesta. Há jogos de tabuleiro (*i.e.* que são manifestados num tabuleiro), jogos de carta, jogos de dados, etc. Dentre os jogos que são manifestados em meio eletrônico, distingüimos os jogos para computador.

2.1.1 Jogos para Computador

Um jogo para computador é um jogo manifestado em mídia eletrônica e executado em um PC (*i.e. Personal Computer*). Jogadores interagem com o jogo comumente através de um monitor (saída), onde são exibidos os elementos gráficos e através de periféricos de entrada como teclado, *joystick* e *mouse*. Os elementos gráficos exibidos variam desde simples textos, como é o caso dos MUDs (*Multi-User Dungeon or Domain or Dimension*), até modelos tridimensionais de objetos com alto grau de realismo. A Figura 1 exibe alguns exemplos de jogos para computador com componentes gráficos muito realistas.

O conceito de *video game* é mais geral e se refere não somente a jogos para computador, mas também a jogos que são executados em outros tipos de dispositivos eletrônicos que não sejam PCs, como os *consoles* (dispositivos dedicados para a execução de jogos, *e.g.* PlayStation [15], XBox [16], Game Boy [17], etc), *arcades* (máquinas dedicadas que vêm, cada uma, com um

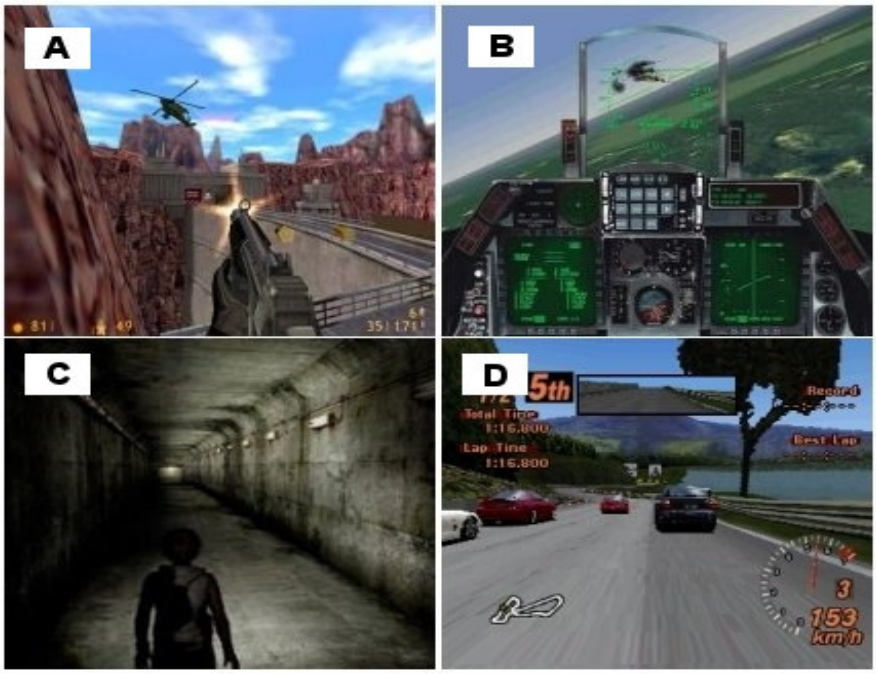


Figura 1. Jogos para computador recentes, exibindo o alto grau de realismo gráfico: **(*a*)** Half-Life [18], um jogo famoso do gênero *First-Person Shooter* (FPS); **(*b*)** Falcon 4.0 [19], um jogo simulador de combate aéreo; **(*c*)** Silent Hill [20], um jogo de terror; **(*d*)** Gran Turismo 2 [21], um jogo de corrida para o *console* PlayStation, da Sony.

único jogo, comuns nos *playtimes*) e dispositivos portáteis (*e.g.* laptops, telefones celulares, PDAs (*Personal Digital Assistants*), calculadoras avançadas, etc).

A classificação de jogos para computador em gêneros é muito imprecisa, particularmente porque os jogos para computador, em geral, não se encaixam somente em um único gênero, mas em vários, ainda que propensos para um deles. O jogo Vidya, por exemplo, pode ser classificado dentro de muitos gêneros, com uma forte propensão para os chamados *God Games*. A seguir exibiremos uma taxonomia de jogos para computador proposta por Chris Crawford [22], famoso *game designer*[2], com alguns acréscimos para maior completude. Existem outras taxonomias, mas essa foi a mais apropriada para nosso trabalho.

2.1.2 Uma Taxonomia de Jogos para Computador

Uma bem elaborada taxonomia de jogos para computador pode se tornar uma ferramenta fundamental de apoio à criatividade para equipes de desenvolvimento de jogos, e em particular para o *game designer*, na exploração de novos estilos e combinações de estilos de jogos. A proposta de uma taxonomia de jogos para computador não é catalogar detalhadamente cada tipo e subtipo de jogo, mas sim fornecer uma organização em gêneros, ainda que imperfeita, para um grande número de objetos relacionados, no caso, jogos para computador.

A taxonomia que iremos apresentar divide os jogos em três grandes grupos – jogos de S&A (*i.e. Skill-and-Action games*), jogos de Estratégia e jogos de Simulação. Cada grupo possui muitos gêneros. Os gêneros de jogos que se encaixam no grupo S&A enfatizam o exercício das habilidades perceptivas e motoras do jogador. Os gêneros que são considerados do grupo de Estratégia dão maior ênfase no exercício das suas habilidades cognitivas. Os gêneros de jogos de Simulação tentam simular uma experiência do mundo real, levando em consideração as suas limitações físicas, inclusive, freqüentemente requerendo do jogador habilidades técnicas específicas, como é o caso de jogos simuladores de vôo, por exemplo.

Jogos de S&A

As pessoas associam a maioria dos jogos com jogos de S&A. Isso porque esse é o grupo em que está a maioria dos jogos. Para se ter idéia, todos os jogos de *arcade* são jogos de S&A e quase todos os jogos do *console* ATARI 2600 [23] são jogos de S&A. Esse grupo possui jogos que são caracterizados por ação em tempo-real, forte ênsfase em efeitos gráficos e sonoros, exigindo do jogador uma boa coordenação perceptiva e tempo de reação rápido. Estão agrupados aqui em seis gêneros: Combate, Labirinto, Esportivos, *Paddle*, Corrida e miscelâneas dos gêneros anteriores.

---

[2] Os *game designers* são, dentro de um grupo de desenvolvimento de jogos para computador, os projetistas de jogo. Eles são responsáveis por grande parte do trabalho para elaboração e desenvolvimento de um jogo, e em especial na criação do documento de *Game Design*, que contém toda a especificação em alto nível do jogo, além de detalhes de arquitetura alvo e *gameplay* (formas de interação jogador-jogo).

- Jogos de Combate

Jogos de combate apresentam confronto direto e, em geral, violento. Os combatentes podem ser jogadores ou personagens controlados por computador. São os tipos de jogos mais comuns, que na década de 80 garantiram a popularidade dos *video games*. Atualmente, existe uma variedade muito grande de jogos de combate, desde os baseados no modelo do Space Invaders (Figura 2.a) aos FPS, como o Half-Life (Figura 2.b).

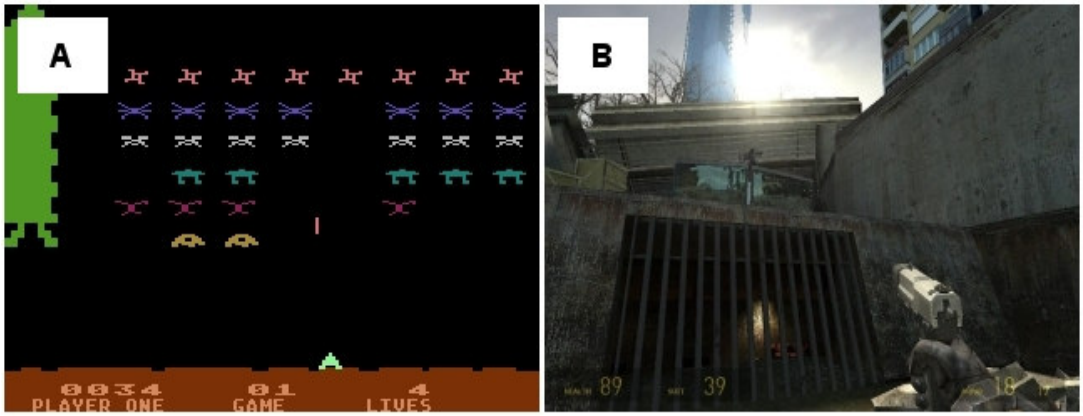


Figura 2. Exemplos de jogos de combate: **(*a*)** Space Invaders; **(*b*)** Half-Life.

- Jogos de Labirinto

Os jogos de labirinto são caracterizados por possuírem um labirinto de caminhos através dos quais o jogador deverá percorrer. O caso de maior sucesso é o PAC-MAN, do qual a maioria dos jogos de labirinto derivam. A Figura 3 é um *screenshot* do jogo PAC-MAN.

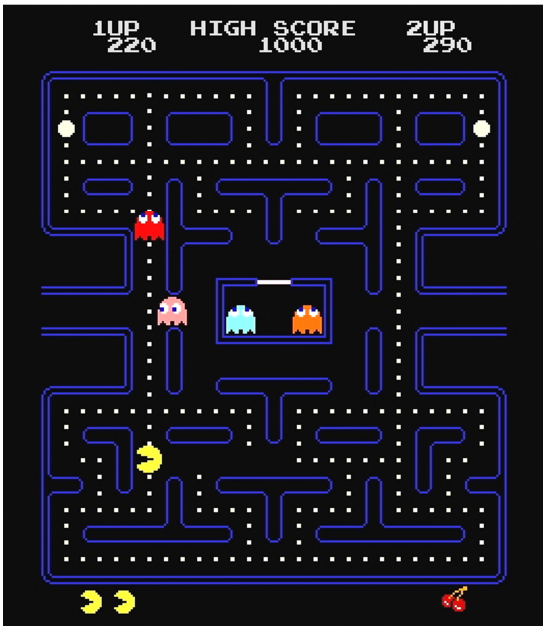


Figura 3. Captura de tela do jogo PAC-MAN.

- Jogos Esportivos

Jogos esportivos para computador tentam representar jogos de esportes populares da vida real no computador. São muitos os jogos baseados em tênis, futebol, boxe, basquete e outros. Esses tipos de jogos tentam alcançar o mercado dos consumidores que, em geral, são conservadores, e já estão habituados com algum esporte em particular na vida real. Por isso, este gênero de jogo é muito popular. Alguns deles dão ênfase na atividade do esporte em si mesmo, enquanto outros enfatizam os aspectos de estratégia do esporte. A Figura 4 exibe exemplos de jogos de esportes.

Figura 4. Exemplos de jogos de esporte recentes: **(*a*)** Pro Evolution Soccer 4 [24], um jogo de futebol; **(*b*)** NBA Jam [25], um jogo de basquete.

- Paddle Games

Paddle games são jogos para computador que são baseados no clássico Pong. Em Pong (Figura 5.a), o jogador deve interceptar um projétil com um aparato, de forma a rebatê-lo, similar ao ping-pong. Outras variedades incluem o jogo Breakout, que é um jogo mono-usuário onde o objetivo do jogador é rebater com um aparato uma bola, de forma a quebrar blocos na parte superior da tela (Figura 5.b).

Figura 5. Exemplos de *paddle games*: **(*a*)** Pong; **(*b*)** Breakout.

- Jogos de Corrida

São jogos que envolvem corrida. Na maioria desses jogos, o jogador que primeiro chegar num destino é o vencedor, como ocorre em corridas tradicionais do mundo real. Outros penalizam jogadores por baterem em obstáculos que são colocados no caminho de propósito. A Figura 1.d exibe um famoso jogo de corrida de automóveis.

- Miscelâneas de jogos de S&A

Há muitos jogos de S&A que não se encaixam exatamente num dos gêneros citados anteriormente, mas sim vários deles ao mesmo tempo. Exemplos desses jogos são o Donkey Kong, da Nintendo, que lembra vagamente um jogo de corrida com obstáculos inteligentes, e Apple Panic, que é uma mistura de jogo de labirinto com elementos de jogos de combate.

Jogos de Estratégia

Jogos de estratégia exploram mais cognição do que manipulação. Isso não exclui dos jogos de S&A alguns elementos de estratégia, que estão presentes em quases todos os jogos. Mas, diferentemente dos jogos de S&A, jogos de estratégia não exploram muito habilidades motoras do jogador e, na grande maioria dos exemplos, não são de tempo-real.

Os jogos de estratégia são também mais longos e quase que exlusivamente restritos a computadores pessoais e estão agrupados aqui em cinco gêneros: Aventura, RPGs, Guerra, Educacionais e Interpessoais.

- Jogos de Aventura

Nesses jogos, o jogador deve se mover num mundo complexo, acumular ferramentas e superar adequadamente cada obstáculo a fim de chegar a um objetivo final.

- RPGs

Os jogos de RPG (*i.e. Role-Playing Game*) para computador têm a intenção de imergir o jogador num universo de fantasia ou ficção científica, comumente baseado num história prévia a partir da qual a trama do jogo se desenvolve. O *avatar*[3] (ou grupo de *avataras*) possui um conjunto de atributos, os quais o caracterizam e são altamente influenciados pelas preferências do jogador. Também é comum tais jogos darem opção ao jogador de selecionar classes de personagem para seu *avatar*.

Ainda em RPGs destacamos um subconjunto desse gênero de jogos para computador designado por MMORPG (*i.e. Massively Multiplayer Online Role-Playing Game*). Trata-se de RPGs que conectam milhares de jogadores através da Internet, cujo campo de interação é um mundo virtual

[3] Um *avatar* é o representante do jogador dentro do um RPG. O jogador terá a função de direcioná-lo para o seu objetivo.

que possui um servidor dedicado. MMORPGs têm se tornado muito populares nos últimos tempos, principalmente por causa da popularização do acesso à Internet e pela conectividade social que tais jogos proporcionam. A Figura 6 exibe o jogo Ultima Online [26], um famoso MMORPG.

Figura 6. *Screen-shot* do jogo Ultima Online.

- Jogos de Guerra

Esse gênero de jogo explora estratégias de guerra do jogador, exibindo interações entre diferentes exércitos, possivelmente controlados por jogadores diferentes. Em geral, vence o jogo quem vence a guerra. A Figura 7 exibe um jogo de estratégias de guerra.

Figura 7. Age of Empires [27], da Microsoft, um famoso jogo de estratégias de guerra.

- Jogos Educacionais

Embora todos os jogos sejam, de alguma forma, educacionais, os jogos assim chamados têm objetivos educacionais explícitos. Tais jogos não são tão populares. Como exemplo, podemos destacar o Rocky's Boots [28], um jogo para crianças sobre lógica Booleana e circuitos digitais.

- Jogos Interpessoais

O foco dos jogos interpessoais está no relacionamento entre induvíduos ou grupos. Quando colocados na Internet, a rede mundial de computadores, realizam sua função de integração de uma forma muito intensa.

Jogos de Simulação

Há uma grande discussão sobre o limite que define onde termina um jogo e onde começa uma simulação (ou vice-versa). Essa discussão sempre vem à tona em jogos que tentam simular parte do mundo real. Nesses jogos, o investimento em realismo físico dos objetos é grande, e isso demanda aplicação das mesmas regras que são usadas para modelagem de ambientes em simulações propriamente ditas. Jogos que possuem uma interseção com simulações são colocados neste grupo que chamamos de jogos de simulação. Muitos dos jogos de simulação são *Serious Games*.

Aqui nós não colocamos jogos de simulação como um grupo de gêneros, mas como um gênero em si mesmo, com um conjunto de sub-gêneros. Os sub-gêneros de jogos de simulação que iremos citar aqui são: jogos de simulação de Vôo, jogos de simulação Militar, jogos de simulação Espacial, jogos de Projeto de Cidade e *God Games*.

- Jogos Simuladores de Vôo

São os mais populares jogos de simulação. Exploram os aspectos mais técnicos em pilotagem de aeronaves, e podem incluir simulações de combate, como é o caso do jogo Falcon 4.0 (Figura 1.b).

- Jogos de Simulação Militar

Existem muitos tipos de jogos dentro deste sub-gênero. Há simuladores de tanques de guerra, de navios e submarinos miltares. Como exemplo, destacamos o jogo Silent Hunter III [29], mostrado na Figura 8.a.

- Jogos de Simulação Espacial

Esse sub-gênero de jogos simulam, em geral, combates no espaço. Nem sempre podem ser considerados como simuladores fiéis da realidade, visto que muitos dos objetos espaciais encontrados nesses jogos não existem. A Figura 8.b exibe a captura de tela de um jogo de simulação espacial gratuito, o Orbiter [30].

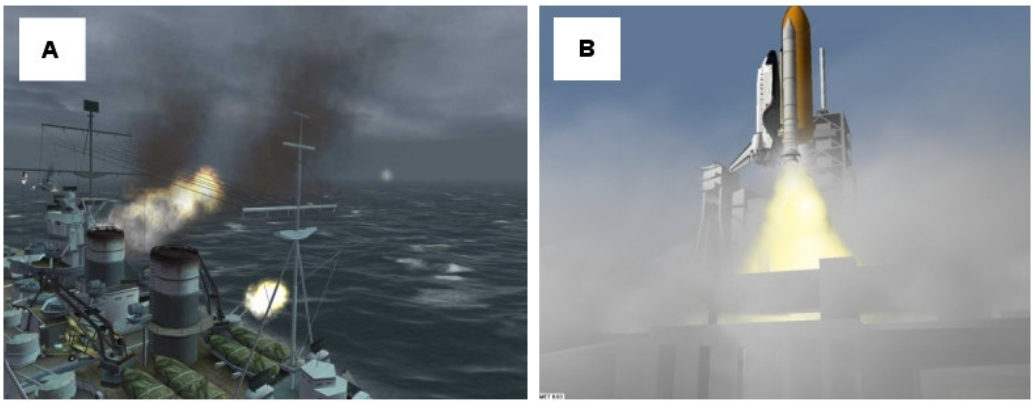


Figura 8. (*a*) Silent Hunter III, um jogo de simulação militar; (*b*) Orbiter, um simulador espacial.

- Jogos de Projetos de Cidades

Tais jogos podem ser considerados um subconjunto dos jogos de simulação de economia. Nesse caso, o jogador é responsável por realizar atividades de planejamento para ampliação de obras em cidades virtuais, atentendo a demanda de serviços desejados pelos seus habitantes (cidadãos virtuais), levando em consideração aspectos econômicos. Em computadores, o exemplo de destaque é o clássico SimCity [31] (ainda que SimCity tenha caraterísticas também de *God Games*).

- God Games

Os jogos classificados como *God Games* são simulações, em geral de populações inteiras e grandes mundos, que colocam o jogador na posição de uma entidade "divina" (com poderes divinos sobre a simulação). O que é mais interessante em *God Games* é que o jogo prossegue mesmo se o jogador não interferir, ou seja, as intervenções são opcionais.

O jogo Vidya é um exemplo de *God Game*, se encaixando nas características citadas. A Figura 9 mostra uma captura de tela de The Sims [32], o caso mais famoso de *God Game*, cujo foco está na simulação de vida cotidiana dos personagens e suas interações sociais.

Vidya, apesar de ter propostas parecidas com as de The Sims, é um jogo muito diferente desse. Primeiramente, porque o modelo de interação jogador-personagem de Vidya é inovador, segundo o qual o jogador não controla diretamente os personagens, mas lhes fornece informações não detalhadas, tendo os personagens autonomia suficiente para tomar a decisão de seguí-la ou não. Além disso, eles aprendem novos comportamentos com o tempo, a partir da própria experiência. Já em The Sims, ainda que também seja um *God Game*, os personagens não têm autonomia nem aprendizado, na medida em que o jogador impõe a sua vontade.

As relações sociais entre os *Jivas* expressas em Vidya também são substancialmente diferentes das que ocorrem entre os personagens de The Sims. Enquanto em Vidya nós temos

comportamentos básicos individuais que fazem emergir comportamentos sociais, em The Sims nós encontramos esse comportamento pré-programado. A presença do ecossistema e de vidas artificiais também tornam diferentes Vidya de The Sims.

Figura 9. The Sims, um famoso *God Game* que explora interações sociais entre personagens.

2.1.3 IA em Jogos para Computador

Ao mesmo tempo que o grau de realismo gráfico em jogos para computador aumenta rapidamente com o tempo, a demanda por comportamento inteligente de nível humano por parte de agentes de jogos se torna maior [9], [10], [11], [33], [34], [35]. Isso induz um alto investimento, que já se iniciou, por parte da indústria de jogos, em IA. Com a disseminação das placas gráficas aceleradoras, grande parte da computação gráfica, que antes era realizada pela UCP, foi migrada para essas placas, deixando assim o processador mais livre para outros tipos de processamento, tornando-se um convite para a aplicação mais intensa de IA.

Com o uso de técnicas de IA em jogos eletrônicos tornou-se possível a criação de personagens e outras estruturas com comportamento inteligente bastante elaborados. IA tornou-se uma parte essencial dos jogos para computador [6].

IA para jogos pode ser definida como o código embutido em jogos para computador que fazem os seus agentes (os que são controlados pelo computador), que podem ser oponentes ou cooperadores, tomarem boas decisões quando são oferecidas múltiplas escolhas para uma situação, resultando em comportamentos relevantes, efetivos e úteis dentro do jogo [36].

Dentre as técnicas de IA, e de apoio à IA, utilizadas em jogos para computador, as mais básicas são Máquinas de Estados Finitas (FSMs) [37], Máquinas de Estado *Fuzzy* (FuSMs) [38], Sistemas Baseados em Mensagens e Sistemas de *Scripting*. As técnicas avançadas de IA para jogos que destacamos são Algoritmos Genéticos [39] e Redes Neurais Artificiais [40]. Outras técnicas que são dignas de nota, ainda que menos freqüentes, são ALife [41], Algoritmos de Planejamento, Sistemas de Produção e Árvores de Decisão [42]. A seguir, definiremos

sucintamente cada uma dessas técnicas e seus principais pontos de aplicação dentro de jogos para computador [36].

FSM

Uma FMS, ou máquina de estados finita, não é propriamente uma técnica de IA, ainda que se encaixe dentro da definição dada de IA para jogos dada nesta Seção, mas uma estrutura de dados composta por três componentes: (i) conjunto de todos os estados inerentes da máquina, (ii) conjunto das diversas condições de entrada e (iii) uma função de transição que serve para conectar os diversos estados da máquina.

Dentro de jogos para computador, FSMs são usadas para modelar a interface de comportamento agentes, manipulando a transição entre estados do mesmo. Nesse sentido, todos os jogos que possuem agentes controlados por computador (racionais ou não) implementam FSMs, sendo distintos somente na função que determina a transição de um estado a outro, ou seja, na eficiência da decisão. Todos os agentes dentro do jogo Vidya, por exemplo, implementam FSMs.

FuSM

FuSMs são FSMs definidas dentro da noção de lógica *fuzzy*[4], ainda que não representem sistemas de lógica *fuzzy* propriamente ditos. Assim como FSMs, FuSMs possuem um conjunto finito de estados. Mas, diferentemente das FSMs, que, estando num estado atual passam para um próximo estado baseadas no *input* (que comumente é a percepção do agente), FuSMs assumem a possibilidade de estar em vários estados ao mesmo tempo, sem codificação de transições. Ao invés disso, cada estado dentro de uma FuSM calcula um nível de ativação, que determina a extensão para o qual o sistema está em qualquer estado dado.

Em jogos para computador, FuSMs são úteis para modelagem do comportamento de agentes que podem estar em mais de um estado num mesmo instante de tempo, e cujos estados são mais do que valores digitais, como ligado/desligado, vivo/morto, aberto/fechado, etc, mas que também podem assumir valores intermediários entre dois extremos, como “quase morto”, “parcialmente aberto”, etc.

Sistemas Baseados em Mensagens

Diferentemente das técnicas anteriormente citadas (FSM e FuSM), sistemas baseados em mensagens não são rigorosamente falando, estruturas para tomada de decisões, mas um protocolo de comunicação entre os diversos elementos dentro de um contexto para troca de mensagens que são úteis para tomadas de decisões. Suponhamos que um objeto A, para realizar uma determinada ação, tenha a pré-condição de um objeto B esteja num determinado estado. Ao invés de A ficar

[4] Lógica *Fuzzy* é um superconjunto da Lógica Booleana convencional com o intuito de abrigar o conceito de “verdades parciais”.

periodicamente "inspencionando" B sobre suas mudanças de estado, o que acarreta num custo computacional, admite-se um esquema em que B, assim que chegar no tal estado, avisa A sobre a mudança, permitindo assim que A tome sua decisão (sem o custo computacional da inspeção periódica).

O sistema de troca de mensagens, ou eventos, entre os diversos objetos dentro de um jogo para computador é muito importante na redução do custo computacional que sempre é crítico em desenvolvimento de jogos. Além disso, agentes inteligentes em jogos para computador são, em geral, reativos, dependendo de uma percepção externa para tomada de decisões. Projetando-se um sistema que é baseado em mensagens, a computação exaustiva por parte dos (possivelmente muitos) agentes dentro do jogo em inspecionar periodicamente mudança de estados de outros objetos é significativamente reduzida.

Sistemas de *Scripting*

Scripting, dentro do contexto de desenvolvimento de jogos, significa usar (ou criar) uma linguagem de programação simplificada para desenvolver elementos de AI, lógica e comportamentos. Ela se torna útil, e mesmo essencial, em grandes projetos de desenvolvimento de jogos para computador, especialmente porque a programação dos comportamentos é feita num macro-nível, sem levar em consideração detalhes mais relevantes em engenharia de *software*.

Apesar da facilidade proporcionada pelo uso ou criação de uma linguagem de *scripting*, a criação da mesma se torna um produto à parte, o que implica num custo maior para o desenvolvimento do jogo.

AG

AG é considerada uma técnica de IA avançada dentro de jogos para computador. Dentro do jogo Vidya, a inteligência do *Jiva* (a classe de personagens principais do jogo) é completamente modelada usando AGs, que é discutido em mais detalhes na Seção 2.2. A idéia básica que permeia AG é a de que uma parte do conjunto completo de soluções para um dado problema, a população de indivíduos do AG, é avaliada, produzindo uma próxima geração que estará mais próxima da melhores soluções. Tudo isso utilizando conceitos de seleção natural, como cruzamento de indivíduos e mutação, para a geração de outros indivíduos melhores (mais adaptáveis). Pode-se dizer que o AG realiza uma "busca evolutiva" no espaço de soluções.

Em jogos para computador, AGs são usados para encontrar novas soluções dentro de situações do jogo, sintonizar parâmetros e comportamento de agentes e também como método de busca de melhor caminho para locomoção dentro de um mundo (*pathfinders*). No Capítulo 3 deste trabalho, mostramos como a IA do Jiva foi projetada através do uso de AGs. No Capítulo 4, mostramos resultados de experimentos realizados, comprovando a eficácia de AG para modelagem de comportamentos de agentes inteligentes em jogos para computador.

RNA

RNA é uma técnica de IA muito usada para o reconhecimento de padrões e predição de tendências. Seu funcionamento intrínseco se baseia na aproximação de função (regressão) para um conjunto de dados de treinamento que caracterizam o problema real. O modelo das RNAs são inspirados biologicamente pelo cérebro, que, assim como no modelo artificial, constitui uma rede neural. Existem muitos modelos conhecidos de RNAs, bem como diversos métodos de aprendizado.

Em jogos para computador, RNAs podem ser usadas em agentes para o aprendizado de comportamento inteligente. Mas, para isso, precisa-se de dados de treinamento da rede, que comumente são adquiridos a partir de ações de exemplo de seres humanos controlando os agentes.

ALife (*Artificial Life*)

ALife é a tentativa de simular a vida. Sem adentrar mais nas questões filosóficas de definição da vida, ALife pode ser definida como o conjunto de estudos que objetivam entender melhor a vida natural, tentando recriar fenômenos biológicos em um ambiente artificial, possivelmente num computador.

Alguns jogos para computador possuem agentes que podem ser considerados como vidas artificiais, visto que habitam um mundo virtual, possuem atributos vitais, possuem um tempo de existência, se reproduzem e lutam pela sobrevivência através da busca de recursos que os sustém. Os seres vivos que habitam o mundo virtual dentro do jogo Vidya, por exemplo, incluindo o *Jiva*, são vidas artificiais, e todos em conjunto formam um ecossistema.

ALife não é muito aplicada em jogos para computador, atualmente, senão na geração de movimentos complexos que vêm acrescer o nível de detalhes do jogo. Em alguns RPGs se usa o conceito de ALife para a geração de ecosistemas de criaturas dentro de algumas partes do mundo virtual.

Algoritmos de Planejamento

Planejamento na realização de ações de agentes é considerar, além dos custos imediatos da realização das ações, possíveis custos futuros. Às vezes, a realização de uma ação que traz benefícios imediatamente pode acarretar em malefícios futuros, ou vice-versa.

Algoritmos que tentam prever e planejar ações de agentes dentro do mundo do jogo são conhecidos como algoritmos de planejamento. Dentro do planejamento de ações, também incluímos o planejamento de menor caminho, que são os algoritmos chamados de *pathfinders*. Como expoentes dentre os algoritmos de planejamento, destacamos o Algoritmo A* [43], que é um *pathfinder* muito conhecido, e a técnica de Programação Dinâmica [44], que é uma variação de Aprendizado por Reforço para achar a melhor política para um agente realizar um curso de ações, considerando perdas ou ganhos imediatos e futuros baseados na experiência.

No jogo Vidya usamos um algoritmo de planejamento próprio para projetar o acúmulo de custos das ações dos *Jiva* em um número finito de passos no futuro.

Sistemas de Produção

Sistemas de produção são sistemas baseados em regras que se eforçam para alcançar conhecimento e comportamento especialista numa determinada área. Sistemas de produção, geralmente, são guiados por objetivo e altamente reativos, consistindo, na maioria das vezes, de avaliações condicionais sobre os estados do sistema.

Em jogos para computador, Sistemas de Produção são usados para especializar agentes reativos. Apesar de puramente reativos, esses agentes podem ter comportamento muito refinado, caso o número de regras definidas no sistema de produção seja muito grande, dando a impressão de complexidade. No jogo Vidya, os agentes, com exceção do *Jiva*, são sistemas reativos.

Árvores de Decisão

Árvores de decisão são estruturas de dados com organização em árvore que têm o objetivo de auxiliar na tomada de decisão. Em um agente puramente reativo, por exemplo, podemos organizar a lista de mapeamentos de estados em ações, que quase sempre são feitos através de *switchs* ou uma seqüência de *if-then*, em uma estrutura de árvore, constituindo uma árvore de decisão.

A idéia que permeia uma árvore de decisão é a organização hierárquica do problema que, em última análise, é mais eficiente em busca. Nos nós de uma árvore de decisão são colocadas as "questões" que o agente deve responder (no caso de um agente reativo), até que se chegue a uma folha que é uma ação atômica a ser realizada.

A criação de árvores de decisão pode ser feita através de dados de treinamento, que são exemplos de mapeamento de entradas em saídas. Elas podem ser úteis tanto em casos de classificação quanto de regressão (generalização), de funções parecidas com as de RNA, porém com as seguintes diferenças: (i) árvores de decisão são mais compreensíveis em estrutura que RNAs; (ii) árvores de decisão consideram uma única variável por vez, enquanto RNA pode considerar mais de uma; (iii) RNA pode ser mais precisa que árvores de decisão.

Em IA para jogos de computador, árvores de decisão podem ser usadas em classificação e regressão. Um caso de classificação comum é a determinação do tipo de comportamento do jogador, tornando a IA mais direcionada para o tipo classificado (tentativa de adaptar o comportamento dos agentes de um jogo às características do jogador). Uma tarefa de regressão é a predição da dificuldade percebida pelo jogador no jogo, ajustando o nível de dificuldade para mais ou para menos, a depender do contexto.

2.2 Técnica Inteligente Usada no Jogo Vidya

A Seção anterior apresentou rapidamente o universo dos jogos, jogos para computador e técnicas de IA usadas em jogos para computador. Nesta Seção, explicaremos em mais detalhes a principal técnica que foi usada no jogo Vidya para a modelagem do comportamento do *Jiva*, um agente autônomo inteligente com características evolutivas, capaz de aprender a partir de sua própria experiência no Mundo de forma a tornar seu comportamento sintonizado com o meio em que habita. Dizemos "a principal técnica" porque outras técnicas também foram usadas como suporte secundário às decisões do *Jiva*, mas é ela mesma o *core* do módulo de decisão. A técnica de que estamos falando é Algoritmos Genéticos.

A aplicação de AGs em jogos eletrônicos foi rapidamente comentada na Seção anterior, contudo ressaltamos que a maneira como AGs estão sendo usados em *Vidya* difere substancialmente de como AGs têm sido usados no desenvolvimento de IA para jogos. Essa diferença é explicada mais à frente, no Capítulo 3.

2.2.1 Algoritmos Genéticos

Algoritmos Genéticos (AG) são uma família de algoritmos inspirados na evolução natural [39], [45], amplamente usados para resolver problemas de otimização e busca. Os problemas de otimização são caracterizados por três elementos principais: (i) a codificação do problemas; (ii) a função objetivo (que se deseja maximizar ou minimizar); (iii) o espaço de soluções para o problema.

AG contêm soluções potenciais para o problema que se pretende resolver. Essas soluções são codificadas no algoritmo como cromossomos (ou indivíduos), formando uma população. Cromossomos são iterativamente submetidos a operações de seleção, cruzamento (ou *cross-over*) e mutação, com o intuito de produzirem novas gerações de cromossomos ainda mais sintonizados em resolver o problema.

Uma das principais vantagens em se usar AG é que eles permitem uma formulação e solução de problemas de busca e otimização de forma simplificada. Nos casos mais simples do algoritmo, cromossomos são codificados como cadeias de bits de tamanho fixo (outros AG mais complexos podem usar cadeias de tamanho variável). Além disso, AG são sempre convergentes e numericamente robustos, pois não produzem soluções aproximadas (não sensíveis a erros de arredondamento).

Antes de usar AG é necessário modelar cada possível solução para o problema em questão em um cromossomo. Isso pode ser feito mapeando-se características que são julgadas relevantes em genes. Uma população inicial de indivíduos – *i.e.* um conjunto de cromossomos – são gerados aleatoriamente. Depois de serem classificados por algum critério de aptidão, eles irão evoluir e guiar o processo de busca por uma ótima solução para o problema. Ao calcular os valores das aptidões, dois indivíduos por vez são selecionados por métodos de amostragem, como o da Roleta [46]. Os dois cromossomos selecionados por vez são então combinados para gerar um novo

indidívuo. Nesse momento, é possível introduzir um operador aleatório – *e.g.* mutação – para evitar convergência prematura, mudando um valor de gene do novo indivíduo. Cruzamento e mutação acontecerão até que uma nova população com o mesmo tamanho da anterior seja gerada. A Figura 10 ilustra todos os passos mencionados.

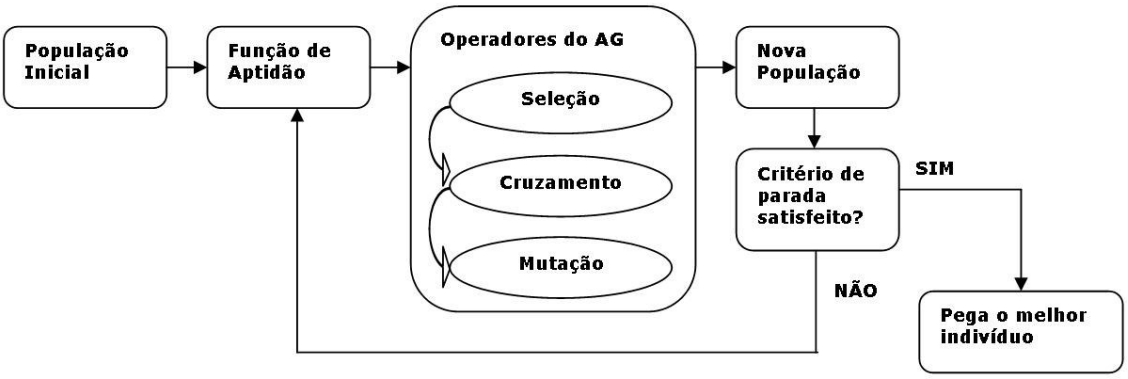


Figura 10. Passos do AG.

Encontrar uma boa solução para um problema pode ser uma operação que consome muito tempo, especialmente se todo o conjunto de soluções é explorado. O AG é capaz de econtrar uma resposta tão boa quanto a necessária, executando o ciclo mencionado um número suficiente de vezes até encontrar uma solução. Note que para cada ciclo há um melhor indivíduo na população atual. Assim, tão logo haja a necessidade de parada do algoritmo (*e.g.* o usuário demanda uma ação a ser tomada), uma solução candidata não aleatória adequada para o problema é prontamente oferecida pelo AG. É importante notar que, em geral, não são necessários muitos ciclos para encontrar uma solução razoável.

As populações de cromossomos são sempre re-avaliadas, para gerar novas populações mais qualificadas. Em cada avaliação, a cada cromossomo é atribuído um valor de aptidão, que irá informar o quão bom ele é em resolver o problema. Os cromossomos mais aptos são selecionados e submetidos ao cruzamento, a fim de gerar novos indivíduos mais aptos, ou seja, melhores soluções.

2.2.1.1 Cromossomos e Genes

Cromossomos são cadeias de números (bits, inteiros ou reais) que codificam soluções para o problema que se pretende resolver com o AG. Cada elemento dessa cadeia é chamado de gene, e codificam características relevantes para a resolução do problema. O propósito do AG é encontrar o melhor cromossomo, ou seja, aquele cuja configuração de genes melhor soluciona o problema. A Figura 11 mostra a estrutura de um cromossomo de um AG simples.

Figura 11. Estrutura de um cromossomo em um AG simples.

2.2.1.2 Método de Seleção da Roleta

Existem muitos métodos de seleção de cromossomos. Destacamos aqui o método de seleção da Roleta. Esse é um método probabilístico, onde os cromossomos mais aptos têm uma probabilidade maior de serem selecionados.

Para melhor visualizar o método, considere um círculo dividido em *PS* regiões – onde *PS* é o tamanho da população – e que a cada região está associado um cromossomo da população. O tamanho de cada região é diretamente proporcional à aptidão do cromossomo associado a ela (Figura 12, esquerda). Então, sobre esse círculo é posto uma "roleta" com *PS* cursores igualmente espaçados (Figura 12, direita). Após um giro da roleta, a posição dos cursores sobre as regiões do círculo determina os cromossomos selecionados para cruzamento.

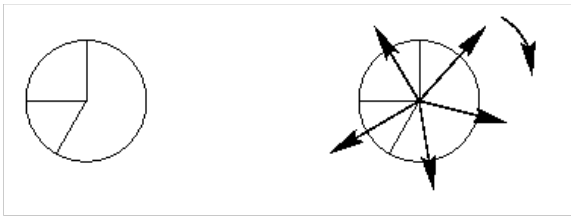


Figura 12. Método da Roleta. Na esquerda, um círculo dividido em regiões proporcionais às aptidões dos indivíduos. Na direita, a roleta que selecionará os indivíduos.

2.2.1.3 Reprodução

Após selecionados, os cromossomos deverão ser cruzados entre si. O cruzamento é realizado da seguinte forma: a lista de cromossomos selecionados é embaralhada (aleatoriamente), gerando uma segunda lista – a lista de parceiros; os cromossomos de índices coincidentes nas duas listas são tomados aos pares e cruzados. A Figura 13 ilustra o cruzamento entre dois indivíduos.

A cadeia de números que forma os cromossomos são partidas durante o cruzamento num ponto pré-definido – o ponto de corte – e um novo cromossomo é gerado tomando-se a parte inicial da cadeia do primeiro cromossomo e a parte final da cadeia do segundo cromossomo.

2.2.1.4 Mutação

A operação de mutação impede que, para certos problemas, o AG convirja prematuramente para mínimos locais, fazendo com que ele varra mais cromossomos que compõem o espaço de estados

de soluções total. A mutação é realizada com uma certa probabilidade sobre os indivíduos de uma população. Quando ela ocorre, números das cadeias numéricas de um ou mais cromossomos da população são alterados aleatoriamente.

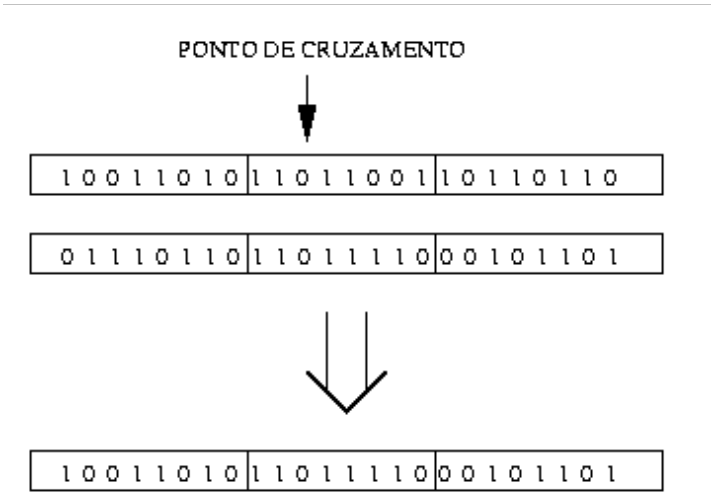


Figura 13. Cruzamento entre dois cromossomos.

2.2.1.5 Parâmetros Importantes no AG

A codificação do cromossomo é muito importante para um bom desempenho do AG. Mas outros parâmetros são também muito importantes para o desempenho. Entre eles, destacamos: (1) tamanho da população; (2) número de gerações; (3) a probabilidade de cruzamento; (4) a probabilidade de mutação – a importância de cada parâmetro depende da classe de problemas que se está tratando. O tamanho da população e o número de gerações são dependentes da complexidade da otimização ou busca que se está realizando. Em qualquer caso, vários experimentos devem ser realizados para se achar a melhor configuração de parâmetros.

2.2.1.6 Aplicações de AG

AG é muito aplicado em diversas áreas científicas [1]. Destacamos as seguintes:

- Geração de Circuitos Analógicos – achar a melhor configuração de parâmetros para um circuito (*e.g.* topologia, tipo de componentes, etc);
- Programação Genética - geração de códigos de programas de computador que resolvam um determinado problema;
- Gerenciamento de redes (e.g., para descobrir rotas ótimas);
- Ciências biológicas - modelagem de comportamentos biológicos para estudo de certas estruturas genéticas.

Capítulo 3

Vidya

Este trabalho é a concretização de um *God Game* chamado Vidya. Além de ser um jogo, Vidya é um ambiente onde é possível a observação de personagens de comportamento social, seres artificialmente inteligentes, chamados *Jivas*. Neste Capítulo, apresentamos Vidya em detalhes, com o foco centrado nos aspectos de modelagem inteligente do *Jiva*, objeto também de pesquisa do jogo.

Em um jogo tradicional, o jogador controla diretamente as ações dos personagens sob sua responsabilidade. Como foi visto no Capítulo 2, *God Games* possuem a característica de que o jogador não controla diretamente um personagem e, mesmo se não interferir, o jogo continua. Em Vidya o jogador poderá, no máximo, fornecer orientações, ou melhor, intuições, para os *Jivas*, numa interface de interação jogador-personagem inovadora. Referenciamos as orientações como intuições porque, dentro do contexto ficctício do jogo, elas surgem como um conhecimento interno em cada *Jiva* que terá o livre-arbítrio para as seguir ou não; uma intuição dentro do jogo é chamada de *vidya*.

O foco principal de Vidya está na aplicação de técnicas de IA para desenvolver a "mente" do *Jiva*, que deve ser capaz de tomar decisões eficientes dentro do Mundo em que ele habita, chamado *Suryaloka*. O fato de *Suryaloka* não ser habitado somente por *Jivas*, mas também por outros seres com os quais o *Jiva* compartilha e disputa recursos naturais muitas vezes escassos, acrescenta muito mais complexidade nos processos de decisão. Isso quer dizer que a tomada de decisões de um *Jiva* leva em consideração, além do efeito de suas ações sobre o Mundo, a incerteza sobre as decisões dos outros seres.

Diferentemente das aplicações de IA em jogos que se vê com freqüência, onde a computação inteligente uma vez aprendida não evolui com o tempo, em Vidya existe a constante evolução "intelectual" dos *Jivas*. Ou seja, que se adapta constantemente dentro do ambiente. Isso permite tomadas de decisão cada vez mais bem elaboradas e, além disso, a migração do módulo de inteligência do *Jiva* para outros seres em outros ambientes.

O jogo inicia convidando o jogador a ser o *Deva* (Deus) de um clã, dentre dois que existem em *Suryaloka*. O jogador tem como tarefa fazer seu clã ser o último sobrevivente. A única forma que terá de ajudá-lo é fornecendo *vidya* para cada um de seus integrantes, contando com suas possíveis tomadas de decisão independentes, devido ao livre-arbítrio inerente do *Jiva*. Quanto mais corretamente o jogador intuir seu clã, melhores ações o *Jiva* realizará em prol de si próprio e do clã, e mais eficientemente o *Jiva* aprenderá a agir. Por correto, entende-se tudo que melhora ou preserva a vida dos *Jivas* e dos seus clãs.

Neste Capítulo, iniciamos explicando aspectos de desenvolvimento do *software* (*i.e.* o jogo Vidya) sem entramos em detalhes privativos do *Game Design*[5] – o Anexo A desta monografia está o *Game Design* de Vidya – onde exibimos uma visão modular de alto-nível e mais detalhadamente sua hierarquia de pacotes (componentes), contudo sem entrar numa descrição a nível de classes de objetos, o que ocuparia um grande volume. Posteriormente, apresentaremos o Mundo *Suryaloka* e seus objetos, falando sobre cada um deles em vários aspectos.

Uma Seção à parte é exclusiva para a caraterização do módulo de decisão inteligente do *Jiva*. Mais à frente é explicado como o *Jiva* age socialmente, quando então revelamos seu modelo de interação social que permite esse tipo de comportamento.

A seguir, comentamos o modelo inovador de interação jogador-personagem proposto por Vidya, em que o jogador não controla diretamente os personagens, mas fornece apenas instruções para eles.

O presente Capítulo não pretende detalhar o *gameplay* (*i.e.* a forma como se joga) de Vidya, mas apresentar o jogo e seus elementos que, em última análise, são um pré-requisito para seu objetivo maior que é a modelagem inteligente do *Jiva*. Vidya deve ser visto como um jogo de simulação, um *God Game*, cujo principal objetivo é a análise de seres artificiais inteligentes (os *Jivas*) tomados individualmente e em sociedade. Em outras palavras, aqui estamos focados no objetivo do jogo, e não no do jogador. Para conhecimento da forma como se joga Vidya, vide o documento de *Gameplay*, anexo a este documento (Anexo B).

3.1 O Software Vidya

Vidya foi completamente desenvolvido na linguagem Java [48]. Java possui muitas APIs já implementadas e é orientada a objetos. Isso torna a modelagem mais simples. Apesar de poucas, existem referências para a elaboração de jogos eficientes em Java, [49], [50], [51], [52], [53]. A principal preocupação em relação à Java é que que seu desempenho geral não é tão bom

[5] *Game Design* é um documento de extrema importância em desenvolvimento de jogos, que define o escopo e base ideológica subjacente de um novo jogo a ser construído. Esse documento é uma especificação de alto nível e conceitual do jogo, exibindo informações como nome, gênero, descrição, história, etc, sem levar muito em consideração aspectos referentes ao meio de representação do jogo (*e.g.* mesa, PC, *mobiles*, etc).

comparado com outras linguagens de programação, como C e C++. Como este trabalho apenas investiga conceitos e não se propõe a desenvolver um produto comercial, julgamos aceitável essa decisão de projeto. Porém, há uma intenção futura de migrar o código fonte de Vidya para C++. A liguagem C++ é considerada a melhor linguagem de programação para jogos *desktop*, quanto ao aspecto de desempenho computacional [54], atualmente, para a qual existem muitos *game engines*[6], bibliotecas gráficas, etc. Além disso, C++ é OO (*Object-Oriented*), como Java, o que facilita a migração.

3.1.1 Visão Modular do Software Vidya

A Figura 14 exibe os módulo de alto nível do jogo Vidya.

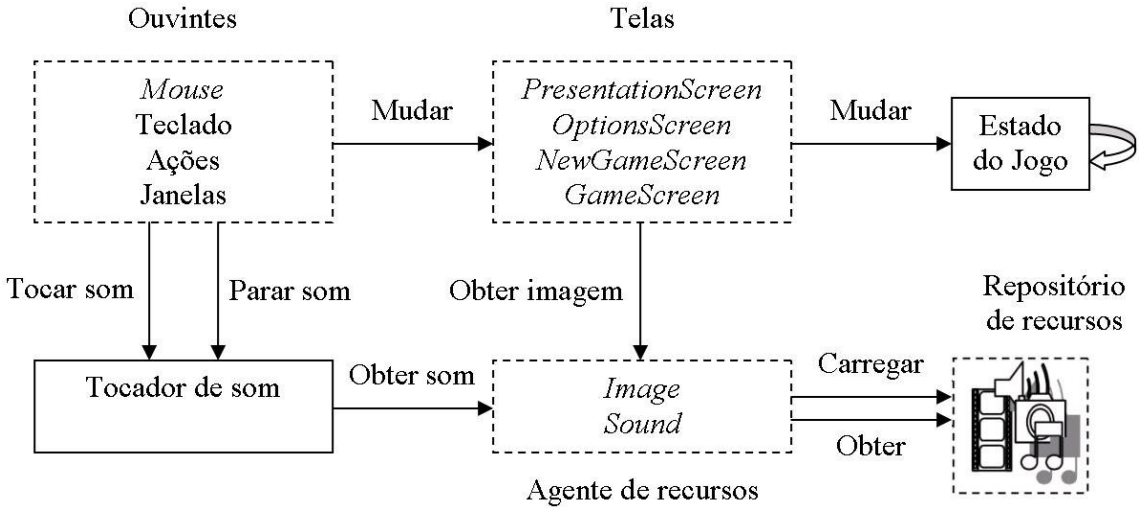


Figura 14. Visão modular de alto nível do jogo Vidya.

Como é visto na Figura 14, em alto nível, Vidya é composto por seis módulos principais: Estado do Jogo, Telas, Ouvintes, Tocador de som, Agente de recursos e Repositório de recursos – que são explicados a seguir.

Estado do Jogo

Dentro desse módulo nós encontramos o motor do jogo (*game engine*), o mundo e seus objetos. O Mundo e cada um dos objetos possuem atributos que mudam no decorrer do desenvolvimento do jogo. A configuração instantânea desses atributos, tomadas em conjunto, é o que chamamos de estado do jogo. O estado do jogo pode ser mudado pelos próprios elementos internos ao jogo e, indiretamente, através das telas do jogo, por eventos gerados pelo usuário.

---

[6] *Game engine* é o sistema global de controle de um jogo. Ele é responsável por intergrar todos os subsistemas, como os de interface gráfica, reprodução de som, IA, etc. Através de um *game engine*, esses módulos se comunicam de forma organizada e consistente [55].

Telas

O módulo "Telas" engloba todas as telas do programa. A seqüência de telas do programa determina seu fluxo navegacional. A Figura 15 exibe tal fluxo.

A tela designada por *PresentationScreen* é a primeira tela do programa, que aguarda o pressionamento de qualquer tecla por parte do usuário. Pressionada uma tecla, segue-se à tela designada por *OptionsScreen*.

A tela *OptionsScreen*, como o nome já sugere, é a tela de opções do jogo. O usuário poderá optar por uma de duas opções disponíveis, a saber, *New Game* e *Exit*. A opção *New Game* dirige o fluxo de navegação para a tela *NewGameScreen*. A opção *Exit* tem a finalidade de fazer o programa terminar (uma janela de confirmação é mostrada ao usuário antes da finalização do programa).

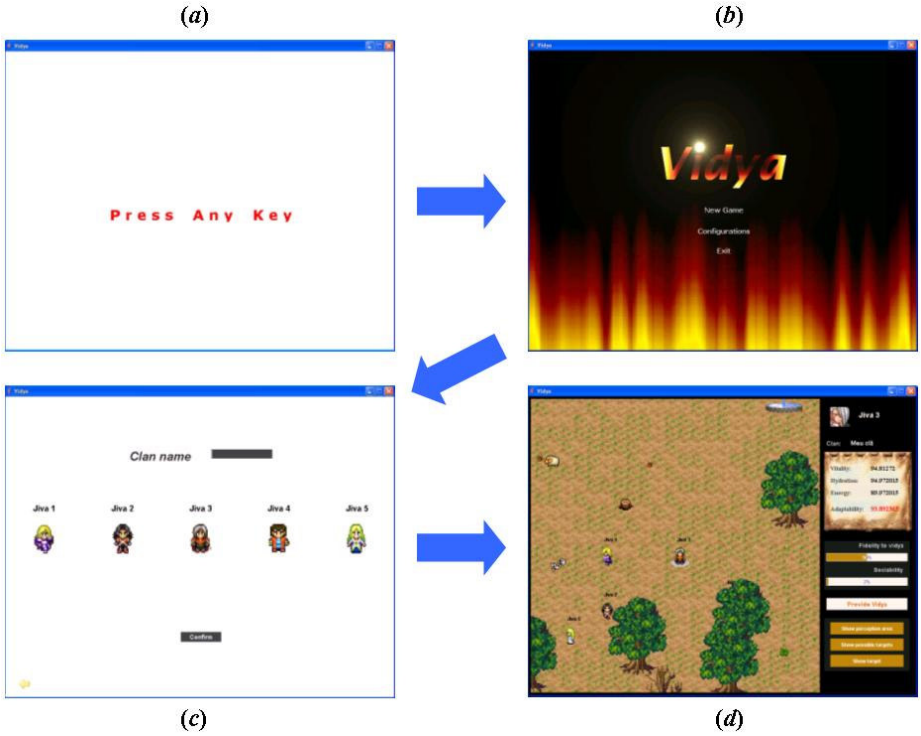


Figura 15. Fluxo navegacional de Vidya. (*a*) A tela *PresentationScreen*; (*b*) A tela *OptionScreen*; (*c*) A tela *NewGameScreen*; (*d*) A tela *GameScreen*.

A tela *NewGameScreen* é a penúltima tela antes do início efetivo do jogo. Ela exibe o clã de cinco *Jivas* que será responsabilidade do jogador. O jogador precisa fornecer um nome para o clã antes de prosseguir. Os *Jivas* já vêm com nomes por *default*, mas é adequado que o usuário mude seus nomes. Clicando no botão *Confirm*, estando com todos os dados preenchidos, o fluxo de navegação é direcionado para a tela *GameScreen*. A seta amarela em *NewGameScreen* faz voltar para a tela *OptionsScreen*.

A tela *GameScreen* é composta por dois painéis. O painel maior, que fica na esquerda da tela, é o painel de jogo. É nesse painel que é visto o desenvolvimento do jogo. O painel da direita superior é o painel de informações de objetos. Quando um dos objetos do jogo é selecionado, suas informações são mostradas neste painel. É também neste painél que o jogador interage com os *Jivas* de seu clã.

A seta rotulada como "mudar", que vai do módulo "Telas" ao módulo "Estado do Jogo" indica que o estado do jogo pode ser mudado através das telas do programa. Para ser mais específico, isso ocorre em eventos que usuário gera a partir da interação com a tela *GameScreen* (através do *mouse* e do teclado). As telas também fazem uso de arquivos de imagem para representar diversos tipos de objetos. Por isso há uma seta rotulada como "obter imagem", que vai do módulo "Telas" ao módulo "Agente de recursos", que é o módulo responsável pela manipulação de imagens e sons.

Ouvintes

"Ouvintes" formam uma classe de objetos que obedecem ao padrão de projeto *Observer*. No padrão *Observer* sempre há um processo ouvinte e muitas prováveis entidades observadoras. Entidades "observadoras" se registram no ouvinte, que fica responsável por notificar essas entidades quando eventos ocorrem.

Em Vidya, há ouvintes para *mouse*, teclado, ações e janelas que, respectivamente, notificam eventos de *mouse*, teclado, objetos de interface gráfica e janelas. Por exemplo, quando o jogador clica com o botão esquerdo do *mouse* sobre um objeto do Mundo, até que esse objeto venha a ser mostrado no painel de informações de objetos, um tratamento adequado do evento gerado pelo clique teve de ser feito. Outro exemplo do uso dos ouvintes ocorre quando a tela de apresentação (*PresentationScreen*) é exibida e fica aguardando por um evento qualquer de teclado para dar continuidade ao fluxo do programa. Quando o usuário tenta fechar a janela, por exemplo, uma janela de confirmação de saída é mostrada; isso só é possível porque há um ouvinte de eventos sobre janelas.

A seta rotulada como "mudar" que vai do módulo "Ouvintes" ao módulo "Telas" indica que através dos ouvintes se pode mudar o estado das telas, ou mudar de telas. Alguns eventos que estão relacionados com mudança de telas podem fazer uso do "Tocador de som", exibido como um módulo na Figura 14. Por exemplo, quando se muda da tela *NewGameScreen* para *GameScreen*, o som de fundo é também mudado. Essa mudança é iniciada nos ouvintes. Por isso estão indicadas duas setas, rotuladas como "tocar som" e "parar som", para tocar e parar sons, respectivamente.

Agente de recursos

Esse módulo realiza a manipulação, a nível de arquivos, de sons e imagens. O armazenamento desses recursos são representados na Figura 14 pelo módulo "Repositório de recursos". As setas

"carregar" e "obter" indicam, respectivamente, carregamento e obtenção de arquivos. O módulo "Agente de recursos" é usado diretamente pelos módulos "Tocador de som" e "Telas".

Tocador de som

Esse módulo representa o tocador de som do jogo, com uma interface de mais alto nível para o módulo "Ouvintes". A seta rotulada como "Obter som" simboliza a obtenção de um arquivo de áudio para manipulação. Esse módulo foi exclusivamente criado para manipulação em alto nível de som devido à maior complexidade desse tipo de mídia, comparativamente a arquivos de imagem (que são, em sua maioria, estáticos).

Repositório de recursos

"Repositório de recursos" representa o centro de armazenamento de todo tipo de mídia utilizada no programa, que são arquivos de áudio e imagem. Trata-se somente de um repositório estruturado de arquivos. O módulo "Agente de recursos" carrega e obtém arquivos desse repositório para manipulação. Na verdade, nenhum outro módulo fará uso direto desse repositório, senão via "Agente de recursos", que foi criado unicamente para esse fim (um *proxy* para o repositório de recursos).

3.1.2 Pacotes de Classes do Software Vidya

A Figura 16 exibe o diagrama de componentes (pacotes) que compõem *Vidya*. Esses componentes, em sua hierarquia, realizam as funcionalidades descritas na sessão 3.1.1 (visão modular do software Vidya). No mais alto nível, *Vidya* é composto por três componentes: *entity*, *io* e *util*.

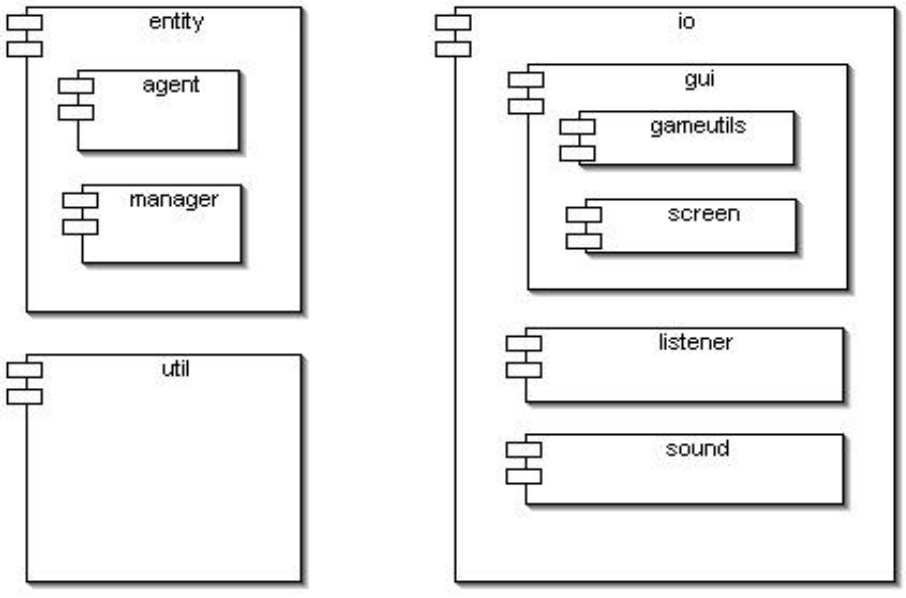


Figura 16. Componentes de alto nível de Vidya: *entity*, *io* e *util*.

Componente *entity*

No componente *entity* estão descritas todas as entidades que, em sua totalidade, definem o estado do jogo. O estado do jogo é o estado do Mundo mais o estado de cada objeto em particular que está inserido no Mundo. Então, em *entity* estão descritos o Mundo e seus objetos. Esse estado é caracterizado pelos valores, num dado instante de tempo, dos atributos do Mundo e dos objetos.

O componente *entity* é composto por todas as entidades do jogo, mais dois subcomponentes que são úteis para essas entidades: *agent* e *manager*.

No subcomponente *agent* estão definidos os agentes de alguns dos seres (artificialmente) vivos do Mundo, cuidando de seus comportamentos. Por exemplo, ovelhas são entidades do jogo e seus comportamentos são modelados segundo uma classe agente para ovelhas. Há agentes para *Jivas*, lobos e vacas também.

No subcomponente *manager* estão definidas as classes de gerenciamento de entidades. Por exemplo, há uma classe para gerenciamento das árvores presentes no Mundo, que tem a função de cuidar do ciclo de vida de todas as árvores, povoar o Mundo com árvores, etc.

Componente *io*

No componente *io* estão definidos todos os pacotes que realizam operações de entrada e saída. As operações de entrada estão refletidas no subcomponente *listener*, que possui classes para realizar "escuta" de eventos de *mouse*, teclado, componentes gráficos (como botões, áreas de texto, rótulos, etc.) e janela, e tomar as devidas ações na ocorrência de tais eventos. As operações de saída estão refletidas nos subcomponentes *gui* e *sound*. Esses constituem os componentes de saída gráfica e sonora, respectivamente.

O componente *listener*, como já foi dito, é composto por todas as classes para notificação de eventos iniciados pelo usuário. Esse pacote realiza todo o tratamento de entrada pelo *mouse*, teclado, componentes gráficos e janela. Cada um desses tipos de entrada gera eventos específicos, cujo tratamento varia de acordo com o momento e ambiente em que foram iniciados. Por exemplo, tratamento de eventos de teclado na tela de opções é diferente do tratamento para eventos de teclado quando se está na tela de jogo. Ocorre da mesma forma para os outros tipos de eventos.

Componente *io.gui* [7]

O componente *gui* é composto por dois subcomponentes: *gameutils* e *screen*. Ambos cuidam da interface gráfica do usuário.

---

[7] Essa nomenclatura indica uma hierarquia de componentes, referenciando, nesse caso, o sub-componente *gui* que está incluso no super-componente *io*.

O subcomponente *gameutils* possui classes de interface gráfica de utilidade em jogos (no caso de Vidya, `Sprite`[8] e `VidyaSpritesFactory`[9]). O subcomponente *screen*, em síntese, é composto por todas as telas possíveis do programa, desde a tela de apresentação até a tela de jogo propriamente dita.

Componente *io.sound*

Ainda dentro do componente *io*, dentre os subcomponentes de saída, nós temos o pacote *sound*, que é responsável por toda a manipulação de reprodução de áudio do jogo. Este pacote possui duas classes: `Sound` e `SoundPlayer`.

A classe `Sound` é uma representação de alto nível de um determinado som. A classe `SoundPlayer` realiza a reprodução de sons, provendo métodos para iniciar e parar reprodução, bem como para liberar recursos de áudio.

Componente *util*

O componente *util* agrega classes de utilidade geral para a realização do jogo. Nesse pacote, encontramos as variáveis globais do jogo, o agente de acesso a recursos de imagem e áudio e algumas estruturas de dados essenciais para *Vidya*.

3.2 O Mundo *Suryaloka* e seus Objetos

Nesta Seção são explicadas as estruturas do Mundo do jogo e de seus objetos, que fazem parte do pacote *entity* (explicado na seção anterior).

Todas as entidades do jogo, ou seja, o Mundo e seus objetos implementam, obrigatoriamente, dois métodos: *update* e *paint*.

Antes de explicar esses métodos obrigatórios, é preciso entender a estrutura básica de um jogo eletrônico, ou o motor do jogo, como é comumente mencionado. Basicamente, um jogo eletrônico é um laço, com ou sem condição de terminação, em que são executadas repetidamente três tarefas: (i) captar entrada do usuário, (ii) atualizar estado do jogo e (iii) representar graficamente (pintar) seu estado atual. Com essas informações fica fácil compreender os métodos *update* e *paint*.

O método *update* atualiza o estado da entidade e o método *paint* pinta o estado atual da entidade. São esses métodos que, juntamente com a escuta de *inputs* fornecidos pelo jogagor (realizada pelos *listeners*), são executados repetidamente no *loop* do jogo, para cada entidade. A

---

[8] Um *sprite* é uma representação gráfica animada para entidades de um jogo, como personagens, utensílios, objetos animados, etc.

[9] *Sprites factory* (fábrica de *sprites*) foi usada em Vidya para a criação de todos os *sprites* do jogo, tornando a obtenção de *sprites* por parte das entidades mais facilitada.

seguir, comentamos o modelo inovador de interação jogador-personagem proposto por Vidya, em que o jogador não controla diretamente os personagens, mas apenas os instrui.

3.2.1 Caracterização do Mundo do Jogo

A entidade (classe) que representa o Mundo do jogo se chama `World`. Ela funciona principalmente como um repositório de objetos e, como toda entidade, também implementa os métodos *update* e *paint*.

Na verdade, o laço do jogo não chama os métodos de atualização e pintura para todas as entidades, mas unicamente para uma instância de `World`. Os métodos de atualização e pintura de `World` chamam, por sua vez, os mesmos métodos para cada objeto que pertence ao seu repositório. Essa estrutura hierárquica é uma boa solução para separação de responsabilidades num jogo eletrônico.

A classe `World` oferece métodos para inserção, remoção e obtenção de objetos, além de outros métodos auxiliares, como por exemplo para detectar e evitar colisão. Outros métodos específicos do jogo também são fornecidos.

O Mundo do jogo é, estruturalmente, um *grid* bidimensional de células. A Figura 17 mostra o Mundo, chamado *Suryaloka*, com linhas que denotam os limites de cada célula. Tanto as dimensões do Mundo (em número de células) quanto as de cada célula (em número de *pixels*) são parametrizáveis. Um objeto no Mundo ocupa uma determinada área, onde a unidade de comprimento dessa área é a célula. Esse modelo estrutural do Mundo facilita bastante a localização de objetos, bem como a detecção de colisão entre eles.

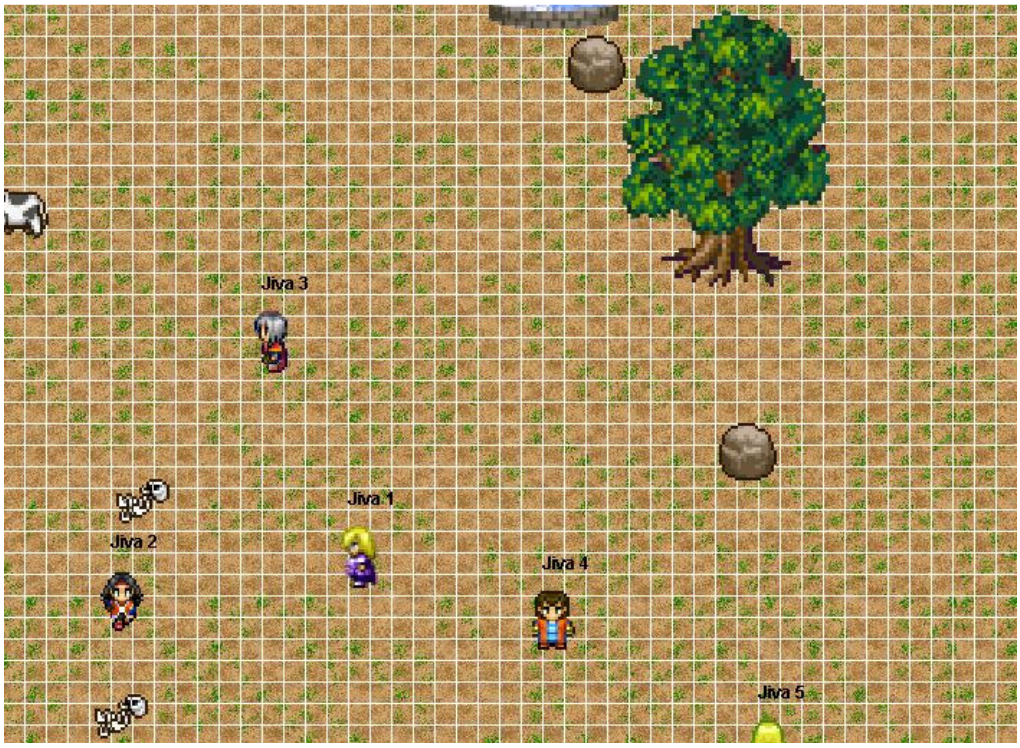


Figura 17. *Screen-shot* de Vidya, mostrando o Mundo do jogo com um *grid* bidimensional de células.

3.2.2 Caracterização dos Objetos do Mundo

Todo objeto que pode fazer parte do Mundo (`World`) deve herdar da classe abstrata `WorldObject`. Por ser uma classe abstrata, não pode ser instanciada, o que é adequado, pois `WorldObject` não representa nenhum objeto em particular. Essa classe é útil para unificar o tratamento para com todos os objetos por parte de `World`, além de ser formalmente uma prática correta.

Cada `WorldObject` possui como atributos um *sprite*, um retângulo de colisão (que delimita sua área de ocupação no Mundo e evita que outros objetos entrem nessa área), sua posição atual no Mundo, sua posição anterior no Mundo (para que, no caso de colisão com outros objetos, possa retornar a ela) e um objeto da classe `WorldObjectDescriptor`.

Posições no Mundo são representadas por duas variáveis, x e y, ou seja, pontos (x, y), onde o ponto (0, 0) é o canto superior esquerdo do Mundo; x cresce para a direita e y cresce para baixo.

A classe `WorldObjectDescriptor` é um simples descritor do objeto, levando consigo somente a posição atual do objeto no Mundo e seu tipo (que é um tipo enumerado chamado `WorldObjectType`). Pode ser vista como uma versão "leve" do objeto em si, evitando que outras entidades manipulem o objeto todo (que possui informações irrelevantes). Por exemplo, quando um ser vivo artificial percebe o Mundo ao seu redor, na verdade ele está obtendo o `WorldObjectDescriptor` de cada objeto dentro de seu limite de percepção, não os próprios objetos.

A enumeração `WorldObjectType`, que pertence ao componente *utils*, enumera todos os tipos possíveis de objetos do Mundo em Vidya. A seguir, explicaremos todos os possíveis tipos de objetos do Mundo.

Pedras

As pedras não possuem nenhuma função dentro do Mundo. Servem como elemento enriquecedor do ambiente gráfico, ocupando um espaço. Nas porções centro-superior e médio-direita da Figura 14 podemos visualizar duas pedras.

Fontes D'água

Suryaloka é uma ilha. A água salgada do mar ao redor de *Suryaloka* é imprópria para o consumo dos seres vivos do Mundo. Mas existem fontes d'água doce em terra, nas quais os seres artificiais bebem água para se manterem vivos. A porção centra-direita da Figura 18 exibe um *Jiva* bebendo água numa fonte.

Plantas

As plantas são os seres vivos mais simples no Mundo do jogo. Possuem um único atributo: sua vitalidade, que é interpretada em Vidya como o tempo restante de vida de um ser. A vitalidade

das plantas decresce linearmente com o tempo e, quando chega em zero, a planta morre, tornando o solo que ocupava fértil.

Mas, antes de morrer, dado um tempo após o seu nascimento, a planta se reproduz, gerando mais quatro plantas no máximo nas proximidades. Uma planta só nasce em solo fértil, logo nem todas as plantas geram quatro novas plantas ao seu redor, mas somente as que estariam numa posição em que o solo é fértil. Os predadores naturais das plantas são as ovelhas e as vacas, que comem plantas para adquirir mais energia.

Figura 18. *Screen-shot* de Vidya, mostrando um *Jiva* bebendo água numa fonte em terra pois a água do mar lhes é imprópria para consumo.

Árvores

As árvores (Figura 19.b) nascem de frutos (Figura 19.a) e, de tempos em tempos, geram frutos (no máximo 4). Cada fruto poderá dar nascimento a uma nova árvore, caso o solo em que ele esteja localizado seja fértil. Se o solo não é fértil, o fruto não gera uma nova árvore, mas o solo se torna fértil logo após a decomposição do fruto no local.

Após um tempo de vida as árvores morrem, adquirindo dentro do jogo o aspecto de árvore morta sem folhas (Figura 19.c). Esse estado de árvore morta evolui para um estado em que só aparece a raiz da árvore morta (Figura 19.d), que após um tempo desaparece por completo. Os predadores naturais de árvores são ovelhas, vacas e *Jivas*, pois comem seus frutos.

Ovelhas

Uma ovelha nasce do cruzamento de duas outras ovelhas de sexos opostos. Dentre os diversos atributos de uma ovelha podemos destacar os seguintes: vitalidade, hidratação, energia e sexo. Quando a hidratação está abaixo de um limiar parametrizável, a ovelha sente sede e então busca uma fonte d'água. Quando seu nível de energia está abaixo também de um limiar, a ovelha sente fome e busca por comida, que podem ser plantas ou frutos (que caem das árvores, como vimos). As ovelhas observam os objetos do Mundo que estão dentro de seu retângulo de percepção.

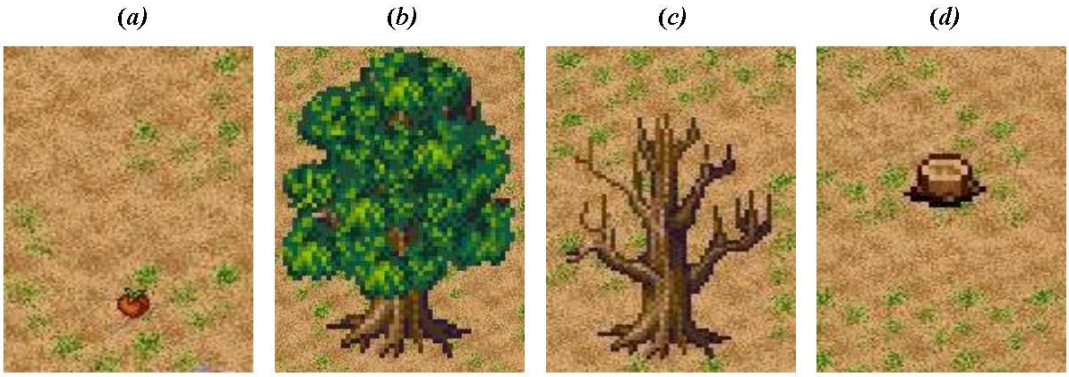


Figura 19. Ciclo de vida de uma árvore em Vidya: (*a*) fruto; (*b*) árvore formada; (*c*) árvore morta; (*d*) raiz de árvore.

Quando uma ovelha macho encontra uma ovelha fêmea em período fértil, cruza com ela. Assim também como uma ovelha fêmea busca um macho para cruzar quando está em período fértil. A fêmea passa então um período prenha, até que dá nascimento a uma nova ovelha habitante do Mundo.

As ovelhas têm como predadores naturais os lobos e os *Jivas*. Sempre que uma ovelha vê algum predador, foge no sentido inverso de sua direção de aproximação. Se vê muitos predadores, foge do mais próximo. Se a ovelha é atacada por um predador, perde vitalidade, podendo chegar à morte. Quando a vitalidade, a hidratação ou a energia de uma ovelha chega em zero, a ovelha morre, aparecendo somente seus restos mortais no Mundo, tornando o solo imediatamente abaixo da ovelha fértil.

A Figura 20 mostra uma ovelha viva e os restos mortais de uma outra ovelha.

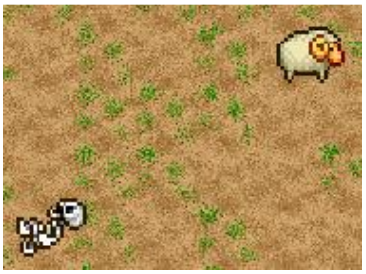


Figura 20. Uma ovelha viva (canto superior direito) e outra ovelha morta (canto inferior esquerdo).

Vacas

As vacas têm o comportamento quase semelhante ao das ovelhas. Se reproduzem da mesma forma, possuem os mesmos atributos, se alimentam de forma igual e têm os mesmos predadores. As diferenças entre vacas e ovelhas é que as vacas possuem mais vitalidade e energia, bem como possuem um maior retângulo de percepção que as ovelhas (vêem mais).

O agente das vacas e das ovelhas (e também dos lobos, que são explicados no próximo tópico) são agentes inteligentes puramente reativos, que se comportam de forma semelhante para situações semelhantes, sem aprendizado no tempo. A Figura 21 mostra uma vaca e outros seres vivos do Mundo.

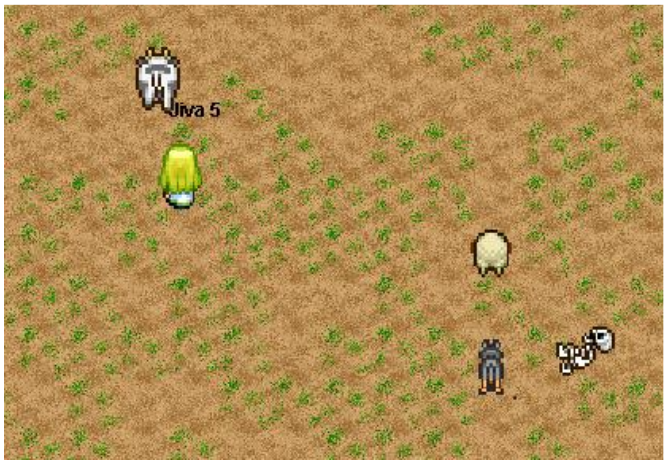


Figura 21. No canto superior esquerdo, uma vaca sendo perseguida por um *Jiva*. No canto inferior direito, uma vaca morta (*sprite* semelhante a de qualquer ser morto, incluindo ovelha, lobo e *Jiva*) e um lobo perseguindo uma ovelha.

Lobos

Um lobo nasce do cruzamento de dois outros lobos de sexos opostos. Possui como principais atributos a vitalidade, hidratação, energia e sexo. Se alimentam de vacas e ovelhas, e eventualmente podem atacar *Jivas* para defender-se (caso o *Jiva* se aproxime demais), podendo chegar a alimentar-se de *Jivas* mortos caso esteja com fome (mas jamais mataria um *Jiva* para se alimentar). Observam os objetos do Mundo dentro de um retângulo de percepção, assim como as ovelhas, vacas e jivas. A Figura 22 mostra dois lobos conjuntamente com outros objetos do Mundo.

Os lobos se reproduzem de forma semelhante às ovelhas e às vacas, mudando somente os períodos de fertilidade e gestação das fêmeas.

Os únicos predadores naturais dos lobos são os *Jivas*, apesar de que a carne de lobo não é preferencial para os *Jivas*. Sempre que vêem *Jivas*, os lobos fogem na direção inversa; mas se os *Jivas* chegam perto demais (5 células de proximidade), os lobos passam a atacar. Quando um dos atributos vitais do lobo chega a zero, o lobo morre, quando então seus restos mortais aparecem no Mundo, na posição em que morreu.

Figura 22. *Screen-shot* de Vidya mostrando dois lobos.

Jivas

O *Jiva*, assim como a ovelha, a vaca e o lobo, também é um agente inteligente. Entretanto, o *Jiva* é um agente inteligente autônomo evolutivo, porque ele é capaz de tomar decisões inteligentes de uma forma independente aprendendo com suas decisões (evoluindo). Além disso, são seres sociais que se comunicam entre si para tomadas de decisões cojuntas baseadas em coação, formando um sistema multiagente. A Figura 23 mostra um clã de *Jivas* agindo no Mundo.

Figura 23. Um clã de *Jivas* agindo cooperativamente durante a caça de uma ovelha.

Diferentemente dos outros seres do Mundo, o *Jiva* não se reproduz. Quando o jogador inicia o jogo, ele tem sob sua responsabilidade cinco *Jivas* de um mesmo clã e a tarefa de ajudá-los a sobreviver dentro das adversidades de *Suryaloka*. O *Jiva* pode se alimentar de frutos (ganham vitalidade com isso), ovelhas, vacas, lobos e de outros *Jivas* que não sejam do seu clã. Contudo, comer lobos e *Jivas* faz com que o *Jiva* perca vitalidade, apesar de algum ganho de energia. Como a vitalidade, na ponderação dos Jivas, tem um peso muito maior que a energia, é razoável que eles dêem preferência a outros tipos de alimento que não sejam lobos e *Jivas*.

Como outros agentes, ele percepe o ambiente, realiza um processamento interno, seleciona uma "boa" ação, e atua no ambiente. A percepção do *Jiva* é parcial, ou seja, ele não percebe o ambiente inteiro, mas somente dentro de um limite variável de percepção (um retângulo de percepção que é dinâmicamente atualizado, além de poder variar de dimensões dependendo das variáveis de estado do Jiva).

O que definitivamente difere o *Jiva* dos outros agentes dentro do jogo Vidya é que ele não é um agente reativo simples. Isso significa que o processamento interno realizado para se obter uma ação adequada para uma dada percepção não tem como base um mapeamento direto de estados para ações, como acontece com os outros seres de Vidya. Existe sim um processamento inteligente *online* (aprendizado contínuo durante ações). Esse processamento inteligente é o que garante a não previsibilidade do comportamento do *Jiva*, o que é uma característica interessante para o jogo proposto.

3.3 O Problema do Módulo de Decisão Inteligente do *Jiva*

É fácil deduzir que o problema básico do *Jiva* é executar a melhor ação no ambiente para uma dada percepção, tentando minimizar uma função de custo (ou maximizar uma função de ganho). As ações do *Jiva* devem ser vistas como operações de controle sobre o ambiente para a criação de condições ideais para a sua sobrevivência.

A tarefa do módulo de decisão inteligente do *Jiva* é selecionar (ou decidir) a melhor ação a ser executada pelo agente do *Jiva*. Essa tarefa de seleção de ação é realizada usando AG (técnica de IA explanada na Seção 2.2). Nessa seção, focamos no comportamento individual do *Jiva*, ou seja, sem levar em consideração as ações que são motivadas pelo seu contexto social. Apesar disso, em última análise, a ação do *Jiva* é decidida individualmente, no sentido de que é o próprio *Jiva* quem toma a decisão de ser egoísta (agir para si) ou altruísta (agir para o grupo).

O *Jiva* tem um conjunto muito grande de ações que podem ser executadas, como é definido mais adiante. A seleção da melhor ação pode consumir muito tempo e processamento computacional usando técnicas tradicionais, em que todas as ações têm de ser avaliadas e qualificadas. Com AG nós diminuímos o espaço de busca, e as escolhas convergem para a melhor ação. Outra característica importante é que já a primeira geração é uma solução plausível para o

problema de decisão. Uma ação é então mapeada de um indivíduo no AG e a população do AG é um subconjunto menor para busca das ações disponíveis.

O *Jiva* percebe um conjunto de objetos do mundo dentro de um quadrado de percepção. Este limite de percepção depende da vitalidade do *Jiva*. No começo, o *Jiva* tem o máximo de vitalidade e nível de percepção, e vai perdendo sua capacidade de percepção proporcionalmente com o decaimento da vitalidade. A vitalidade do *Jiva* é interpretada, aqui, como o tempo restante de vida do *Jiva*. Outros fatores podem vir a interferir também na sua capacidade perceptiva.

O quadrado de percepção do *Jiva* define um conjunto de células que o *Jiva* pode perceber. Essas células também definem o espaço completo de posições futuras possíveis para o *Jiva* (as possíveis posições que o *Jiva* pode ocupar no próximo instante de tempo) na primeira vez que um estado de percepção é alcançado. A Figura 24 mostra o quadrado de percepção do *Jiva*, os objetos que são percebidos e o conjunto de células do quadrado.

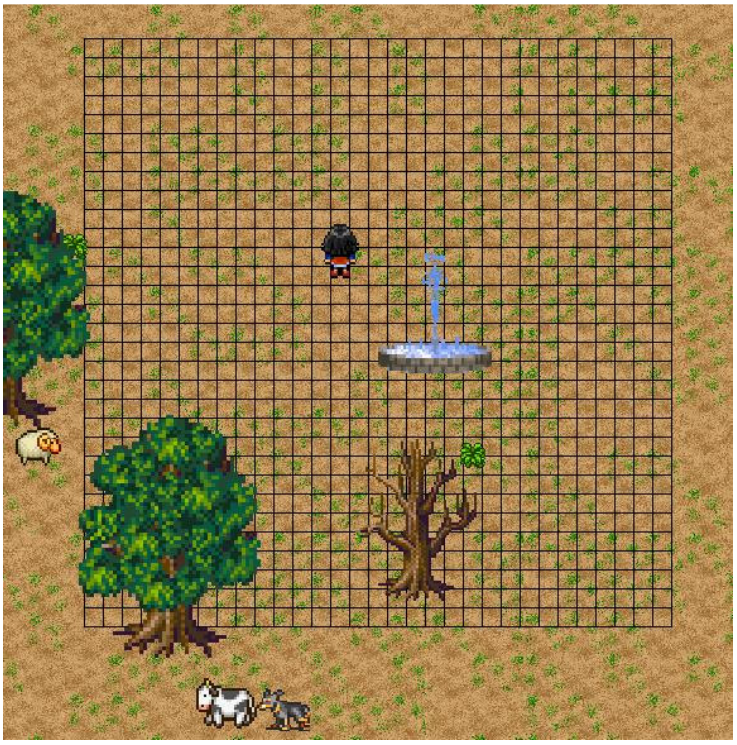


Figura 24. *Screen-shot* do mundo do jogo, mostrando o quadrado de percepção do *Jiva*. Os objetos dentro do quadrado estão sendo percebidos (uma fonte d'água, uma planta, uma árvore morta e uma árvore). Os pequenos quadrados que compõem a percepção são as céulas de percepção. Fora do quadrado de percepção, na parte inferior da imagem, um lobo (predador) está caçando uma vaca (sua presa).

A percepção do *Jiva* é caracterizada por um conjunto de objetos do Mundo. Cada objeto na percepção é caracterizado pelo seu tipo (vide tipos de objetos do mundo na Seção 3.2.2) e posicionamento (relativo à posição do *Jiva*).

No presente trabalho, o *Jiva* pode perceber objetos oclusos, ou seja, ele pode ver todos os objetos dentro de seu quadrado de percepção, mesmo se alguns objetos estiverem atrás de outros (escondidos). Incerteza causada por oclusão de objetos é um tópico relevante [56], no entanto, nós deixamos isso para um trabalho futuro.

O agente do *Jiva*, após ter percebido o ambiente e ter criado uma representação interna da percepção, selecionará uma boa ação para aquela percepção. Uma ação é definida como a célula de destino para a qual o *Jiva* irá se mover. Então, nós caracterizamos uma ação pelo par de coordenadas (x, y), significando que o *Jiva* irá se mover para aquela posição no próximo passo de tempo, como mostrado na Figura 25.

Figura 25. Célula de destino para a qual o *Jiva* irá migrar. Na célula de destino (pintada de vermelho, na parte superior da imagem) há uma fruto, e o *Jiva* vai comê-lo.

O valor semântico associado a uma ação é aqui assumido como implícito e depende do objeto que está ocupando a célula de destino. Por exemplo, se o *Jiva* migra para uma célula onde está um fruto (Figura 25), o valor semântico associado a essa ação é "comer o fruto". Senão, se na célula de destino está um lobo, o efeito semântico é "ferir o lobo". Se a célula de destino está vazia (nenhum objeto está lá), o valor semântico associado é apenas "caminhar até a posição de destino". Para cada tipo de objeto na célula de destino há um diferente valor semântico associado. Para alguns tipos de objetos, o valor semântico associado pode ser o mesmo (*e.g.* ovelha e vaca).

As implicações semânticas das ações nos atributos vitais do *Jiva* também são dependentes dos objetos que ocupam a célula de destino (*e.g.* comer um fruto aumenta a vitalidade e a energia, enquanto comer um lobo, apesar do mesmo proporcionar um ganho de energia maior que o do fruto, faz o *Jiva* perder vitalidade).

3.4 Resolvendo o Problema de Decisão Inteligente do *Jiva* com Computação Evolutiva

Uma única instância de AG não pode resolver o problema como posto na Seção anterior, *i.e.* selecionar a melhor ação para todas as possíveis situações. Isso porque uma boa ação para uma dada percepção pode não ser (provavelmente não será) boa para uma outra percepção. Cada percepção é um problema de decisão diferente. Então instâncias diferentes de AG têm de ser criadas para cada uma dessas diferentes percepções. O módulo de decisão inteligente do *Jiva* possui uma estrutura que mapeia percepções em instâncias de AGs num mapeamento de um-para-um.

Cada instância de AG tem a função de selecionar a melhor ação para uma dada percepção, seguindo o método de seleção natural que caracteriza um AG. Isso inclui a avaliação, cruzamento e mutação de indivíduos.

Quando avaliar uma ação, o AG deve se preocupar não somente com o custo imediato (a curto prazo) que surge da realização dessa ação, mas inferir um custo a longo prazo para ela. Algumas vezes nós vamos referenciar essa avaliação a longo prazo aqui como uma avaliação "*F* passos no futuro", onde *F* é a quantidade de passos que o AG pode ver no futuro para fazer a estimativa do custo a longo prazo. Esse custo a longo prazo é um algoritmo de planejamento (algoritmos de planejamento foram definidos na Seção 2.1.3) e é a forma que o *Jiva* tem de considerar custos futuros também na sua tomada de decisão. Esse custo a longo prazo pode ser inferido porque o módulo de decisão inteligente do *Jiva* pode simular o progresso das ações no futuro. Claro, o custo a longo prazo estimado tem de ser confrontado com a experiência do *Jiva*, ou seja, o custo real obtido por interagir com o ambiente.

Usando Algoritmos Genéticos, nós temos um método convergente (*i.e.* que tende para a melhor solução) com uma menor busca. Supondo um GA com uma população de 40 indivíduos, aplicado ao problema exposto no parágrafo anterior, nós temos uma redução significativa de 90% no espaço de busca. A Figura 26 mostra a distribuição espacial inicial da população para uma dada percepção do *Jiva* e o indíviduo vencedor (célula de destino, em vermelho).

Localizado em um estado, o algoritmo do *Jiva* chama a instância de GA associada ao estado para selecionar uma ação. Cada vez que esse estado é alcançado, a instância de GA avança uma geração. Note-se que mesmo na primeira geração já existe uma solução convergente. Esse fato é possível pelas tarefas comuns em todos os AG: avaliar, cruzar e mutar indivíduos.

Antes de tudo, a população é avaliada para a seleção da melhor ação que será executada pelo *Jiva*. A fase de avaliação só está completa quando a ação é executada e o *Jiva* tem um custo imediato real obtido pela experiência. Então, a fase de avaliação é dividida em dois momentos: antes da seleção e após a ação.

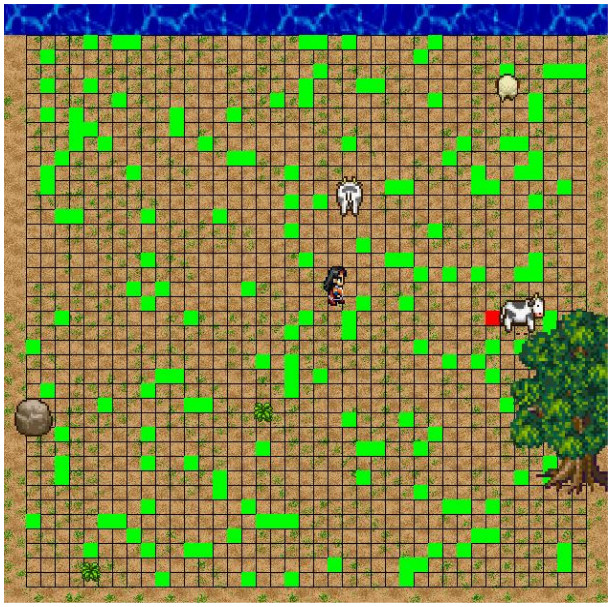


Figura 26. Distribuição espacial inicial da população para uma dada percepção. Podemos ver as células que estão dentro da área de percepção de dimensões 41 × 41, a distribuição de população (168 indivíduos, ou seja, 10% do espaço total de busca, em células verdes) e a célula de destino (canto médio-direito, em vermelho), selecionada através de uma simulação analítica considerando 5 passos no futuro. O *Jiva* (centro) está caçando uma vaca (direita).

Antes da seleção, cada ação é internamente simulada *F* passos no futuro. A ação que obtiver a melhor qualificação (maior ganho – maior aptidão) na simulação é então selecionada. A Figura 27 exibe o algoritmo de simulação para uma ação (em pseudo-código). Depois de obter a qualificação (custo a longo prazo inferido) para cada ação, a ação que tiver a maior qualificação será selecionada e executada pelo *Jiva*.

A ação vencedora foi selecionada tendo somente como base a simulação interna que o *Jiva* realiza para saber qual a melhor ação, levando em consideração *F* passos no futuro (custo simulado a longo prazo). Contudo, essa simulação não considera a experiência de vida do *Jiva*. Somente após ter sido executada é que a ação é reavaliada com base no custo real de interação com o Mundo. Esse custo real tem o papel de reforçar (caso positivo) ou enfraquecer (caso negativo) a aptidão da ação realizada, de forma que ela perpetue as suas características ou não para as próximas gerações do AG, respectivamente. É dessa forma que o *Jiva* aprende a melhor agir no ambiente com o passar do tempo.

```
parameters: currentAction, currentState, F
return:     qualification

var qualification := 0

iterate 1 to F:
    /* Qualification update.                    */
    /* costToGo returns the cost to realize     */
    /* the current action in the current state. */
    qualification := qualification +
                costToGo(currentState, currentAction)

    /* New state reached */
    var newState := updateState(currentState,
                                currentAction)
    /* GA instance for the new state reached */
    var gaInstance := getGAFor(newState)
    /* Best action for the GA instance */
    var newAction := getBestActionFor(gaInstance)

    /* Makes current action the new action */
    currentAction := newAction
    /* Makes current state the new state */
    currentState := newState

/* Returns accumulated qualification */
return qualification
```

Figura 27. Algoritmo para a simulação (em pseudo-código) de uma ação *F* passos no futuro.

O custo para a realização de uma determinada ação é obtido subtraindo-se a condição vital atual do *Jiva* e sua condição vital prévia, imediatamente antes da realização da ação. O que chamamos de condição vital do *Jiva* (CV) é uma média ponderada da vitalidade (V – com um peso de 50%), hidratação (H – com um peso de 30%) e energia (E – com um peso de 20%) do *Jiva*, como mostra a Equação 1. Esses pesos foram atribuídos *ad hoc,* mas podem vir a ser parametrizados no futuro.

$$\mathbf{CV} = 0{,}5 \times \mathbf{V} + 0{,}3 \times \mathbf{H} + 0{,}2 \times \mathbf{E} \qquad \textbf{(Eq. 1)}$$

Depois dessa pós-avaliação, a ação vencedora poderá permanecer sendo a melhor ação ou poderá perder o posto para outra ação melhor qualificada. Então, essa pós-avaliação tenta corrigir erros de inferência na simulação, re-ordenando, em termos de aptidão, os indivíduos da

população do AG para o dado estado. Esses indivíduos estão agora prontos para a fase de cruzamentos e mutações.

Uma ação para o *Jiva* é caminhar para uma célula de destino. Então, ações são representadas por um par de coordenadas (x, y), indicando esse destino, como já foi mencionado. O método de seleção usado para escolher pares de ações (indivíduos do AG) que irão cruzar entre si foi o da Roleta. Pares de ações são selecionados e cruzados, gerando novas ações que irão compor a próxima geração da população. O operador de mutação é também aplicado à população. A probabilidade de mutação é, no caso do *Jiva*, 1/32, ou seja, em média, para cada 32 ações 1 é mutada.

O uso de AG no módulo de decisão inteligente do *Jiva* objetiva reduzir consideravelmente o espaço de busca na seleção de ações de uma maneira convergente, usando princípios de seleção natural e gerando boas soluções logo na primeira geração de indivíduos.

Cada vez que um estado é alcançado, uma nova geração de melhores ações sobrepõe a geração anterior. As ações antigas são avaliadas considerando seus prováveis custos de longo prazo (*F* passos no futuro, onde *F* é um parâmetro do jogo), conseguido através de simulações de progresso, e a ação selecionada (a que obteve a melhor qualificação) é re-qualificada pelo seu custo imediato real após ser executada. Então, essas ações antigas são submetidas às operações de cruzamento e mutação, resultando em uma nova geração de ações melhores qualificadas.

Há uma forte tendência em jogos para computador que os mundos dos jogos se tornem cada vez mais complexos [11] (isso é visto na última geração de jogos para computador). Essa crescente complexidade tem dificultado a programação explícita de comportamentos de agentes (nos casos de agentes puramente reativos) ou, quando IA está de fato envolvida, a obtenção de bases de conhecimento representativas do mundo do jogo específico, porque o número de situações diferentes que acontecem em mundos de jogos verdadeiramente complexos é muito grande, considerando inclusive situações causadas por jogadores que são, pelo menos na primeira análise, não-determinísticos.

A criação de agentes adaptativos capazes de aprender em situações novas é uma boa forma de contornar essa dificuldade. O agente do *Jiva* e módulo de decisão inteligente é uma instância do uso eficiente de Computação Evolutiva em jogos para computador, especificamente na modelagem de comportamento de agentes autônomos. Esse módulo de decisão inteligente poderá ser reusado em outros jogos para computador, em agentes autônomos que necessitem ser adaptativos nas adversidades do mundo do jogo e capazes de aprender em situações novas.

3.5 Comportamento Social dos *Jivas*

Até então nada foi comentado sobre o comportamento social do *Jivas*, que é também um dos objetos de estudo deste trabalho. De fato, os *Jivas* de um mesmo clã formam um sistema multiagente. Isso só é possível mediante um modelo de compartilhamento de informações que, no caso dos *Jivas*, está baseado num protocolo de comunicação.

Segundo Piaget, há dois tipos de relações sociais – a coação social e a cooperação [57]. Ele define a coação social como toda relação entre dois ou mais indivíduos na qual intervêm elementos de autoridade ou prestígio. E a cooperação como a relação entre dois ou mais indivíduos que acreditam ser iguais entre si, sem nenhum tipo de hierarquia social.

A idéia central que permeia o comportamento social dos *Jivas* é o conceito de coação, obviamente aplicado a um nível bem mais básico de operação que o da sociedade humana. A implementação da relação de cooperação entre os *Jivas* é deixado com um trabalho para o futuro.

Em qualquer relação de coação há a hierarquia. Aqui, usamos um único contexto hierárquico para qualificar a posição de um *Jiva*. Diz-se que um *Jiva* é superior a um outro se é mais adaptável. A adaptabilidade de um *Jiva* no Mundo do jogo é medida pela sua condição vital (CV – cujo cálculo está demonstrado na Equação 1). Quanto maior a condição vital de um *Jiva*, maior é sua posição dentro da hierarquia social.

Como vimos, a coação social pode surgir devido a ordens que provenham de alguma autoridade, ou devido à influência de indivíduos de maior prestígio social. Inicialmente, pensamos no modelo de autoritarismo. Contudo, sob uma reflexão maior, verificamos que esse modelo tiraria a flexibilidade do sistema e faria com que os *Jivas* mais fracos só "pensassem" quando estivessem sozinhos (sem companhia de outros *Jivas* do mesmo clã) ou quando se tornassem mais fortes. Isso comprometeria a autonomia do indivíduo, o que não é desejável.

Então, ficamos com o segundo modelo, no qual os *Jivas* atribuem prestígio social aos mais fortes, e consideram as decisões deles quando vão tomar suas próprias decisões. Isso dá ao *Jiva* a possibilidade de agir ou coagir socialmente, com autonomia de escolha e, o mais importante, sem degenerar seu próprio módulo de decisão inteligente. O módulo de decisão inteligente do *Jiva* antecede a coação social, visto que, dentre as diversas possibilidades de ação, estão já inclusas as que são provenientes das relações de coação.

A comunicação entre os *Jivas* ocorre no sentido dos *Jivas* mais adaptáveis para os menos adaptáveis. Tomando aqui a adaptabilidade como o prestígio social, podemos dizer os *Jivas* de menor prestígio consideram sempre as ações dos *Jivas* de maior prestígio que estão dentro de seu limite de percepção, antes mesmo de tomar suas próprias decisões. E, quanto mais positivas forem as recompensas pela coação, maior será o nível de sociabilidade para um *Jiva*, ou seja, mais ele considerará as ações realizadas pelos *Jivas* de maior prestígio que estão dentro de seu limite perceptivo.

Tecnicamente, quando um *Jiva* está numa determinada configuração de percepção e nela identifica outros *Jivas* do seu clã, ele busca os mais adaptáveis que ele, observa suas decisões e inclui as decisões deles como opções dentro da instância do AG para a dada percepção. Quanto maior é o nível de sociabilidade do *Jiva*, mais propenso ele é escolher as ações dos *Jivas* mais adaptáveis que estão dentro da percepção. Obviamente, ele também comunicará sua melhor decisão para os *Jivas* menos adaptáveis que ele que estão por perto.

Os *Jivas* são seres, por enquanto, não-cooperativos. A cooperação só surge porque o comportamento emergente de uma sociedade é substancialmente superior ao dos indivíduos em

particular, o que aumenta suas chances de sobrevivência. A sociedade atual de *Jivas* pode ser um ambiente propício para o estudo de jogos não-cooperativos [14], dentro do contexto da Teoria dos Jogos [13]. E um ponto bem interessante a ser tocado neste sentido seria o estudo mais específico das disposições de equilíbrio (ou equilíbrios de Nash) [14], se houver, que surgem na dinâmica social dos *Jivas* e no ecossistema no qual ele está inserido, considerando suas ações e coações (*i.e.* a influência de suas ações no equilíbrio social e no ecossistema global).

3.6 Um Novo Modelo de Interação entre Jogador e Personagem

Como já foi notado, o personagem principal no jogo Vidya é o *Jiva*. Ele é o ponto focal entre o jogador e o jogo.

Tradicionalmente, interação entre jogador e personagens em jogos para computador acontece de uma forma muito direta. Através dos controles, o jogador pode operar quase completamente (senão completamente) o comportamento dos *avataras*.

No jogo Vidya, o jogador é o *deva* de um clã de *Jivas*, cuja função é mentorar o clã e promover sua sobrevivência no mundo do jogo. Essa é uma tarefa difícil, porque os *Jivas* são agentes inteligentes autônomos, e podem escolher a ação a ser executada, a despeito da ação sugerida pelo *deva* (o jogador).

As instruções não detalhadas que o jogador fornece ao clã de *Jivas* (individualmente para cada integrante) é conhecida no jogo como *vidya*. Uma *vidya* aparece dentro de cada *Jiva* como uma "intuição". O objetivo do jogador é fornecer boas *vidyas* para os *Jivas*, que os ajudarão a sobreviver.

Quando o jogador intercepta um *Jiva* para prover *vidya*, o *Jiva* pára de agir e então o jogador pode sugerir uma célula de destino para ele seguir. Entre outras células de destino possíveis, que são os indivíduos da população da instância do GA para aquela percepção particular, a *vidya* irá seletivamente competir pela melhor qualificação. Se ela estiver, pelo menos, entre as melhores qualificadas, irá passar suas características para a próxima geração, e ajudará o aprendizado a longo prazo do *Jiva*. Esse é o principal objetivo da *vidya*.

Capítulo 4

Experimentos e Resultados

Os experimentos e resultados aqui inclusos foram usados para avaliar dois aspectos do Vidya: (1) o comportamento do algoritmo evolutivo inteligente proposto no Capítulo anterior como forma de dotar *Jivas* de autonomia e (2) a aceitação do jogo Vidya por parte de eventuais seus jogadores.

4.1 Avaliação do Algoritmo Evolutivo Inteligente Proposto

O algoritmo evolutivo inteligente proposto na Seção 3.4 tem como objetivo a implementação do comportamento inteligente do *Jiva*, e é baseado em Algoritmos Genéticos. No algoritmo proposto, todas as ações são mapeadas em indivíduos de uma dada população. A aptidão de cada indivíduo é obtida através de uma simulação *F* passos no futuro e através da experiência imediata do *Jiva* [58]. Então, as melhores ações passarão suas características para a próxima geração (de soluções – ações).

O algoritmo foi avaliado sob dois aspectos: (i) o aspecto de desempenho computacional e (ii) o aspecto de inteligência proporcionada por sua adoção. Nesse último caso, utilizou-se a adaptabilidade observável do *Jiva* como um meio indireto de avaliar a inteligência do algoritmo. No caso específico do algoritmo inteligente proposto, essa é a melhor forma de avaliar a sua inteligência, pois não há uma base de dados de teste para inferência da taxa de erro do mesmo. Em algoritmos em que as fases de treinamento, testes e aplicação são bem definidas (aplicações que usam Redes Neurais Artificiais, por exemplo), a suas taxa de erro são obtidas diretamente.

Dentre os parâmetros do sistema, identificamos dois parâmetros principais: o tamanho da população do AG – *PS*; e o número de passos no futuro que a simulação pode antever – *F*. A partir daí, variamos esses dois parâmetros para observar o comportamento do algoritmo quanto a ações possíveis.

O conjunto de indivíduos, do qual a população do AG é um subconjunto, tem tamanho igual à área perceptiva do *Jiva*. Nós assumimos uma área de percepção constante de 961 células (quadrado 31 × 31) nesses experimentos, de forma que eles pudessem ser comparados entre si. Os valores para *PS* assumidos foram 10%, 20% e 40% de todas as ações possíveis, ou seja, *PS* = 96, *PS* = 192 e *PS* = 384, respectivamente. Os valores para *F* assumidos foram *F* = 2, *F* = 4 e *F* = 6.

Destacamos que os valores absolutos dos resultados não são tão importantes aqui, haja visto não serem diretamente o objetivo do trabalho. Na verdade, os experimentos propõem uma análise comparativa do mesmo algoritmo com diferentes valores assumidos pelos parâmetros *PS* e *F*. Conjuntamente esses reultados demonstram apenas boas configurações de parâmetros que melhoram o desempenho (computacional e inteligente) do *Jiva*.

O *hardware* usado nas simulações foi um computador pessoal comum, AMD DURON™ 1.6 GHz, com meros 352MB de RAM.

4.1.1 Desempenho Computacional do Algoritmo

O gráfico da Figura 28 mostra o desempenho computacional (tempo de processamento), em milisegundos, do algoritmo proposto para os parâmetros selecionados; ou seja, variações em *F* (passos antevistos no futuro) combinado com *PS* (tamanho da população do GA).

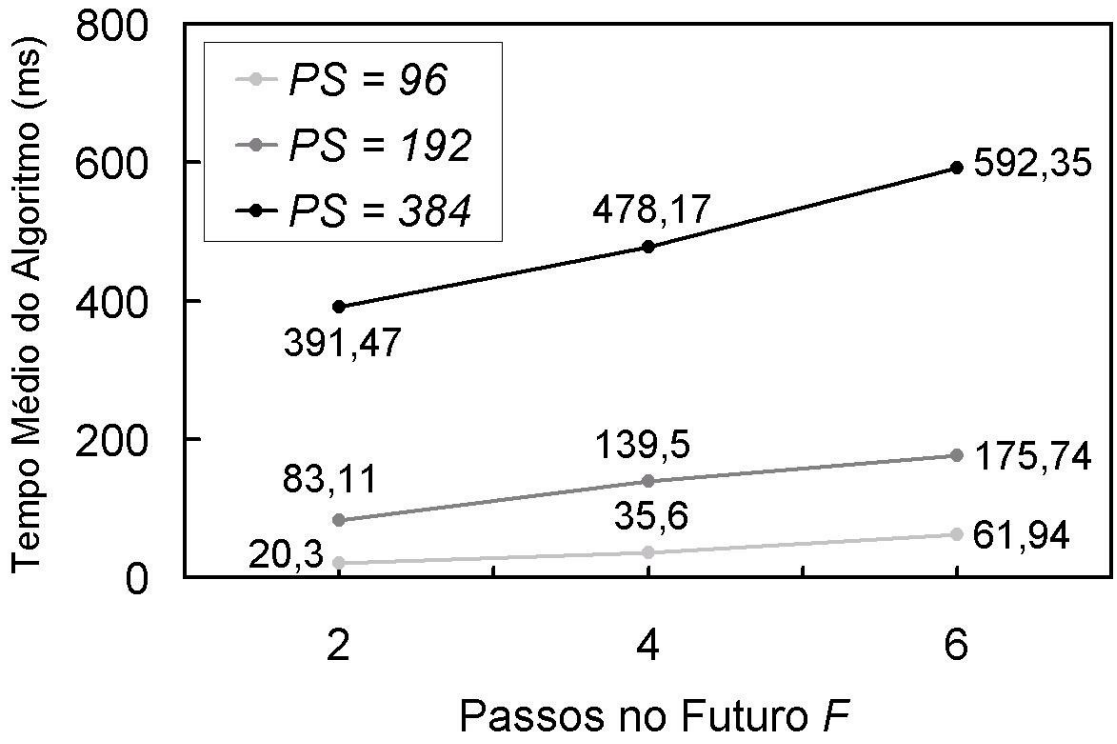


Figura 28. Variações de desempenho do algoritmo devido à mudanças nos parâmetros selecionados.

As três linhas do gráfico da Figura 28 são tendências do algoritmo quanto ao desempenho. Para cada tendência, o tempo de processamento aumenta em função de *F*. O crescimento observado, levando em consideração as combinações de variações no experimento realizado, tende a ser linear.

Cada ponto no gráfico da Figura 28 é uma média aritmética de 100 execuções do algoritmo. Para obter esses dados, tivemos que adicionar rotinas extras ao código principal. No entanto, os resultados referidos acima estão relacionados somente com o código examinado (o algoritmo inteligente).

O comportamento exibido é previsível para o algoritmo, no que diz respeito à carga de processamento que deveria crescer com *F*. O fato do crescimento ter sido linear é um bom indicador da escalabiliadade da solução. As diferenças nos valores medidos para populações de diferentes tamanhos, para os mesmo valores de *F*, são previsíveis também, e têm também uma tendência de crescimento igualmente linear.

Os valores absolutos obtidos nos experimentos também revelam que o tempo de processamento como um todo do algoritmo não é grande (para os casos de teste, não chegaram a exceder, no mais complexo dos casos simulados, 600ms no pior caso), e é perfeitamente aplicável em problemas reais que tenham limitações de tempo de resposta.

4.1.2 Medições de Inteligência do *Jiva*

Medir inteligência é difícil qualquer que seja o domínio e contexto. Para realizarmos essa tarefa neste trabalho, tivemos que usar um método indireto. Para o caso do módulo de decisão inteligente do *Jiva*, avaliamos o ganho (que pode ser positivo ou negativo) obtido pelo *Jiva* durante um período de tempo interagindo com o ambiente. Ou seja, usamos a capacidade de adaptação do *Jiva* no jogo Vidya para prover, indiretamente, o desempenho qualitativo do algoritmo - *i.e.* o comportamento inteligente.

Tomando por base que as características mais relevantes do *Jiva*, aquelas que mais interessam para a avaliação da adaptabilidade do *Jiva* são: vitalidade, hidratação e energia, utilizamos uma métrica que integrasse esses valores. Essa métrica foi a média ponderada desses três atributos – condição vital do *Jiva*, como já foi explicado na Seção 3.4. Usamos os valores absolutos da condição vital do *Jiva* para medir sua adaptabilidade no ambiente.

A idéia utilizada foi que, se medíssemos o ganho obtido na condição geral pelo *Jiva* durante um período de tempo, nós poderíamos dizer que esse valor representa aproximadamente a sua capacidade de sobrevivência (ou seja, sua adaptabilidade, indiretamente sua inteligência). Isso porque a métrica proposta indica quão bem ele agiu na obtenção de recursos adicionais para sua própria sobrevivência. Note-se que, em nossos experimentos, não avaliamos o comportamento social dos *Jivas*, mas tão somente o comportamento individual do *Jiva* (*i.e.* preocupa-mo-nos com a sua sobrevivência individual). A análise de qualidade do comportamento social emergente dos *Jivas* foi deixado como um trabalho futuro.

O gráfico da Figura 29 mostra o ganho em condição vital obtido pelo *Jiva* para as possíveis variações de parâmetros *F* e *PS* assumidas, em um período de 10 min. Cada valor no gráfico é uma média aritmética de 3 execuções. Uma morte do *Jiva* em alguns desses testes representam uma ganho de -100 (perda), porque 100 é a máxima condição vital inicial do *Jiva*. *E.g.*, para *PS* = 96 e *F* = 6, o ganho de -100 no gráfico mostra que o *Jiva* morreu nos 3 testes executados.

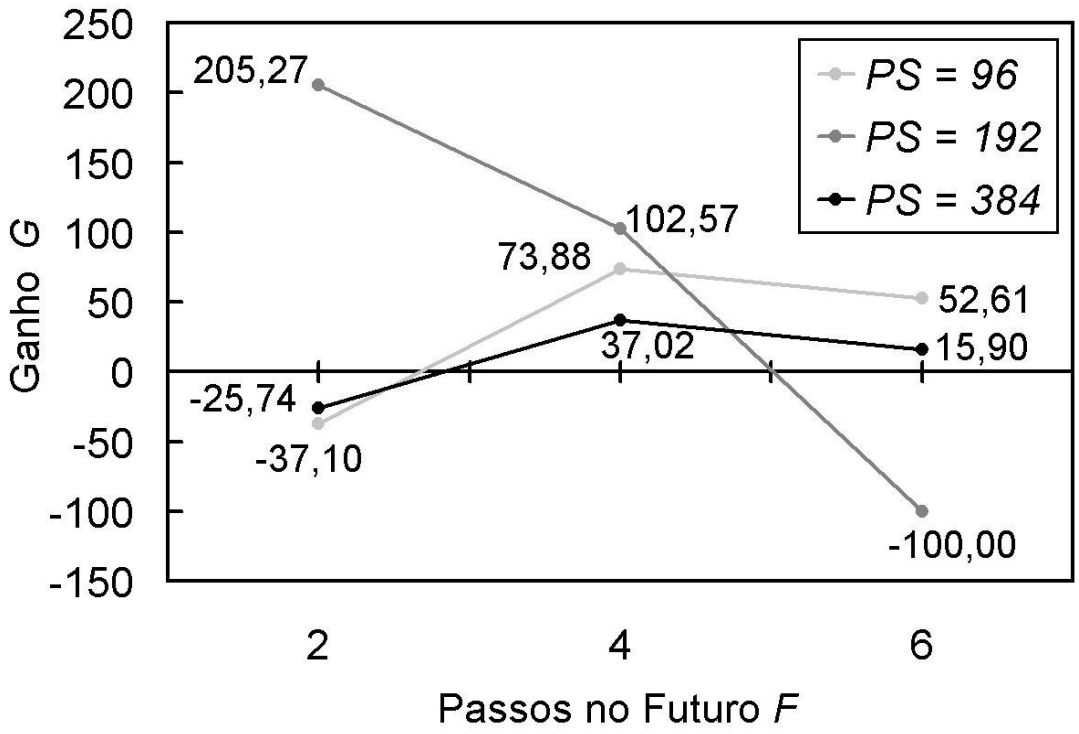


Figura 29. Ganhos na condição vital do *Jiva* para as possíveis variações de parâmetros.

Teoricamente, quanto maior é *PS* e *F*, maior deveria ser o ganho em condição vital do *Jiva*. Mas essa expectativa foi refutada pelos experimentos.

Ao analisarmos o ganho médio para variações em *PS,* observamos que: para *PS* = 96, houve um ganho médio de 29,8 pontos na condição vital. Para *PS* = 192, houve um maior desvio padrão, com o melhor e o pior resultado, resultando em um ganho médio de 69,28 pontos na condição vital. E finalmente para *PS* = 384, nós temos um ganho médio de 32,15 pontos na condição vital do *Jiva*. Concluímos então que, dadas as condições ambientais investigadas, o melhor valor médio para *PS* é 192.

Passamos então a analisar o ganho médio para variações em *F*. Para *F* = 2 observamos um ganho médio de 47,48 pontos; para *F* = 4 observamos um ganho médio de 71,16 pontos; e para *F* = 6 nós temos um ganho médio de -10,49. Baseados nestas informações, nós concluímos que o melhor valor médio para *F* é 4.

Como um todo, a melhor combinação de parâmetros, considerando a sinergia entre eles, é *PS* = 192 e *F* =2.

Esse resultado é interessante porque é contra-intuitivo. Quando *F* cresce, o *Jiva* pode ver mais passos no futuro, mas essa análise não leva em consideração o custo imediato para realizar uma ação específica. Hipotetizamos que essa ação, mesmo se ela produz o maior ganho a longo prazo, pode conduzir o *Jiva* para situações onde o custo a curto prazo é muito grande, causando sua morte (como foram os casos para $F = 6$). Isso pode explicar porque a melhor configuração de parâmetros dadas as condições ambientais investigadas tem um valor pequeno para *F* ($F = 2$), e o melhor valor médio para $F = 4$ não é o maior.

4.1.3 Compromisso entre Inteligência e Desempenho Computacional

O maior valor para *PS* que nós experimentos ($PS = 384$) também não ajuda a condição vital dos *Jivas*. Nosso entendimento foi de que nesse caso o *Jiva* "pensa" demais e conseqüentemente se torna mais lento na tomada de decisão. É claro que a agilidade na tomada de decisões inteligentes é um fator igualmente determinante para a sobrevivência de qualquer ser vivo (artificial ou biológico). Assim, o *Jiva* é penalizado no seu ganho vital quanto mais demora para tomar decisões, o que ocorre se colocamos uma população grande no AG.

A configuração de parâmetros $PS = 192$ e $F = 2$ foi a melhor dentre todas as investigadas para a adaptabilidade do *Jiva*. Além disso, estabelece-se um compromisso com o desempenho computacional, que para $PS = 192$ está na média (como mostra a Figura 26) e para $F = 2$ é o melhor caso.

4.2 Avaliação de Aceitação de Vidya

Em adição às necessárias avaliações de eficiência (*i.e.* desempenho no tempo) e também da avaliação de eficácia (*i.e.* capacidade de adaptação – inteligência) dos algoritmos produzidos neste trabalho – ambas avaliações descritas nas Subseções anteriores, tomamos o cuidado de elaborar questionário de usabilidade da aplicação; que foi respondido por usuários potenciais após jogarem Vidya. O propósito desse questionário foi de avaliar a aceitação do jogo por parte dos usuários, no que diz respeito a vários aspectos que serão explicados adiante.

É importante notar que essa avaliação não acrescenta na qualificação do algoritmo inteligente do *Jiva*, mas do jogo como um todo, considerando conjuntamente todos os seus elementos constituintes. Usamos como base um método de avaliação explanado em [59], com as devidas adaptações. O Questionário de Avaliação do Jogo Vidya está no Anexo D deste documento, e está organizado de forma que a primeira questão do questionário visou a separar os usuários em classes. Distinguimos três classes de usuários para esta avaliação:

- Casuais, os que jogam apenas como distração;
- Adeptos, os que jogam seriamente e sempre que podem;
- Não-jogadores.

O número de avaliações coletadas foi de 24, sendo que 12 delas foram de usuários casuais e os outros 12 de usuários diferenciados, a saber: 7 foram de adeptos e 5 de não-jogadores. Os usuários foram voluntários para jogar Vidya durante 15 min e tiveram 10 min para responder ao questionário. Os elementos do jogo que foram objeto do questionário foram:

- Qualidade da proposta do jogo;
- Diversão proporcionada pelo jogo;
- Qualidade da interação com o jogo;
- Qualidade dos gráficos e sons do jogo;
- Dinâmica do ecossistema do Mundo do jogo;
- Sensação de "ser deus", proporcionada pelo jogo;
- Capacidade de pôr estratégias em prática.

A versão atual de Vidya, desenvolvida em Java, não levou em consideração aspectos de desempenho. Consideramos que a jogabilidade foi bastante afetada pelo desempenho ainda não otimizado do programa. O desempenho pouco satisfatório é decorrência do ecossistema complexo, onde os seres vivos possuem ciclos completos de vida. Todas as tarefas tomadas em conjunto, computacionalmente custosas, não foram racionalizadas por não serem objeto deste trabalho. Mesmo assim, o jogo pôde ser apreciado com relativo sucesso pelos usuários-voluntários.

As perguntas realizadas no questionário possuem quatro respostas objetivas de múltipla escolha, ordenadas crescentemente por qualidade do objeto da questão. A quantificação das respostas foi a seguinte a primeira opção foi associada à nota 0, a segunda opção foi associada a 4, a terceira opção foi associada a 7 e a quarta opção foi associada a 10.

Os resultados que iremos relatar a seguir são médias aritméticas das notas atribuídas pelos usuários de uma mesma classe para cada elemento do jogo. No caso de elementos analisados (*e.g.* qualidade da interação com o jogo) possuírem mais de uma questão associada, a nota assumida é uma média aritmética das notas obtidas nas questões.

A seguir, exibiremos os resultados da avaliação dos diversos usuários para cada elemento do jogo.

4.2.1 Qualidade da Proposta do Jogo

A Tabela 1 mostra as notas médias atribuídas pelos usuários na avaliação da qualidade da proposta do jogo.

Tabela 1. Avaliação da qualidade da proposta do jogo.

Classe de jogador	Nota
Adeptos	9,1
Casuais	9
Não-jogadores	8,8
MÉDIA	**8,97**
DESVIO PADRÃO	**0.15**

A proposta do jogo foi muito bem aceita entre os diversos tipos de usuários. Podemos dizer, inclusive, que o desvio padrão para os três tipos de usuários foi nominalmente próximo a zero, o que indica uma opinião aproximadamente entre os tipos.

4.2.2 Diversão Proporcionada pelo Jogo

A Tabela 2 mostra as notas médias atribuídas pelos usuários na avaliação da diversão proporcionada pelo jogo.

Tabela 2. Avaliação da diversão proporcionada pelo jogo.

Classe de jogador	Nota
Adeptos	5,1
Casuais	6,2
Não-jogadores	5
MÉDIA	**5,44**
DESVIO PADRÃO	**0,67**

O jogo não se mostrou muito divertido para os três tipos de usuários. Entre os tipos, os casuais foram os que mais acharam divertido o jogo. Isso se deve essencialmente ao caráter mais de simulação e pesquisa do que necessariamente o de proporcionar divertimento, por parte de Vidya. Ainda assim, resolvemos avaliar este elemento.

4.2.3 Qualidade da Interação com o Jogo

A Tabela 3 mostra as notas médias atribuídas pelos usuários na avaliação da qualidade da interação com o jogo.

Tabela 3. Avaliação da qualidade da interação com o jogo.

Classe de jogador	Nota
Adeptos	6,5
Casuais	5,9
Não-jogadores	6,1
MÉDIA	**6,17**
DESVIO PADRÃO	**0,31**

O desempenho baixo do programa veio acarretar numa avaliação mediana da interação com o jogo e com o *Jiva*, acrescido pelo fato de o modelo de interação ser novo. Os Adeptos, que melhor avaliaram a qualidade da interação no jogo, demonstraram com isso que mais facilmente se adaptam a novos modelos de interação.

No presente trabalho, nenhum estudo de interação humano-computador foi realizada. A implementação de um modo de interação entre jogador e jogo inovador não foi adotado arbitrariamente, mas surgiu como uma conseqüência da necessidade de independência característica do *Jiva*. Portanto, os mecanismos de interação não foram o foco do trabalho.

4.2.4 Qualidade dos Gráficos e Sons do Jogo

A Tabela 4 mostra as notas médias atribuídas pelos usuários na avaliação da qualidade dos gráficos e dos sons do jogo.

Tabela 4. Avaliação da qualidade dos gráficos e sons do jogo.

Classe de jogador	Nota
Adeptos	6,8
Casuais	7,6
Não-jogadores	7,7
MÉDIA	**7,37**
DESVIO PADRÃO	**0,49**

Os componentes multimídia do jogo (imagem e som) foram bem avaliados. Os Adeptos se mostraram mais rigorosos quanto a esse aspecto, seguidos pelos Casuais e por fim os Não-jogadores, que deram a maior nota. Certamente, os elementos gráficos e de som também não foram o alvo de Vidya. Comparado com as mais recentes tecnologias de imagem e som para a criação de jogos para computador, Vidya deixa muito a desejar. Porém, obteve uma média de 7,4 entre os tipos de usuários, o que se encaixa na boa qualidade.

4.2.5 Dinâmica do Ecossistema do Mundo do Jogo

A Tabela 5 mostra as notas médias atribuídas pelos usuários na avaliação da dinâmica do ecossistema do Mundo do jogo.

Os jogadores Adeptos foram mais atentos aos detalhes dinâmicos do jogo, incluindo o andamento do Mundo. Como já foi demonstrado, Vidya é em muito um jogo de simulação de ecossistema, com ciclos de vidas bem definidos, porém complexos, para os seus diversos seres vivos. A avaliação da dinâmica do ecossistema do Mundo do jogo foi muito boa, e mais bem notada pelo jogador mais experiente, como é o caso dos Adeptos, que deram a maior nota.

Tabela 5. Avaliação da dinâmica do ecossistema do Mundo do jogo.

Classe de jogador	Nota
Adeptos	8,3
Casuais	8
Não-jogadores	7
MÉDIA	**7,77**
DESVIO PADRÃO	**0,68**

4.2.6 Sensação de "ser deus" Proporcionada pelo Jogo

A Tabela 6 mostra as notas médias atribuídas pelos usuários na avaliação da capacidade do jogo em dar a sensação ao usuário de ser o deus de um clã de *Jivas*.

Tabela 6. Avaliação da capacidade do jogo em dar a sensação ao usuário de ser deus.

Classe de jogador	Nota
Adeptos	3,4
Casuais	4,41
Não-jogadores	5,2
MÉDIA	**4,37**
DESVIO PADRÃO	**0,9**

Os usuários estão bem acostumados com a interação direta com o personagem, e portanto sem o hábito de encararem a possibilidade de serem deus de um mundo onde há livre-arbítrio. O desejo imediato dos jogadores é controlar o jogo, e se esse controle não é claro, ou é passível de "desrespeito" por parte do personagem, conjecturamos que isso pode acarretar numa má aceitação do usuário. Além disso, a idéia de ser deus de um clã de personagens implica numa sensação de tudo poder, como se o livre-arbítrio dos personagens fosse interrompido quando a sua vontade se impõe. Em Vidya ocorre o inverso. O jogador-deus pode intuir o personagem, mas sem retirar seu livre-arbítrio. Aliás, deve aprender a fazer isto de forma indireta, daí a inovação proposta aqui. Isto é facilmente visto pela pouca sensação dada ao usuário de ser um deus (que interfere fortemente nos personagens) de um clã. Como o teste de usabilidade durou poucos minutos, é possível que em mais longos períodos de interação essa habilidade não-trivial (de induzir e não conduzir) seja alcançada.

4.2.7 Capacidade de Pôr Estratégias em Prática

A Tabela 7 mostra as notas médias atribuídas pelos usuários na avaliação da capacidade do jogo em permitir que o usuário ponha estratégias em prática.

As notas prestadas pelos usuários são conclusivas quanto ao fato de que os jogadores não conseguiram pôr em prática estratégias para seu clã. Para isso, contribuíram o baixo desempenho do jogo, reiteramos que o inovador modelo de interação jogador-personagem e a autonomia do

Jiva, que muitas vezes "desobedece" a vontade de seu deus (o jogador). Mas isto ocorre no curto prazo, em mais longos períodos de jogo há uma melhor percepção deste fato.

Tabela 7. Avaliação da capacidade do jogo em permitir estratégias do usuário.

Classe de jogador	Nota
Adeptos	2,7
Casuais	3,6
Não-jogadores	4,4
MÉDIA	**3,57**
DESVIO PADRÃO	**0,85**

4.2.8 Comentários Gerais sobre a Avaliação

O propósito de Vidya é, antes de tudo, contribuir para a área de IA em jogos de computador na geração de um ser autônomo social inteligente, o *Jiva*, mais um novo modelo de interação jogador-jogo, através do qual o jogador não controla diretamente o *Jiva*. Além disso, ser uma plataforma para simulação de ecossistemas e observação de dinâmicas populacionais e comportamentos emergentes de sociedades. Mais ainda, dar a sensação ao jogador de ser o deus de um clã de seres que possuem "livre-arbítrio" nas suas ponderações. O elemento de estratégia dentro do jogo é permitir que o jogador elabore meios de sintonizar o comportamento de seu clã numa qualidade ótima, a fim de fazê-lo sobreviver.

O aprendizado do modelo de interação jogador-personagem demonstrou ser uma curva de crescimento lento, visto que a autonomia do *Jiva* constitui uma dificuldade significante na manipulação do jogador. Tal autonomia também impactou na elaboração de estratégias por parte do usuário.

Como um jogo simulador de dinâmica de ecossistemas, Vidya foi bem avaliado, considerando o questionário usado.

A aceitação da proposta do jogo foi muito boa, o que significa uma carência dos usuários por jogos do gênero. Na prática, contudo, surgiram dificuldades para a interação eficiente do usuário com o jogo. A avaliação contou com poucos participantes e, por este motivo, pode não ser uma representação fiel da opinião do público em geral quanto aos diversos aspectos do jogo. Apesar disso, julgamos relevante a sua contribuição para o nosso trabalho, no sentido de informar uma deficiência na interação entre jogador, jogo e personagens, em Vidya, a fim de melhorá-la no futuro.

O caráter de pesquisa em IA para jogos sobressaiu-se em relação aos aspectos lúdicos de Vidya e desempenho computacional do jogo. Por isso, a avaliação de diversão proporcionada teve um resultado sub-mediano. Mesmo assim, destacamos que a avaliação geral do jogo atingiu um patamar de 6,24 numa escala de 10, ao não se assumir prevalência de fatores avaliativos entre si.

Capítulo 5

Conclusões e Trabalhos Futuros

5.1 Contribuições

Este trabalho foi fortemente motivado pela possibilidade de criação de um jogo de simulação com a finalidade de servir como plataforma para observação de equilíbrios em ecossistemas, dinâmicas populacionais e comportamentos sociais emergentes. Vidya é um passo inicial neste sentido, e sua principal contribuição nesse contexto de pesquisa foi a construção de um ecossistema com leis econômicas (fluxo de recursos naturais) claras e a elaboração de um ser social inteligente autônomo, quando o *Jiva* coage juntamente com seus companheiros de clã.

A outra importante motivação para o trabalho veio na demanda contínua e crescente de aplicação de IA em jogos, na produção de agentes inteligentes cada vez mais realísticos do ponto de vista humano. Ou seja, agentes capazes de se adaptarem a diversos ambientes e jogadores. Com o advento e popularização das placas aceleradoras gráficas isto passou de uma simples necessidade para um convite, visto que muito do processamento central que antes era dedicado à instruções gráficas está agora disponível para uso de IA nos jogos para computador.

O *Jiva* é um agente inteligente autônomo completamente modelado com Computação Evolutiva. O uso do algoritmo proposto, modelado com AG, provou ser uma boa escolha porque não somente produz bons resultados em prover inteligência para o *Jiva*, mas também possui um bom desempenho computacional que o torna hábil em aplicações que, inclusive, possuem requerimentos de tempo de resposta. É importante notar que esta última observação, também vai de encontro às espectativas usuais de que AG são lentos. Nessa seara, capitalizamos no fato de que mesmo soluções (gerações) iniciais de AG já são plausíveis. No contexto do nosso jogo isto já reduz muito a incerteza das próximas ações a tomar pelo Jiva.

Com respeito ao desempenho computacional, percebemos que o tempo de processamento aumenta linearmente com o tamanho do problema, para as variações dos parâmetros *PS* e *F* feitas nos experimentos. Mesmo em valores absolutos, o algoritmo não excedeu 600 ms de tempo de

processamento, no pior caso, para um passo de tempo – o que é aceitável, dadas a complexidade relativa do ambiente.

Com respeito ao desempenho inteligente, os resultados também foram muito promissores. Verificamos *tradeoffs* nas variações dos parâmetros do algoritmo, inclusive resultados contra-intuitivos, porém explicáveis. Se *F* aumenta, apesar do algoritmo "ver mais" no futuro, mais lento ele se torna, o que impacta diretamente e de forma decisiva na adaptabilidade do *Jiva*. Os resultados de adaptabilidade do *Jiva* através da variação de parâmetros do algoritmo também foram úteis para determinar a melhor configuração de parâmetros que corresponde a um compromisso entre inteligência e desempenho computacional.

Postulamos que esse mesmo módulo de decisão inteligente do *Jiva* poderá ser reusado em outros cenários de jogos para computador, em agentes que tenham os mesmos requerimentos de autonomia, inteligência e comportamento social.

Uma terceira contribuição de Vidya, ainda em jogos eletrônicos, foi a criação de um modelo de interação jogador-personagem inovador. Em jogos tradicionais, os jogadores comumente interagem com o personagem de uma forma muito direta, controlando completamente os seus movimentos. Em Vidya, o jogador não atua diretamente nos *Jivas*, mas fornece instruções não detalhadas, as *vidyas*, que surgem na mente artificial deles como uma "intuição". Contudo, eles têm autonomia para seguir ou não as intuições fornecidas pelo jogador.

5.2 Discussões

Obviamente, Vidya ainda não está pronto para ser uma plataforma para uso efetivo em aplicações reais das ciências sociais e cognitivas (individualmente ou de forma integrada) no estudo de comportamentos sociais emergentes e dinâmica de populações. Antes disso, o jogo Vidya deve ser visto como um protótipo que ainda precisa de muitos ajustes e implementações para que esse objetivo maior seja alcançado; e as expectativas para que isso aconteça são promissoras. Com este trabalho demos o passo inicial na elaboração do *Jiva* como indivíduo inteligente de uma sociedade e na criação do ambiente de simulação de ecossistema.

Em Sociologia, está claro que o movimento das massas sociais e suas tendências são uma emergência dos comportamentos de seus indivíduos. Um fenômeno dos mais interessantes é que os comportamentos inteligentes emergentes das sociedades, geralmente, podem ser qualitativamente superiores aos dos seus indivíduos: "O conjunto é mais que a soma das partes" – esse é o mesmo tipo de fenômeno que ocorre, quando a mente humana emerge das redes neurais naturais que constituem o sistema nervoso do homem.

É certo que as simulações (e também modelos analíticos) estarão cada vez mais presentes para não só explicar fenômenos como os descritos acima, mas também realizar previsões de comportamentos normais e não-normais. No contexto do sistema nervoso, por exemplo, imaginemos a utilidade de uma previsão de evolução de doenças degenerativas nervosas. Seguindo o mesmo raciocínio, imaginemos a utilidade de realizar previsões, através de

simulações, de estados futuros de ecossistemas e sociedades reais (humanas ou não). Para isso é preciso fazer a ponte entre a célula do sistema e o sistema como um todo. Nesta proposição, o *Jiva* seria a célula que podermos tentar utilizar para fazermos previsões, e quiçá, adaptá-lo para previsões em sociedades humanas.

No que diz respeito a jogos para computador, não há dúvidas que o módulo de decisão do *Jiva* foi uma contribuição importante, dados os resultados obtidos. A demanda por agentes humanamente inteligentes em jogos eletrônicos é grande e para que isso ocorra é necessário inserir elementos essencialmente humanos como reatividade inteligente (não-monotónica) em situações de novidade, criatividade, comportamento social e adaptação contínua. Argüimos que o *Jiva* e seu módulo de IA são uma boa contribuição nessa direção.

5.3 Trabalhos Futuros

A construção de Vidya trouxe muitas idéias na sua esteira, das quais as mais importantes estão listadas abaixo:

- Reusar a inteligência do *Jiva* em outros jogos de computador e cenários de jogos;
- Analisar comportamentos emergentes das sociedades dos *Jivas*;
- Estudar ecossistemas e sociedades mais complexas usando o ambiente proposto;
- Estudar dinâmica e pontos de equilíbrio (equilíbrios de Nash) de populações e de ecossistemas formados por indivíduos não-cooperativos;
- Implementar relações sociais de cooperação entre os *Jivas*, incluindo sua reprodução com transmissão de traços de personalidades dos pais para sua prole;
- Lidar com incertezas devido a percepção e tratamento de oclusões de elementos do Mundo;
- Aumentar a parametrização do jogo, por exemplo, (i) permitindo características distintas em clãs heterogêneos, (ii) permitir alterações em pesos de vitalidade, hidratação e energia para comporem a condição vital do Jiva e (iii) jogadores escolhendo número de Jivas sob sua “divindade”;
- Aumentar a complexidade dos fatores que definem capacidades perceptivas e de longevidade dos seres de Vidya;
- Programar Vidya numa liguagem de programação mais eficiente, por exemplo C++, já que essa linguagem é a mais usada atualmente para desenvolvimento de jogos para computador;
- Melhorar os mecanismos de interação jogador-jogo de Vidya, tornando-o mais simples e intuitivo, de forma a tornar o crescimento da curva de aprendizado do jogo mais rápido.

Bibliografia

[1] Sun Microsystems. *Java 2 Platform, Micro Edition*. Disponível em: http://java.sun.com/javame/index.jsp (Acessado em 19/11/2006).

[2] PITA, M.R.S., SILVA, C.E., ARRAES, D., PEDROSA, D., TRINTA, F., WANDERLEY, I., PORTELA, M., MACHADO, M., BORGES, R., RAMALHO, G. *Uma Plataforma para Jogos Móveis Massivamente Multiusuário*. SBGAMES, 2005.

[3] CESAR. http://www.cesar.org.br/ (Acessado em: 19/11/2006).

[4] MEANTIME. http://www.meantime.com.br/ (Acessado em: 19/11/2006).

[5] BATTAIOLA, A.L. *Jogos por Computador – Histórico, Relevância Tecnológica e Mercadológica, Tendências e Técnicas de Implementação*. Anais da XIX Jornada de Atualização em Informática, XIX Congresso Nacional da Sociedade Brasileira de Computação, 2000.

[6] NAREYEK, A. *AI in computer games*. Queue, Game Development: Serious Business, Serious Coding, volume 1, issue 10, ACM Press, 2004, New York.

[7] LENT, M. e LAIRD, J. *Developing an Artificial Intelligence Engine*. Proceedings of the Game Developers' Conference, 1999, San Jose, CA.

[8] BLACKMAN, S. *Serious Games...and Less!*. ACM SIGGRAPH Computer Graphics, volume 39, issue 1, pp. 12-16, 2005.

[9] LAIRD, J. e LENT, M. *Human-level AI's Killer Application: Interactive Computer Games*. Proceeding of National Conference on Artificial Intelligence, AAAI Press, 2000, Menlo Park, CA.

[10] BRIDGER, B.C. e GROSKOPF, C.S. *Fundamentals of Artificial Intelligence in Computer Games*. Proceedings of 38th Annual on Southeast Regional Conference, ACM Press, 2000, Clenson, South Carolina.

[11] LAIRD, J. *Using Computer Games to Develop Advanced AI*. Computer, volume 34, issue 7, 2001, pp. 70-75.

[12] KRAMMER, W. *What is a Game?*. The Games Jornal, 2000, Disponível em: http://www.thegamesjournal.com/articles/WhatIsaGame.shtml (Acessado em 07/11/2006).

[13] NEUMANN, J. e MORGENSTERN, O. *Theory of Games and Economic Behavior*. Princeton: Princeton University Press, 1944.

[14] NASH, J.F. *Non-Cooperative Games*. Annals of Mathematics 54, pp. 286-295, 1951.

[15] PLAYSTATION. http://www.us.playstation.com/ (Acessado em: 19/11/2006).

[16] XBOX. http://www.xbox.com/ (Acessado em: 19/11/2006).

[17] GAMEBOY ADVANCE. http://www.gameboyadvance.com/ (Acessado em: 19/11/2006).

[18] HALF LIFE 2. http://half-life2.com/ (Acessado em: 19/11/2006).

[19] FALCON 4.0 HEADQUARTERS. http://www.f4hq.com/default.php?page=default (Acessado em: 19/11/2006).

[20] SILENT HILL. http://www.sonypictures.com/movies/silenthill/ (Acessado em: 19/11/2006).
[21] GRAN TURISMO 2. http://www.us.playstation.com/Content/OGS/SCUS-94455/Site/main.html (Acessado em: 19/11/2006).
[22] CRAWFORD, C. *The Art of Computer Game Design* (1982). Osborne: Mcgraw-Hill Osborne Media, 1984, pp. 25-35.
[23] ATARI. *Atari Flashback 2 – Owner's Manual, Model CX-2600*. Disponível em: http://www.atari.com/us/images/games/FBK2/manual/main.htm (Acessado em: 19/11/2006).
[24] PRO EVOLUTION SOCCER 4. http://www.pes4.net/ (Acessado em: 19/11/2006).
[25] NBA JAM. http://www.us.playstation.com/PS2/Games/SLUS-20648 (Acessado em: 19/11/2006).
[26] ULTIMA ONLINE. http://www.uo.com/ (Acessado em: 19/11/2006).
[27] AGE OF EMPIRES. http://www.microsoft.com/games/empires/ (Acessado em: 19/11/2006).
[28] ROCKY'S BOOTS. http://www.warrenrobinett.com/rockysboots/ (Acessado em: 19/11/2006).
[29] SILENT HUNTER III. http://www.silenthunteriii.com/ (Acessado em: 19/11/2006).
[30] ORBITER. http://orbit.medphys.ucl.ac.uk/ (Acessado em: 19/11/2006).
[31] SIMCITY. http://simcity.ea.com/ (Acessado em: 19/11/2006).
[32] THE SIMS. http://thesims.ea.com/ (Acessado em: 19/11/2006).
[33] LAIRD, J. *Research in Human-level AI Using Computer Games*. Communications of ACM, volume 45, issue 1, 2002.
[34] HICKS, J.R. e DRISCOLL, J.A. *Evolution of Artificial Agents in a Realistic Virtual Environment*. Proceedings of the 43rd Annual on Southeast Regional Conference, volume 2, ACM Press, 2005, Kennesaw, Georgia, pp. 365-369.
[35] BUCKLAN, M. *AI Techniques for Game Programming*. Premier Press, 2002.
[36] SCHWAB, B. *AI Game Engine Programming*. Massachusetts: Charles River Media, 2004.
[37] YANNAKAKIS, M. e LEE, D. *Testing Finite State Machines*. Proceedings of the twenty-third annual ACM symposium on Theory of computing, ACM Press, 1991.
[38] YANNAKAKIS, G.N. *AI Game Development*. Computers in Entertainment (CIE), ACM Press, 2005.
[39] GOLDBERG, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison-Wesley, 1989.
[40] PRINCIPE, J.C., EULIANO, N.R. e LEFEBVRE, W.C. *Neural and Adaptive Systems: Fundamentals Through Simulations*. New York: Willey & Sons, 2000.
[41] MAES. P. *Artificial Life Meets Entertainment: Lifelike Autonomous Agents*. Communications if the ACM, volume 38, issue 11, ACM Press, 1995, pp. 108-114.
[42] MORET, B. *Decision Trees and Diagrams*. ACM Computing Surveys (CSUR), volume 14, issue 4, ACM Press, 1982, pp. 593-623.
[43] KOENIG, S. e LIKHACHEV, M. *Real-time Adaptive A**. Proceedings of the fifth international joint conference on Autonomous agents and multiagents systems (AAMAS), ACM Press, 2006.
[44] HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.
[45] NANG, L. e MATSUO, K. *A Survey on Parallel Genetic Algorithms*. Journal of the Society of Instrument and Control Engineering, volume 3, issue 33, 1994, pp. 500-509.
[46] JINGHUI, Z. *Comparision of Performance Between Different Selection Strategies on Simple Genetic Algorithms*. CIMCA, volume 2, 2005, pp. 1115-1121.

[47] MIRANDA, M.N. *Algoritmos Genéticos: Fundamentos e Aplicações*. Disponível em: http://www.gta.ufrj.br/~marcio/genetic.html (Acessado em: 26/12/2006).

[48] Sun Microsystems. *Java 2 Platform, Standard Edition*. Disponível em: http://java.sun.com/javase/index.jsp (Acessado em 09/11/2006).

[49] HOLDER, W. *Java Game Programming for Dummies*. Foster City: IDG Books Worldwide, 1998.

[50] CLINGMAN, D., KENDALL, S. e MESDAGHI, S. *Pratical Java Game Programming (Game Development Series)*. Charles River Media, 2004.

[51] FAN, J. *Black Art of Java Game Programming*. Sams, Macmillan Computer Publishing, 1996.

[52] DAVISON, A. *Killer Game Programming in Java*. O'Reilly Media, 2005.

[53] DAVISON. A. *Games Programming with Java and Java3D*. Disponível em: http://fivedots.coe.psu.ac.th/~ad/jg/intro (Acessado em 09/11/2006).

[54] BETHKE, E. *Game Development and Production*. Wordware Publishing, 2003, pp. 129-130.

[55] LAMOTHE, A. *Tricks of 3D Game Programming Gurus*. Indiana: Sams Publishing, 2003, pp. 495-496.

[56] FUKUSHIMA, K. *Recognition of Occluded Patterns: a Neural Network Model*. IJCNN, 2000.

[57] FREITAS, L.B.L. *Piaget and the Moral Conscience: an Evolving Kantism?*. Psicologia: Reflexão e Crítica, volume 15, no. 2, Porto Alegre, 2002, pp. 303-308, Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0102-79722002000200008&lng=en&nrm=isso (Acessado em 10/11/2006).

[58] PITA, M.R.S., MADEIRO, S.S. e NETO, F.B.L. *Vidya: A God Game Based on Intelligent Agents Whose Actions are Devised Through Evolutionary Computation*. IEEE Symposium on Computational Intelligence and Games 2007 (submitted), 2006.

[59] BUEDE, O.V., LUZARDO, R. e SILVEIRA, M.S. *Rumo a uma Avaliação Específica de Qualidade de Interação para Jogos Eletrônicos*. SBGAMES, 2005..

Anexo A

Documento de Game Design

Vidya

Versão 01.00 - 11/09/2005

Marcelo Pita

1. Introdução

Este documento tem como objetivo descrever o jogo que será artefato produzido no meu Trabalho de Graduação.

Servirá como guia para o desenvolvimento do jogo proposto, e é a documentação mínima do produto gerado.

1.1. Convenções, termos e abreviações

Termo	Descrição
God Games	Jogos que dão ao jogador um papel de entidade divina.
Tamagotchi	Jogo (classe de jogo) onde jogador cuida de seu personagem.
MOG	Multiplayer Online Games.
RPG	Role-Playing Games.
Ecossistema	Ambiente em equilíbrio habitado por uma ou mais formas de vida.
Kosmo	Universo, considerando o nível de organização deste.
Intuição	Conhecimento interno, esotérico.
Comportamentos sociais emergentes	Comportamentos não vistos na individualidade de cada participante de uma sociedade, mas somente vistos em conjunto.
The Sims	God Game desenvolvido para PC
Pokemon / Digimon	Jogo baseado nos seriados de televisão Pokemon e Digimon, seguindo o estilo Tamagotchi.
Multiplayer	Multiusuário
Windows	Classe de sistemas operacionais da Microsoft

2. Descrição do Jogo

2.1. Nome

Vidya

2.2. Gênero

Em Vidya são identificadas características dos seguintes gêneros:

- God Games
- Tamagotchi
- Multi-usuário (MOG)
- RPGs

2.3. Descrição

Vidya foi feito para exploração e observação dos mecanismos de aprendizagem e sobrevivência numa sociedade heterogênea, composta por diferentes clãs de criaturas humanóides, chamadas Jivas.

O jogador de Vidya está convidado a ser o Deva (deus) de um clã de Jivas, e terá como tarefa intuir cada integrante deste clã. Apesar disto, o Jiva tem livre-arbítrio para seguir ou não a Intuição Divina fornecida, chamada de vidya.

A vidya tem como objetivo direcionar as ações de cada integrante do clã, de forma a garantir um melhor e mais rápido aprendizado, para que o grupo sobreviva por mais tempo.

Numa sessão de jogo, 2 clãs lutam pela sobrevivência e supremacia, num mundo chamado Suryaloka. O clã sobrevivente é considerado o vencedor.

Vários conflitos são encontrados neste cenário:

- Conflitos entre clãs;
- Conflitos entre indivíduos de um mesmo clã;
- Decisão do indivíduo entre agir para si e agir para o grupo (egoísmo e altruísmo) – questões éticas;
- Decisão dos Jivas em seguir a vidya ou não.

Suryaloka é um mundo e ao mesmo tempo um ecossistema. Como em todo ecossistema, deve haver um equilíbrio econômico e conflitos. A raiz de todos os conflitos está na disputa pelos recursos naturais de Suryaloka.

2.4. História

Suryaloka é um mundo em desenvolvimento, recém-nascido. Chegou o momento do aparecimento de vida inteligente no mundo. Vivasvan, o Criador de Suryaloka, resolveu então criar o Jiva, um ser vivo humanóide com poder de decidir sobre seu próprio destino.

Porém, Vivasvan está com uma grande dúvida: qual grupo de Jivas deve dar início à população de seres inteligentes no mundo?

Vivasvan decidiu que um clã dar início à nova raça, mas que para isso o mesmo deveria passar por uma seleção natural juntamente com outro clã e outros seres vivos. Para que eles não estejam completamente desamparados nesta tarefa, Vivasvan dará a eles a oportunidade de serem intuídos por Devas (Devas) do Kosmo. Dois Devas intuirão dois clãs de Jivas. O clã sobrevivente dará início à raça de seres inteligentes no mundo.

Então, Vivasvan convocou dois grandes hierarcas, Devas do Kosmo, para conduzir dois clãs de Jivas. Dentre os Devas, você foi um dos escolhidos. Agora, é seu papel fornecer vidya para os integrantes de seu clã, fazendo-os sobreviver, e mais que isso, torná-los famoso e soberano no mundo de Suryaloka.

2.5. Personagens

Suryaloka é habitado principalmente por Jivas, seus personagens principais. Os Jivas são criaturas humanóides (com forma humana) com tendências inatas a formarem clãs. Nas atuais condições de Suryaloka, a vida é difícil pela escassez de recursos.

Suryaloka possui outras formas de vida, além de Jivas, algumas das quais servem de alimento para ele. Há plantas, árvores que dão frutos, ovelhas, vacas e lobos.

Os Jivas mais bem-sucedidos são os que usam sua inteligência mais eficazmente para adaptação e sobrevivência em Suryaloka. Uma das armas dos Jivas é a ação grupal, através de clãs.

Mas os Jivas não estão sós. Eles contam com a vidya. Cabe a eles seguí-las ou não, de acordo com as diversas situações que se desenvolverão e com o aprendizado adquirido no decorrer de suas vidas.

Os Jivas possuem três atributos:

- Vitalidade: Energia vital dos Jivas. À medida que crescem, os Jivas vão perdendo a vitalidade naturalmente e por ocasião de danos ao corpo, até que morram. A função de sobrevivência dos Jivas tenta maximizar a vitalidade. Os Jivas ganham mais vitalidade quando comem frutas.
- Hidratação: Nível de água corporal do Jiva. O Jiva precisa de água para sobreviver e realizar as atividades. Ele recupera sua água corporal bebendo diretamente das fontes de água doce.
- Energia: Nível de energia do Jiva. O Jiva precisa de energia para realizar ações, e pode recuperar energia se alimentando.

2.6. Objetivos do jogo

Vidya tem como objetivo principal ser uma plataforma para observação de comportamentos sociais emergentes das sociedades que os Jivas formam, observação da interatividade e dinâmica entre os indivíduos dessa sociedade e verificação de equilíbrio em ecossistemas. Além disso, ele objetiva dar a sensação ao jogador de ser Deva de seu clã de Jivas e contribuir com o agente do Jiva para a comunidade de IA para jogos de computador.

Obviamente, tudo isto estará sendo vivido num ambiente fantasioso, com seres fantasiosos, com elementos fantasiosos, num mundo chamado Suryaloka.

Além disso, Vidya tem como objetivo despertar habilidades de estratégia no jogador.

2.7. Objetivo do jogador

O jogador é o Deva de um clã de Jivas. Terá como objetivo principal fornecer aos integrantes de seu clã.

O objetivo do clã, e de cada Jiva em específico é a sobrevivência. Para isso, os Jivas lutarão entre si para obter os recursos naturais escassos de Suryaloka.

O jogador deve fornecer intuições corretas, para que os Jivas possam cada vez tomar boas decisões e aprender de forma mais rápida.

Quanto mais intuições corretas forem fornecidas, mais chances têm o grupo de sobreviver. Por ação correta entende-se aquela aumenta a adaptabilidade dos Jivas, que é uma média ponderada de seus atributos.
O jogador tem como objetivo final fazer seu clã ser o clã sobrevivente, ou seja, eliminar o outro clã de Jivas. Numa mesma sessão de jogo, 2 clãs de, no máximo, 5 Jivas estão presentes.

2.8. Jogos similares

Não há ainda jogo similar a Vidya. Mas Vidya possui características existentes nos seguintes jogos:

- Black & White
- The Sims
- Tamagotchi
- Pokemon / Digimon

2.9. Número de jogadores

Monousuário.

2.10. Plataformas alvo

- Dispositivos: computadores pessoais (PC's)
- Sistema operacional: Windows

2.11. Características essenciais

- Personagens humanóides chamados Jivas
- Clãs de Jivas
- Jivas são agentes inteligentes autônomos
- Comportamentos sociais dos Jivas
- Ecossistema formado por vários seres artificiais diferentes
- Jogador (Deva) fornece intuição para o líder de seu clã
- Novo modelo de interação jogador-personagem

Anexo B

Documento de Gameplay

Vidya

Versão 01.00 - 14/11/2006

Marcelo Pita

1. Objetivo do Documento

Este documento tem como objetivo descrever o funcionamento do jogo Vidya e as forma que o jogador tem de interagir com ele.

2. Descrição do Jogo

Vidya foi feito para exploração e observação dos mecanismos de aprendizagem e sobrevivência numa sociedade heterogênea, composta por diferentes clãs de criaturas humanóides, chamadas Jivas.

O jogador de Vidya está convidado a ser o Deva (deus) de um clã de Jivas, e terá como tarefa intuir cada integrante deste clã. Apesar disto, o Jiva tem livre-arbítrio para seguir ou não a Intuição Divina fornecida, chamada de vidya.

A vidya tem como objetivo direcionar as ações de cada integrante do clã, de forma a garantir um melhor e mais rápido aprendizado, para que o grupo sobreviva por mais tempo.

Numa sessão de jogo, 2 clãs lutam pela sobrevivência e supremacia, num mundo chamado Suryaloka. O clã sobrevivente é considerado o vencedor.

Vários conflitos são encontrados neste cenário:

- Conflitos entre clãs;
- Conflitos entre indivíduos de um mesmo clã;
- Decisão do indivíduo entre agir para si e agir para o grupo (egoísmo e altruísmo) – questões éticas;
- Decisão dos Jivas em seguir a vidya ou não.

Suryaloka é um mundo e ao mesmo tempo um ecossistema. Como em todo ecossistema, deve haver um equilíbrio econômico e conflitos. A raiz de todos os conflitos está na disputa pelos recursos naturais de Suryaloka.

3. Fluxo Navegacional

Nesta seção, apresentaremos o fluxo navegacional de Vidya, ou seja, o fluxo de telas apresentadas ao jogador no decorrer de uma seção de jogo.

3.1. Tela de Apresentação

Esta é a primeira tela do jogo Vidya. Trata-se de uma tela em preto com a frase "*Press Any Key*" em vermelho no centro (Figura 1), esperando que o usuário pressione algum tecla. Feito isso, o fluxo segue para a tela de opções.

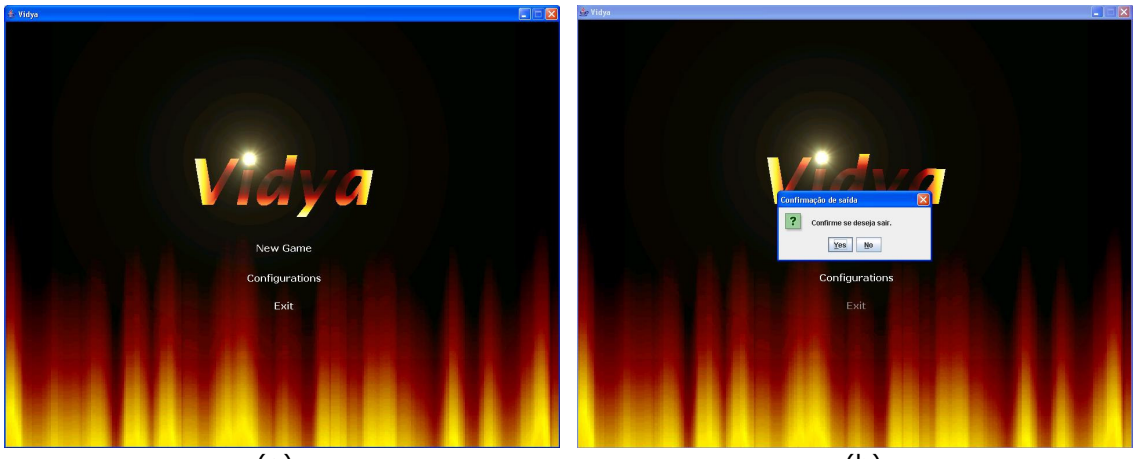
Figura 1. Tela de apresentação.

3.2. Tela de Opções

A tela de opções (Figura 2.a) lista as seguintes opções para o usuário:

- *New Game* (iniciar um jogo novo);
- *Configurations* (configurar parâmetros para o jogo);
- *Exit* (sair do jogo).

Clicando em *New Game* o fluxo é direcionado para a tela para novo jogo. Clicando em *Configurations* a tela de configurações é exibida. Clicando em *Exit*, o programa exibe uma tela de confirmação de saída (Figura 2.b), que se respondida afirmativamente faz abortar o jogo.



(a) (b)

Figura 2. (a) Tela de opções; (b) Tela de opções pedindo confirmação de saída.

3.3. Tela de Configurações

Não implementada na versão atual de Vidya.

3.4. Tela para Novo Jogo

Esta tela é exibida na Figura 3. Nesta tela o clã que será de responsabilidade do jogador dentro do Mundo é apresentado. São cinco *Jivas*, cujos nomes podem ser editados. O nome do clã deve ser fornecido também. O botão "*Confirm*" conduz ao início do jogo propriamente dito, caso o usuário não tenha editado nada errado (não fornecer algum nome para *Jiva* ou para o clã). A seta amarela, no canto esquerdo inferior da tela, conduz à tela de opções.

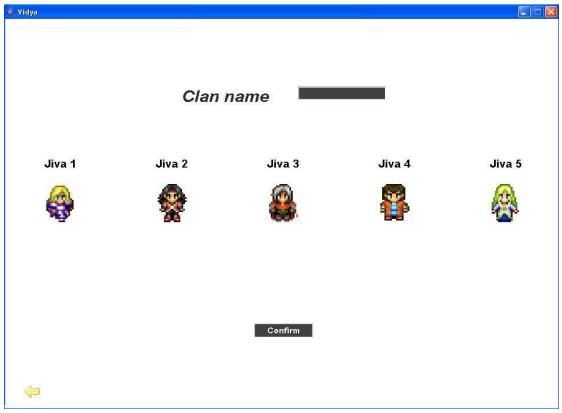


Figura 3. Tela para novo jogo.

3.5. Tela de Jogo

A tela de jogo aparece quando o usuário fornece o nome do clã e dos personagens e aperta no botão "*Confirm*" na tela para novo jogo. A tela de jogo é composta por dois ambientes, como mostra a Figura 4.

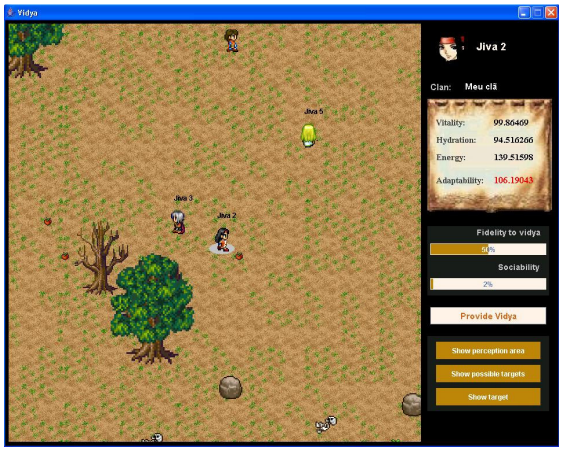


Figura 4. Tela de jogo.

O ambiente maior, à esquerda, é onde aparece o desenvolvimento do Mundo do jogo. O ambiente menor, à direita, é o painel de informações de personagens. Quando um personagem é selecionado, seus atributos e opções operacionais são mostrados neste ambiente.

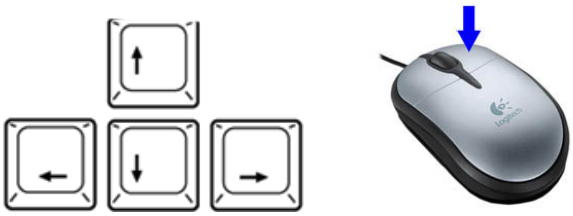
Logo quando inicia o jogo, o foco é colocado sobre o clã de *Jivas* do jogador, que fica no lado inferior direito do Mundo. O clã oponente está localizado no canto superior esquerdo do Mundo do jogo.

O Mundo é ocupado por muitos tipos de objetos, dentre os quais alguns são vidas artificiais.

4. Movendo-se dentro do Mundo do Jogo

O jogador pode mover o foco de exibição para outras partes do Mundo. Há duas maneiras do usuário mover-se dentro do Mundo do jogo: (i) através das setas do teclado e (ii) através do botão direito do *mouse*.

Usando as setas do teclado (Figura 5.a), o usuário move-se passo a passo na direção requerida dentro do Mundo do jogo. Usando o botão direito do mouse (Figura 5.b), o usuário clica no ponto para o qual quer migrar, quando então o foco é direcionado para a posição desejada.



(a) (b)

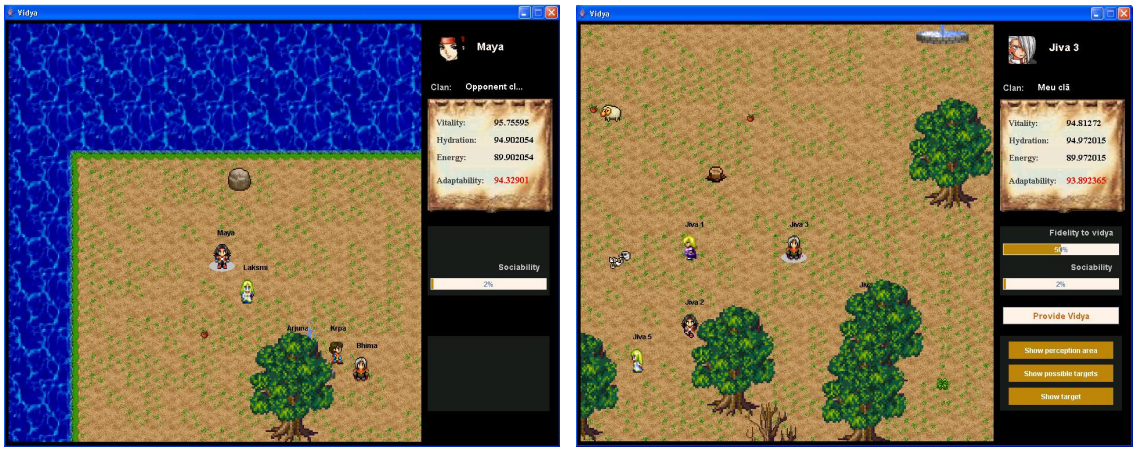
Figura 5. (a) Setas do teclado; (b) Botão direito do *mouse*.

Quando um objeto no mundo é selecionado, o foco de exibição é automaticamente direcionado para ele.

5. Selecionando Objetos do Mundo

Objetos no mundo podem ser selecionados. Para selecionar um objeto, o jogador deve clicar nele com o botão esquerdo do *mouse* (o outro botão não indicado do *mouse* exibido na Figura 5.b). Todos os objetos são selecionáveis, mas na versão atual de Vidya somente os *Jivas* quando selecionados exibem seus atributos no painel de informações de personagens.

Um *Jiva* do clã oponente, quando selecionado, mostra somente algumas informações no painel de informação de personagens (Figura 6.a). Já quando o *Jiva* é do clã do jogador é selecionado, além dos atributos comuns, outros atributos e opções operacionais são exibidos (Figura 6.b).



(a) (b)

Figura 6. (a) *Jiva* do clã oponente selecionado; (b) *Jiva* do clã do jogador selecionado.

6. Atributos do Jiva

Quando um *Jiva* do clã do jogador é selecionado, todos os seus atributos mais algumas opções operacionais são exibidos no painel de informações de personagem (Figura 7).

Figura 7. Informações sobre um *Jiva*.

Além de uma imagem da face, o nome e o nome do clã do *Jiva*, os atributos "vitalidade", "hidratação", "energia", "adaptabilidade", "fidelidade a *vidya*" e "sociabilidade" são exibidos. Mais ainda, alguns botões estão disponíveis para prover *vidya* e exibir ou esconder área de percepção, possíveis posições alvo e posição alvo escolhida.

6.1. Vitalidade, Hidratação e Energia

Estes três atributos são comuns a alguns seres vivos dentro do Mundo. A vitalidade é encarada como o tempo de vida restante do *Jiva*, a hidratação seu nível de água e a energia o nível de nutrição.

6.2. Adaptabilidade

A adaptabilidade é uma medida indireta de inteligência do *Jiva*. É função da vitalidade, hidratação e energia (uma média ponderada).

6.3. Fidelidade a Vidya

Indica o quanto que o *Jiva* leva em consideração a *vidya* fornecida pelo jogador.

6.4. Sociabilidade

Indica o quanto que o *Jiva* leva em consideração as ações dos *Jivas* mais fortes mais próximos. A sociabilidade, dentro do contexto da versão atual de Vidya, estabelece relações (hierárquicas) de coação entre os *Jivas* de um mesmo clã.

6.5. Provendo de *vidya*

Selecionado o *Jiva* do seu clã, o jogador poderá "intuí-lo" a mover-se para uma determinada posição no Mundo, a *vidya*. O *Jiva*, contudo, tem poder para seguir ou não a intuição. Para fornecer *vidya* a um *Jiva*, basta que, selecionando-o, o jogador clique no botão "*Provide Vidya*". Neste momento, o *Jiva* fica em posição de espera, como se estivesse "tendo uma intuição", quando então o jogador deverá clicar numa posição para a qual ele ache que o *Jiva* deverá migrar. Isso é mostrado na Figura 8.

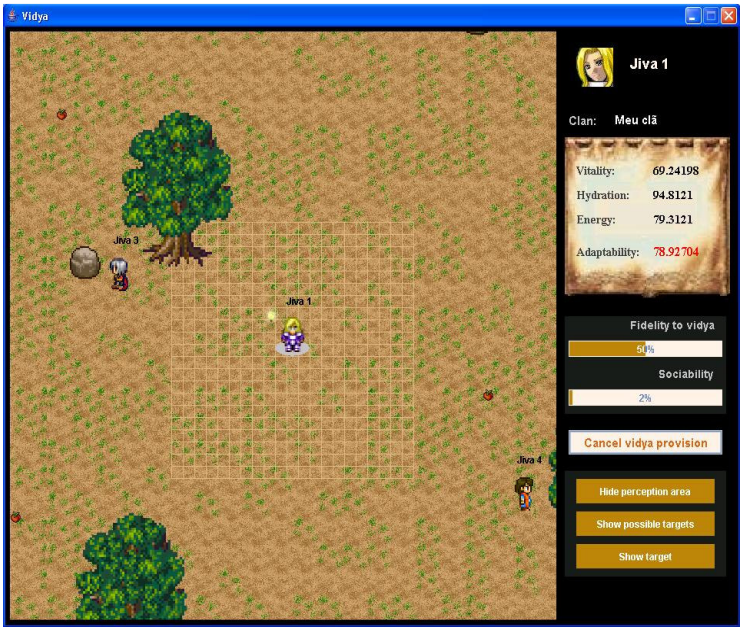


Figura 8. Provendo *vidya* para um Jiva.

O jogador deverá clicar numa das células exibidas da área exibida. Se ele tentar clicar numa célula de fora da área, um aviso sonoro é acionado, indicando operação proibida. A operação de prover *vidya* pode ser cancelada através do mesmo botão, agora nomeado "*Cancel vidya provision*".

6.6. Exibindo / Escondendo Alcance de Percepção

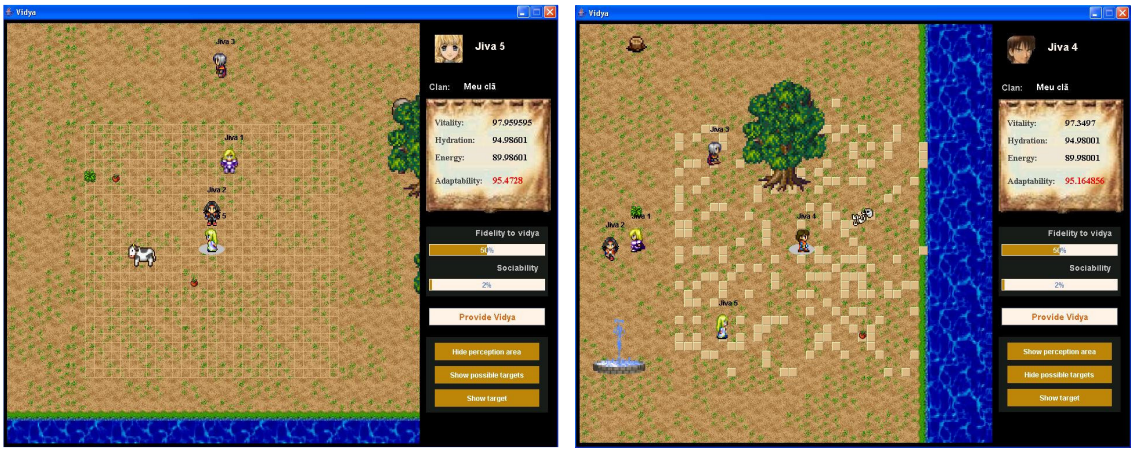
O alcance de percepção é a área dentro da qual o *Jiva* consegue ver objetos (Figura 9.a). Fora desta área, nada é visto pelo *Jiva*, indicando que ele não tomará decisões baseadas em objetos externos a ela. Para fazer com que o *Jiva* exiba esta área de percepção, basta clicar no botão "*Show perception area*", e para esconder basta clicar novamente no mesmo botão, agora com nome "*Hide perception area*".

6.7. Exibindo / Escondendo Possíveis Posições Alvo

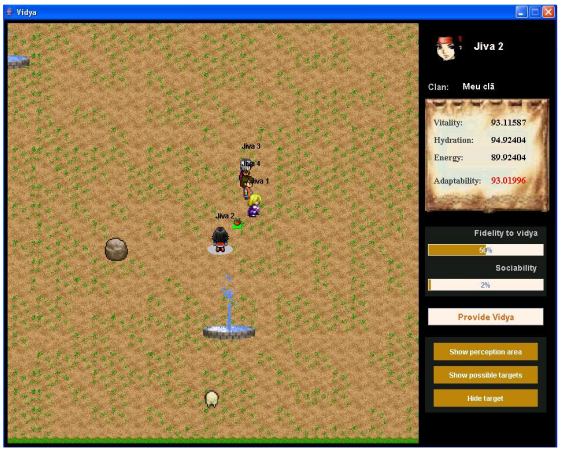
As possíveis posições alvos são os lugares que o *Jiva* considera antes de tomar a decisão de migrar para um deles (Figura 9.b). Para fazer com que o *Jiva* exiba ou esconda essas posições possíveis, deve-se clicar no botão de rótulo "Show possible targets".

6.8. Exibindo / Escondendo Posição Alvo

A posição alvo é a posição escolhida pelo *Jiva* para migrar num instante de tempo, dentre as diversas posições possíveis (Figura 9.c). Para exibir ou esconder esta posição alvo, o jogador deve clicar no botão "*Show target*".



(a) (b)



(c)

Figura 9. (a) Alcance de percepção do *Jiva*; (b) Possíveis posições alvo do *Jiva*; (c) Posição alvo do *Jiva* (célula em verde).

7. Fim do Jogo

O jogo acaba quando todos os *Jivas* de um dos clãs morrerem. Se os *Jivas* do clã oponente morrerem primeiro, o jogador é o vencedor; senão quem vence é o oponente.

Anexo C

Vidya: A God Game Based on Intelligent Agents Whose Actions are Devised Through Evolutionary Computation

Marcelo Rodrigo de Souza Pita, Salomão Sampaio Madeiro, and Fernando Buarque de Lima Neto

***Abstract*—*Vidya* is a strategy computer game, god-style, that can be seen as an environment where virtual beings compete among themselves for the natural resources and strive within an ecosystem. Although in this game the player cannot directly control the agents, he can give some intuitions to them. Together with these intuitions the intelligent agents, called *Jivas* – the most developed species of the ecosystem, devise actions through evolutionary computation. The game allows also the observation of all interactions among the various beings inhabiting *Vidya*. Interactions happen in a quasi-autonomous manner which grants the game with an interesting dynamics. The evolved *Jiva*'s intelligence, which build-up during the game, can be reused in other game scenarios. This work helped on further understanding of some emergent autonomous behaviors and parameterization of intelligent agents that live in closely coupled ecosystems.**

I. INTRODUCTION

VIDYA is a strategy computer game, posed as a environment where virtual beings compete for available natural resources of an ecosystem. Among many types of beings that inhabit the virtual world, there is a special one called *Jiva*; they are intelligent agents designed to be autonomous and survive against all the odds. In the *Vidya* game, the player is invited to be a *Jivas*' clan *Deva* (*i.e.* their god), providing guidance, but not detailed one, to every clan member. These instructions appear in the *Jiva*'s mind as intuitions, known in the game as *vidyas*, which are useful for helping the *Jivas* to take their decisions. Due to the *Jivas*' "autonomy", they are capable of deciding which actions to perform for various situations in the world (having or not the *vidya* provided by player).

Artificial Intelligence (AI) techniques [1] are heavily used in computer games [2]. They are today one of the main factors of success or failure of any non-casual game. Two main factors contribute to this in electronic games. First, following the appearance of graphical accelerator boards, much of central processing potential became available for AI computations. Second, newer games are continuously required to have greater degrees of realism [3], [4]. The realism referred here applies not only in the graphical design and physical modeling of the game, but also in the evoked behavior of the agent.

This means that a good non-casual game that explores computational intelligence not only has to be non-monotonic behavior-wise, but has to possess plastic beauty as well; desirably an even balance should be obtained between these two "ingredients".

In the *Vidya* game the algorithms are written in a way that intelligent behavior is at the center of the game's processing demand, especially when controlling the behavior of the *Jivas*. There are others agents in the game, namely, animals like sheep, wolfs, cows, that express non-monotonic behavior. But in these agents' minds, behaviors are reactive processes. In other words, for these lower beings of *Vidya*, behaviors are state-to-actions static mappings, without evolution. As opposed to that, *Jiva*'s intelligence evolves thru time and increases its knowledge about the world. They autonomously learn by their own experience as well as godly received inspiration (*i.e.* the *vidyas* provided by the player). This happens because all *Jivas* learn by self-analyzing

their own behavior, evaluating whether a given action at a given time was appropriate. In the long run, this helps gathering insights for Evolutionary Computation [5].

Apart from the ludic aspects of *Vidya*, the result of the intelligence modeling of *Jiva* is the creation of evolutionary intelligent agents, which act and learn in a quasi-autonomous way.

Furthermore, as a glimpse of future extension, *Vidya* could be easily adapted to be a simulation environment to help on the understanding of other living beings social behaviors and ecosystems.

II. Intelligent Techniques Utilized

Genetic Algorithm (GA) is an intelligent technique widely used for solving complex search and optimization problems based on principles of evolution of nature [6], [7]. Population, fitness, chromosomes, genes, reproduction, mutation and offspring are important concepts brought from Genetics & Nature straight into this technique.

The Population is a set of individuals or chromosomes, here possible solutions for the Jiva's decision problem. The fitness function calculates a value to classify how good an individual is according to some criterion. Reproduction (crossover), selection and mutation are GA operators applied directly on the population to generate a new one. By using those concepts it's sufficient to implement a powerful search in the input space.

Before using GA it is necessary to model every possible solution for the problem at hand into a chromosome. This can be done by mapping judged relevant characteristics into genes. An initial population of individuals, *i.e.* set of chromosomes, is generated randomly. After being classified by some fitness criterion, they will evolve and guide de search process to an optimal solution for the problem. After fitness value calculation, two individuals per time are selected due to methods such as the Roulette Wheel [8] or the Tournament [8]. Following that their chromosomes are combined to generate a new individual. At this time, it's possible to introduce a random operator, *e.g.* mutation, to avoid premature convergence, by changing one gene value of that new individual. Crossover and mutation will happen until a new population with the same size is generated. Figure 1 illustrates all mentioned steps.

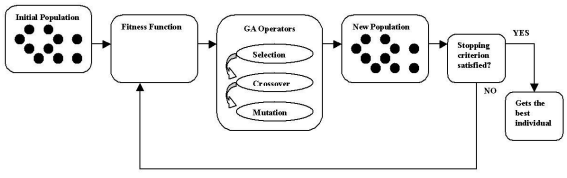


Fig. 1. Genetic Algorithm steps overview.

Finding a good solution to a problem can be a very time consuming operation, especially whether one tries to explore the entire solutions surface. GA is capable of finding an answer as good as needed by executing the mentioned cycle a number of times sufficient to pick-up a solution. Note that for every cycle, there is one best individual in the current population. Thus, as soon as there is a need for the algorithm to stop (*e.g.* the user demands an action to be taken) , a non-random candidate solution, a suitable for the problem, is readily offered by the G.A. . It is important to be noticed that, in general, it doesn't take many cycles to find a reasonable solution.

III. Vidya

A. The Vidya Game

Vidya is a single-player strategy-god game [9] game that includes AI and a new player-character interaction model. *Vidya* is also a test-bed for investigations that aims at advancing development of autonomous intelligent agent through Evolutionary Computing.

The intelligent decision module of *Vidya* agents was projected to be used in other games, in characters that need more autonomy, in agents of ecosystem simulations, or even in studies of emergent social behaviors.

The *Vidya* game environment is shared by many types of beings (as show Figure 2) that compete for available natural resources, sometimes not abundant. The main characters in this environment are the *Jivas*, they are autonomous intelligent agents that are endowed with evolutionary properties. The social unit of the *Jivas* is their clan.

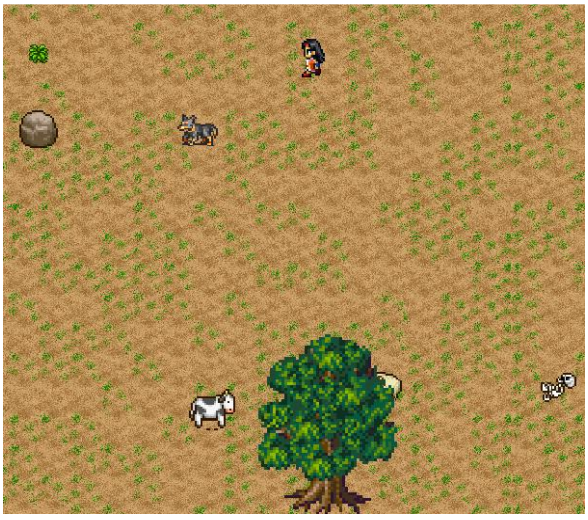


Fig. 2. Screen-shot of the game world, showing the *Jiva* (upper-center) and other beings like fox, rock, cow and tree.

The player performs the role of a clan's *Deva*, and has to instruct the *Jivas* of one clan, by helping them to survive in the game world. This is not an easy task, especially because *Jivas* are autonomous beings, and can choose to follow the players instruction, the *vidyas*, or simply choose to ignore the so-perceived as "divine intuition".

B. The Vidya Software

Vidya was completely developed in Java language [10]. Java is easy for programming, have plenty of Application Programming Interfaces (APIs) already implemented and is Object-Oriented (OO), what makes modeling much easier. The main concern regarding Java is it notorious not so good performance when compared to other programming languages, like C and C++.

There is an intention of the authors to migrate the *Vidya* source code into C++. The C++ language is considered the best general programming language for desktop computer games today, for which there is many game engines, graphical libraries, etc. Besides, C++ is OO, like Java, which ease the migration. Figure 3 shows the high level software architecture of the *Vidya* game.

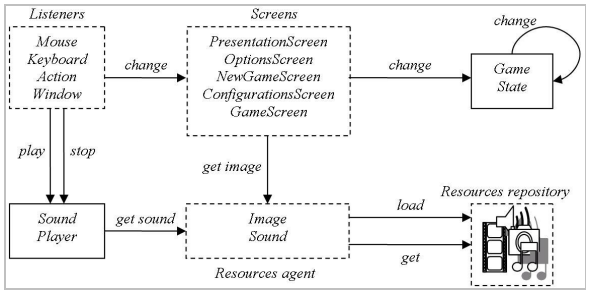


Fig. 3. High-level software architecture of the Vidya game.

The *Listeners* module listen for player's input, generating events that, sometimes exchange screens, sometimes change the game state (delegated by *GameScreen* screen). Events are still generated to control the *Sound Player*.

The *Resources repository* is a set of image and sound files that are accessed through *Resources agent*. The *Screens* and *Sound Player* modules access (via *Resources agent*) image and sound resources, respectively.

Inside the *Game State* module we have the game motor, the world and its objects, of which we find the *Jiva*, with its intelligent decision module.

C. The Jiva's Intelligent Decision Module

Jiva (and other beings inhabiting the game world) follows the agent's model. Specifically, *Jiva* is an autonomous intelligent agent, because it is capable of taking intelligent decisions in an independent way.

Like all agents, *Jiva* perceives the environment, performs an inner processing, to selecting a good action, and act on this environment. The *Jiva*'s perception is partial, that is, it does not perceive the whole environment, but only inside of a variable perception limit.

It is easy deduce that the basic problem of the *Jiva* is to perform the best action on the environment for a given perception, trying to minimize a cost function (or maximize a gain function). *Jiva*'s actions might be seen as control operations that intend to benefit from the environment in order to create ideal conditions for its own surviving.

The task of the *Jiva*'s intelligent decision module is responsible for select (or decide) the best action to be performed by the *Jiva*'s agent. This action selection task is realized using GA (the AI technique explained in section II).

Jiva has a very large set of available actions that can be performed, and selection of the best one can be time spent using traditional techniques, in which all actions have to be evaluated and qualified. With GA we minimize the search space, and the choices converge to the best action, having still a good solution at the first generation. An action is, then, mapped to an individual in the GA. The GA's population is a subset of the available actions.

Jiva perceives a set of world objects inside a perception square. This perception limit depends on *Jiva*'s vitality. In the beginning, the *Jiva* has the maximum vitality and perception level, and goes losing perception capacity in proportion to vitality's decay. *Jiva*'s vitality can be interpreted as the remaining life time of the *Jiva*.

The perception square of the *Jiva* defines a set of cell that *Jiva* can perceive. These cells also define the complete possible future positions space of the *Jiva* (the possible positions that *Jiva* can fill at the next time step) at the first time a perception state is reached. Figure 4 shows the perception square of the *Jiva*, the objects that are perceived and the cells set of the square.

The *Jiva*'s perception is characterized by a set of world objects. Each object in the perception is characterized by its type and positioning (relative to the *Jiva* position).

In the present work, the *Jiva* is able to perceive occluded objects, that is, it can see all objects that are inside its perception square, even whether some objects are behind other objects. Uncertainties caused by occlusion of objects is an interesting topic [11], however, we left it for future work.

Jiva's agent, after it has perceived the environment and created an inner representation of perception, will select a good action for that perception.

An action is defined as the destiny cell to which the *Jiva* will move to. So, we characterize an action by a coordinate pair (x, y), meaning that the *Jiva* will move to that position in the next time step, as shown in Figure 5.

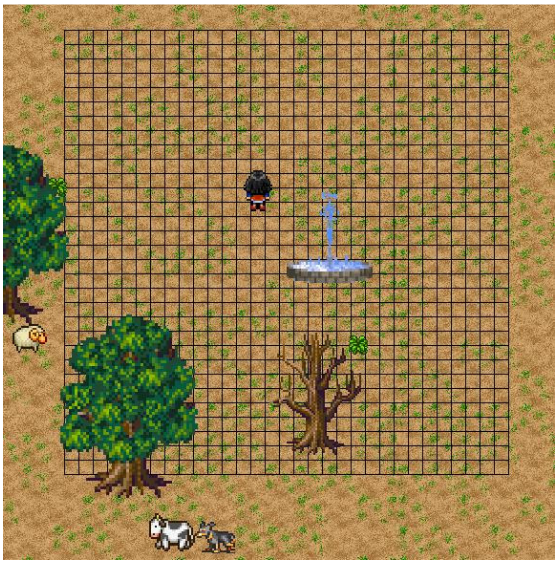


Fig. 4. Screen-shot of the game world, showing the perception square of a *Jiva*. The objects inside the square are being perceived (a water font, a small plant, a dead tree and a tree). The small squares that compose the perception are the perception cells. Outside of the perception square, in the bottom of image, a wolf (predator) is hunting a cow (prey).

Fig. 5. Destiny cell, for which *Jiva* will migrate. In destiny cell (top) there is a fruit, and the *Jiva* will eat it.

The semantic value associated to an action depends on the object that is occupying the destiny cell. For example, if a *Jiva* migrate to a cell where a fruit is (Figure 5), the semantic value associated to this action is "eating the fruit". Else, if in the destiny cell a wolf is, the semantic effect is "hurting the wolf". If the destiny cell is empty (no object is there), the semantic value of the action is just "walking until the destiny position". For each object type in the destiny cell there is a different semantic value associated. For some object types, the semantic value associated may be the same (*e.g.* sheep and cow).

The semantic implications of actions in the vital properties of the *Jiva* are also dependents on the objects that occupying the destiny cell (*e.g.*, eating a fruit increases the vitality and some of energy, while eating dead wolf, though increase more in energy, decreases vitality).

D. *Solving the Problem Through Evolutionary Computation*

A unique GA instance cannot solve the problem of selecting the best action for all possible situations: a good action for a given perception may not (probably will not) be good to another perception. Each different perception is a different problem, and then different GA instances have to be created for each one of these different perceptions. The *Jiva*'s intelligent decision module has a structure that maps perceptions on instances of GAs (a one-to-one mapping).

Each GA instance has the role of selecting the best action for a given perception, following the natural selection method that characterizes a GA. This includes the evaluation, crossing and mutation of individuals.

When evaluating an action, the GA might care about not only its immediate cost, but infer a long term cost for it. Sometimes we will refer to this long term evaluation as an evaluation "*F* steps in the future", where *F* is the quantity of steps that the GA will see in the future to estimate a long term cost.

The long term cost can be inferred because the *Jiva*'s intelligent decision module can simulate the progress of an action in the future. Of course, the estimated long term cost have to be confronted with *Jiva*'s experience, that is, the real cost obtained by interacting with the environment.

The path of *Jiva* in the game world is internally represented as a states machine (Figure 6), where each state is a different perception configuration of the *Jiva*'s agent.

Associated to each state (s_i) there is an intelligent processing element, that is an instance of a GA (GA_i). The task of this element is selecting the best action (a_{ij}) for the state. Due a performed action, the *Jiva* migrate to another position in the game world and has a new perception, for which there is another GA instance to "solve" it.

When the *Jiva* reaches a perception state, it has a set of positions to go (the set of possible actions). Calculating the cost to go for each destiny cell is a

heavy task, because the number of possible destiny cells is very large. For example, if we have a perception square of 20 × 20 cells and we would like to know its cost 3 steps in the future ($F = 3$), 400 cells would be evaluated, having for each cell a triple analysis (the 3 steps in the future). After this task we would still have to do a comparative analysis between these 400 cells to choose one.

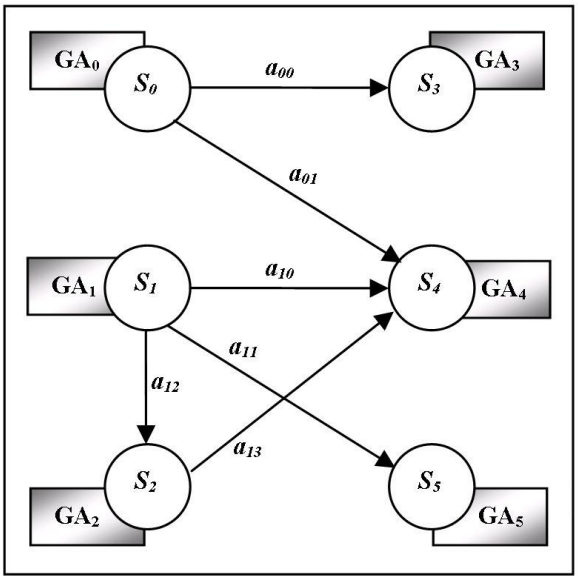


Fig. 6. States machine that represents the path of *Jiva* in the game world. Each state is a different perception configuration, for which there is a GA instance that will select the best action for that state.

Using Genetic Algorithms, we have a convergent method (converging to the best solution) with a smaller search space. Supposing a GA with a population size of 40 individuals, for the problem exposed in the last paragraph, we have a meaning reduction of 90% in the search space. Figure 7 shows the initial spatial distribution of the population for a given perception of the *Jiva*.

Localized in a state, the *Jiva* calls the associated GA instance to select an action. Each time this state is reached, the GA instance advances one generation. In the first generation we still have a converging solution. This advance is characterized by the common tasks in all GA: evaluating individuals, crossing and mutating individuals.

Before all, evaluation of the population is done for selecting the best action that will be performed by the *Jiva*. The evaluation phase only is completed when the action is performed and the *Jiva* has a real immediate cost obtained by experience. So, the evaluation phase is divided in two times: before the selection and after the action.

Before the selection, each action is internally simulated F steps in the future. The action that has the best qualification in the simulation is the selected. Figure 8 shows the simulation algorithm for one action (pseudo-code). After obtain the qualification (inferred cost) for each action, the action that has the greater qualification will be selected and performed by the *Jiva*.

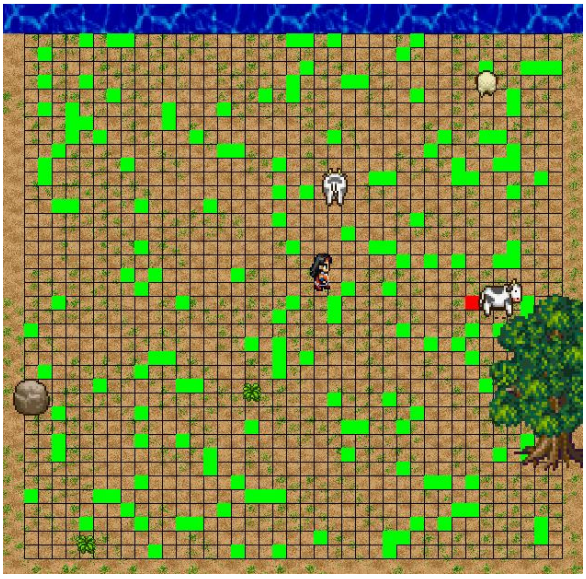


Fig. 7. Initial spatial distribution of the population for a given perception. We can see the cells that are inside the 41×41 perception area, the population distribution (168 individuals, that is, 10% of the total search space, in light gray cells) and the destiny cell (dark gray cell), selected through an analytic simulation considering 5 steps in the future. The *Jiva* (center) is hunting a cow (right).

```
parameters: currentAction, currentState, F
return:     qualification

var qualification := 0

iterate 1 to F:
    /* Qualification update.                    */
    /* costToGo returns the cost to realize     */
    /* the current action in the current state. */
    qualification := qualification +
                costToGo(currentState, currentAction)

    /* New state reached */
    var newState := updateState(currentState,
                                currentAction)
    /* GA instance for the new state reached */
    var gaInstance := getGAFor(newState)
    /* Best action for the GA instance */
    var newAction := getBestActionFor(gaInstance)

    /* Makes current action the new action */
    currentAction := newAction
    /* Makes current state the new state */
    currentState := newState

/* Returns accumulated qualification */
return qualification
```

Fig. 8. Algorithm for the simulation (in pseudo-code) of an action F steps in the future.

For the time being, *Jiva*'s experience is out of scene. Now, the winner action, after to be performed, will be evaluated again. The immediate cost obtained by the *Jiva* in realizing the recent action will be summed to the qualification of the action, re-qualifying it. After this post-evaluation, the winner action may keep on being the best one, or may lose the first position to another action best qualified. So, this post-evaluation try to correct inference errors in the simulation, re-ordering, in terms of qualification, the individuals of the population. These individuals are, now, ready to the crossings and mutations phase.

An action for the *Jiva* is walking to a destiny cell. So, actions are represented by a coordinate pair (x, y), indicating this destination.

The selection method used to choose pair of actions that will cross is the Roulette Wheel selection method [8]. Pairs of actions are selected and crossed, generating new actions that will compose the next generation of the population.

The mutation operator is also applied to the population. The mutation probability is, in the *Jiva* case, 1/32, that is, in average, for each 32 actions 1 is mutated.

The use of GA in the *Jiva*'s intelligent decision module aims to reduce considerably the search space for selecting actions in a convergent fashion, using natural selection principles and generating good solution still in the first generation of individuals.

Each time a state is reached, a new generation of better new actions replaces the old ones in the population. The old actions are evaluated considering their probable long term qualification (*F* steps in the future, where *F* is a valid parameter of the game), achieved through progress simulations, and the selected action (that has the best qualification) is re-qualified by its immediate cost after to be performed. So, these old actions are submitted to crossings and mutations operations, resulting in a new generation of actions better qualified.

The trends in computer games are that the game worlds will became more and more complex [12] (this is seen in the ultimate generation of computer games). This crescent complexity has turned hard the explicit programming of agents' behaviors (in the cases of purely reactive agents), or, when AI is evolved, the obtaining of representative knowledge database of the specific game world, because the number of different situations that happens in truly complex game worlds is very large, considering, inclusively, situations caused by players that are, at least in first analysis, non-deterministic.

Creation of adaptive agents capable to learn in novelty situations is a good way to contour this difficulty.

The *Jiva*'s agent and its intelligent decision module is an instance of the efficient use Evolutionary Computing in computer games, specifically in the behaviors of their autonomous agents. This intelligent decision module can be reused in other computer games, in autonomous agents that need to be adaptive at game world adversities and capable to learn in novelty situations.

E. *A New Player-Character Interaction Model*

As mentioned, the main character in the *Vidya* game is the *Jiva*. It is the focal-point between the player and the game.

Traditionally, interaction between player and characters in a computer game happens in a very direct way. Through controls, the player can operate almost completely (else completely) the behavior of the characters (also referenced as avatars).

In the *Vidya* game, the player is a clan's *deva*, whose role is to mentor the clan and to promote surviving in the game world. This is a hard task, because *Jivas* are autonomous intelligent agents, and can choose the action to perform, despite the action suggested by the *deva* (the player).

The non detailed instruction the player gives for the *Jivas*' clan (individually for each integrant) is known in the game as *vidya*. A *vidya* appear inside of each *Jiva* as an "intuition". The goal of the player is provide good *vidyas* to the *Jivas*, which will help them to survive.

When the player intercepts the *Jiva* to provide *vidya*, the game pauses and then the player can suggest a destiny cell that the *Jiva* must follow. Among others possible destiny cells, which are the individuals of the population of the GA instance for that particular perception, the *vidya* will selectively compete for the best qualification. The most qualified ones will pass their characteristics for the next generation. If the action suggested by the player is a good action, it will help the long term learning of the *Jiva*. This is the main goal of the *vidya*.

IV. Experiments and Results

The experiments included in this section were used to evaluate the intelligent evolutionary algorithm proposed in the last section. Basically, this algorithm aims to implement the *Jiva*'s intelligent behavior, which is based on Genetic Algorithms. In the proposed algorithm all *Jiva*'s actions are mapped on individuals of a given population. The feature of each individual is obtained through simulation of n-step future action. Then, the so-called best actions will pass on its characteristics to the next generation.

The focus of all experiments is at computational performance evaluation of the algorithm as an indirect method for evaluating intelligence. The idea is to assess intelligence thru analyses of the

adaptability of the *Jiva*.

We can identify two main parameters within the algorithm: (i) the GA's population size – *PS* and , (ii) the number of future steps that the simulations can see – *F*. We varied these two parameters to observe the algorithm behavior according to possible actions.

The set of individuals, for which the GA's population is a subset, has a size equals to the *Jiva*'s perceptive area. We assumed a constant perception area of 961 cells (31 × 31 square) in these experiments, so that they can be compared among themselves. The *PS* values assumed to were 10%, 20% and 40% of all possible actions, that is, *PS* = 96, *PS* = 192 and *PS* = 384, respectively. The *F* values assumed were *F* = 2, *F* = 4 and *F* = 6.

Absolute values expressed in the results are not so important here. Actually, the experiments propose a comparative analysis of the same algorithm with different values assumed by parameters (*PS* and *F*). They show the best parameters configuration that maximizes the *Jiva*'s performance (computational and intelligent).

The hardware used was a regular personal computer AMD Duron™ 1.6 GHz, with mere 352MB of RAM.

A. *Performance of the Algorithm*

The graphic of Figure 9 shows the performance, in milliseconds, of the proposed algorithm for the selected parameters; that is, variations for *F* (steps into the future) and *PS* (population size of the GA).

The three lines shown in Figure 9 are trends evoked by the algorithm related to its computational

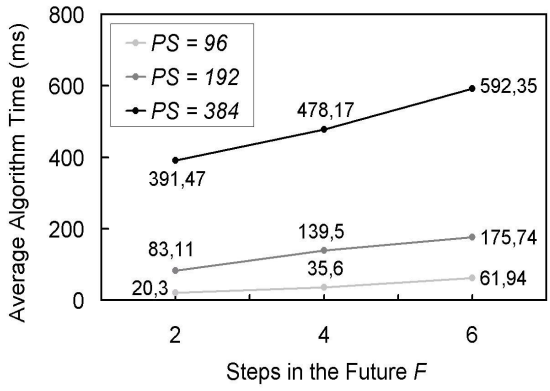


Fig. 9. Performance change of the proposed algorithm due to selected parameters variations.

performance. For every trend, processing time augment as function of *F*. The observed growth is practically linear.

Each point in the graphic was obtained by arithmetic average over 100 executions. In order to obtain this data we have to add extra routines to the main code. However, the results referred above are only related to the examined code.

This behavior is foreseeable for the algorithm, respecting the processing load that is ought to grow linearly with *F*. The differences in the measured values across populations of different sizes, for the same values of *F*, are foreseeable too, and have an equally linear growth trend.

The absolute values obtained in the experiments also reveal that the overall computational performance of the algorithm is not large (for the test cases, don't exceed 600ms), and is perfectly applicable in real problems that have response time limitations.

B. *Jiva's Intelligence Measurements*

Measuring intelligence is difficult. To it, we have to use an indirect method that, in our case (the Jiva's intelligence decision module) is evaluate the gain (that can be positive or negative) obtained by the *Jiva*, during a period of time interacting upon the environment. That is, we are using the *Jiva*'s adaptation capability in the *Vidya* game to measure the algorithm performance in provide intelligent behavior.

The most relevant *Jiva*'s features, the ones that interest most for evaluation of the *Jiva*'s adaptability, are: 'vitality', 'hydration' and 'energy'. A weighed average of these values, which can be obtained instantaneously from every *Jiva*, produces what we refer as 'general vital condition'; this value changes thru time. In accordance to absolute values of the general vital condition, values assumed for vitality, hydration and energy were 50%, 30% and 20%, respectively..

If we measure the gain obtained in the general vital condition by the *Jiva* after a period of time, we can state that this value represents approximately the *Jiva*'s increased capability for surviving. Moreover, this also means how well it has performed to obtain additional resources for its own survival. Note that in our experiments we do not evaluate the social behavior of the *Jivas*, which probably is qualitatively superior, but only the *Jiva*'s individual behavior (*i.e.* relating to its individual survival). The analysis of emergent social behavior of the *Jivas* is left as a future work.

The graphic of Figure 10 shows the gain in general vital condition obtained by a *Jiva* for the possible parameters variations (*F* and *PS*), in a period of 10 min. Each value in the graphic is an arithmetic average of 3 executions. A death in some of these tests represents a loss of -100 (negative gain), because 100 is the *Jiva*'s maximum initial general vital condition. E.g., for *PS* = 96 and *F* = 6, the gain of -100 shows that *Jiva* died in the three tests.

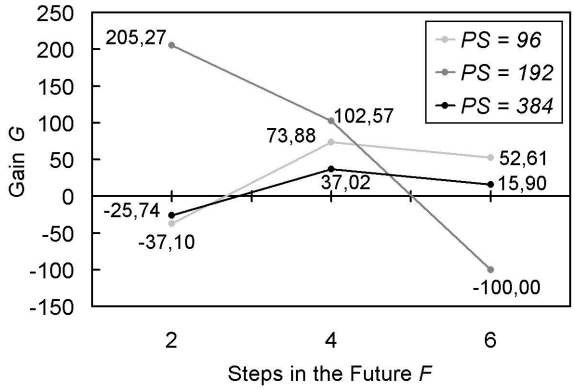


Fig. 10. Gains in the general vital condition of the *Jiva* for the possible parameters variations.

Theoretically, the larger is *PS* and *F*, the larger should be the gain in general vital condition of the *Jiva*. But, this expectation w disproved by the experiments.

Let's analyze the average gains for variations in *PS*. For *PS* = 96, we have an average gain of 29.8 points in the general vital condition. For *PS* = 192, we have the greater standard deviation, with the best and the worst values, resulting in an average gain of 69.28 points for the general vital condition. For *PS* = 384, we have an average gain of 32.15 points in the general vital condition of the *Jiva*. Then, the best value for *PS* is 192.

Let's now analyze the average gains for variations in *F*. For *F* = 2 we have an average gain of 47.48 points; for *F* = 4, we have an average gain of 71.16 points; and, for *F* = 6, we have an average gain of -10.49. Based on this information, we conclude that the best value for *F* is 4.

Overall, the best parameter combination, considering the synergy among them, was is *PS* = 92 and *F* = 2.

This result is interesting because is counter-intuitive. When *F* grows, the *Jiva* can see more steps in the future but , this analysis don't take into account the immediate cost of realize a specific action. This action, even if it produces the best gain in long term, may conduce the *Jiva* to situations where the short-term cost is very large, causing its death. This elucidates why the best configuration has a small value for *F* = 2, and the best average value for *F* = 4 is not the greater.

C. *Compromise between Intelligence and Performance*

The greater value for *PS* in the experiments (*PS* = 384) also do not help, because the *Jiva* "think" more and, consequently, becomes slower.

The parameters configuration *PS* = *92* and *F* = 2 was found to be the best parameters combination for *Jiva*'s adaptability. In addition to establish a compromise with the computational performance that, for *PS* = 92 is on the average (as shown in Figure 8) and for *F* = 2 (the best case).

V. CONCLUSION

This paper introduced the *Vidya* game and put forward a particular autonomous intelligent agent, the *Jiva.* It is an artificial living being capable to take non-monotonic behavioral decisions and learning by self-analyzing its own behavior in the world. The game also offers a new player-character interaction model, where the *Jivas* are not fully controlled by the player.

Computer games are continuously requiring best artificially intelligent agents, and the *Jiva*'s AI module intends to contribute towards this demand, especially at modeling the *Jiva*'s intelligence through Evolutionary Computation.

The use of Genetic Algorithms here proved to be a good choice because not only it produces good results in improving Jiva's intelligence (which was assessed through simulations on the *Jiva*'s adaptability), but also the computational performance was good in supporting agents decision processes.

Regarding computation performance, we have noticed that processing time increased linearly with the problem size, namely, increase of *PS* and *F* parameters. Even in absolute values, the algorithm did not exceeded 600ms of processing time for one step – which is acceptable given de relative complexity of the environment.

Regarding intelligence performance, the results were also very promising. We have learned about tradeoffs in increasing or decreasing the population size of the AG according to future steps in the inner actions simulations. We noticed that the number of future steps in the inner actions simulations is favorable towards the game objectives. That is, the fastest the *Jiva* thinks the better.

Finally, the *Jiva*'s adaptability results due to parameter setting were also useful in determining the best configuration that corresponds to a compromise between computational intelligence and performance.

We left as future works the following tasks:

- To reuse *Jiva*'s intelligence in other computer games and game scenarios;
- To analyze emergent behaviors of *Jivas* societies;
- To study more complex ecosystems and societies behaviors using the proposed environment;
- To tackle uncertainties due occluded perception;
- To program *Vidya* in faster programming languages, such as C++, for higher performances.

ACKNOWLEDGMENT

This work was partially sponsored by CNPq, the Brazilian federal agency that supports scientific and technological development, under Grant 200295-98.5.

REFERENCES

[1] S. Russell and P. Norving, *Artificial Intelligence: a modern approach*. New Jersey: Prentice-Hall, 1995.

[2] A. Nareyek, "AI in computer games," in Queue, Game Development: Serious Business, Serious Coding, vol. 1, issue 10, ACM Press, New York, 2004.

[3] B. C. Bridger and C. S. Groskopf, "Fundamentals of artificial intelligence in computer games," *Proceedings of the 38th annual on Southeast regional conference*, ACM Press, Clemson, South Carolina, 2000.

[4] J.E. Laird and M. van Lent, "Interactive Computer Games: Human-Level AI's Killer Application," in *Proc. Nat'l Conf. A.I.,* AAAI Press, Menlo Park, Calif., 2000.

[5] Z. Michalewicz and M. Michalewicz, "Evolutionary computation techniques and their applications," in IEEE International Conference on Intelligent Processing Systems, 1997.

[6] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. New York: Addison-Wesley, 1989.

[7] L. Nang and K. Matsuo, "A survey on the parallel genetic algorithms," in JSICE, 1994.

[8] Z. Jinghui, "Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms," in CIMCA, vol. 2, 2005, pp. 1115 – 1121.

[9] C. Crowford, *The art of computer game design*. Washington State University, 1982.

[10] Sun Microsystems. *Java 2 Platform, Standard Edition*. Available: http://java.sun.com/javase/index.jsp (30/10/2006).

[11] K. Fukushima, "Recognition of occluded patterns: a neural network model," in IJCNN, 2000.

[12] J. E. Laird, "Using computer game to develop advanced AI," in Computer, vol. 34, issue 7, 2001, pp. 70-75.

Anexo D

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Departamento de Sistemas Computacionais

TRABALHO DE CONCLUSÃO DE CURSO

"Vidya – um jogo de estratégia que inclui intuição e Inteligência Artificial"

Aluno: Marcelo Rodrigo de Souza Pita
Orientador: Professor Fernando Buarque de Lima Neto, PhD

Questionário de Avaliação do Jogo Vidya

Que tipo de jogador você se considera?

- ☐ Casual. Jogo apenas para me distrair.
- ☐ Adepto. Jogo sempre que posso e encaro com muita seriedade.
- ☐ Não me considero um jogador.

Antes de jogar Vidya, leia a seguinte descrição do jogo:

Vidya é um jogo de simulação de ecossistema, aonde vivem muitos seres vivos artificiais que se reproduzem e disputam recursos naturais disponíveis. Dentre estes vários tipos de seres, há um especial chamado *Jiva*. Os *Jivas* são seres vivos inteligentes que têm autonomia para tomar suas próprias decisões. Além disso, eles se reúnem em clãs e agem socialmente dentro desse contexto.

Em Vidya, o jogador é o deus de um clã de *Jivas*. Como deus, não poderá interferir diretamente sobre os *Jivas*, mas sim fornecer apenas intuições a eles, as chamadas *vidyas*. Apesar disso, os *Jivas*, que são seres autônomos, podem escolher entre seguir ou não as *vidyas* que o jogador fornece para eles. Quanto mais boas *vidyas* (que ajudem os *Jivas* a aprender e sobreviver) o jogador fornecer, mais fiel os *Jivas* se tornam a elas.

No mundo em que se desenvolve o jogo, há dois clãs: o do jogador e o do computador – a tarefa do jogador é fazer seu clã o último sobrevivente dentre os dois. Para isso, o jogador deverá pensar em estratégias para vencer o clã inimigo que ameaça a existência do seu clã dentro do mundo. Além disso, também deverá ajudar os *Jivas* do seu clã a viver num ambiente cheio de conflitos.

Após ter jogado Vidya, responda as seguintes questões:

Como você avalia a proposta do jogo?

- ☐ Não gostei da idéia do jogo.
- ☐ Gostei de pouquíssimas coisas na idéia.
- ☐ Pouquíssimas coisas eu não gostei.
- ☐ Achei a idéia muito boa.

Quão divertido (prazeroso em jogar) você considera o jogo?

- ☐ Sem diversão alguma.
- ☐ Achei pouquíssimas coisas divertidas.
- ☐ Algumas poucas coisas não são divertidas.
- ☐ O jogo é muito divertido.

O que você achou da forma de interagir com os *Jivas*, incluindo a autonomia deles?

- ☐ Simplesmente não gostei. Acharia melhor se os controlasse diretamente.
- ☐ Achei interessante, mas só por um tempo.
- ☐ No começo estranhei, mas com o tempo fui gostando cada vez mais.
- ☐ Achei muito interessante desde o princípio.

A interação com o jogo é:

- ☐ Muito ineficiente. Praticamente não compreendi como interagir e a interface não é prática.
- ☐ Pouco eficiente. Aprendi / Consegui manipular somente algumas coisas.
- ☐ Eficiente. Poucas coisas não consegui manipular direito.
- ☐ Muito eficiente. Facilmente compreensível e prática.

Os elementos gráficos do jogo são: (considere como padrão um jogo 2D tradicional)

- ☐ Muito ruins. Têm que mudar completamente.
- ☐ Ruins, mas alguns elementos são aceitáveis.
- ☐ Bons, mas alguns elementos estão ruins.
- ☐ Muito bons.

Os elementos sonoros do jogo são:

- ☐ Muito ruins. Têm que mudar completamente.
- ☐ Ruins, mas alguns elementos são aceitáveis.
- ☐ Bons, mas alguns elementos estão ruins.
- ☐ Muito bons.

Como você avalia a dinâmica do ecossistema presente no mundo do jogo?

- ☐ Muito ruim. Achei tudo muito parado, sem vida.
- ☐ Algumas poucas coisas são legais e dinâmicas, mas no geral está ruim.
- ☐ Eu não gostei de algumas poucas coisas, mas está bom no geral.
- ☐ Achei muito interessante o desenvolvimento do ecossistema.

Você se sentiu de fato o deus de um clã de *Jivas*? Aprendeu a manipular eles?

- ☐ De jeito nenhum. Não me senti um deus, nem mesmo consegui manipulá-los.
- ☐ Em parte sim. Mas senti muita dificuldade em interagir com eles.
- ☐ Sim, ainda que algumas poucas vezes tenha sentido dificuldade em manipulá-los.
- ☐ Me senti um deus, de fato. Aprendi a manipular os *Jivas* bem.

Você conseguiu pôr em prática estratégias durante o jogo?

- ☐ Não consegui absolutamente.
- ☐ Consegui pouquíssimas vezes.
- ☐ Consegui muitas vezes. Contudo, algumas vezes não consegui.
- ☐ Consegui sempre pôr em prática todas as minhas estratégias, apesar de algumas terem dado errado.

Agradecemos a sua participação!