FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO "EURÍPIDES DE MARÍLIA" – UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL PELEGRINO GARCIA

UMA COMPARAÇÃO ENTRE FERRAMENTAS DE GERAÇÃO DE COMPORTAMENTOS AUTÔNOMOS EM AMBIENTES VIRTUAIS

MARÍLIA
2005

FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO "EURÍPIDES DE MARÍLIA" – UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL PELEGRINO GARCIA

UMA COMPARAÇÃO ENTRE FERRAMENTAS DE GERAÇÃO DE COMPORTAMENTOS AUTÔNOMOS EM AMBIENTES VIRTUAIS

Monografia apresentada como pré-requisito de conclusão do curso de Bacharelado em Ciência da Computação, da Universidade Eurípides de Marília – UNIVEM.

Orientador:
Prof. Dr. José Remo Ferreira Brega

MARÍLIA
2005

GARCIA, Rafael P.

Uma comparação entre ferramentas de geração de comportamentos autônomos em um ambiente virtual. Rafael Pelegrino Garcia; orientador: José Remo Ferreira Brega. Marília, SP: [s.n.], 2005.

Monografia (Graduação em Ciência da Computação)- Curso de Ciência da Computação, Fundação de Ensino “Eurípides Soares da Rocha”, Mantenedora do Centro Universitário Eurípides de Marília – UNIVEM.

1. Agentes Autônomos 2. Mundo Virtual 3. Comportamento

CDD: 006

Rafael Pelegrino Garcia

"UMA COMPARAÇÃO ENTRE FERRAMENTAS DE GERAÇÃO DE COMPORTAMENTOS AUTÔNOMOS EM AMBIENTES VIRTUAIS"

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: _____(_____)

Orientador: José Remo Ferreira Brega _____

1º. Examinador: Antonio Carlos Sementille _____

2º. Examinador: Ildeberto Aparecido Rodello _____

Marília, 07 de dezembro de 2005.

Agradecimentos

Aqui, no término dessa minha caminhada, deixo meus agradecimentos, sobretudo a Deus, que me iluminou e me deu forças para a escala de mais este degrau.

Aos meus pais e a minha irmã por sempre estarem ao meu lado.

Ao professor Remo, que foi meu orientador, e me ofereceu a oportunidade para crescer.

E a todos aqueles que direta ou indiretamente contribuíram para que esse sonho se tornasse uma realidade.

Muito obrigado.

Rafael

Garcia, Rafael P. Uma comparação entre ferramentas de geração de comportamentos autônomos em um ambiente virtual. 2005. 61 f. Monografia. Bacharelado em Ciência da Computação – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2005.

RESUMO

O trabalho a seguir tem como principal objetivo, a geração de comportamentos de agentes autônomos em ambientes virtuais. Define-se o conceito de agentes autônomos, as principais aplicações que utilizam esse paradigma, e estuda algumas plataformas utilizadas para uma tentativa de simulação de um mundo virtual. Nota-se uma certa dificuldade em encontrar bibliografia sobre o assunto, mas o que está relatado no presente trabalho, foi considerado o mais importante e preciso. Os resultados obtidos foram inferiores aos esperados, devido às plataformas utilizadas não estarem completamente funcionais.

Palavras-chave: agentes autônomos, mundo virtual, comportamento.

Garcia, Rafael P. A comparing between tools of creating autonomous steering behaviors in a virtual environment. 2005. 61 f. Monografia. Bacharelado em Ciência da Computação – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2005.

ABSTRACT

The work to proceed has as main objective to create steering behaviors for autonomous characters in a virtual environment. It is defined the autonomous agents concept, the main applications that uses this paradigm, and studies some platforms used for a virtual world simulation attempt. A certain difficulty was noticed in finding bibliography on the subject, but what is told in the present work, the most important and necessary considered. The achieved results were inferiors than expected, due to the utilized platforms not being fully functional.

Keywords: autonomous characters, virtual world, steering behaviors.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

Introdução

Este trabalho foi desenvolvido com o propósito de simular um ambiente virtual habitado por agentes autônomos. Para isso estudou-se o conceito de agentes, aplicações que utilizam esse recurso, e as plataformas utilizadas para a tentativa de simulação.

O principal objetivo é compreender a função dos agentes autônomos, como surgiu esse paradigma e como ele pode otimizar aplicações futuras. Considera-se útil descrever o que é um agente. Agente é um sistema computacional que se encontra em um ambiente definido, e interage com esse ambiente, assim realizando instruções e tarefas com o propósito para qual foi designado.

A evolução desse tema envolve muitas áreas de conhecimento, como a inteligência artificial, a engenharia de software e o desenvolvimento de redes. Objetos de estudo dessas áreas, os agentes provam ser uma grande inovação no contexto de programação, com o paradigma de programação orientada a agentes.

Em cada capítulo pretende-se descrever de uma maneira clara e objetiva cada um dos elementos estudados. No primeiro capítulo, há a discussão do que é um agente, suas principais características, as maneiras de serem utilizados, suas diferenças em relação a objetos e ambientes possíveis para sua implementação.

No segundo capítulo, relata-se algumas aplicações que envolvem agentes, ou que trabalham com esse recurso, descrevendo sua área de atuação e como são utilizados.

Por fim, no terceiro capítulo há uma abordagem sobre as plataformas que foram usadas nesse estudo, seus principais elementos e motivos pelos quais não se obteve sucesso na simulação. Descreve-se cada ferramenta, ressaltando seus pontos fortes e fracos.

O tema foi escolhido por haver grande interesse em saber como os agentes autônomos são criados e em criar uma base para novos estudos nessa

área, que, sem dúvida, vem se expandindo e incorpora-se a quase todos os tipos de aplicações.

Com este trabalho pretende-se definir os tipos de agentes existentes, e o que se pode esperar nas próximas gerações de software em relação a esse recurso.

CAPÍTULO 1 - AGENTES AUTÔNOMOS

1.1. Definição

O conceito, a especificação e a implementação de agentes autônomos é uma derivação direta da Inteligência Artificial cuja investigação científica teve grandes desenvolvimentos nos últimos anos. Os agentes são aplicados nas mais diversas áreas que variam desde a interação homem-máquina até complexos processos de controle industrial. Devido ao elevado número de aplicações e à relativa abertura do conceito, existem diversas definições sobre o conceito de agente e não existe um consenso entre os autores da área sobre esta matéria. No entanto, uma primeira e simples definição de agente é:

> *"Um sistema computacional que habita um dado ambiente, sente e age nesse ambiente, e ao fazê-lo realiza um conjunto de objetivos ou tarefas para o qual foi projetado".(Maes, 1996).*

O crescimento da pesquisa realizada no campo dos agentes está ligado ao fato de ser um consenso geral dos investigadores que os agentes constituem um paradigma de software apropriado para desenvolver aplicações para ambientes abertos, heterogêneos e distribuídos como, por exemplo, a Internet. Desta forma, o crescimento do número e dimensão de ambientes com estas características constitui uma forte motivação para a investigação em agentes. A adequação dos agentes a processos de resolução de problemas cuja perspectiva centralizada não demonstra ter capacidade de os resolver satisfatoriamente é outra razão para o crescimento da investigação realizada em agentes. Por outro lado, os sistemas compostos por múltiplos agentes (Sistemas Multi-Agente) constituem uma metáfora natural para perceber, simular ou construir um vasto conjunto de sociedades artificiais.

Dados os fatos, discute-se o conceito de agente. Inicialmente analisa-se a controvérsia em torno da definição de agente e as principais definições de agente propostas por investigadores da área. Também se discute os principais atributos

que um agente deve possuir tais como a autonomia, reatividade, pró-atividade, mobilidade, comunicação, cooperação e aprendizagem. Serão ainda analisados os possíveis ambientes em que os agentes se podem incluir e os domínios que inspiraram o aparecimento da investigação na área dos agentes inteligentes.

1.2. Áreas Científicas que Inspiraram os Agentes

A área da investigação designada por Agentes Autônomos surgiu inspirada nas áreas científicas da Inteligência Artificial, Engenharia de Software, Sistemas Distribuídos e Redes de Computadores, Sociologia, Teoria dos Jogos e Economia. A influência destas áreas no campo dos agentes autônomos situa-se em vários níveis.

1.2.1. Inteligência Artificial

Micro-aspectos, como a resolução de problemas, raciocínio lógico, representação e utilização de conhecimento, planejamento, aprendizagem, entre outros.

1.2.2. Engenharia de Software

O agente como uma abstração, programação orientada por agentes.

1.2.3. Sistemas Distribuídos e Redes de Computadores

Arquiteturas de agentes, Sistemas Multi-Agente, comunicação e coordenação.

1.2.4. Sociologia

Macro-aspectos como a formação de sociedades virtuais e a interação entre agentes.

1.2.5. Teoria dos Jogos e Economia

Negociação, resolução de conflitos, mecanismos de mercado.

Embora a Inteligência Artificial tenha exercido inicialmente uma influência muito forte sobre o campo dos Agentes Autônomos e Sistemas Multi-Agente, verifica-se hoje em dia que este campo já evoluiu muito, não mais podendo ser considerado uma sub-área da Inteligência Artificial.

1.3. Agentes

Ao longo dos últimos anos verificou-se uma grande explosão na investigação na área dos agentes inteligentes ou agentes autônomos. No entanto, a controvérsia que envolve esta área continua a ser elevada. Este fato deve-se essencialmente a:

- Inexistência de um paradigma de programação bem definido para sistemas distribuídos;
- O termo agente ser vulgarmente utilizado para descrever software em geral devido às definições vagas e contraditórias de que é objeto;
- O paradigma dos agentes tentar resolver o problema da ascensão do “mundo fechado” na orientação a objetos;
- Ao interesse da comunicação social no assunto que resultou na extrapolação da área científica para o público em geral, sem que o seu significado fosse corretamente explicado.

Desta forma o termo agente é por vezes utilizado para descrever coisas com um nível de complexidade muito reduzido tais como um simples termostato (ou outro instrumento de controle e/ou regulagem) ou um *daemon* de software (como um servidor de impressão ou um servidor http).

Embora não exista uma definição consensual do conceito de agente, existe a noção de que a autonomia é essencial num agente. Baseados nesta noção central de autonomia, pode-se adotar a seguinte definição de agente:

> *"Um agente é um sistema computacional, situado num dado ambiente, que tem a percepção desse ambiente através de sensores, tem capacidade de decisão, age de forma autônoma nesse ambiente através de dispositivos interativos, e possui capacidade de comunicação de alto-nível com outros agentes e/ou humanos, de forma a desempenhar uma dada função para a qual foi projetado." (Wooldridge e Jennings, 1995)*

A definição apresentada diz-nos que um agente é um sistema computacional que se encontra situado num dado ambiente. Este ambiente pode ser uma parte do mundo real (universidade, fábrica, hospital, campo de futebol, etc.), um ambiente simulado ou mesmo um computador. Os agentes mais simples são os agentes de software. No entanto, os agentes podem ter uma existência física (possuindo um corpo), designando-se nesse caso por agentes robóticos.

Independentemente do tipo de agente e ambiente, essenciais na definição de agente que adotamos, é a capacidade do mesmo se integrar ao ambiente e nele agir de forma autônoma. Desta forma, o agente deve possuir sensores e dispositivos interativos apropriados ao seu ambiente, e à execução das tarefas para as quais foi projetado.

Um humano interage com um ambiente através dos seus olhos, ouvidos, do olfato, do paladar e do tato. Age nesse ambiente utilizando os seus atuadores: braços, pernas, cordas vocais, etc. Os sensores de um agente robótico podem incluir câmeras, microfones, sensores de proximidade (incluindo, entre outros, infravermelhos e ultra-sons), sensores de tato e aceleração, etc. Usualmente os dispositivos dos robôs são braços e pernas robóticas, motores e rodas, e alto-falantes. (Figura 1.1)

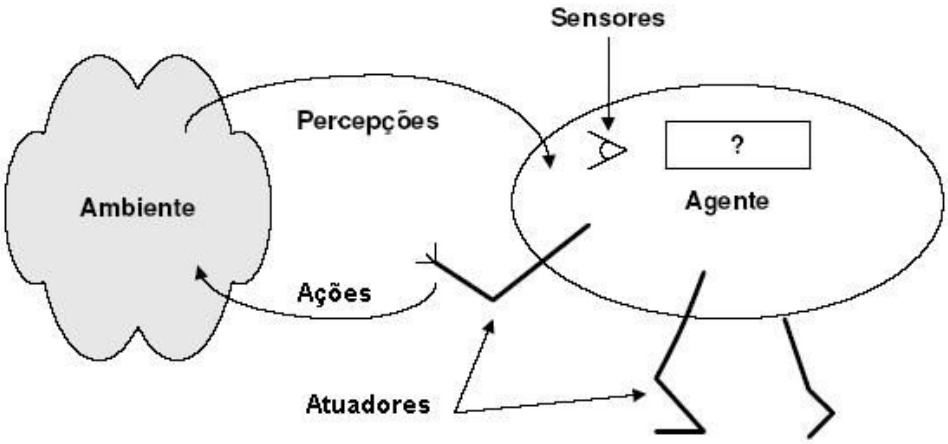

Figura 1.1: Esquema Típico de um Agente

Nos agentes de software é mais difícil definir o que são os sensores e atuadores do agente. Por exemplo, para um agente que joga uma partida de xadrez, os sensores deverão permiti-lo determinar a posição das peças no tabuleiro e os dispositivos interativos serão capazes de agir no tabuleiro realizando jogadas. A definição exata da forma de funcionamento dos sensores e dispositivos do agente pode, no entanto, apresentar diversas alternativas.

1.4. Definições de Agente

Existem diversas definições de agente que, no entanto, são normalmente muito específicas relativamente ao que definem ou, pelo contrário, extremamente gerais. Entre as mais conhecidas, encontram-se:

> *"Um agente é algo que pode ser visto como tendo a percepção de um ambiente através de sensores e agindo nesse ambiente através de atuadores" (Russel e Norvig, 1995)*

> *"Um agente é uma peça de hardware ou um sistema computacional baseado em software que goza das seguintes propriedades: autonomia, reatividade, pró-atividade e habilidade social". (Wooldridge e Jennings, 1995)*

> *"… Sistemas computacionais que vivem em ambientes complexos e dinâmicos, sentem e agem autonomamente nesse ambiente e ao fazerem-no realizam um conjunto de objetivos e tarefas para o qual foram projetados" (Maes, 1996)*
>
> *"... Um agente é uma entidade persistente de software, dedicada a um objetivo específico. A persistência distingue os agentes de sub-rotinas pois estes têm as suas próprias idéias sobre como realizar tarefas e têm as suas próprias agendas. O objetivo específico distingue-os de aplicações multi-função; os agentes são tipicamente muito menores." (Smith et al., 1994)*
>
> *"Os agentes inteligentes desempenham continuamente três funções: percepção das condições dinâmicas do ambiente; ação para afetar as condições desse ambiente; raciocínio para interpretar as percepções, resolver problemas, realizar inferências e determinar ações."(Hayes-Roth, 1995)*
>
> *"Um agente autônomo é um sistema situado e que faz parte de um dado ambiente, que sente esse ambiente e age nele ao longo do tempo, de forma a realizar a sua própria agenda e de forma a afetar o que sentirá no futuro" (Franklin e Graesser,1996)*

Nestas definições destacam-se os fatos de os agentes serem capazes de reconhecerem seu ambiente e agirem nesse mesmo ambiente, representarem um dado utilizador ou realizarem um dado conjunto de tarefas, serem persistentes, capazes de se comunicarem utilizando uma linguagem de alto-nível, e autônomo.

Uma das definições de agente mais conhecidas e aceitas na comunidade é apresentada por Wooldridge e Jennings (1995) que definem um agente como uma peça de hardware ou (mais usual) um sistema computacional baseado em software que goza das seguintes propriedades.

1.4.1. Autonomia

Os agentes operam sem a intervenção direta de humanos ou outros agentes e possuem algum tipo de controle sobre as suas ações e estado interno.

1.4.2. Reatividade

Os agentes têm a percepção do seu ambiente e respondem rapidamente às alterações que nele ocorrem.

1.4.3. Pró-Atividade

Os agentes não se limitam a agir em resposta ao seu ambiente. Eles são capazes de tomar a iniciativa e exibir comportamento direcionado por objetivos.

1.4.4. Habilidade Social

Os agentes são capazes de interagir com outros agentes (e possivelmente com humanos) através de uma dada linguagem de comunicação de agentes.

Estas propriedades são mais complexas de obter do que pode parecer à primeira vista. A autonomia, embora consensual na comunidade dos Agentes e essencial na nossa definição adotada de agente, nunca pode ser totalmente obtida. É evidente que o agente tem de ser criado e colocado em funcionamento originalmente por um humano (ou por um outro agente). A idéia de que o funcionamento do agente não terá um término também não é totalmente válida.

Evidentemente que, à luz da ciência atual, o agente terá um tempo de vida limitado e um final de operação. Por outro lado, embora seja essencial para poder ser considerado um agente, que o seu funcionamento seja efetuado sem operação direta de um humano, usualmente a interação com o humano é desejável. Assim, é usual a construção de agentes que embora se possam comportar de forma autônoma, são também capazes de acatar ordens ou instruções de humanos, se tal for desejável.

Construir agentes puramente reativos é muito simples, mas não é desejável. Um agente puramente reativo reagiria consecutivamente às mudanças no ambiente sem procurar atingir os seus objetivos de médio ou longo prazo, isto

é, de exibir comportamento orientado por objetivos. O interessante é definir agentes capazes de balancear o comportamento reativo com o comportamento pró-ativo. No entanto, a dificuldade em balancear estes dois tipos de comportamento é elevada.

A pró-atividade, isto é, a obtenção de comportamentos orientados por objetivos, é simples de conseguir em sistemas funcionais. No entanto, esta simplicidade só se verifica se for considerado que o ambiente é estático, isto é, não muda durante a execução de um dado procedimento ou função. Além disso, o agente tem que dispor de toda a informação que necessita para executar esse procedimento ou função, sem incerteza no ambiente. Para ambientes dinâmicos e não totalmente acessíveis, os agentes têm de ser capazes de reagir às mudanças no ambiente e raciocinar se os objetivos originais ainda são válidos, em face das mudanças no ambiente enquanto executam um dado procedimento. Isto significa que os agentes têm de ser reativos e conseqüentemente capazes de reagir rapidamente às mudanças no ambiente.

A capacidade social de um agente está relacionada com a sua capacidade de trocar mensagens de alto-nível (e não só bytes de dados sem um significado associado) e efetuar processos de interação social com outros agentes (e/ou humanos) semelhantes aos utilizados pelos humanos no seu dia a dia. Estes processos incluem a coordenação, cooperação, competição e negociação. De forma a efetuá-los, é necessário raciocinar acerca dos objetivos dos outros agentes (eventualmente humanos) presentes no ambiente ou, pelo menos ter a noção da sua existência. É também necessário compreender que eles são, também, agentes autônomos e que não partilham necessariamente dos nossos objetivos. Desta forma pode ser necessário negociar e cooperar com os outros agentes eventualmente trocando informação. Por exemplo, de forma a convencer um agente a ser cooperativo pode ser necessário efetuar um pagamento ou oferecer um determinado bem ou serviço. Podem, inclusivamente, existir agentes no ambiente com objetivos totalmente opostos e não ser possível, conseqüentemente, efetuar qualquer processo cooperativo que os inclua.

O balanceamento da capacidade social com as capacidades pró-ativas e reativas é também de enorme importância. Esta importância é ainda maior no caso da realização de trabalho cooperativo por um conjunto de agentes que partilham de um objetivo comum. Neste caso, cada agente necessita de balancear a reação a eventos no ambiente com a decisão individual necessária para desempenhar tarefas individuais e com o comportamento social necessário para desempenhar tarefas coletivas. Um exemplo típico consiste no futebol robótico em que os agentes de um time têm como objetivo marcar gols de um lado do campo e evitar que sejam marcados gols do outro lado. Os agentes adversários têm exatamente os objetivos opostos. Desta forma é impossível qualquer tipo de atividade cooperativa com esses agentes sendo necessário definir formas de competição entre eles.

Além das propriedades essenciais mencionadas, Wooldridge e Jennings (1995) definem ainda uma noção forte de agente, que deriva essencialmente da área da Inteligência Artificial, e envereda por uma visão antropomórfica, onde um agente é visto como uma entidade cognitiva e com consciência, que é capaz de exibir sentimentos, percepções e emoções, à semelhança dos humanos. Assim, nesta noção, as seguintes propriedades podem também ser desejáveis nos agentes (Wooldridge e Jennings, 1995):

1.4.5. Mobilidade

Capacidade de um agente se movimentar de um local para outro. Usualmente esta capacidade é mencionada no contexto de agentes de software e como tal a movimentação verifica-se no interior de uma rede de computadores.

1.4.6. Verdade

Um agente deve sempre ser verdadeiro e não comunicar informação falsa propositadamente.

1.4.7. Benevolência

Os agentes não devem assumir um comportamento contra-produtivo e devem sempre tentar fazer aquilo que lhes é solicitado.

1.4.8. Conhecimento e Crença

Possuir conhecimento consiste em possuir uma coleção de informação dinâmica, assim como, possuir também capacidade de raciocínio sobre essa informação. É necessário definir qual a melhor estratégia de raciocínio a aplicar numa determinada situação, e porquê (meta-conhecimento).

Uma crença representa a noção atual que o agente possui sobre determinado fato. As crenças são geralmente dinâmicas, isto é, podem alterar o seu valor de verdade com o tempo.

1.4.9. Intenções e Obrigações

Intenções são objetivos de longo prazo do agente. Resultam em padrões de comportamento que levam à execução de um determinado conjunto de ações individuais. As obrigações estão relacionadas com compromissos que o agente assumiu anteriormente. A partir do momento em que o agente expressou a sua disponibilidade para executar determinada tarefa, é responsável por realizar as ações necessárias para essa execução.

1.4.10. Racionalidade

Um agente agirá de forma a atingir os seus objetivos e não agirá de forma a impedir que esses mesmos objetivos sejam atingidos. Em cada instante, em face do seu conhecimento e de acordo com as suas capacidades, tentará executar a melhor ação para cumprir esses objetivos.

Usualmente, além das propriedades apresentadas, são também referidas como propriedades comuns dos agentes:

1.4.11. Inteligência

O estado de um agente é formalizado por conhecimento (i.e. crenças, objetivos, planos) e ele interage com outros agentes utilizando uma linguagem simbólica. Possui capacidade de raciocínio abstrato e de resolução de novos problemas e adaptação a novas situações.

1.4.12. Continuidade Temporal

O agente é um processo que é executado continuamente ao longo do tempo. Usualmente são utilizados também os termos: persistente ou “com uma vida longa” para designar a continuidade temporal dos agentes.

1.4.13. Caráter

O agente possui uma personalidade credível e eventualmente possui também um “estado emocional”.

1.4.14. Capacidade de Aprendizagem

Capacidades de aprendizagem que fazem com que o agente adquira novo conhecimento e altere o seu comportamento baseado na sua experiência prévia.

Nota-se que existem diversas definições do conceito de agente nem sempre concordantes. A autonomia é consensual como uma capacidade necessária para um agente. A reatividade, pró-atividade e exibição de comportamento social são também importantes e consideradas na maioria das definições de agente. Outras capacidades como a inteligência e a racionalidade,

embora importantes, estão já um pouco contidas nas anteriores. A aprendizagem, embora importante em alguns sistemas, pode não ser desejável em outros.

Exemplos disto incluem sistemas em que embora a tarefa a desempenhar seja complexa, é crítico o agente não falhar. Como tal, não é particularmente útil aprender com as suas falhas, pois supomos que ele não falharia o que resultaria em conseqüências indesejadas. Por exemplo, um agente criado para efetuar a aterrissagem de um avião. Neste caso, pretende-se que o agente seja capaz de realizar a sua tarefa de forma autônoma e inteligente, mas pode ser argumentado que não será desejável que ele possua aprendizagem, pois interessa ser capaz de prever as ações que o agente irá executar na realização desta tarefa. Pelo que se conclui que será bastante mais difícil aos humanos confiarem no agente se este efetuar ações imprevisíveis.

1.5. Atributos dos Agentes

Foi discutido anteriormente que um agente pode ser definido a partir de um conjunto de características básicas. O conjunto destas características pode ainda ser utilizado como forma de agrupar os agentes em classes ou tipologias. Deve ser referido que um agente não precisa possuir todas estas características ou atributos, embora as suas capacidades estejam diretamente associadas à presença delas. A escolha de quais os atributos que devem estar presentes num dado agente depende da funcionalidade que o projetista pretende dar a esse mesmo agente.

1.5.1. Autonomia

A autonomia é provavelmente a característica mais consensual na definição do conceito de agente pelos investigadores da área dos agentes inteligentes. Refere-se ao princípio de que os agentes podem agir baseados nas suas próprias regras de decisão, sem existir a necessidade de serem guiados por humanos. Os agentes possuem estados e metas internas, agindo de forma a

atingir estas metas a favor dos seus utilizadores. Alguns autores defendem que a autonomia aumenta com o aumento da pró-atividade, i.e. da capacidade dos agentes agirem por iniciativa própria, sem a necessidade de agirem em virtude de mudanças no ambiente ou a pedido de um humano ou outro agente. Huhns e Singh (1997) classificam a autonomia dos agentes em cinco tipos distintos:

1.5.1.1. Autonomia Absoluta

O agente possui controle completo sobre as suas percepções, raciocínio e ações, e é muito pouco previsível.

1.5.1.2. Autonomia Social

Um agente conhece os outros agentes presentes no sistema e é sociável, exercendo a sua autonomia em certas circunstâncias. A autonomia tem uma ligação natural com a coordenação ou com outras noções de alto nível tais como compromissos.

1.5.1.3. Autonomia de Interface

Em grande parte dos sistemas, onde a autonomia absoluta é impossível, a autonomia máxima possível para o agente é a autonomia respeitante à sua forma de interface com o exterior.

1.5.1.4. Autonomia de Execução

Liberdade que o agente possui na execução de ações no ambiente.

1.5.1.5. Autonomia de Projeto

Grau de autonomia dos projetistas do agente na sua construção. Quanto maior for a autonomia de projeto, maior poderá ser a heterogeneidade dos agentes.

Os agentes detentores de um elevado grau de autonomia podem manter a sua agenda independente das agendas dos seus utilizadores. Para tal, os agentes apresentam três requisitos básicos: capacidade de execução de ações periódicas, execução espontânea e iniciativa própria. Estes requisitos habilitam o agente autônomo a realizar ações preemptivas, independentes, que não são canceladas por qualquer evento exterior, e que poderão eventualmente beneficiar o seu utilizador.

1.5.2. Mobilidade

A mobilidade é a capacidade de um agente se poder movimentar de um local para outro. No caso dos agentes de software refere-se à capacidade de estes se poderem movimentar através de uma rede de computadores. Este atributo é particularmente interessante para agentes que auxiliam os seus utilizadores na pesquisa de informações, principalmente na Internet. No entanto, este atributo pode causar sérios problemas de sobrecarga na rede, uma vez que implica a necessidade dos agentes transitarem entre diferentes máquinas através dessa mesma rede. Dependendo do balanceamento entre a dimensão do código do agente e a dimensão dos dados que este troca com o exterior, a mobilidade pode até ser vantajosa em termos de sobrecarga da rede. Suponha-se, por exemplo, o caso em que o agente é de dimensão reduzida (em termos do seu código) e a sua função consiste em realizar pesquisas de informação que implicam uma análise de largas quantidades de informação até à seleção da informação mais adequada. Neste caso, seria desejável o agente movimentar-se para junto da fonte dos dados, de forma a que, durante a pesquisa, os dados

atravessassem uma reduzida parcela da rede. No regresso, somente necessitaria de trazer o seu código fonte e os dados selecionados, diminuindo claramente o tráfego total na rede.

Um agente móvel pode conter defeitos sérios no seu código ou mesmo transportar um vírus. Sendo assim, a implantação de agentes móveis deve ser acompanhada de processos de autorização e validação, além da garantia de que a memória e os recursos da máquina estarão sempre protegidos. Isto implica o desenvolvimento de plataformas especiais para agentes móveis que permitam a navegação controlada destes agentes na rede.

A mobilidade pode também ser considerada numa outra perspectiva, mais abrangente, no caso dos agentes físicos (agentes que possuem um corpo). Neste caso, a mobilidade refere-se à capacidade do agente físico (robô) se deslocar fisicamente no seu ambiente, movimentando-se de uma localização para outra. Esta mobilidade física coloca um vasto conjunto de questões e desafios de investigação, incluindo: integração sensorial, percepção ativa, determinação de trajetórias, navegação robusta, metodologias de localização própria e mapeamento do ambiente.

1.5.3. Reatividade

A reatividade é a capacidade de um agente reagir rapidamente a mudanças no seu ambiente. Para tal, o agente deve ser capaz de se aperceber do seu ambiente e atuar sobre ele. Este atributo está também presente direta ou indiretamente em quase todas as definições de agente (Wooldridge e Jennings, 1994), (Russel e Norvig, 1995), (Franklin e Graesser, 1996).

O termo reatividade é, no entanto, utilizado na literatura com pelo menos dois significados distintos. O primeiro refere-se à capacidade de um agente reagir às mudanças no seu ambiente rapidamente. O segundo refere-se à capacidade dos agentes decidirem as suas ações sem consultarem um modelo interno do mundo. Wooldridge e Jennings (1995) usam o primeiro significado de reatividade e referem agentes que reconhecem seu ambiente e respondem de forma rápida às

mudanças que ocorrem nesse ambiente. Russel e Norvig (1995), por oposição, referem-se a reatividade quando os agentes utilizam planejamento reativo e regras de condição e ação para definir o seu comportamento.

Embora a reatividade (no sentido de reação às mudanças do ambiente) seja desejável, um agente puramente reativo, reage consecutivamente às mudanças no ambiente sem procurar atingir os seus objetivos de médio ou longo prazo. Se estes objetivos não forem triviais, ou seja, a tarefa a desempenhar pelo agente for complexa, será muito difícil a um agente reativo realizá-la. De forma a que o agente possa exibir um comportamento orientado por objetivos interessa definir metodologias que permitam balancear o comportamento reativo com as restantes capacidades do agente.

O interesse pela reatividade e pelos agentes reativos levou ao aparecimento de uma tendência de investigação designada por “escola reativa” (Ferber, 1999). Os investigadores desta “escola” afirmam que não é necessário que os agentes individuais sejam deliberativos e possuam inteligência para que um Sistema Multi-Agente demonstre inteligência como um todo (Ferber, 1999).

Exemplos de sociedades biológicas tais como colônias de formigas foram utilizados de forma a provar que agentes reativos muito simples podem, em conjunto, comportarem-se de forma inteligente, resolvendo tarefas que podem ser classificadas como complexas.

1.5.4. Pró-Atividade

Este atributo pode ser também denominado de capacidade de iniciativa, uma vez que representa um comportamento independente. As ações são selecionadas de acordo com os objetivos gerais do agente e não simplesmente devido a mudanças que ocorrem no ambiente do agente. A pró-atividade é essencial para que os agentes possam exibir comportamento orientado por objetivos, os quais podem ser gerados internamente.

1.5.5. Comunicação

Quando existe mais do que um agente presente no ambiente, há uma necessidade óbvia de dispor de capacidade de comunicação entre estes. Entretanto, o conceito de capacidade de comunicação não se restringe apenas à capacidade de troca de informações entre agentes. Franklin e Graesser (1996) afirmam que agentes podem comunicar com outras entidades além dos agentes, incluindo humanos e o seu próprio ambiente.

De forma a dispor de capacidade de comunicação com o exterior, os agentes necessitam de dispositivos de interação apropriados ao envio dessas mensagens (placa de rede, colunas de som, etc.) e conhecimento que lhes permita utilizar protocolos de transporte dessas mensagens. Por outro lado, e a um nível de análise mais elevado, os agentes necessitam de uma linguagem que possa também ser entendida pelos outros agentes presentes no ambiente. Precisam ainda de partilhar a semântica do que é veiculado. A disponibilidade de atuadores apropriados, protocolos e linguagem para efetuar a comunicação é essencial para dispor de outras capacidades mais elaboradas, construídas com base na capacidade de comunicação, como sejam a habilidade social e a capacidade de cooperação.

1.5.6. Habilidade Social

A habilidade social está diretamente associada com a característica de capacidade de comunicação, uma vez que representa a capacidade de interagir com outros agentes. Genesereth e Ketchpel (1994) afirmam que um agente tem de ser capaz de comunicar numa linguagem de alto nível, e indicam a necessidade da existência de uma linguagem comum para a comunicação de agentes, de forma a que estes possam trocar informação. No entanto, embora a utilidade da existência desta linguagem seja reconhecida por todos os investigadores da área, o problema é definir qual será a linguagem ideal.

Atualmente, não existe uma linguagem padrão aceita mundialmente, para a comunicação entre agentes. A habilidade social implica ainda mais do que a existência de uma linguagem comum. Os agentes devem ser capazes de compartilhar a semântica das mensagens e utilizar o mesmo vocabulário de acordo com a aplicação em causa.

1.5.7. Cooperação

Cooperação pode ser entendida como a capacidade que os agentes têm de trabalhar em conjunto de forma a concluírem tarefas de interesse comum. Acredita-se que a cooperação entre agentes é fundamental, sendo a razão principal para a existência de um ambiente multi-agente. Tal como os humanos, os agentes têm que combinar os seus esforços de forma a atingir objetivos comuns que não podem ser atingidos individualmente.

Para permitir esta cooperação, o agente deve ser dotado de uma certa habilidade social, capacitando-o a interagir com outros agentes e possivelmente humanos, através de alguma linguagem de comunicação (Wooldridge e Jennings, 1994). Deve ainda dispor de metodologias apropriadas para realizar essa cooperação, de acordo com a tarefa cooperativa que irá ser executada pelo conjunto de agentes. As estratégias propostas na literatura para efetuar a cooperação entre agentes incluem estratégias fixas, aprendizagem de trabalho em equipe, organizações hierárquicas, normas e leis sociais, etc.

Usualmente a comunicação é vital para a realização da cooperação, embora em alguns casos seja possível a cooperação sem comunicação. Esta comunicação é usualmente realizada a um nível abstrato e elevado, utilizando linguagens e protocolos apropriados para a comunicação inter-agente.

1.5.8. Aprendizagem

Uma característica que é freqüentemente confundida com a inteligência é a capacidade de aprender. Um agente só pode possuir uma autonomia completa quando possuir a capacidade de avaliar as variações de seu ambiente externo, e propor a ação mais correta. No mundo real, o ambiente muda e não se pode esperar que, se o agente possuir continuidade temporal, o ambiente que ele vá encontrar dentro de alguns anos seja semelhante ao ambiente para o qual foi inicialmente projetado. Acresce a isso o fato de que será muito difícil ao projetista prever, na fase de projeto do agente, a variação futura do ambiente onde este irá operar. Assim, mesmo quando um agente não reconhece a existência de uma ação com elevada probabilidade de sucesso para ser executada, é esperado que ele procure, encontre e experimente alternativas. A questão não é acertar sempre, mas aprender continuamente por experiência, seja através de sucessos ou de fracassos. No entanto, a inclusão desta capacidade de aprendizagem não é trivial em agentes.

Por vezes é difícil identificar a falha ou sucesso da ação e mais complexo ainda identificar o motivo que levou a essa falha ou sucesso. Além disso, mesmo após uma correta identificação dos motivos de uma dada falha, muitas vezes é também difícil utilizar esta informação de forma a encontrar planos alternativos com maior probabilidade de sucesso.

A aprendizagem pode ser também um processo interativo em que um treinador ensina um dado agente (ou conjunto de agentes) a executar uma determinada tarefa individual ou cooperativa. Nestes casos, o treinador pode fornecer ao agente o conhecimento, através de uma seqüência de instruções ou informá-lo apenas quando o agente não possuir o conhecimento necessário para executar a sua. O treinador pode ainda possuir uma visão mais geral sobre a tarefa, e capacidade de percepção e conhecimento específico que lhe permita detectar situações de falha ou sucesso com maior facilidade do que o agente que executa a tarefa. No caso da aprendizagem multi-agente, os agentes aprendem a realizar uma tarefa que envolve mais do que um agente na sua execução. Neste

caso, a figura do treinador é novamente bastante importante, pois uma visão global da situação permite detectar falhas e construir planos de execução da tarefa com maior facilidade.

1.6. Agentes vs. Objetos

Um dos grandes desafios que se coloca à comunidade de investigadores em Agentes Inteligentes consiste em estabelecer claramente a distinção entre os conceitos de Agente e Objeto, mostrando a inovação que existe por trás do conceito de Agente (Wooldridge, 2002).

O conceito de Objeto para a comunidade de Programação Orientada por Objetos consiste numa entidade computacional que encapsula um estado e é capaz de executar um conjunto de ações (métodos) sobre este estado e comunicar através de mensagens.

Embora as semelhanças entre os conceitos sejam evidentes, as diferenças são também visíveis: os Agentes possuem capacidade de decisão autônoma, comportamento reativo, pró-ativo e social, fluxo de controle próprio e distinto dos restantes agentes que compõem um dado Sistema Multi-Agente.

Embora os objetos possuam autonomia sobre o seu estado (dados) não possuem autonomia sobre o seu comportamento, ou seja, se disponibilizarem um determinado método, sempre que outro objeto o invoque este estará disponível. Pelo contrário, os agentes possuem controle sobre o seu comportamento e, como tal, um outro agente terá de solicitar ao agente que execute uma dada ação. Este pode decidir, em cada situação concreta, efetuar ou não a ação solicitada. No modelo orientado a objetos a decisão é efetuada pelo objeto que solicita a execução do método.

Relativamente à pró-atividade, reatividade e comportamento social, o modelo orientado a objetos não contém qualquer referência a estes tipos de comportamentos. A distinção relativa ao fluxo de controle próprio baseia-se essencialmente no fluxo de controle único do modelo orientado a objetos padrão. Neste modelo o fluxo único de controle invalida a noção de entidade autônoma.

No entanto, os recentes objetos ativos diminuem esta distinção. Os objetos ativos são supostos terem o seu fluxo de controle próprio sem necessariamente terem a capacidade de exibir um comportamento autônomo.

Embora existam distinções importantes entre agentes e objetos, é claro que o modelo orientado a objetos é bastante adequado à implementação de agentes. Desta forma, linguagens como Java (Deitel e Deitel, 1999), o C++ (Bronson, 1997) e suas extensões, encontram-se entre as mais utilizadas na implementação de agentes autônomos.

Numa extensão a este raciocínio os agentes podem ser vistos como um novo paradigma de programação: a Programação Orientada por Agentes. Ao longo da história da Engenharia de Software as novas metodologias têm vindo a surgir no sentido de incrementar a localização e encapsulamento das unidades básicas de programação. O surgimento do paradigma da Programação Orientada a Agentes confirma esta tendência. (Tabela 1.1)

	Programação Monolítica	***Programação Estruturada***	***Programação Orientada a Objetos***	***Programação Orientada a Agentes***
Comportamento das Unidades	Externa	Local	Local	Local
Estado das Unidades	Externa	Externa	Local	Local
Invocação das Unidades	Externa	Externa (chamada)	Externa (mensagem)	Local (regras e objetivos)

Tabela 1.1: Evolução dos Paradigmas de Programação.

A tendência verificada na evolução da engenharia de software (tabela 1) vai ao sentido de um cada vez maior encapsulamento. As unidades de programação têm-se tornado cada vez mais autônomas e a programação orientada por agentes incrementa ainda mais esta autonomia dando às unidades de programação, agora designadas por agentes, controle sobre o seu próprio comportamento.

1.7. Ambientes

Um agente é um sistema computacional que “vive” num determinado ambiente, reconhecendo-se e agindo nesse ambiente. As características do ambiente em que o agente “habita” são determinantes na definição da arquitetura do agente e da sua forma de operação. Antes de projetar um agente é necessário uma análise cuidada das percepções e ações possíveis e da complexidade da tarefa a executar pelo agente. Neste contexto a distinção entre ambientes reais e simulados não implica que um ambiente real seja mais complexo do que um ambiente simulado. Por vezes, agentes de software construídos para ambientes simulados possuem um conjunto possível de percepções e ações mais rico do que agentes robóticos construídos para operar em determinado ambiente real.

Seguindo uma classificação baseada no trabalho de Russel e Norvig (1995), pode-se classificar as propriedades dos ambientes segundo as seguintes classes:

1.7.1. Acessível vs. Inacessível

Um ambiente acessível é aquele onde um agente consegue obter, através dos seus sensores, informação atualizada, precisa e completa sobre o ambiente. Grande parte dos ambientes típicos não são acessíveis neste sentido. Entre os ambientes inacessíveis destacam-se a Internet e todos os ambientes físicos reais.

1.7.2. Determinístico vs. Não determinístico

Num ambiente determinístico cada ação tem um efeito único garantido, não existindo qualquer incerteza quanto ao resultado da sua execução.

1.7.3. Estático vs. Dinâmico

Um ambiente estático é suposto permanecer inalterado enquanto o agente decide a próxima ação a executar. Em contraste, num ambiente dinâmico, outros agentes encontram-se a agir no ambiente ao mesmo tempo. Todos os ambientes físicos do mundo real e a Internet são dinâmicos neste sentido.

1.7.4. Discreto vs. Contínuo

Um ambiente diz-se discreto se existe um número finito de percepções e ações possíveis para o agente e contínuo caso contrário. Um ambiente pode ser contínuo no que diz respeito às percepções do agente e discreto no que diz respeito às ações.

Quanto mais acessível um ambiente for, mais simples será projetar e construir agentes para nele operarem. A qualidade das decisões efetuadas por um agente está obviamente ligada à quantidade e qualidade da informação que este possui, sendo desta forma muito mais simples tomar as decisões corretas em ambientes acessíveis. Também é importante dizer que a escala de acessibilidade não pode ser considerada uma escala discreta (acessível / não acessível). Em ambientes não acessíveis é importante considerar o grau de acessibilidade do ambiente ao agente. Por exemplo, para um agente robótico colocado num ambiente físico, será muito mais simples tomar decisões corretas se possuir câmaras, sensores de infravermelhos e outros equipamentos sensoriais avançados, de forma a ter uma razoável acessibilidade ao estado do seu mundo. Neste caso, é também importante considerar a precisão da informação recolhida pelos sensores do agente.

Os sensores do agente podem ter acesso à grande maioria da informação necessária para a operação desse agente, mas não serem capazes de obter essa informação de forma precisa. A existência de erros na percepção do agente torna as suas tarefas bem mais complexas de serem executadas.

O não determinismo de um ambiente é uma grande fonte de complexidade para o mesmo, impondo sérias restrições à capacidade de um agente para efetuar previsões quanto ao estado futuro do mundo. Esta característica ressalta dados importantes do mundo real, como a possibilidade de as ações executadas por um dado agente falharem ou não terem um efeito único, bem determinado e garantido. Desta forma é muito mais complexo construir agentes para ambientes não determinísticos, pois é necessário providenciar capacidades de recuperação e tolerância à falhas. No entanto, mesmo ambientes determinísticos tais como os jogos de tabuleiro podem ser complexos.

O dinamismo de um ambiente é uma das fontes mais significativas de complexidade.Grande parte da investigação realizada no início da inteligência artificial, principalmente a realizada no âmbito do desenvolvimento de algoritmos de planejamento, estava preocupada em determinar a seqüência de ações que permitia a um dado agente, partindo de um estado inicial especificado, atingir um dado estado objetivo também especificado. O desenvolvimento deste tipo de algoritmos assumia implicitamente que o ambiente era estático, ou seja, que não seria alterado por qualquer outro agente no fluxo de execução do plano (Wooldridge 2002). No entanto, os ambientes físicos do mundo real e grande parte dos ambientes computacionais não são estáticos e este tipo de planejamento rígido não pode neles ser aplicado.

Em ambientes dinâmicos, os agentes não podem assumir que se não executarem alguma ação durante um dado intervalo de tempo, o ambiente vai permanecer inalterado. Soma-se a isto, o fato de que os agentes têm inclusivamente de entrar em consideração com o tempo de raciocínio na decisão das ações a executar. O efeito de uma dada ação no instante t(1) poderá ser totalmente diferente do efeito da mesma ação no instante t(2).

O dinamismo implica também que os agentes têm que executar ações apropriadas para recolherem informação do ambiente. Neste contexto é importante o agente possuir sensores flexíveis. Outra característica dos ambientes dinâmicos é a presença de outros agentes que agem no ambiente e que podem interferir nas ações efetuadas pelo agente. Aqui, o agente deixa de estar isolado e

surge a necessidade de se coordenar, comunicar, cooperar, competir ou negociar com estes agentes adicionais.

Um ambiente estático é muito mais simples, pois o agente pode recolher informação sobre uma dada característica do ambiente unicamente uma vez e assumir que este ambiente não será modificado, a não ser pelas suas próprias ações. Nos ambientes estáticos, os agentes não têm também necessidade de se sincronizarem ou coordenarem as suas ações com as de outros agentes no ambiente nem de interagirem de qualquer outra forma com outros agentes.

A última característica dos ambientes diz respeito a estes serem discretos ou contínuos. Um ambiente discreto é aquele que só pode assumir um conjunto predefinido de estados discretos. Por oposição, num ambiente contínuo é impossível enumerar todos os estados possíveis. Desta forma, o Xadrez e outros jogos de tabuleiro, constituem ambientes discretos, pois existe um número finito (embora muito grande) de estados possíveis do necessário para o jogar.

Ambientes físicos do mundo real são por natureza intrinsecamente contínuos. A continuidade de um ambiente pode-se verificar a três níveis:

1.7.5. Estado do Mundo (Ambiente)

O estado do mundo pode ser discreto ou contínuo. Por exemplo, no caso de um ambiente para um jogo de tabuleiro, como o Xadrez ou as Damas, o ambiente será discreto. Em diversos ambientes simulados, o estado do mundo é também discreto, no sentido em que é representado por variáveis discretas e existe um número finito de estados possíveis para esse ambiente.

1.7.6. Percepções do Agente

Embora um dado ambiente possa ser contínuo, as percepções de um agente inserido nesse ambiente podem ser discretas. Este é o caso, por exemplo, de um robô com sensores de proximidade que se encontra num ambiente físico.

1.7.7. Ações do Agente

Tal como no caso das percepções, as capacidades de ação de um agente que se encontra num dado ambiente contínuo podem ser unicamente discretas, i.e. o agente dispor de um conjunto limitado de ações.

Os ambientes discretos são bastante mais simples para os agentes por diversos motivos. O motivo principal está diretamente relacionado com o fato dos sistemas computacionais serem por natureza discretos e, como tal, para tratarem ambientes contínuos, terem que efetuar algum tipo de discretização e aproximação. É este o caso, por exemplo, na percepção através de uma câmara, em que é necessário efetuar a digitalização da imagem previamente ao seu processamento através do sistema computacional. Embora a precisão na aproximação digital possa ser elevada, em qualquer tipo de digitalização, existe sempre informação que é perdida ou impossível de ser tratada em tempo útil. Desta forma, num ambiente contínuo os agentes irão sempre raciocinar com informação aproximada.

Analisando as propriedades dos ambientes é fácil concluir que os ambientes mais complexos para os agentes são os ambientes inacessíveis, não determinísticos, dinâmicos e contínuos. Ambientes com estas propriedades são designados ambientes abertos. Embora estes ambientes sejam os mais complexos, verifica-se que os ambientes físicos do mundo real são intrinsecamente abertos. Como tal, e numa perspectiva de futuro desenvolvimento de agentes para operarem no mundo real, interessa ser capaz de definir metodologias que permitam o desenvolvimento de agentes para operarem em domínios que têm a maioria ou mesmo todas as características dos ambientes abertos.

CAPÍTULO 2 – APLICAÇÕES DE AGENTES AUTÔNOMOS

Para entender melhor o conceito de agentes autônomos, este capítulo tratará de propostas e aplicações desse recurso, analisando suas principais características, e, demonstrando como os agentes podem facilitar e aperfeiçoar as mais variadas tarefas.

Em muitas aplicações industriais, uma quantidade substancial de tempo, esforços e dinheiro são gastos no desenvolvimento de sistemas computacionais complexos e sofisticados. Estes sistemas são sempre vistos de uma maneira isolada, como ilhas de automação, quando na realidade, eles deveriam ser vistos como componentes de uma grande função. O maior benefício dessa perspectiva é que subsistemas podem ser integrados em uma comunidade coerente, na qual, eles trabalham juntos para alcançar os objetivos da aplicação como um todo. Pelo fato de estarem integrados, o orçamento disponível para o desenvolvimento de sistemas de informação pode ser estipulado mais adiante - os agentes podem compartilhar uma versão de dados consistente e atualizada, funcionalidades básicas e objetivos.

Dois componentes são necessários para planejar uma comunidade integrada bem estruturada: uma estrutura que forneça assistência para a interação entre os sub-componentes constituintes e a metodologia, a qual fornece uma forma de estruturação para estas interações.

2.1. ARCHON

O ARCHON é o maior projeto na área de Inteligência Artificial Distribuída da Europa. Ele foi planejado para uma arquitetura de propósito geral, e sua metodologia tem sido usada para suportar o desenvolvimento de sistemas de Inteligência Artificial Distribuída nos mais diferentes domínios da indústria. Alguns exemplos de aplicações para as quais ele foi satisfatoriamente aplicado incluem: fornecimento e distribuição de eletricidade; transmissão e distribuição de eletricidade, controle de forno de cimento, controle de aceleradores, controle de

robôs. O tipo de cooperação comunitária utilizado tem um regime de controle descentralizado e agentes para solução de problemas individuais, os quais são semi autônomos.

As entidades responsáveis para a solução de problemas individuais no ARCHON são os agentes. Estes têm a habilidade de controlar suas próprias resoluções de problemas e de interagir com outros membros da comunidade. As interações tipicamente envolvem comunicação e cooperação entre agentes a fim de engrandecer suas soluções de problemas individuais e para melhorar a solução de todas as aplicações do problema. Cada agente consiste de um ARCHON Layer (AL) e um aplicativo conhecido como sistema inteligente. Sistemas inteligentes com fins de construção podem fazer uso das funcionalidades do ARCHON para melhorar seus métodos de resolução de problemas. Entretanto, sistemas inteligentes já existentes podem ser incorporados, como pouca adaptação, e podem ser experimentar benefícios similares. Este último ponto é muito importante porque em muitos casos o desenvolvimento de uma aplicação inteira pode ser considerado muito caro.

2.1.1. Utilização do ARCHON

O gerenciamento de energia é um processo de monitoração e controle do ciclo de geração, transporte e distribuição de energia elétrica para consumidores industriais e domésticos. Para minimizar as perdas durante o transporte de energia, a voltagem é alta (132 kV ou mais), antes de ser colocada em uma rede de transporte. Finalmente, a voltagem é diminuída e a eletricidade é enviada aos consumidores usando uma rede de distribuição. Para assegurar que a rede de transporte permanência dentro de nível de segurança, ela é equipada com um sofisticado sistema de aquisição de dados e muitos aplicativos convencionais, os quais ajudam o operador a analisar todo o sistema. Esta operação de rede é monitorada a partir de um DCR e quando um evento inesperado ocorre, centenas de alarmes são disparados automaticamente através do sistema. Sobre estas circunstâncias, o operador tem que contar com o seu conhecimento para analisar

a informação, diagnosticar a situação e tomar a atitude correta, a fim de recolocar a rede em um estado de segurança. Para reduzir a carga cognitiva do operador em tais circunstâncias, e para ajudá-lo a tomar a melhor decisão rapidamente, foi decidido que vários sistemas de suporte a decisão deveriam ser desenvolvidos. Estes sistemas eram então, interconectados e subseqüentemente, expandidos usando a tecnologia ARCHON.

2.1.2. Especificação dos Agentes

Em condições normais de trabalho, o gerenciamento da rede pelo operador no DCR consiste principalmente em mudanças de topologias, geração de escalonamento e controle do intercâmbio de energia com outros utilitários. No entanto, durante situações de emergência, o gerenciamento começa a ficar consideravelmente mais difícil, por causa do número de restrições que precisam ser consideradas e na qualidade insuficiente da informação disponível para a tomada de decisão. Situações de emergência originam-se tipicamente de curtos circuitos em uma linha, em um bus bar, ou em um transformador. Eles podem ser ocasionados pelo mal funcionamento do equipamento, ou por sobrecargas subseqüentes. A situação pode piorar, se as poderosas estações começarem a se desconectar, causando uma instabilidade na capacidade da rede. Conseqüentemente, ações para restaurar o serviço têm de ser tomadas rápida e efetivamente. Só então, começa-se a diminuir relativamente os problemas e evita-se uma catástrofe. Nestas circunstâncias, as ações que o operador pode tomar, consistem basicamente de operações nos breakers, mudanças de topologia, ativação e desativação de automatismos, e reles de proteção. Para distúrbios maiores, no entanto, ações nas plantas da rede podem ser necessárias.

Desta descrição, do trabalho da engenharia de controle, uma análise top-down identifica que um sistema de suporte à decisão compreensiva pode realizar as seguintes tarefas:

- Detectar a existência de distúrbios algumas vezes a ativação de reles de proteção e breakers podem ser causados por uma rotina

de manutenção e não devem ser confundidos com uma situação de distúrbios genuínos.

- Determinar a causa, local e tipo do distúrbio, incluindo a identificação de qualquer equipamento que esteja permanentemente danificado.
- Analisar a situação da rede uma vez que ela alcança um estado regular e preparar um plano de restauração para retornar a rede ao seu estado original de operação.

Aliando está análise top-down com a perspectiva bottom-up de exame dos sistemas existentes, decidiu-se encapsular os seguintes sistemas pré-existentes como agentes sistemas especialistas de análise de alarme e sistemas de controle da interface. A disponibilidade de mensagens cronológicas de alarme necessita de um novo sistema de diagnóstico, o qual será desenvolvido como um agente. Finalmente, este estará sempre conhecendo a área inicial fora de serviço (área de black-out), para ajudar nas pesquisas de restrições de falhas nos equipamentos, no entanto, foi desconsiderado o custo efetivo para desenvolver o sistema stand-alone dedicado, já que a performance original do sistema especialista de análise de alarmes foi considerada satisfatória. No entanto, com o uso dessa tecnologia, muito da infra-estrutura básica implementada para esta funcionalidade estará agora disponível para outros agentes, e pode ser considerada economicamente viável para desenvolver um sistema capaz de produzir esta informação.

Concluindo, o sistema DAI desta aplicação consiste em cinco agentes rodando em cinco máquinas diferentes.

2.1.2.1. BAI (Identificador da área de blackout)

Identifica quais elementos da rede estão fora de serviço, sendo que o elemento na falha atual deve estar dentro desta região. Ele usa mensagens de alarme não cronológicas e trabalha em cooperação com o BRS para melhorar a eficiência do diagnóstico.

2.1.2.2. CSI (Interface do Sistema de Controle pré-existente)

Seus objetivos são adquirir e distribuir dados da rede para outros agentes, servir de interface para sistemas de gerenciamento convencionais, monitorar o processo de restauração para detectar algum desvio inesperado. Interage com o BAI, BRS e UIA.

2.1.2.3. BRS (Supervisor de Breakers e Relês)

Detecta a ocorrência de um distúrbio, determina o tipo de falha e sua expansão, gera uma lista ordenada de tipos de falhas, valida estas hipóteses e identifica o equipamento que apresenta mal funcionamento.

2.1.2.4. SRA (Agente de Restauração de Serviço)

Viabiliza um plano de restauração de serviço a fim de retornar a rede à um estado estável.

2.1.2.5. UIA (Agente de Interface do Usuário)

Implementa a interface entre usuários e a comunidade de agentes.

2.2. Aglets

Agentes Inteligentes são entidades autônomas dotadas de uma base de conhecimento e capazes de interagir com o meio em que estão, tomando assim, decisões que irão auxiliar ou até mesmo substituir o trabalho de um agente humano. Applets são basicamente conjuntos de código de programa que podem ser descarregados, instanciados e executados em WEB browsers.

O Aglet é um agente para Internet com a capacidade adicional de ser transportado pela rede. São objetos Java que podem se mover de um host ao outro, parar a execução, despachar-se para um host remoto e executar-se lá autonomamente, traçando seu próprio itinerário.

Para implementar um Aglet pode ser usada uma plataforma para programação de agentes móveis da IBM denominada Aglet Workbench. Trata-se de um ambiente para Windows destinado à programação de Aglets em Java. A linguagem utilizada, portanto, é Java por ser esta uma linguagem que permite a criação de aplicativos (applets, servlets e aglets) independentes da plataforma em que serão executados (fundamental para o propósito de um aglet), facilitando a movimentação de um sistema de um computador para outro.

2.3. Agentes Inteligentes para Simulação Interativa de Ambientes

A simulação interativa de ambientes representa uma tecnologia com extensa possibilidade de aplicação nas áreas de educação, manufatura, entretenimento e treinamento. Por sua vez, estes ambientes permitem a construção e estudo de agentes inteligentes autônomos. Esse projeto pretende desenvolver agentes inteligentes capazes de imitar seres humanos, atuando em papéis específicos dentro de um ambiente simulado. Atualmente estão concentrados no desenvolvimento de pilotos inteligentes autônomos, em batalha aérea simulada. Estes agentes são dinâmicos, interativos, e atuam em ambientes multi-agentes, proporcionando desafios interessantes no que concerne à pesquisa de capacidades específicas de agentes, bem como a integração destas capacidades na criação de agentes-pilotos "completos".

2.4. Ambiente para Assessoria e Suporte Empresarial Virtual através da Internet

O objetivo geral desse ambiente consiste na implantação de um laboratório virtual de suporte técnico e de serviços estabelecendo um canal direto

de comunicação através da Internet entre a Universidade e a iniciativa privada. Tal canal de comunicação direta visa o constante crescimento de serviços de suporte à iniciativa privada por parte da Universidade. Os seguintes serviços de assessoria via Internet, seriam disponibilizados:

- Sistema de suporte técnico e assessoria em Planejamento e Controle da Produção;
- Redução dos custos financeiros em relação à administração do capital de giro (CG);
- Consultor Inteligente sobre decisões de compra; Sistema de apoio à decisão para a Previsão de Vendas e Fluxo de Caixa; Análise Gerencial de Custos.

Neste contexto, destaca-se o uso de agentes inteligentes no módulo: Consultor Inteligente sobre Decisões de Compra.

2.4.1. Consultor Inteligente sobre Decisões de Compra

O sistema inteligente proposto oferece três serviços básicos: negociação inteligente, assessoria inteligente e sistema de consulta de consulta de informações.

O serviço de informação provê treinamento prático através de cursos e apostilhas, palestras de consultores, informações sobre ofertas regionais e consulta direta à Universidade, que desta forma terá acesso direto as principais dúvidas e reais necessidades das empresas.

A assessoria inteligente funciona como um consultor inteligente na tomada de decisão, levando em consideração os seguintes aspectos:

- Condições financeiras oferecidas pelo fornecedor (prazos, descontos, ofertas de mix, ICM, etc.);
- Níveis de estoque desejáveis pela empresa;
- Histórico do fornecedor (segundo informações do próprio sistema);
- Previsões de variação de preço;
- Regras de preferência do usuário e

- Escolha de alternativas considerando ofertas de "mix" de produtos ao invés de um único produto ou serviço.

Caso o usuário deseje, o sistema apresenta uma nova classificação das alternativas feitas através de um sistema especialista, ponderando os aspectos citados anteriormente com perguntas referentes à liquidez, disponibilidades, e grau de risco aceitável pelo usuário.

A negociação inteligente funciona como um elemento auxiliar do serviço de assessoria. Uma vez fornecidos pelo cliente certas informações de busca; através do uso de agentes inteligentes de software, é feita uma procura em diferentes bases de dados e sites. A informação resultante (previamente filtrada pelo agente) é direcionada à banco de dados para uma nova reorganização com base em um novo questionamento ao usuário, de pose da informação pesquisada.

2.4.1.1. ClickTheButton

É um serviço automático de busca de preços que roda em background no PC e é disparado por um click quando um produto em uma pagina web faz parte do conjunto de 300 sites que ele suporta. Automaticamente ele busca os melhores preços e condições de pagamento.

2.4.1.2. Active Buyer's Guide

É um serviço gratuito de recomendação que encontra o melhor produto baseado na necessidade individual do usuário. O cérebro por trás deste serviço baseia-se na Adaptative Recommendation Technology que automatiza o processo de compra. A ART é composta por três módulos principais: Módulo de Decisão, Máquina de Análise e Avaliador de Produtos.

2.4.1.3. IChoose

É um bot de compras que funciona assim: quando o usuário encontra algo que quer comprar, antes de efetuar a compra, ele dispara o agente iChoose que tentará encontrar uma ou mais alternativas.

2.4.1.4. Smartshop.com

Um serviço shopbot que permite aos consumidores: conhecer novos produtos de centenas de fabricantes e vendedores; comparar com uma vasta variedade de características baseadas na preferência individual de cada um sobre performance, qualidade, garantias; encontrar o melhor custo-benefício baseado em preço, entrega, promoções disponíveis e descontos. Isso sem que tenha que visitar cada um dos sites que disponibilizam os produtos.

2.5. Assistente de Correio Eletrônico - Maxims

O Maxims é um agente que ajuda o usuário com seu e-mail. Ele aprende a priorizar, apagar, responder, organizar e arquivar mensagens de mail recebidas pelo usuário. Maxims é implementado em Macintosh Commom Lisp. Ele comunica-se com o pacote comercial de e-mails EUDORA usando Apple Events. A principal técnica de aprendizado utilizada pelo Maxims é o raciocínio baseado em memórias. Se o usuário realiza uma ação, o agente memoriza toda a situação gerada, por exemplo, se o usuário salva uma mensagem particular após tê-la lido, o agente adiciona a descrição desta situação e a ação tomada pelo usuário em sua memória de exemplos. Quando uma nova situação ocorre, o agente compara-a com seu banco de memória e verifica qual ação tomar.

2.6. Agente de Escalonamento de Reuniões

Este agente assiste o usuário no que diz respeito ao escalonamento de reuniões (aceitar, rejeitar, negociar horários, etc.). O comportamento do usuário é repetitivo, mas muito diferente de indivíduo para indivíduo. Algumas pessoas preferem reuniões pela manhã outras à tarde, diferentes pessoas possuem diferentes critérios para classificar uma reunião como importante. O agente aprende o comportamento do usuário e seus critérios de classificação a fim de marcar as reuniões. Testes comprovam que o uso de agentes tem sido útil e satisfatório. Usuários dizem se sentir em uma situação confortável delegando tarefas para os agentes. Os testes revelam que é importante prover o agente com um amplo conjunto de características para descrever situações, quanto mais características possuir o agente, melhor será sua performance.

CAPÍTULO 3 – PLATAFORMAS UTILIZADAS

3.1. Introdução

O objetivo da simulação era o de criar agentes puramente reativos, que pudessem interagir entre si, ou com o próprio ambiente. Por isso, procuraram-se plataformas que trabalhassem com o conceito de agentes (ou veículos), e que implementassem comportamentos baseados em regras.

Para tornar esse estudo uma base para novas pesquisas, procurou-se reduzir os custos e trabalhar com softwares livres (ou *open source*), e através de busca na Internet, o primeiro projeto de destaque encontrado foi o OpenSteer.

3.2. OpenSteer

Primeira plataforma estudada para o desenvolvimento da simulação, onde pretendíamos criar agentes reativos que imitassem o comportamento humano.

O OpenSteer foi inicialmente desenvolvido por Craig Reynolds em 2002, no grupo de pesquisa e desenvolvimento da Sony Computer Entertainment America.

É uma biblioteca open source de componentes, feita para auxiliar na construção de comportamentos de direção para agentes autônomos, tanto em jogos quanto em outras aplicações multi-agente. Esses agentes podem representar personagens, veículos, ou outros tipos de agentes móveis. Essa plataforma foi primeiramente desenvolvida no sistema operacional Linux e mais tarde, foi adaptada para o Windows e Mac OS X.

O OpenSteer fornece ferramentas para geração de comportamento, definidas em um agente móvel abstrato chamado *vehicle*. Códigos exemplos são oferecidos, incluindo uma implementação simples de um veículo e exemplos de combinações de comportamentos simples para a produção de outros mais complexos.

Para permitir flexibilidade de integração com *engines* existentes, o OpenSteer pode adicionar sua funcionalidade, tanto por camadas como por herança. Além disso, a biblioteca possui uma aplicação interativa chamada OpenSteerDemo (Figura 3.1), que demonstra vários tipos de comportamentos. O OpenSteer é escrito em C++ e utiliza OpenGL em sua parte gráfica. Todas as chamadas ao OpenGL são agregadas em um módulo, caso seja necessário a substituição da API gráfica.

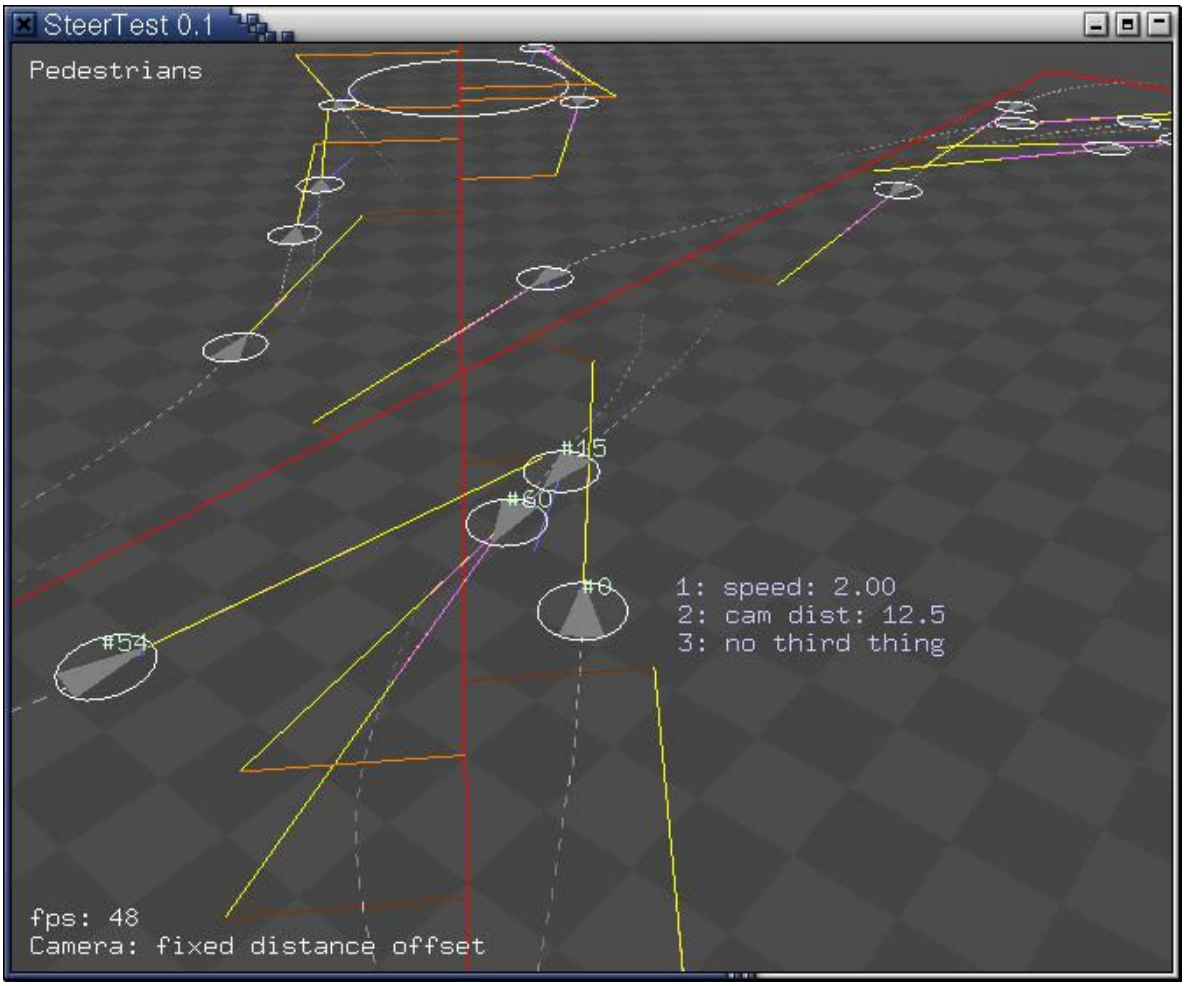

Figura 3.1: OpenSteerDemo com o esquema "Capture the Flag"

Contudo, o projeto ainda está em desenvolvimento, não está maduro ou estável, seu design atual deverá sofrer mudanças e reorganizações, porém seus desenvolvedores prometem evitar mudanças sem sentido.

O grande problema encontrado com essa plataforma foi o de não possuir suporte a modelos tridimensionais, o que dificultaria a implementação, já que para criar os avatares (inicialmente foi definida a modelagem em Java3D), e integrar os comportamentos do OpenSteer teríamos que utilizar JNI (Java Native Interface), que realiza a integração entre Java e C++. Porém o projeto não seria portável, ou seja, seria sempre preciso transportar os códigos em C++ para manter o ambiente funcionando.

3.3. Fuze3D

O Fuze3D é, na verdade, uma *engine* para jogos, feita com componentes *open source,* e é parte de um projeto maior denominado *Machinimascope*. Totalmente construído em Java nos fez acreditar que seria uma plataforma utilizável, completa e que atenderia aos requisitos da simulação de um ambiente virtual.

Implementando as classes do OpenSteer (Figura 3.2), esse projeto parecia muito promissor, já que possuía a proposta de criação de jogos e aplicações similares, sem nenhum custo.

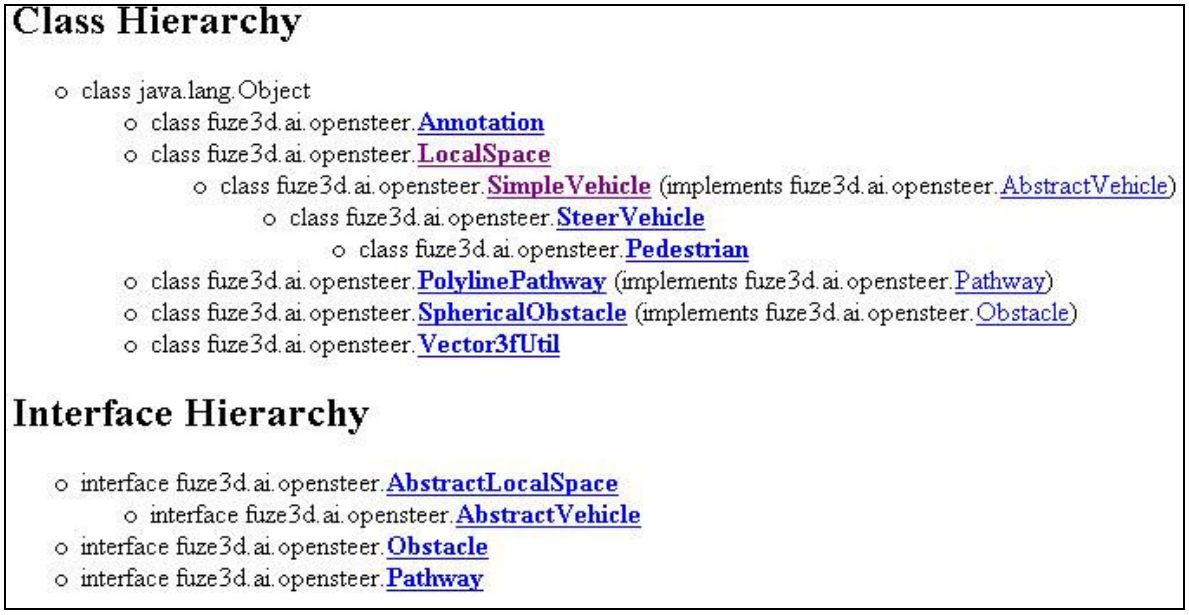

Figura 3.2: Classes do OpenSteer implementadas no Fuze3D

O Fuze3D utiliza um componente chamado OpenMind, para construir o grafo de cena, as animações dos personagens e implementação da física do ambiente. Para realizar a visualização o pacote utilizado se chama JOGL (*Java OpenGL*), que utiliza funções OpenGL 1.5 com as extensões atuais, inclusive shaders, porém em Java. Outras bibliotecas utilizadas são Cal3D e JGraph, que incrementam a plataforma com recursos de modelagem.

Contudo, a utilização dessa ferramenta não foi possível, devido a conflito entre as versões do Fuze3D e do pacote responsável pela geração dos quadros (*frames*) JOGL, apesar de termos entrado em contato com a autor do projeto, uma solução efetiva não foi obtida, pois o mesmo apenas reconheceu a incompatibilidade das versões e prometeu uma atualização no Fuze3D, que segundo ele, seria feita rapidamente, porém, essa promessa não se cumpriu, tornando impossível a utilização da plataforma.

3.4. 3D GameStudio

A plataforma 3D GameStudio foi encontrada após muita pesquisa, e escolhida pela sua principal característica: criação rápida e completa, de aplicações tridimensionais com interação em tempo-real, especialmente jogos. Com suporte a criação de modelos e terrenos, se encaixava dentro das expectativas da simulação.

A ferramenta possui sua *engine* como núcleo do sistema de desenvolvimento – ela gera as imagens 3D e controla o comportamento do mundo virtual. Devido a combinação de árvore BSP e renderizador de terrenos, a *engine* do GameStudio pode manipular cenários abertos e fechados, com o mesmo desempenho. Possui uma *engine* de iluminação que suporta sombras e fontes de luz móveis.

A construção de um mundo virtual com essa ferramenta é baseada em três etapas: edição de modelos (*Model Editor*), produção de scripts (*Script Editor*) e construção de níveis (*Level Editor*).

3.4.1. Editor de Modelos (Model Editor)

O editor de modelos, é usado para criar ou converter modelos 3D e terrenos para o 3D GameStudio. Possui interface amigável que facilita a criação de animações para os modelos. Através dessa ferramenta, pode-se criar os agentes, aplicar pele (*skin*), e produzir as animações referentes ao comportamento que desejamos obter (Figura 3.3). Os arquivos de modelos são salvos com extensão MDL5, MDL7, ou HMP (terrenos).

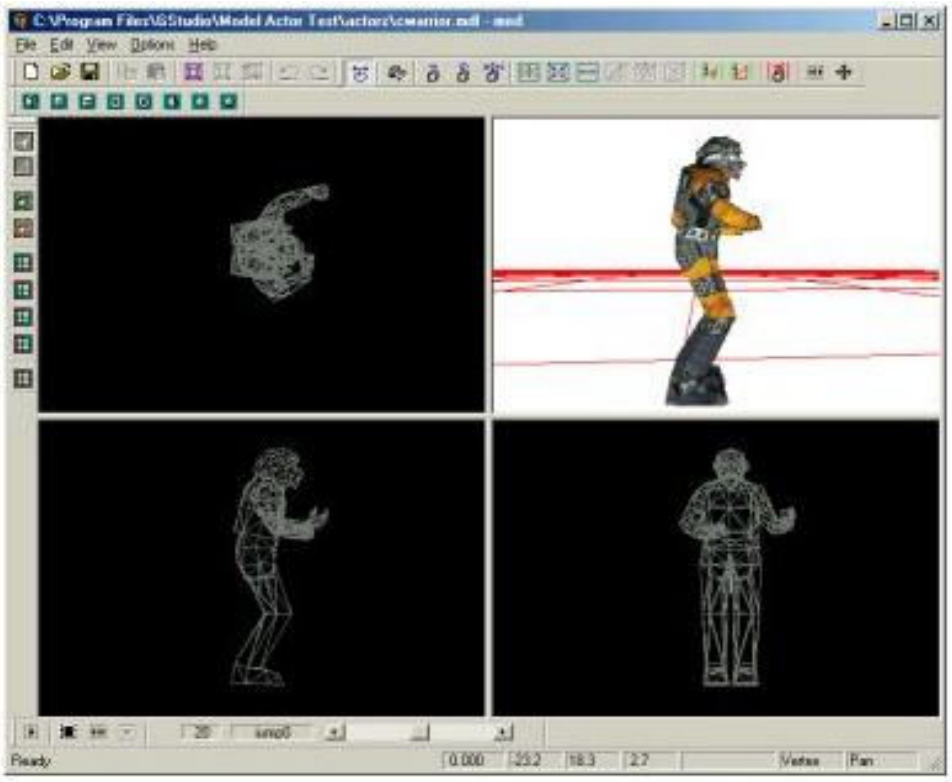

Figura 3.3: Criação e Animação de Humanóide

3.4.2. Editor de Script (Script Editor)

Baseado em linguagem C++, o editor de scripts pode ser usado para criar scripts para qualquer mundo virtual que queiramos construir, desde os mais simples, até grandes e complexos ambientes. Essa linguagem, definida pelos desenvolvedores dessa ferramenta, é denominada C-Script (Figura 3.4). Ela também nos oferece a possibilidade de adicionar apenas pequenos efeitos no

mundo virtual, ou construir toda a aplicação em C-Script. Os arquivos são gerados com a extensão WDL.

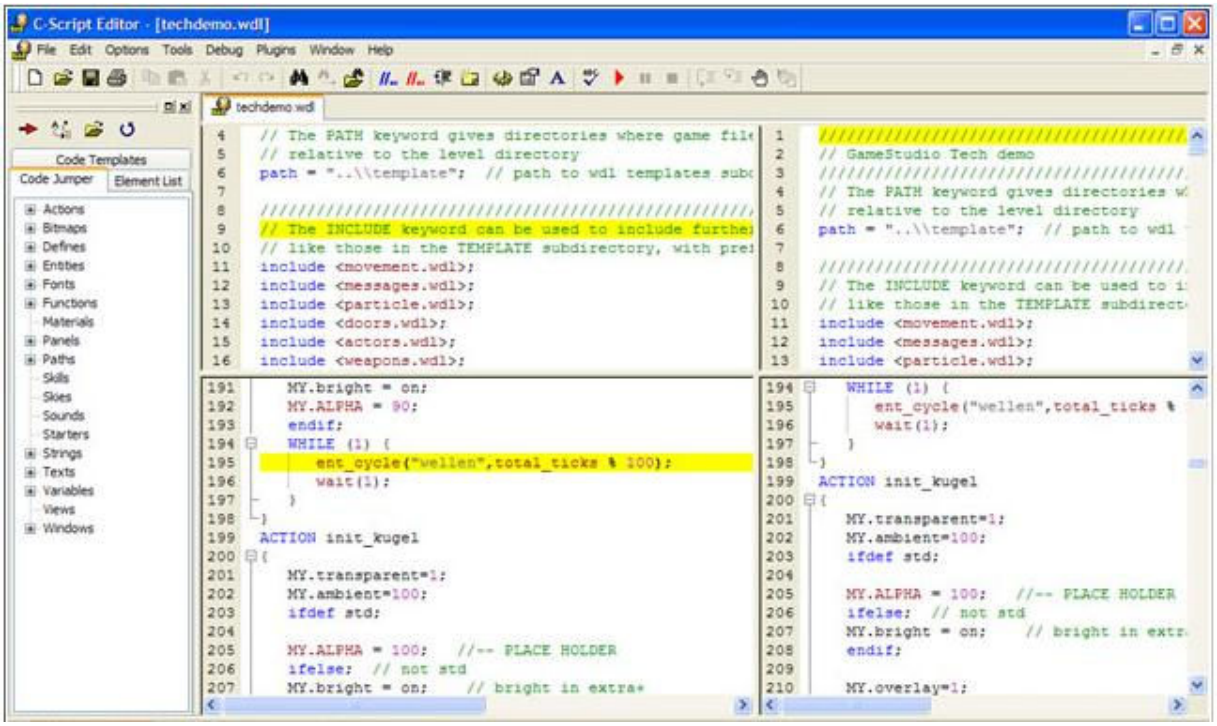

Figura 3.4: Exemplo de C-Script

3.4.2.1. C-Script

O C-Script possui características de modernas linguagens orientadas a objeto, sendo uma versão simplificada da linguagem profissional C++. Tem como principal atração, a facilidade em criar muitas características do ambiente com pouca programação.

A linguagem consiste de definições e funções. Definições criam objetos e instanciam suas propriedades iniciais; funções determinam o comportamento dos objetos, mudando suas propriedades durante a execução da aplicação, dependendo de certas condições.

A sintaxe do script é a mesma usada em Java, Javascript, C e C++, porém simplificada. Para um objeto existir, ele pode ser criado diretamente no mundo (*Level Editor*), ou pode ser definido no script, declarando seu tipo, nome e valores iniciais.

3.4.3. Editor de Níveis (Level Editor)

É a ferramenta para criação de mundos virtuais, que são chamados de níveis. Funciona também como IDE, o “centro de controle”, onde mundos, modelos e scripts são linkados, para obter o resultado final (Figura 3.5).

O editor de níveis gera mundos baseados em arvores BSP, para *engines* do GameStudio, com extensões WMP, e também importa níveis feitos em *Qooles*, *Quake* ou *Worldcraf,* com extensões QLE ou MAP.Tem como característica o paradigma de edição orientada a objeto. Essa metodologia esconde informação desnecessária para o usuário, oferecendo transparência. Com isso, é possível construir um mundo baseado em objetos lógicos ao invés de milhares de blocos e vértices. Também são introduzidas modularidade e portabilidade, já que o desenvolvedor pode construir seções independentes de um mundo, sem se preocupar com o resto do ambiente. Além disso, é possível compilar bibliotecas de objetos complexos e reutilizáveis.

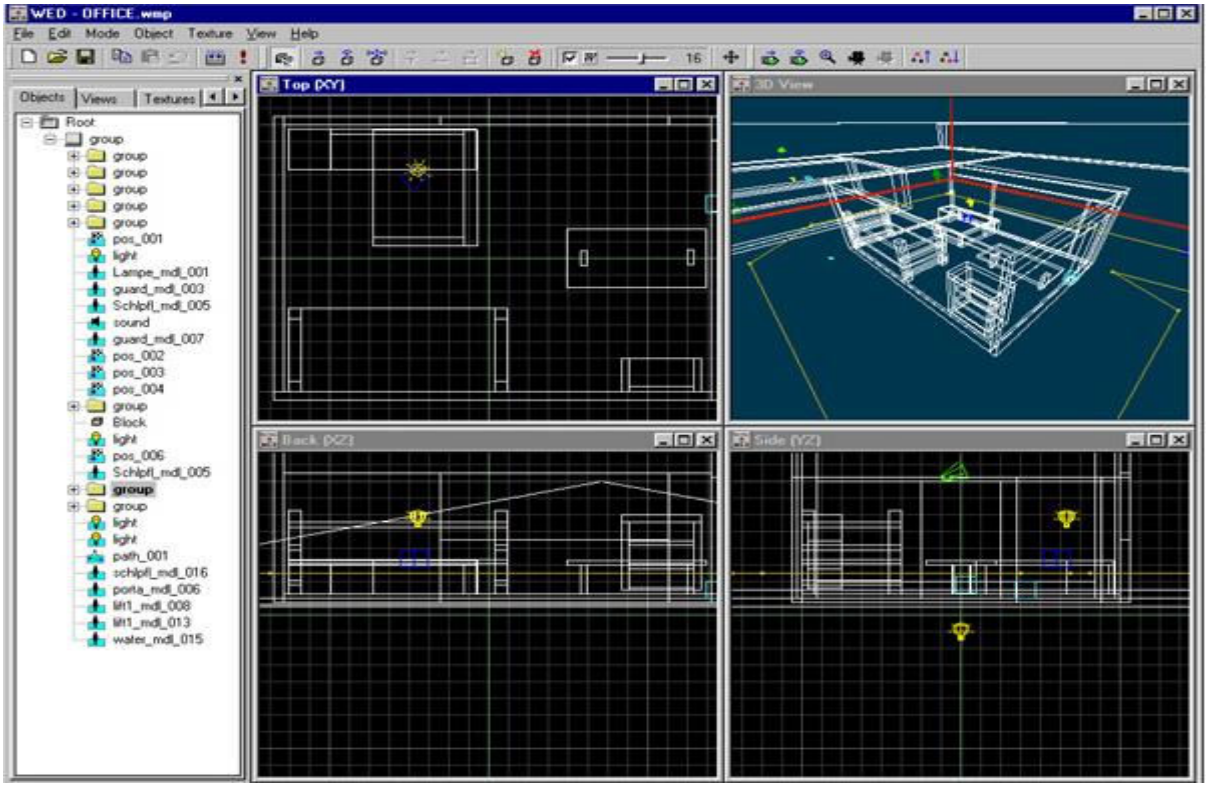

Figura 3.5: Exemplo de construção de nível

Há dois elementos primários em um mundo 3D construído com o Gamestudio: blocos e entidades. O primeiro é apenas um objeto sólido, texturizado, em qualquer forma convexa; para o segundo elemento existem quatro tipos de entidades: terreno, modelo, mapa e *sprite*. Cada um é utilizado para diferentes propostas.

3.4.3.1. Terreno

Um terreno consiste de um *grid* retangular com texturas aplicadas sobre ele. Como o próprio nome diz, terrenos são utilizados tanto em grandes como em pequenas partes de um ambiente, que formam terrenos irregulares. Pode ser criado com o editor de modelos, ou importado de programas construtores de terreno. Vários terrenos pequenos obtém um melhor desempenho do que um terreno imenso.

3.4.3.2. Modelo

Um modelo é um objeto 3D animado constituído por uma “rede” conectada de partes tridimensionais, com uma textura (pele) aplicada. Entidades desse tipo são utilizadas para pequenos objetos – atores, animais, ou similares. Podem ser criados através do editor de modelos (*Model Editor*). Modelos são as únicas entidades que podem utilizar sombras dinâmicas.

3.4.3.3. Mapa

Um mapa é um objeto tridimensional rígido. Pode ser usado em partes do mundo virtual que se movem como um todo, como portas, plataformas ou veículos. Podem ser criados utilizando o editor de níveis.

3.4.3.4. Sprite

Um *sprite* é um objeto 2D plano, que pode ser colocado em uma parede como um efeito visual, ou sempre ficar de frente para a câmera – como árvores ou elementos de cenário. Pode ser criado em programas comuns de manipulação de imagens. *Sprites* são renderizados muito mais rápido que modelos ou mapas e podem ser usados para explosões, luzes, chamas, e afins.

Por ser uma plataforma comercial, trabalhamos apenas com a versão *trial* do 3D GameStudio, o que implicou em uma série de limitações, observadas apenas após o estudo da ferramenta, como por exemplo, possibilidade de utilizar apenas scripts disponibilizados junto com a versão *trial,* também havia limitação no tamanho do ambiente e quantidade de entidades possíveis. Tudo isso nos impediu de criar a simulação inicialmente idealizada, cabendo apenas apresentar a ferramenta e mostrar as funções disponíveis, somente sem restrições na versão completa (*Standard*).

3.4. Análise das plataformas

Pode-se afirmar que cada plataforma utilizada possui suas próprias vantagens e desvantagens, o OpenSteer, por exemplo, possui uma implementação simples e objetiva de comportamentos autônomos, porém não oferece a capacidade de modelar os agentes. Nesta seção, destacam-se os principais pontos de cada ferramenta, e como cada uma delas utilizam o conceito de agente.

Como citado anteriormente, o OpenSteer integra os comportamentos aos agentes de forma muito simples, além de oferecer recursos para diversas situações. Sua maior desvantagem, é que ele não propicia modelagem de agentes para o ambiente, trabalhando apenas com formas bidimensionais.

O Fuze3d possui em sua definição, todas as classes relativas ao OpenSteer, com a vantagem de trabalhar com a criação dos personagens, porém,

por possui bibliotecas externas que não são atualizadas, enfrentam-se problemas de compatibilidade e instabilidade.

O 3D GameStudio é uma plataforma completa, voltada para a criação de jogos em ambientes virtuais, implementando os comportamentos através de scripts, e criando um único executável, o que o torna bastante portável. Contudo, se trata de uma ferramenta comercial, dificultando o acesso a uma versão completa e totalmente funcional.

Embora sem a efetiva manipulação das ferramentas, de modo a construir a simulação, podem-se analisar as plataformas pelos recursos oferecidos nas descrições e o que realmente foi constatado (Tabela 3.1).

Características	**Plataformas analisadas**		
	OpenSteer	***Fuze3D***	***3D GameStudio***
Modelagem de agentes	Não	Sim	Sim
Modelagem de ambientes	Não	Sim	Sim
Incorporação de regras aos agentes	Sim	Sim	Sim
Open source	Sim	Sim	Não
Dependências externas *(third-party)*	Não	Sim	Não
Disponibilidade de exemplos *(demo)*	Sim	Não	Sim
Linguagem utilizada	C++	Java	C-Script

Tabela 3.1: Comparativo das plataformas utilizadas

Conclusão

Através da leitura do vários textos utilizados para a execução deste trabalho, pode-se perceber que o estudo sobre os agentes autônomos vem se expandindo muito, embora seja um pouco restrito em relação à linguagem.

Por ser um novo paradigma, a programação orientada a agentes deve evoluir muito nos próximos anos, trazendo maiores facilidades para a implementação de agentes autônomos e mais recursos.

Existem aplicações no mercado que já utilizam o conceito de agentes autônomos, como relatado nessa pesquisa, e, verificou-se que é possível usar agentes, nas mais variadas tarefas, desde um organizador de correio eletrônico, até um complexo sistema industrial.

A pesquisa de plataformas que foi realizada mostrou que existem projetos e ferramentas que trabalham a criação de comportamentos, para incorporá-los a agentes autônomos, porém, não foi obtido sucesso na simulação devido a problemas e limitações presentes nas mesmas.

O aprimoramento que essa pesquisa proporcionou foi excepcional, pôde-se aprender novas técnicas de programação, assim como conhecer características importantes na construção de softwares. O trabalho construído poderá servir de base para novas pesquisas no campo de realidade virtual e inteligência artificial, facilitando a introdução desse assunto a novos pesquisadores.

O que pode ser sugerido é a busca por melhores plataformas, que sejam preferencialmente *open source*, para não haver problemas semelhantes aos encontrados nesse projeto, em relação a limitações de ferramentas comerciais.

Referências Bibliográficas

BRONSON, Gary. Program Development and Design Using C++, West, 1997.

CONITEC DataSystems. 3D GameStudio. Disponível em: < http://www.3dgamestudio.com/>. Acessado em: 25 de Outubro de 2005.

CYPHER, Allen; SMITH, David C.; SPOHRER, Jim; "KidSim: Programming Agents Without a Programming Language" Communications of the ACM, p. 55-67, 1994.

DEITEL, Harvey M.; DEITEL, Paul J. Java Como Programar: Quarta Edição. Prentice Hall, 1999.

FERBER, J. Multi-agent Systems: An Introduction to Distributed Artificial Intelligence, Addison Wesley Longman, Inglaterra, 1999.

FRANKLIN, Stan; GRAESSER, Art. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

GENESERETH, Michael R.; KETCHPEL Steven P. Software Agents In Communication of the ACM - Special issue on Intelligent Agent. 1994 - Vol. 37, 1994.

HAYES-ROTH, Barbara. "An Architecture for Adaptive Intelligent Systems," Artificial Intelligence: Special Issue on Agents and Interactivity, ed.72, p. 329-365, 1995.

HUHNS, Michael N.; SINGH, Munindar P. Readings in Agents. Morgan Kaufman, EUA, 1997.

JENNINGS, Nicholas; WOOLDRIDGE, Michael. Agent Theories, Architectures, and Languages: A Survey ,1994.

JENNINGS, Nicholas; WOOLDRIDGE, Michael. Formalizing the cooperative problem solving process. Thirteenth International Workshop on Distributed Artificial Intelligence, p. 403-417, 1994.

JENNINGS, Nicholas; WOOLDRIDGE, Michael. Towards a Theory of Cooperative Problem Solving. Sixth European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds, 1995.

LLOYD, Dave. Fuze3D. Disponível em: <http://www.shortfuze.co.uk/index.htm>. Inglaterra. Acessado em: 21 de Maio de 2005.

MAES, Pattie; Intelligent Software: Easing the Burdens that Computers Put on People. IEEE Expert 11(6): 62-63, 1996.

NORVIG, Peter; RUSSELL, Stuart. Artificial Intelligence: A Modern Approach. Prentice Hall, 1995.

REYNOLDS, Craig. OpenSteer. Disponível em: <http://opensteer.sourceforge.net>. Acessado em: 10 de Março de 2005.

WOOLDRIGDE, Michael. Introduction to Multi-Agent Systems, 2002.