UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

VIRTUAL WORLD BUILDER: AGENTES MÓVEIS COM REALIDADE AUMENTADA

RENATO CIVIDINI MATTHIESEN

ORIENTADOR: PROF. DR. NIVALDI CALONEGO JUNIOR

Piracicaba

2006

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

VIRTUAL WORLD BUILDER: AGENTES MÓVEIS COM REALIDADE AUMENTADA

RENATO CIVIDINI MATTHIESEN

ORIENTADOR: PROF. DR. NIVALDI CALONEGO JUNIOR

Dissertação apresentada ao programa de Mestrado em Ciência da Computação da Faculdade de Ciências Matemáticas e da Natureza da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do título de Mestre em Ciência da Computação.

Piracicaba

2006

Matthiesen, Renato Cividini

VIRTUAL WORLD BUILDER: AGENTES MÓVEIS COM REALIDADE AUMENTADA.

Orientador: Prof. Dr. Nivaldi Calonego Junior

Dissertação (Mestrado) – Universidade Metodista de Piracicaba – Faculdade de Ciências Exatas e da Natureza

Programa de Mestrado em Ciência da Computação

1. Aglets. 2. Realidade Virtual. 3. Realidade Aumentada, 4.Sistemas Distribuídos. 5. Agentes Móveis em Java. I.Matthiesen, Renato Cividini.

Calonego Junior, Nivaldi. II Universidade Metodista de Piracicaba, Faculdade de Ciências Matemáticas e da Natureza

Programa de Mestrado em Ciência da Computação. III. Titulo.

VIRTUAL WORLD BUILDER: AGENTES MÓVEIS COM REALIDADE AUMENTADA

Autor: Renato Cividini Matthiesen

Orientador: Prof. Dr. Nivaldi Calonego Junior

Dissertação de Mestrado defendida e aprovada em 28 de Abril de 2006 pela Banca Examinadora constituída pelos Professores:

_____

Prof. Dr. Nivaldi Calonego Junior

Universidade Metodista de Piracicaba

_____

Prof. Dr. Ricardo Luiz de Freitas

Pontifícia Universidade Católica de Campinas

_____

Prof. Dr. Cláudio Kirner

Universidade Metodista de Piracicaba

“Nós estamos aqui para fazer alguma diferença no universo. Senão, porque estaríamos aqui. Estamos criando uma nova consciência, como artistas, ou poetas...”

Noah Wyle interpretando Steve Jobs – Filme Piratas do Vale do Silício

“É difícil dizer o que é impossível, pois a fantasia de ontem é a esperança de hoje e a realidade de amanhã”.

(Robert H. Goddard)

“Senhor, dai-nos coragem para mudar as coisas que podem ser mudadas, paciência para aceitar as imutáveis e sabedoria para distinguir umas das outras”.

(Abraham Lincoln)

“Eu não procuro. Eu acho”.

(Pablo Picasso)

AGRADECIMENTOS

Agradeço primeiramente a Deus, que pela sua infinita bondade, me permitiu trilhar este caminho.

Agradeço a minha mãe, Tilú, que sempre me amou e zelou pelo meu bem.

Agradeço à minha esposa Regiane, que durante os momentos de incerteza, me deu forças e me fez acreditar que era possível realizar este trabalho.

Agradeço a AEHDA, que acreditou em seu colaborador e contribuiu com o apoio necessário para ele executar o seu projeto de vida.

Agradeço muito ao meu orientador, Nivaldi Calonego Junior, que, com toda sua competência, seriedade e paciência me ajudou a entender e a desenvolver as atividades deste trabalho de dissertação de mestrado.

RESUMO

A Realidade Aumentada (RA) é uma área da Realidade Virtual (RV) que utiliza técnicas avançadas de interface computacional para realizar a sobreposição de objetos virtuais no mundo real. Agentes móveis é uma tecnologia que representa um modelo de programação distribuída, onde o código executável e seu estado podem ser transportados entre *hosts* em uma rede de computadores.

Este trabalho tem como objetivo o estudo, projeto e implementação de uma estrutura baseada em agentes móveis para interagir em uma cena de realidade aumentada. O projeto foi baseado na metodologia e programação orientada a objetos e a programação fez uso da linguagem Java e da biblioteca "Aglets" para implementação dos agentes móveis. Foi utilizada uma versão da biblioteca "ARToolkit" para fazer a apresentação dos objetos virtuais na cena do mundo virtual.

Como resultado e contribuição, obteve-se uma estrutura chamada "VWBuilder", composta por três agentes. Um agente realiza a busca de objetos virtuais distribuídos em uma rede de computadores, outro agente faz o *download* destes objetos e um terceiro realiza a carga dos objetos virtuais em uma cena controlada pelo aplicativo de realidade aumentada. Esta estrutura pode ser reutilizada em aplicações para a construção de mundos virtuais em sistemas distribuídos de realidade aumentada implementados com a linguagem Java.

PALAVRAS-CHAVE: Aglets, Realidade Virtual, Realidade Aumentada, Sistemas Distribuídos, Agentes Móveis em Java.

ABSTRACT

The Augmented Reality (AR) is a Virtual Reality (VR) area that uses advanced computational interfaces techniques to do a superposition of virtual objects in the real world. Mobile agents is a technology that represents a distributed programming model, where the executable code and its state can be transported among hosts in a computer network.

This work aims a study, project and implementation of a structure based in mobile agents to interact in a augmented reality scene. The project was based in the object oriented programming methodology and the programming used the Java as a programming language and the "Aglets" as a library to implement the mobile agents. The "ARToolkit" library was used to present the virtual objects in the virtual world scene.

As a results and contribution, a structure called "VWBuilder" was obtained which is composed by three agents. One agent does the search of distributed virtual objects in a network computer, another agent does the download of this objects and a third agent does the load of virtual objects in a scene controlled by the augmented reality software. This structure can be reused in applications for build virtual worlds in distributed systems of augmented reality applications with Java language.

KEYWORDS: *Aglets*, *Virtual Reality*, *Augmented Reality, Distributed System*, *Java Mobile Agents*.

SUMÁRIO

LISTA ILUSTRAÇÕES

LISTA ABREVIAÇÕES E SIGLAS

3D – Três dimensões

ADSL – *Asynchronous Digital Serial Line*

API – *Application Program Interface*

AAPI – *Aglets Application Programming Interface*

ASDK – *Aglets Software Development Kit*

ATM – *Asynchronous Transfer Mode*

AVC – Ambiente Virtual Colaborativo

CAD – *Computer Aided Design*

CAVE – *Cave Automatic Virtual Environment*

CGI – *Common Gateway Interface*

COM – *Component Object Model*

CORBA – *Common Object Request Broker Architecture*

CVE – *Collaborative Virtual Environments*

DCOM – *Distributed Component Object Model*

EAI – *External Authoring Interface*

E/S – Entrada e Saída

FIPA – *Foundation for Intelligent Physical Agents*

FTP – *File Transfer Protocol*

GPL – *General Public Licence*

GUI – *Graphic User Interface*

HMD – *Helmet Mounted Displays*

HTML – *Hypertext Markup Language*

HTTP – *Hypertext Transfer Protocol*

IBM – *International Business Machine*

IP – *Internet Protocol*

IR – *Information Retrieval*

ISO – *International Standard Organization*

JAAPI – *Java Aglets Application Programming Interface*

JDK – *Java Development Kit*

JVM – *Java Virtual Machine*

LAN – *Local Area Network*

LOD – *Level of Detail*

MAN – *Metropolitan Área Network*

MASIF – *Multi Agent System Interoperability Foundation*

MPI – *Message Passing Interface*

NVE – *Network Virtual Environment*

OMG – *Object Management Group*

PDA – *Personal Digital Assistant*

PVM – *Paralell Virtual Machine*

RMI – *Remote* Method Invocation

RPC – *Remote Procedure Call*

RA – Realidade Aumentada

RV – Realidade Virtual

SAI – *Script Authoring Interface*

SGML – *Standard Generalized Markup Language*

SQL – *Structured Queue Language*

SNMP – *Simple Network Management Protocol*

TCP – *Transmission Control Protocol*

UCL – *Universal Communication Language*

UDP – *User Datagram Protocol*

UML – *Unified Modelling Language*

URL – *Universal Research Locator*

VCL – *Virtual Collaborative Environment*

VRML – *Virtual Reality Modeling Language*

W3D – *Web 3D Consortium*

WAN – *Wide Area Networks*

WIMP – *Window, Icon, Menu and Pointer*

WWW – *World Wide Web*

X3D – *Extensible 3D*

XML – *Extensible Markup Language*

LISTA DE TABELAS

1 INTRODUÇÃO

A tecnologia da informação é parte do processo evolutivo dos meios de comunicação que no princípio da civilização usava desenhos nas paredes de cavernas, em placas de barros ou casca de árvores para transmitir informações. Posteriormente, ainda em um passado remoto, surgiu o papel, primeiro grande avanço na forma de apresentar informações. Com o advento da eletrônica, dos computadores e das redes, a informação passou a ser armazenada eletronicamente e apresentada em equipamentos como a televisão e os computadores. As formas de captura, tratamento e transmissão de informações sofreram modificações significativas devido ao desenvolvimento dessas tecnologias. O final do século XX vivenciou a explosão do desenvolvimento das telecomunicações, que possibilitou a proximidade virtual entre pessoas no mundo. Vivenciou, também, a modificação dos padrões de interface homem-computador, que passou de sistemas complexos baseados em chaves eletrônicas para sistemas gráficos baseados em janelas.

O ser humano não se contenta mais em transmitir palavras ou simples desenhos. Imitar a realidade e utilizar gráficos 3D para apresentar dados, tornou-se uma das maneiras mais eficientes de transportar informações (LEA, 1996). Ambientes virtuais 3D baseados na Web e multiusuários foram pensados como a interface do futuro (JUNG, 1999) e podem ter representações de usuários através de um avatar (agente), que compõem mundos virtuais, comunicam-se e interagem com outros objetos. Além disso, houve avanços no uso de outros aparelhos sensitivos do homem no estabelecimento de canais de interação com o computador. O uso combinado desses elementos fez com que fosse cunhado o termo Realidade Virtual.

A Realidade Virtual (RV) apresenta-se como uma maneira contemporânea de se apresentar informações. Ela baseia-se na construção de mundos virtuais, onde o apelo para utilização de imagens tridimensionais é uma de suas principais características. A RV é uma técnica avançada de interface que permite ao usuário imergir, navegar e interagir em ambientes

virtuais em 3D gerados por computador em tempo real, utilizando canais multi-sensoriais (BURDEA, 1994).

Inicialmente, a RV tratou do uso de simuladores para a produção de ambientes que tivessem comportamento análogo ao ambiente real. Mas, atualmente, RV é tratada como um *continuum* que abrange da Virtualidade Aumentada indo até a Realidade Aumentada (RA), isto é, de elementos reais inseridos em ambientes virtuais (onde a interface é virtual), até o seu oposto, que é o ambiente virtual povoado com elementos reais (onde a interface é real). Quando a percepção não permite a distinção entre a realidade aumentada e a virtualidade aumentada, tem-se a realidade misturada (figura 01). Nela o virtual e o real se misturam em mundos virtuais e possibilitam a construção de ambientes, antes, apenas imagináveis.

Figura 1 - Realidade Misturada

O desenvolvimento de RV e de RA, permite a geração de mundos virtuais interativos em que se represente informação real e virtual, sob a forma de um cenário. Esta técnica de se apresentar informações pode ser aplicada nas mais variadas áreas do conhecimento. Em muitos casos, a RV revoluciona a forma de interação das pessoas com sistemas complexos tratados com o uso de computadores, propiciando maior desempenho e economizando custos. Dentre as várias áreas, pode-se citar as seguintes:

- **Entretenimento**: jogos virtuais; turismo virtual; cinema virtual;

- **Visualização científica**: superfícies planetárias; túnel de vento; síntese molecular; simulações nucleares;
- **Aplicações médicas/saúde**: simulação cirúrgica; planejamento de radioterapia; saúde virtual; ensino (anatomia); tratamento (deficientes);
- **Arquitetura e projeto**: CAD (*Computer Aided Design*); projeto de artefatos; planejamento; decoração; avaliação acústica;
- **Educação**: laboratórios virtuais; exploração planetária; educação à distância; educação de excepcionais;
- **Treinamento**: simuladores de vôo; planejamento de operações militares;
- **Artes**: pintura; escultura virtual; música; museu virtual;
- **Controle da informação**: visualização financeira; visualização da informação; informação virtual;
- **Telepresença/telerobótica**: controle de sistemas remotos; teleconferência; professor virtual; espectador remoto.

Nessas condições observa-se que nos sistemas de RV e RA, os objetos que compõem o mundo virtual são carregados via leitura do arquivo de grafo de cena (arquivo VRML), armazenado localmente no sistema. Neste contexto, foi observado o problema de que, para a construção destes mundos virtuais, o sistema fica limitado ao carregamento dos objetos disponíveis somente localmente. Muitas vezes restritos a poucos objetos virtuais. Não se apresenta a possibilidade de carregar objetos virtuais desenvolvidos e distribuídos em outros *hosts* de uma rede de computadores. O compartilhamento ou a possibilidade de carregar objetos virtuais distribuídos, desenvolvidos por outros usuários em diferentes sistemas ou ainda em outros ambientes virtuais é inexistente.

Para buscar objetos virtuais distribuídos, é pesquisado nesta dissertação, mecanismos de busca e apresentação de objetos virtuais, e dentre as tecnologias de exploração e busca de objetos para o tratamento da informação, destacam-se os agentes móveis (BAX, 2001). Eles representam um passo na utilização de códigos em execução na Internet, que introduzidos na

rede, podem ser transportados em tempo de execução juntamente com suas informações (IBM, 2003). Essa tecnologia desempenha papel importante na solução de problemas específicos ao permitir a exploração dos recursos computacionais disponíveis no local para onde o código é migrado. Alinhado com a busca de objetos virtuais distribuídos, pretende-se testar a possibilidade de utilização da tecnologia de agentes móveis (com a biblioteca “Aglets”) no domínio de sistemas colaborativos de realidade aumentada.

Como solução para o problema do compartilhamento de objetos virtuais distribuídos e como interesse em verificar o real uso da tecnologia de agentes móveis em RA, é apresentado um protótipo para mapeamento, *download* e carregamento de objetos virtuais distribuídos em uma rede de computadores explorando um novo modelo de programação. O resultado é a discussão relativa à construção de ambientes de RA com o uso de agentes móveis como uma nova alternativa aos modelos colaborativos de mundos virtuais. Este protótipo pode ser utilizado para a construção de ambientes virtuais através da busca ou compartilhamento de objetos virtuais ou na facilidade do desenvolvimento de aplicações distribuídas em rede.

Este trabalho discute aspectos do desenvolvimento e da experimentação do *software* desenvolvido à luz das metodologias utilizadas para o desenvolvimento de um protótipo, considerando diferentes modelos de programação.

O projeto e desenvolvimento é orientado a objetos, usando o modelo de programação baseado em agentes móveis e na linguagem Java para a implementação dos agentes, e usando o modelo cliente-servidor com programação em C++ para a implementação da conexão entre o agente móvel e a aplicação de RA. (ver item 3.1 O Projeto do VWBuilder).

As informações necessárias para a instalação e configuração dos ambientes de programação dos Aglets (agentes móveis em Java) e da biblioteca ARToolkit (para rastreamento dos objetos virtuais no ambiente de realidade aumentada) estão descritas detalhadamente nos Anexos 3 e 4 deste trabalho.

A metodologia de projeto utilizada foi baseada em Orientação a Objetos apoiando-se no fato de que ela permite que se dê importância à estrutura do sistema, e não somente a suas funções, além de permitir a reutilização de códigos. Entre as características deste modelo, pode-se citar: (i) Polimorfismo, permite que vários objetos possam ser executados através da solicitação de uma única operação; (ii) Herança, faz o compartilhamento de atributos e operações de classes mais abstratas. (iii) Encapsulamento, possibilita que objetos tenham diferentes visibilidades e sejam utilizados em diferentes situações.

Para a descrição do projeto, foi utilizada a tecnologia *Unified Modeling Language* (UML) (RUMBAUGH, 2000). Deste modelo foram utilizados quatro diagramas para a descrição do sistema: (i) Diagrama de Classes, que reproduz a estrutura do sistema, apresentando as classes e seus relacionamentos; (ii) Diagrama de Caso de Uso, representa um conjunto de atores, casos de uso e os relacionamentos entre eles; (iii) Diagrama de Seqüência, enfatiza a ordenação temporal em que as mensagens são trocadas entre os objetos de um sistema, e (iv) Diagrama de Estados, representa os estados possíveis de um objeto em particular. Para a criação dos diagramas foram utilizadas as ferramentas JUDE[1] e Posseidon[2] for UML. Estes softwares permitiram gerar parte dos diagramas do projeto *VWBuilder*.

O trabalho está organizado em 4 capítulos. Este capítulo faz uma introdução descrevendo o trabalho de maneira geral e traz o problema e os objetivos do desenvolvimento da dissertação. O capítulo 2 faz uma fundamentação teórica, abordando os domínios de: Realidade Virtual e Agentes Móveis. O capítulo 3 apresenta a descrição do protótipo, seus diagramas, explicações sobre suas classes e a funcionalidade da estrutura do VWBuilder. Para finalizar, o capítulo 4 apresenta os resultados obtidos demonstrando a utilização do protótipo e as considerações finais.

[1] JUDE – Java and UML Developers' Environment http://jude.change-vision.com/jude-web/index.html.
[2] Posseidon for UML 4.1. http://gentleware.com/index.php.

2 REALIDADE AUMENTADA COM AGENTES MÓVEIS

Com o objetivo de situar o leitor no domínio da aplicação, é realizado, neste capítulo uma fundamentação teórica das tecnologias de Realidade Aumentada, no contexto da Realidade Virtual e de Agentes Móveis, no contexto de Sistemas Distribuídos. Também são tratados trabalhos correlatos e ambientes de desenvolvimento (“Aglets” e “ARToolkit”) utilizados na implementação do protótipo.

2.1 Realidade Virtual

Morton Heilig sintetiza o porquê do uso da RV em:

> “se o computador deve alcançar todo o seu potencial como professor, curandeiro, ou inspirador, ele precisa dialogar com o homem da mesma maneira que esse dialogou com a natureza, durante milhões de anos. Isto é, o computador deve falar com o homem, estimulando todos os seus sentidos e permitindo que ele reaja com todas as suas respostas motoras” (HAMIT, 1993).

Esta é a base para a necessidade da utilização da RV em aplicações humanas. A RV objetiva aproveitar o potencial dos computadores, permitindo o crescimento pessoal do próprio homem. Para isso é necessário estender os princípios das interfaces homem-máquina.

Considerando-se que 70% dos receptores do sentido humano encontram-se nos olhos, uma abordagem especial deve ser empregada em relação ao estímulo à visão humana. Comparando o uso de aplicações comuns em sistemas de RV e em sistemas baseados na tecnologia WIMP (*Window, Icon, Menu and Pointer*), alguns aspectos podem ser observados:

- **O 3D é um superconjunto do 2D**. Tudo o que pode ser feito em uma tela plana é aplicável em um mundo tridimensional.

- **A sobreposição de janelas esconde a informação**. O resultado direto de utilizar muitos dados é a necessidade de compartilhar o espaço de trabalho bidimensional. O usuário gasta parte de seu tempo organizando a sua área

de trabalho e buscando documentos. Uma solução para esse problema foi a utilização de ícones[3] para representar os documentos. Em um ambiente virtual, um documento pode ser colocado em um determinado local e obter uma iconização espacial. Desta forma, não há necessidade de reorganizar o mundo virtual para se obter os dados.

- **A distância não é importante em um mundo virtual**. A distância perceptiva de um objeto não apresenta ligação com a quantidade de tempo que ele leva para ser acessado. Um comando, do tipo "vá para", transporta instantaneamente o usuário para o novo local.

A partir dessas observações, nota-se que a utilização de mundos virtuais apresenta vantagens em relação as GUIs (*Graphical User Interfaces*). Porém, a imersão em um ambiente virtual considera mais do que os estímulos à visão humana. O aplicativo de RV é uma simulação animada que permite definir e exibir objetos 3D, alterar seu ponto de referência, campo de visão, manipular e interagir com os objetos. A esses aplicativos é permitido permear objetos contidos no mundo virtual com comportamentos pré-definidos e propriedades físicas, programados para ativarem algum tipo de *feedback* visual, auditivo ou tátil quando um evento específico ocorre.

Além da imersão simbólica e imersão através de ações, os avanços na tecnologia de interfaces, baseadas em RV, induzem um senso de imersão física em um contexto sintético, que envolve a manipulação sensorial, a fim de possibilitar a suspensão da percepção de que o indivíduo está envolto por um mundo virtual. A impressão é de estar em uma nova realidade, ao invés de estar olhando através de uma janela do monitor para um ambiente sintético.

A Realidade Virtual (RV) vem experimentando um desenvolvimento acelerado e indicando perspectivas promissoras para os diversos segmentos vinculados com a área (RAPOSO, 2005). Historicamente, uma das primeiras revoluções na computação foi dada com a mudança de interface em ambiente texto para ambiente gráfico. Esta mudança e os aprimoramentos de *hardware*

[3] Ícone: pequena imagem gráfica que representa, na tela, um objeto que pode ser manipulado.

possibilitaram maior investimento em aplicações gráficas, que culminou na computação gráfica, e evoluiu de imagens estáticas com baixa resolução para animações 3D de alto realismo. O uso de simuladores sofisticados, viabiliza a RV, que gera os Ambientes Virtuais (*Virtual Environments*) ou Mundos Virtuais (*Virtual Worlds*). Uma característica da RV é permitir entrar em um mundo virtual e interagir, em tempo real, com o ambiente.

O aparecimento da RV, aduz ao surgimento das primeiras pinturas renascentistas no final da Idade Média quando a introdução da perspectiva gerou imagens mais realistas a partir do estímulo à sensação de profundidade. Este maior realismo causou um forte impacto na população da época e, desde então, passou a redefinir a arte de representar o real (LEITE JR, 2000). Nos sistemas computacionais atuais, a RV utiliza o computador como novo paradigma de interface com o usuário que não está mais em frente ao monitor, e sim imerso na interface. (AUKSTAKALNIS, 1994).

Para implementar RV, pode-se apoiar no argumento de que imagens geradas pelos computadores têm maior atenção que os algoritmos que a produzem (FERNANDESa, 2000). Imagens, filmes e textos que exploram os limites da interação humana em mundos virtuais 3D e multidimensionais contribuem de forma mais intensa para disseminar o conceito do ciberespaço.

A RV pode ser definida como a forma mais avançada de interface do usuário de computador (HANDCOCK, 1995). Com aplicação na maioria das áreas do conhecimento, e com um grande investimento das indústrias na produção de *hardware*, *software* e dispositivos de E/S especiais, a RV vem experimentando um desenvolvimento acelerado nos últimos anos e indicando perspectivas promissoras para diversos segmentos (KIRNER, 2001).

Entre outras definições, define-se RV como uma técnica avançada de interface, onde o usuário pode realizar imersão, navegação e interação em um ambiente sintético tridimensional, gerado por computador em tempo real. Utiliza canais multi-sensoriais, o uso de computadores e interfaces para criar efeitos em mundos tridimensionais contendo interação de objetos com senso de presença tridimensional (BRYSON, 1996).

A RV envolve controle tridimensional interativo de processos computacionais. O usuário entra no espaço virtual das aplicações, visualiza, manipula e explora os dados da aplicação em tempo real, usando seus sentidos e movimentos naturais do corpo. Uma vantagem desse tipo de interface é que o conhecimento intuitivo do usuário, a respeito do mundo físico, pode ser transferido para manipular o mundo virtual. Para suportar esse tipo de interação, o usuário utiliza dispositivos não convencionais como capacete de visualização (HMD – *Helmet Mounted Displays*), *mouses* especiais, luvas (*Data Gloves*), entre outros. Estes dispositivos dão ao usuário, a impressão de que a aplicação está sendo executada em um ambiente tridimensional real, permite a exploração do ambiente e a manipulação natural dos objetos com o uso das mãos, para apontar, pegar, e realizar ações.

A RV pode ser imersiva ou não imersiva. Do ponto de vista da visualização, a RV imersiva é baseada no uso de capacete ou de salas de projeção nas paredes (CAVE – *Cave Automatic Virtual Environment*), enquanto a RV não imersiva, baseia-se no uso de monitores convencionais ou estereoscópicos. De qualquer maneira, os dispositivos baseados nos outros sentidos acabam dando algum grau de imersão à RV.

Uma aplicação que utiliza RV deve proporcionar imersão, interação, e estimular a imaginação. (BURDEA, 1994). Isto é, um transporte para o contexto “virtual” destes três fatores que representam a realidade e cotidiano através de uma imersão total em um ambiente completamente interativo, modificado pela imaginação. Pode-se traçar um paralelo entre o significado da realidade e o conceito de RV. Quanto maior o nível de imersão, interação e de imaginação aplicado a um sistema, mais próximo chegamos da sintetização de uma nova realidade, a RV.

2.1.1 Programação em Realidade Virtual

A programação de sistemas de RV, devido à sua complexidade, requer múltiplos conhecimentos como: programação em tempo real, orientação a objetos, redes, modelagem geométrica, modelagem física e programação multitarefa. Para facilitar essa tarefa, algumas empresas e universidades

produziram sistemas de desenvolvimento de RV, conhecidos como *VR Toolkits*. Eles são bibliotecas de funções orientadas a objeto, voltadas para especificações de RV, onde um objeto simulado passa a ser uma classe e herda seus atributos padrão.

A *Virtual Reality Modeling Language* (VRML) é uma linguagem para modelagem de RV independente de plataforma, que permite a criação de cenários 3D, para visualizar e interagir com objetos 3D. Apresenta percepção à navegação na Web através de descrições de cenas onde o usuário se encontra. A documentação de especificação do VRML (WEB3D, 2004) afirma que VRML é uma das formas mais generalizadas de visualização e navegação em mundos virtuais. A linguagem foi concebida para descrever simulações interativas de múltiplos participantes, em mundos virtuais disponibilizados na Internet e ligados com a *World Wide Web* (WWW), através de um *browser.* A primeira versão da linguagem (VRML 1.0) não possibilitou muita interação do usuário com o mundo virtual. Nas versões futuras foram acrescentadas características como animação, movimentos de corpos, som e interação entre múltiplos usuários em tempo real. O *VRML 2.0 draft #3*, chamada *Moving Worlds* (padronizada pela *International Standard Organization*, como VRML97), é uma coleção de objetos, chamados nós (*nodes*) com características específicas (MORCRETE, 1999). A classe “sensors”, por exemplo, pode ser utilizada para realizar uma ligação do mundo virtual a um evento externo. Os sensores são “nós” que geram eventos, permitem a interação e animação de uma cena. A dinâmica da cena é produzida através de roteamento de eventos (ROUTE). Os principais sensores são: “TimeSensor” e “TouchSensor”. Estes sensores disparam determinadas ações quando acionados.

Para possibilitar que usuários ou programas tenham interação com cenas, fazendo uso de operações não suportadas apenas com o VRML, utiliza-se outras linguagens em conjunto na programação de cenas. Pode-se utilizar linguagens como o Java, Java3D, C++, Tcl/Tk, Prolog, X3D.

Existem extensões da linguagem como o VRML+ (FERNANDESb, 2000) que oferecem ao programador possibilidades como a declaração de

variáveis, estruturas de repetição, instruções condicionais e acesso à base de dados. O VRML *Specifications,* oferece maneiras de acessar informações de banco de dados utilizando SQL (*Structured Queue Language*) diretamente do *browser* ou através de uma conexão com servidor Web. Outra tecnologia utilizada em mundos virtuais para acessar base de dados é o SQL3D, uma maneira de utilizar dados em aplicações 3D interoperando com CORBA[4].

O *Extensible Markup Language* (XML) é uma especificação do W3C[5]. É uma nova versão do *Standard Generalized Markup Language* (SGML)[6] desenhada especificamente para definições de *tags*, e oferecer funcionalidade e significado em documentos *Hyper Text Markup Language* (HTML). O 3DXML é um meio de descrever sites Web e outras estruturas de informações XML e publicar estas informações em VRML. Esta tecnologia utiliza CGI ou Java para ligar as informações com os mundos virtuais.

Uma especificação para a tecnologia XML, considerada a nova versão do VRML, é o X3D (*eXtensible 3D*) (HOLMES, 2002). O *VRML Consortium* é atualmente o Web3D *Consortium*[7] e define os aspectos de tecnologias 3D utilizadas na Internet. Lançado em 2001, o X3D propõe resolver as limitações do VRML97 utilizando sintaxes do XML. Nesta versão, novas características podem ser adicionadas como *streaming*, ou níveis, nos mundos virtuais.

O Java3D[8] é uma interface de programação para aplicativos utilizada para escrever aplicações gráficas tridimensionais e *applets* (HERRMANN, 2000). Oferece ao desenvolvedor um alto nível para construção e manipulação de geometria 3D para construção de aplicações que podem descrever mundos virtuais integrados à Internet. As *applets* escritas com a API Java3D, tem acesso direto às classes escritas em Java. Foi projetado com

[4] CORBA – *Commom Object Request Brooker Architecture*. São especificações de acesso a sistemas distribuídos. Uma abordagem sobre CORBA é encontrada em http://www.corba.org
[5] W3C – *World Wide Web Consortium* – Consórcio internacional de companhias envolvidas com o desenvolvimento de padrões para a Internet e Web.
[6] SGML – É uma meta-linguagem para organização de elementos em documentos.
[7] Especificações sobre o Web3D Consortium pode ser encontradas em http://www.web3d.org
[8] A API do Java 3D foi desenvolvida pelo Java 3D Consortium (SGI, Intel, Apple Computer, e Sun Microsystems). Detalhes estão em http://java.sun.com/products/java-media/3D

objetivos especiais para oferecer alta performance como: (i) grande conjunto de características para criação de mundos 3D; (ii) alto nível de orientação a objetos para construção de aplicações e *applets,* e (iii) suporte a carregadores *runtime* para permitir ao Java 3D suportar outros formatos (CAD, VRML).

O Java3D é uma API orientada a objetos. Constrói elementos gráficos separados e conectados entre si através de uma estrutura de árvore chamada de grafo de cena. A aplicação manipula estes objetos através de "nós" (*nodes*) e métodos. Possui três diferentes modos: imediato, retido e compilado de renderização de cenas, oferecendo maior liberdade e otimização na renderização de aplicações.

2.1.2 A VRML como recurso de programação em RV

O poder da VRML vem de sua capacidade de suportar cenas 3D dinâmicas e interativas, e sua flexibilidade decorre de um conjunto de "nós" que habilitam interações das cenas com a Web. O *Event Model* oferece possibilidade para que eventos carreguem seu estado entre entidades e cena, e definam o relacionamento entre uma entidade externa e a cena VRML.

Embora a VRML não seja uma linguagem orientada a objetos, ela possui artifícios que a torna semelhante à orientação a objetos. Por exemplo, a utilização do identificador PROTO permite a criação de objetos definidos e delimitados (DA LUZ, 2002).

Para animar cenas 3D em mundos virtuais, necessita-se de um mecanismo para alterar o estado das entidades na cena. O VRML 2.0 suporta este mecanismo através do *Event Model*.

Para alterar uma cena pode-se utilizar três abordagens (LEA, 1996): (i) **API**, onde as cenas são manipuladas por *procedures* ou chamadas à métodos; (ii) ***Language***, pela própria linguagem e (iii) ***Event-based***, que são eventos de uma linguagem externa. Os *Event-based* podem ser realizados através da técnica de SAI (*Script Authoring Interface*) ou EAI (*External Authoring Interface*) que se baseiam em "nós" (*nodes*) como elemento básico nomeado em um acena pelo "nó" DEF.

A técnica de EAI, apresenta um conceito de interação de objetos de ambientes externos com um mundo virtual. É uma técnica que utiliza linguagem independente e permite manipular objetos externos às cenas 3D (SILVA, 2001). É um anexo da especificação do VRML97. Está implementado em alguns *softwares* como por exemplo no *plug-in* para *browser* “Cortona” da Parallel Graphics®, na plataforma Java3D® e nas especificações do X3D. Com o EAI é possível interagir com a cena 3D através da criação de controles e elementos em uma *applet*. O programador trabalha da mesma forma que um *script* interno, utilizando “EventIn” e “EventOut”.

Os eventos oferecem a base para o modelo de execução no VRML. Evento é uma mensagem que contém dados e utiliza-se de um “gatilho” para iniciar a execução. A vantagem do modelo baseado em eventos (*event-based*) é que ele separa os componentes essenciais do modelo de execução.

Os eventos que irão gerar mudanças na cena são: “interpolator” e “sensor”. O “Interpolator” define valores para geração de outros valores intermediários entre os valores dos “nós” (*nodes*).

O comportamento dinâmico pode ser dado pelo “Script node”, que é capaz de rotear eventos dentro de uma cena e de um “nó” externo, associando o “nó” com um arquivo apontado pelo comando “url”. Utiliza-se, então, campos definidos que oferecem o *link* entre a cena VRML e o código Java.

Alguns comandos de interação entre o VRML e o Java são: “TouchSensor”: usado para receber eventos através do mouse, e oferecer interatividade à cena; “TimeSensor”: utilizado para realizar animações simples e repetitivas; “ProximitySensor”: apresenta a posição do navegador de um “nó”; “Switch”: oferece a possibilidade de agregar nós filhos (*children*) em um único “nó”; “Collision”: define a ocorrência de colisão entre “nós”; “LOD”: define o nível de detalhamento de um objeto; “Viewpoint”: define uma câmera pré-alocada em uma dada posição; “Inline”: carrega um arquivo “.wrl”; “PROTO”: agrega um conjunto de nós e campos em um outro “nó” definido, e “EXTERNPROTO”: mecanismo de compartilhamento de protótipos.

2.1.3 A linguagem Java

A linguagem Java possibilita que o processamento deixe de ser realizado apenas no lado do servidor como nas CGI e *servlets*, passando a ser executado também no cliente, no caso das *applets*. É uma linguagem orientada a objetos, robusta, interpretada, segura, portável e *multithread*, amplamente utilizada em computação baseada em redes (PAPASTAVROU, 2000). Permite que o código seja parcialmente transportado entre clientes e servidores como no método RMI (detalhes na página 28). Neste trabalho é abordado a programação de agentes móveis, e com relação a este tipo de programação, Lange(1998) e Oshima(2003) apresentam características que fazem com que Java possa realizar este tipo de programação: (i) orientada a objetos; (ii) execução segura; (iii) trabalha em ambientes heterogêneos; (iv) utiliza carregamento dinâmico de classes; (v) suporta programação *multithread*; (vi) suporta *s*erialização de objetos e (vii) é reflexiva.

No modelo de segurança da linguagem Java, um programa é executado dentro de uma *Sand Box* (caixa de areia) onde, para cada usuário, é definido diferentes níveis de segurança (OAKS, 1999). Isto impede que alguns programas executem ações que possam danificar o sistema onde está sendo executado. Java também especifica paralelismo por *multithread* e permite que programas pesquisem buscando informações e colaborem com programas que executem em outros computadores. Os recursos fundamentais de rede estão no pacote "**java.net**", onde oferece comunicação baseada em *sockets*. O pacote "**java.rmi**" e o pacote "**org.omg**" permitem a execução dos programas distribuídos via RPC, RMI e CORBA (DEITEL, 2001).

Apesar de a linguagem Java poder ser utilizada para a programação de agentes móveis, por si própria, não oferece total suporte para execução dos agentes móveis. Entre suas deficiências estão: (i) inadequado suporte para controle de fontes; (ii) falta de proteção em referências; (iii) falta de propriedade de referências; e (iv) falta de suporte a reinício de execução. Para poder empregar a tecnologia de agentes móveis, utiliza-se pacotes específicos com as classes que implementam os métodos necessários para mover o código.

Um exemplo de API, ou seja, de pacote de classes para agentes móveis é a plataforma “Aglets”. Esta plataforma será utilizada na implementação dos agentes móveis no protótipo desta dissertação.

Com a linguagem Java os programadores podem desenvolver aplicações distribuídas sem se preocupar com as plataformas de suporte (*write once, run anywhere*). É necessário apenas que as aplicações suportem o *Java Virtual Machine* (JVM) (ROQUE, 1999).

2.1.4 Java com VRML

Algumas tarefas não são possíveis de se realizar apenas com VRML, como: acessar e manipular *streams* de entrada e de saída (acessar e escrever dados no disco, escrever interfaces para manipular cenas e utilizar recurso de rede com aplicações distribuídas), acessar arquivos (precisa-se utilizar Java para guardar o estado da cena), visualizar dados em um disco. A comunicação do VRML com o Java é realizada através do *browser* ou do renderizador. Java é integrada à aplicações WWW e pode utilizar de comunicação através de *sockets*. Neste trabalho a comunicação é realizada diretamente com o renderizador, não necessitando do *browser*.

Com os agentes, um usuário pode se comunicar com o ambiente ou com outros agentes por ações, textos ou áudio em um ambiente virtual distribuído através do *browser* sem a preocupação de programar uma série de métodos (*sockets*, protocolos) (LEA, 1996).

2.2 Realidade Aumentada

A Realidade Aumentada (RA) é vista como uma variação da RV. Em RV, o usuário é imerso em um ambiente sintético e não participa do mundo real a sua volta. A RA permite que o usuário veja o mundo real com objetos virtuais superpostos ou combinados com ele. A RA suplementa a realidade, ao invés de substituí-la (RAPOSO, 2005). Para o usuário, os objetos reais e os virtuais coexistem no mesmo espaço. A figura 2 (BERRE, 2002) mostra um exemplo de aplicação de RA onde objetos reais e virtuais compõem um ambiente, com as informações virtuais se sobrepondo ao mundo real.

Figura 2 - Exemplo de visualização de um mundo virtual em RA

A possibilidade de combinar representações virtuais com o mundo real permite dar ao usuário informações adicionais sobre o mundo real que não podem ser obtidas pelos sentidos humanos. As aplicações possíveis para a RA envolvem, por exemplo, o conserto de componentes internos de um sistema mecânico, cujas informações de manutenção são virtualmente mostradas sobre as peças a serem reparadas. Na medicina, a RA também pode ser útil, ao indicar, por exemplo, onde o cirurgião deve fazer a incisão em um paciente. Os militares também têm empregado a RA no intuito de prover à tropa informações vitais sobre seus arredores (SINGHAL, 1999).

Para que as imagens do mundo real e virtual possam ser registradas (fundidas na posição correta) é necessário que a posição e orientação da câmera seja rastreada constantemente (*tracking*). A maioria das aplicações de RA utiliza técnicas de Visão Computacional para realizar o rastreamento.

A realidade misturada (formada pela realidade aumentada e a virtualidade aumentada) pode ser classificada em 6 grupos principais segundo o tipo de *display* utilizado (MILGRAN, 1994). Eles são: i) realidade aumentada com monitor (não imersiva) que sobrepõe objetos virtuais no mundo real; ii) realidade aumentada com capacete (HMD) com visão ótica direta (*see-though*); iii) Realidade aumentada com capacete (HMD) com visão de câmera de vídeo montada no capacete; iv) virtualidade aumentada com monitor, sobrepondo objetos reais obtidos por vídeo ou textura no mundo virtual; v) virtualidade

aumentada imersiva ou parcialmente imersiva, baseada em capacete (HMD) ou telas grandes, sobrepondo objetos reais obtidos por vídeo ou textura no mundo virtual; vi) virtualidade aumentada parcialmente imersiva com interação de objetos reais, como a mão, no mundo virtual. Além disso, uma definição mais precisa de realidade misturada envolve: a combinação do real com o virtual; a interação em tempo real; o posicionamento tridimensional do real e virtual (AZUMA, 2001).

Os capacetes de RA com tecnologia óptica funcionam através da colocação de combinadores ópticos translúcidos e parcialmente reflexivos na frente dos olhos do usuário. Esses combinadores permitem ao usuário enxergar o mundo real através dele e ver imagens virtuais geradas por saídas de vídeo acopladas no capacete e refletidas nos combinadores.

Os dispositivos de RA com tecnologia de vídeo funcionam através da combinação de um capacete ou aparelho similar e uma ou duas câmeras de vídeo montadas na cabeça do usuário. As câmeras de vídeo fornecem ao usuário a visão do mundo real. A imagem da câmera é combinada com a imagem virtual e o resultado é enviado para um visor. A figura 3 (BERRE, 2002) ilustra a tecnologia de vídeo montada junto ao olho do usuário.

Figura 3 - Dispositivo de vídeo para RA

Os ambientes de RA podem também ser baseados em monitor ou utilizar sistemas de projeção, ao invés do capacete de RA. Uma ou mais câmeras, estáticas ou móveis, filmam o ambiente real. A imagem de vídeo do

mundo real e a imagem virtual são combinadas da mesma forma dos capacetes de RA com tecnologia de vídeo, porém a imagem resultante é apresentada no monitor ou nas telas de projeção.

Os sistemas ópticos possuem apenas um fluxo de dados, o virtual, enquanto os sistemas de vídeo devem tratar da captura da imagem de vídeo e da geração da imagem virtual. Os sistemas de vídeo adicionam um atraso na visualização da imagem real da ordem de dezenas de milisegundos. A imagem de vídeo deve estar bem sincronizada com a imagem virtual para que não ocorra distorção temporal. O sistema de vídeo apresenta ainda uma distorção proveniente da câmera que deve ser corrigida. Os sistemas ópticos podem apresentar distorção dependendo das lentes utilizadas, porém são bem menos perceptíveis. Os sistemas de vídeo ainda apresentam desvantagens na resolução da imagem (limitada pela resolução da câmera) e na segurança do usuário (se o sistema falhar, por falta de energia, por exemplo, o usuário perde completamente a visão).

Para o alinhamento dos objetos virtuais com os objetos reais nos sistemas ópticos uma única informação de posição é passada por um dispositivo rastreador que acompanha o movimento da cabeça do usuário. No sistema de vídeo, técnicas de Visão Computacional permitem recuperar a posição e orientação da câmera.

2.2.1 Rastreamento Baseado em Marcas

O ARToolkit foi a biblioteca de rastreamento utilizada para fazer a apresentação dos objetos virtuais trazidos para o repositório de objetos pela estrutura de busca baseada nos agentes junto às imagens do mundo real neste trabalho. É uma biblioteca em linguagem C que permite aos programadores desenvolver aplicações de RA (KATO, 2003). Inclui bibliotecas de rastreamento e disponibiliza o seu código fonte completo, tornando possível portar o código em diversas plataformas ou adaptá-las para resolver as especificações de suas aplicações. Atualmente o ARToolkit pode ser executado nas plataformas SGI, Irix, PC Linux, PC Windows95/98/NT/2000/XP e MacOS X.

A versão atual do ARToolkit suporta tanto RA com visão direta por vídeo ou visão óptica. A visão óptica direta faz uso de modelos de computação gráfica (os objetos virtuais) sobrepostos diretamente às imagens do mundo real percebidas pelo usuário.

O ARToolkit é livre para uso em aplicações não-comerciais e é distribuído com código aberto sob licença GPL. Existem várias versões do *software* disponíveis atualmente para *download*. A versão 2.52 (utilizada neste trabalho) inclui suporte ao VRML com melhorias como a possibilidade de iluminação dos modelos, suporte para *Vision SDK*[9] e ao *DirectShow*[10].

2.2.2 Funcionamento do ARToolkit

O ARToolkit é uma biblioteca de rastreamento de marcas (*patterns*). Uma de suas aplicações, o "simpleVRML.cpp" realiza a apresentação do mundo virtual combinando imagens capturadas por uma câmera (WebCan) misturado com um objeto virtual (arquivo ".wrl"). Para isto usa técnicas de Visão Computacional para calcular o ponto de vista real da câmera em relação a um marcador no mundo real. Há vários passos, conforme mostram as figuras abaixo. Primeiro a imagem de vídeo é transformada em uma imagem binária (P&B) baseada no valor do limiar de intensidade. Depois, busca-se nesta imagem por regiões quadradas. O ARToolkit encontra todos os quadrados na imagem binária, muitos dos quais não correspondem a marcadores de referência. Para cada quadrado, o desenho padrão dentro dele é capturado e comparado com gabaritos pré-treinados chamados de marcas (*patterns)* (Figura 4). Se houver similaridade, o ARToolkit considera que encontrou um dos marcadores de referência. O ARToolkit usa, então, o tamanho conhecido do quadrado e a orientação do padrão encontrado para calcular a posição real da câmera em relação a posição real do marcador. Uma matriz 3x4 conterá as coordenadas reais da câmera em relação ao marcador. Esta matriz é usada

[9] Microsoft Vision SDK é uma biblioteca de programação de para processamento de imagens em tempo real em computadores com sistema operacional Windows. Maiores informações podem ser encontradas em http://research.microsoft.com/projects/VisSDK.

[10] Microsoft DirectShow é uma estrutura genérica para desenvolvimento de sistemas de *streaming* e processamento de imagens. Maiores informações podem ser obtidas em http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwmt/html/ds_wm_faq.asp?frame=true

para calcular a posição das coordenadas da câmera virtual. Se as coordenadas virtuais e reais da câmera forem iguais, o modelo de computação gráfica pode ser desenhado precisamente sobre o marcador real.

Figura 4 - Marca ou *Pattern*

Para executar o ARToolkit, deve-se entrar no diretório “\bin” e executar o arquivo “simpleVRML.exe”. Será carregada a tela de propriedades chamada “Property Sheet”. Nela pode-se configurar a taxa de quadros de saída de vídeo, o espaço de cores e compactação e o tamanho da saída de vídeo. A figura 5 apresenta a tela de “Propriedades de Property Sheet”.

Figura 5 - Tela do “Propriedades de Property Sheet”

Clicando em “OK” nesta tela será carregada uma janela de comandos com informações textuais sobre os arquivos que estão sendo utilizados para gerar a imagem. Em seguida aparecerá a janela de visualização da cena com a imagem real capturada pela câmera e a imagem do objeto virtual carregado sobre a marca conforme mostra a figura 6.

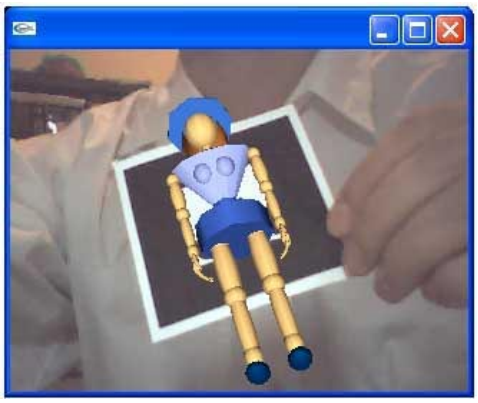


Figura 6 - Imagem do mundo virtual renderizada sobre imagem do mundo real

Com o objetivo de mostrar detalhes da seqüência de funcionamento do ARToolkit, é apresentado abaixo, na figura 7 (CALONEGO JR, 2004) um diagrama com a seqüência de processos do *software*.

Figura 7 - Diagrama descritivo dos passos da detecção dos marcadores e o posicionamento dos objetos virtuais sobre os marcadores detectados na cena do mundo virtual

2.2.3 Associação da imagem real com o objeto virtual

Para inserir uma imagem virtual (objeto virtual) junto à imagem real da câmera, cada marca é associada com uma imagem (arquivo ".wrl"). O mapeamento entre a imagem e o seu *pattern* (marca) é gravado no diretório "\bin\Data\vrml_data". O código abaixo representa um exemplo do arquivo "vrml_data". Pode-se observar neste código as referências realizadas entre o arquivo "Wrl\robot.dat" e o arquivo "Data\patt.hiro". A associação entre os arquivos no ARToolkit é apresentada na figura 8 (CALONEGO JR, 2004).

```
#the number of patterns to be recognized
2
#pattern 1
VRML   Wrl/robot.dat
Data/patt.hiro
80.0
0.0 0.0

#pattern 2
VRML   Wrl/cone.dat
Data/patt.kanji
80.0
0.0 0.0
```

Por sua vez, o arquivo “robot.dat” (apresentado abaixo) faz referência ao arquivo “Wrl\robot.wrl”, que é o objeto virtual que será carregado na cena sob o marcador “patt.hiro”.

```
Wrl/robot.wrl
0.0 0.0 0.0  # Translation - x,y,z from center of tracking pattern
0.0 0.0 0.0 0.0     # Rotation angle + axis, eg 90.0 1.0 0.0 0.0
1 1 1        # Scale x,y,z

LIGHT DIR           # Direction Light - specified in VRML format
0.5               # Ambient
1.0               # Intensity
1.0  1.0  1.0     # RGB
0.0  0.0  1.0     # Direction
```

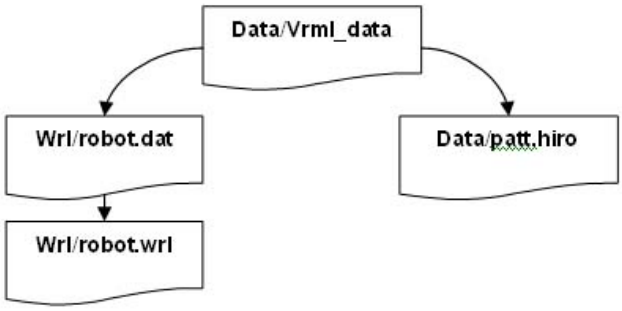


Figura 8 - Associação entre arquivos no ARToolkit.

Detalhes sobre a instalação podem ser encontrados no anexo 4 deste trabalho. Detalhes completos sobre os aspectos técnicos no ARToolkit podem ser encontrados em (CALONEGO JR, 2004).

2.3 Agentes Móveis em Sistemas Distribuídos

Os sistemas de comunicação tiveram seu início marcado com a utilização de desenhos nas paredes das cavernas e o uso de tambores, utilizados nas florestas da África, América e China. Passou posteriormente, já com o uso da eletricidade e eletrônica na construção de equipamentos como o

telégrafo de Samuel Morse (1838), o telégrafo sem fio de Marconi (1899) e o telefone de Graham Bell (1876). A construção dos primeiros microcomputadores (1972) formou o sistema base para a utilização em massa, duas décadas mais tarde, das redes de computadores e os atuais sistemas baseados na World Wide Web (WWW).

Inicialmente a WWW era uma interface baseada em texto. Posteriormente a *Hypertext Markup Language* (HTML) forneceu melhorias na interface com o usuário. Em 1993 o visualizador “Mosaic” foi lançado nos EUA e, além de texto, era possível colocar imagens nos documentos. A questão que surgiu era até onde esta interface 2D poderia resistir aos avanços tecnológicos.

Nos anos 90, através da abertura da Internet e da massificação do uso de rede de computadores, fundiram-se o uso dos computadores aos sistemas de telecomunicações. Através da Internet, as redes de computadores começaram a oferecer serviços com acesso a informações remotas, comunicação pessoal e diversão interativa. Acredita-se que esta evolução só se compara à invenção da imprensa. Houve a instalação das redes de telefonia em escala mundial, a evolução do rádio, da televisão, da indústria de computadores, o lançamento de satélites e a popularização da Internet.

A evolução das redes culminou no desenvolvimento de um mundo paralelo denominado ciberespaço (*cyberspace).* O termo ciberespaço é amplamente utilizado, e deve ser visto como um espaço real fora do espaço físico, como um mundo imaginário existente dentro dos computadores e das redes de comunicação (GREENOP, 1999). É utilizado como estrutura base para tecnologias como: realidade virtual, agentes, portais Web, *games*, comunidades e negócios.

A Internet tornou-se uma estrutura essencial para organizações e pessoas. É uma conexão de múltiplas redes que se comunicam, oferecendo serviços sobre um conjunto de protocolos de comunicação: TCP/IP (*Transmission Control Protocol/Internet Protocol*). Seu tráfego é realizado

através de linhas de comunicação chamadas *Backbones*[11] e conectadas por roteadores e comutações.

Atualmente, várias tecnologias são utilizadas no processo de comunicação dos sistemas em rede. A tecnologia de agentes móveis, objeto de estudo deste trabalho vem se mostrando útil para desenvolvimento de aplicações em diversas áreas. Entre elas destacam-se: (i) gerenciamento de sistemas e redes; (ii) acesso e gerenciamento de informações; (iii) sistemas colaborativos; (iv) comércio eletrônico; (v) realidade virtual entre outras áreas. Neste capítulo, discute-se as características de agentes móveis como modelo de programação para um sistema distribuído.

2.3.1 Sistemas Distribuídos

Uma definição para sistemas distribuídos é: um conjunto de componentes: *hardware* (*hosts*) e *software* através de uma infra-estrutura de comunicação, que cooperam e se coordenam entre si através da troca de mensagens, e eventualmente, pela partilha de memória comum, para execução de aplicações distribuídas (COULOURIS, 2001).

Entre os principais objetivos de se construir sistemas distribuídos pode-se citar: economia, poder de processamento, velocidade, distribuição da aplicação, confiabilidade e crescimento incremental (TANENBAUM, 2003).

No contexto de desenvolvimento de software, várias técnicas são utilizadas para oferecer programação distribuída como *sockets* e “Remote Procedure Call” (RPC). Com o surgimento de tecnologias de orientação à objetos, plataformas distribuídas, reutilização de código, portabilidade entre outros, foram desenvolvidas tecnologias de objetos distribuídos das quais se destacam: “Common Object Request Broker Architecture” (CORBA), “Distributed Component Object Model” (DCOM), “Remote Mothod Invocation” (RMI), “Parallel Virtual Machine” (PVM) e “Message Passing Interface” (MPI) (ROQUE, 1999). A figura 9 ilustra algumas plataformas para desenvolvimento de sistemas distribuídos e modelos de aplicação.

[11] *Backbone* – Linhas de comunicação que representam as principais ligações das redes de computadores.

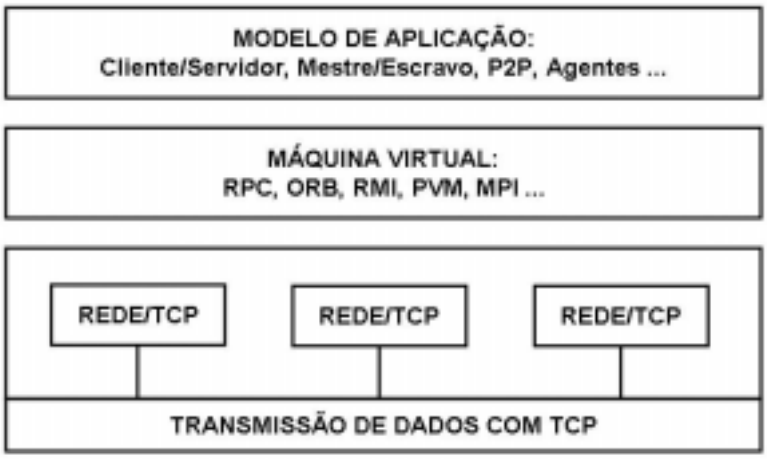


Figura 9 - Plataformas de comunicação em sistemas distribuídos

O primeiro modelo de programação distribuída foi o modelo de passagem de mensagem (*Message Passing*) (MINAR, 1998). A idéia deste modelo é que se um programa deseja se comunicar com outro, ele realiza esta tarefa através do envio de uma mensagem para o outro programa. A passagem de mensagem forma o coração de todos os sistemas de rede e é a base de funcionamento dos agentes móveis.

Trabalhar com objetos distribuídos apresenta um caminho para descrever programas na rede. No entanto, sistemas de objetos distribuídos são estáticos. A utilização de agentes adiciona autonomia, pró-atividade e adaptabilidade aos objetos do sistema (MINAR, 1998).

2.3.1.1 Transmissão de Dados em Sistemas Distribuídos

A transmissão de dados em Sistemas Distribuídos pode utilizar diferentes padrões. Será discutido com detalhes apenas o mecanismo *socket* por ser a base da plataforma de desenvolvimento do protótipo.

Sockets são programas que atuam como uma porta de um canal de comunicação. Permite que processos troquem mensagens com outros processos, local, ou remotamente (SINGHAL, 1999). O termo é usado para *softwares* usados na Internet, para especificar a conexão entre o *software* da camada de aplicação (HTTP, FTP, etc) e a camada de transporte (TCP, UDP).

Na comunicação local por *sockets* (Figura 10), são utilizados programas com funções específicas para esta tarefa (trocar mensagens) controladas pelo Sistema Operacional. Este tipo de comunicação é utilizado por processos de usuário ou sistemas que são executados em um mesmo *hardware*. Permite ao programador definir como e quando será efetuada a comunicação entre processos (troca de mensagens).

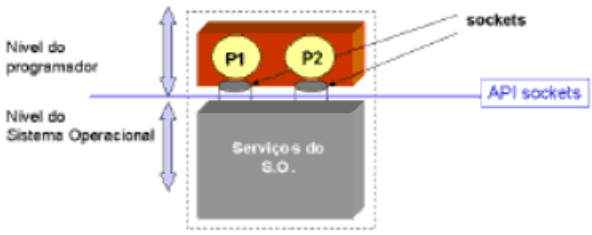


Figura 10 - *Sockets* em comunicação local

Para se realizar uma comunicação remota, são utilizados programas que possuem funções específicas para troca de mensagens entre processos. Além da interação do Sistema Operacional, pode-se utilizar mecanismos de comunicação por rede. Este tipo de comunicação é efetuado por processos que executados em *hardwares* distintos, ou em sistemas alocados fisicamente em locais diferentes (Figura 11).

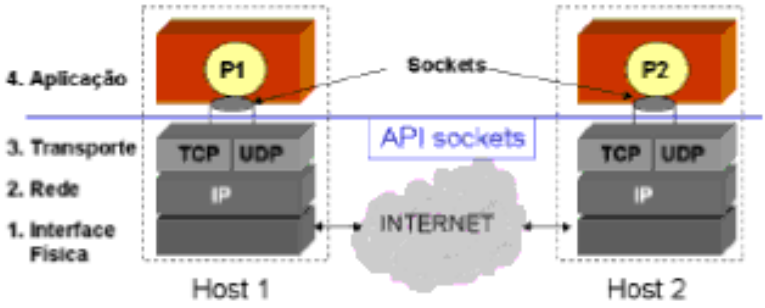


Figura 11 - *Sockets* em comunicação remota

Sockets constitui uma importante ferramenta de comunicação. É importante ressaltar que este tipo de implementação só se torna realmente efetivo se os clientes que a utilizam e, especialmente os servidores, forem construídos de forma robusta. Para uma comunicação mais complexa, apresentam-se outros mecanismos de comunicação.

Apesar do mecanismo de *sockets* oferecer comunicação robusta, é necessário cuidar da heterogeneidade dos sistemas de computação, que devem, de algum modo, interagir suavemente a despeito de suas diferenças (DEITEL, 2005). Um dos desafios primordiais no projeto de sistemas distribuídos é gerenciar a comunicação entre os computadores. Os mecanismos RPC, RMI, CORBA, DCOM, PVM e MPI dentre outros, foram criados para garantir a interoperabilidade nos sistemas. Segue uma breve descrição destes mecanismos.

"Remote Procedure Call" (RPC) ou Chamada à Procedimento Remoto: criada para proporcionar uma abordagem estruturada, de alto nível, à comunicação interprocessos em sistemas distribuídos. Permite que um processo que esteja em execução em um computador invoque um procedimento de um processo que esteja em execução em outro computador. O computador que emite a chamada ao procedimento remoto envia seus parâmetros pela rede para a máquina onde reside o procedimento onde será executado o processo. O resultado é transmitido ao cliente (DEITEL, 2005).

"Remote Method Invocation" (RMI) Invocação à Método Remoto: é a versão RPC para Java. Habilita um processo Java que está executando em um computador, invoca um método de um objeto em um computador remoto usando a mesma sintaxe de uma chamada a método local. Permite que programadores que utilizem a linguagem Java implementem sistemas distribuídos sem ter de programar *sockets* explicitamente. O RMI usa a JVM para criar a máquina virtual (DEITEL, 2005).

"Common Object Request Broker Architecture" (CORBA) é uma especificação padrão de sistemas distribuídos com ampla aceitação, concebida no início da década de 1990 pelo OMG. É um padrão aberto, elaborado para habilitar interoperação entre programas em sistemas heterogêneos. Similar a RMI, suporta objetos com parâmetros ou valores de retorno em procedimentos remotos durante comunicação interprocessos. A sua vantagem é que é independente de linguagem e plataforma. Utiliza o ORB (*Object Request Broker*) para criar a comunicação com JVM (DEITEL, 2005).

“Distributed Componente Object Model” (DCOM): foi desenvolvido na década 90 e incluído no Windows95, é a extensão distribuída do modelo de objeto componente (COM – “Componente Object Model”) da Microsoft para facilitar o desenvolvimento em componente em ambiente Windows. Como CORBA, DCOM é acessado via interfaces, o que permite que objetos DCOM sejam escritos em várias linguagens de programação e em diversas plataformas (DEITEL, 2005).

“Parallel Virtual Machine” (PVM): é um conjunto de ferramentas de *software* e bibliotecas que permite que uma rede de computadores heterogêneos possa ser utilizada para a realização de computação paralela ou concorrente. Essas máquinas formam uma única máquina virtual que pode ser composta de estações de trabalho, (ou servidores) de vários tipos e em lugares físicos diferentes. É independente de linguagem e baseado na idéia de que uma aplicação consiste de várias tarefas, cada uma responsável por uma parte do problema (CACERES, 2006).

“Message Passing Interface” (MPI): baseado em troca de mensagens, surgiu em 1994 sendo eficiente para executar processos em máquinas diferentes, especificando somente o funcionamento lógico das operações. Objetiva a portabilidade de sistemas e estabelece um padrão de troca de mensagens fornecendo um conjunto de rotinas que podem implementar o *hardware* em sistemas paralelos aumentando sua escalabilidade (CACERES, 2006).

2.3.1.2 Modelos de Programação na Computação Distribuída

Modelos de programação estão diretamente relacionados com a tecnologia do *software*. Segundo Deitel (2005) podem ser:

- **Cliente-servidor**: o servidor oferece um conjunto de serviços que fornecem acesso a recursos. A computação é realizada pelo servidor que recebe o pedido do cliente, computa os dados e devolve o resultado. Quando o cliente precisa de recursos, solicita e recebe do servidor a resposta através da rede. A maioria dos sistemas distribuídos tem se baseado nesse modelo.

Ele é suportado por tecnologias como as RPC, CORBA, RMI e DCOM.

- ***Peer to Peer*:** as aplicações são implementadas de forma descentralizada possuem um mecanismo de troca de mensagens independente de servidores. É uma tecnologia híbrida, pois também pode fazer uso de servidores para realizar o mapeamento e a autenticação de usuários.

- **Execução remota**: um componente A possui o código para executar os serviços que necessita, mas faltam os recursos necessários. O componente envia o código da tarefa para um componente B (em um outro local), onde estão disponíveis os recursos necessários à execução do serviço. O componente B efetua o processamento e envia os resultados obtidos para o componente A. Um exemplo é o código gerado por um processador de textos *PostScript* e executado pela impressora.

- **Código por demanda**: o cliente tem recursos, mas não sabe trabalhar sobre eles, então importa código de um serviço e utiliza este código para operar sobre os recursos. O cliente não precisa ter o código instalado antes de começar a processar, quando o cliente precisar de alguma funcionalidade, ele importa o código de um servidor. São utilizados *applets*, executados no lado do cliente e *servlets*, no lado do servidor.

- **Agentes móveis**: a aplicação (agente) migra de um computador para outro. Uma aplicação não precisa de servidores enviando-lhe código ou resultados, cada agente tem acesso aos recursos da maneira que deseja.

No modelo cliente-servidor os componentes computacionais e o código das aplicações não podem ser transferidos após sua criação. Já os modelos de execução remota e código sob demanda fornecem a capacidade de mobilidade ao código processado pelos componentes, permitindo a execução de código em uma localidade remota. Os agentes móveis fornecem mobilidade, não somente ao código processado, mas também aos próprios componentes que efetuam a execução do serviço.

O modelo dos agentes móveis se diferencia dos demais modelos porque envolve a mobilidade de um componente computacional completo para o processamento de serviço. O componente (código) é movido para um local remoto, junto com seu estado, para a execução. É uma tecnologia mais flexível que as outras formas de mobilidade de código (BREWINGTON, 1999).

Dentre as várias definições de agentes, duas são consideradas neste trabalho. Para Lange (1998) são programas que auxiliam pessoas e agem em seu benefício. Segundo Pereira (2000), agente é uma entidade que percebe seu ambiente através de sensores e atua neste ambiente através de reagentes. A figura 12 ilustra o conceito de agente que se utiliza de sensores para agir e reagir a ações sob um ambiente.

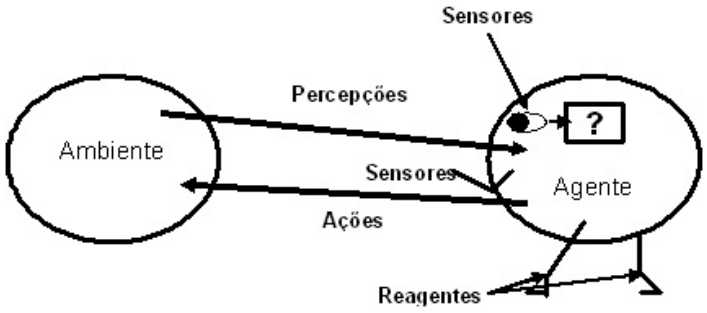


Figura 12 - Interação de agentes com o ambiente através de sensores

As aplicações baseadas em agentes podem ser compostas por agentes cooperativos. Estes agentes podem executar tarefas em conjunto, ou seja, realizando tarefas em cooperação uns com os outros. Os agentes são pesquisados em várias áreas da Ciência da Computação. Em Sistemas Distribuídos, os agentes móveis podem migrar de uma máquina para outra em uma rede (LANGE, 1998, FREITAS, 2002).

Os Agentes têm atraído a atenção dos desenvolvedores por: (i) constituírem um meio de introduzir inteligência nas interfaces, possibilitando que usuários, tirem o máximo de proveito das aplicações; (ii) podem ser utilizados para personalizar aplicações dos usuários; (iii) podem gerenciar a recuperação, disseminação e pesquisa de informações através das redes de computadores; (iv) são aplicáveis de várias maneiras em sistemas de comércio

eletrônico, agindo em favor de consumidores e fornecedores e (v) não beneficiam apenas os indivíduos, podendo auxiliar no funcionamento mais eficiente de organizações como um todo.

Apresenta-se abaixo algumas das características que tornam um elemento de *software* um agente (KNAPIC, 1998):

- **Autonomia**: agentes operam sem a intervenção direta de humanos ou outros agentes e possuem um controle sobre seu estado;
- **Habilidade Social:** agentes interagem com outros agentes (ou humanos) por alguma linguagem de comunicação;
- **Reatividade:** agentes percebem seu ambiente e respondem a mudanças que ocorrem neste ambiente;
- **Mobilidade:** movem-se para outros ambientes;
- **Continuidade Temporal:** apresentam ações contínuas;

Com base nas características dos agentes, pode-se identificar outras características que tornam uma aplicação adequada para a utilização de agentes (SCHIAVONI, 1998):

- **Adaptação**: o agente necessita desenvolver habilidades para executá-la aprendendo melhores ou novos meios;
- **Pesquisa**: o agente deve considerar uma grande quantidade de possíveis soluções, escolhendo a mais adequada de acordo com sua experiência;
- **Assincronia**: a tarefa tem um intervalo significativo entre seu início e fim que pode ser dividido no tempo de processamento das informações ou a falta de informações vitais em um determinado momento.
- **Demonstração**: envolve aprendizado e treinamento. Inclui ensinar os usuários a usar ferramentas de *software* e fornecer explicações;
- **Ajuda**: a tarefa requer grau de cooperação entre o usuário e o agente. O agente pode dar “dicas" sobre como utilizar melhor os recursos do sistema;

Analisando as definições e características sobre agentes, buscou-se neste capítulo, situar o leitor no domínio de agentes, para que, no próximo capítulo possa se fazer uma descrição dos agentes móveis, objeto de estudo e de implementação desta dissertação.

2.3.2 Agentes Móveis

O termo “Agente Móvel” vem de NWANA (1998), que categoriza os agentes em:

- **Colaborativos:** enfatizam autonomia e cooperação e podem aprender. As características fundamentais destes agentes incluem: autonomia, habilidade social, responsabilidade e pró-atividade;

- **De Interface**: enfatizam autonomia e aprendizado para executar tarefas. Um agente de interface é uma espécie de assistente pessoal que colabora com o usuário em seu ambiente de trabalho;

- **Móveis:** são capazes de atuar sobre uma rede de comunicação ou de telecomunicações, interagindo em ambientes heterogêneos, buscando e colhendo informações entre uma de suas funções;

- **De Informação/Internet:** executam o papel de administrar, manipular ou coletar informações de fontes distribuídas;

- **Reativos:** respondem a um estímulo do estado presente do ambiente no qual estão inseridos. São agentes que interagem com outros agentes;

- **Híbridos:** são aqueles que combinam duas ou mais características de agentes mesclando-as em apenas um agente;

- **Heterogêneos:** é um conjunto de dois ou mais agentes, podendo ser híbridos que interagem de maneira integrada;

- **Inteligentes:** são agentes que possuem características múltiplas e utiliza-se de técnicas de inteligência artificial.

Agentes móveis constituem um novo modelo de programação distribuída que oferece uma maneira alternativa para o projeto de implementação de aplicações distribuídas. Este modelo se fundamenta na idéia de que, em vez de transmitir somente dados entre computadores de uma rede, como em sistemas distribuídos tradicionais, o código executável também pode ser transmitido (NAGAMUTA, 1999 e LANGE, 1998, NUNES, 2002).

Agentes móveis são entidades de software autônomas, capazes de viajar através de uma rede e executar tarefas em nome do usuário (LANGE,1998; PAPASTAVROU, 2000; EID, 2005). Outra definição mais detalhada é: agentes móveis são programas que podem ser enviados de um computador para outro para ser executado. Ao chegar ao computador remoto, ele apresenta suas credenciais e obtém acesso para os serviços e dados locais (LANGE, 1998).

O primeiro modelo de agente móvel foi introduzido em 1995, denominado “Agent TCL” (Dartmouth College – Hanover). Nele a execução de um agente é suspensa em qualquer ponto e seu código e estado são transportados para outra máquina onde é terminado seu processamento.

Os agentes móveis oferecem uma estrutura para o desenvolvimento de serviços de rede, enfatizando que a agregação de todas as suas características torna o modelo poderoso, se comparado aos métodos alternativos.

Para que um agente seja despachado ou chegue a locais da rede é necessário que haja um sistema para enviá-lo, recebê-lo e interpretar seus comandos e funções. As plataformas constituem os locais (contextos) para onde os agentes irão migrar. Cada plataforma deve estar instalada em um *host* onde as aplicações irão trocar informações e serviços.

A segurança é um dos problemas relacionados a agentes móveis. Uma das principais questões é como criar mecanismos que garantam a proteção e impeçam a ação de agentes maliciosos. Outro problema é a padronização: as diferenças entre arquiteturas e implementações dos sistemas de agentes móveis impedem a interoperabilidade e a expansão da tecnologia de agentes móveis. Algumas organizações promovem padronizações para a

tecnologia de agentes móveis. Duas delas são: MASIF (*Multi Agent System Interoperability Fundation*) do grupo OMG (*Object Management Group*) e a FIPA (*Fundation for Intelligent Pysical Agents*) (LANGE, 1998).

A OMG tem desenvolvido trabalhos a respeito da padronização de agentes para interoperabilidade (NUNES, 2002). O MASIF possui como objetivo permitir que um agente móvel possa deslocar-se entre agências com perfil semelhante (tipo de agência, de autenticação e método de serialização) através de um conjunto de interfaces CORBA normalizadas. Contrariamente, a FIPA se propõe a padronizar os sistemas de agentes a respeito da comunicação entre agentes, até mesmo na escolha da linguagem. A FIPA define a comunicação entre sistemas de agentes em termos da linguagem de comunicação entre os agentes, e o MASIF define a interação de agentes através de mecanismos como RPC e RMI.

Segurança e interoperabilidade são assuntos importantes dentro do domínio de agentes móveis. Porém, a proposta desta dissertação é apresentar uma aplicação que faça uso de uma plataforma de agentes, como uma maneira alternativa de transmitir códigos executáveis em uma rede, independentemente de problemas de segurança ou de interoperabilidade.

Os agentes, quanto a sua mobilidade, podem ser estáticos ou móveis. Agentes estáticos (estacionários) são aqueles que executam operações somente em um mesmo endereço (*host*). Agentes móveis são aqueles que não possuem limitação de local de execução, e podem migrar entre endereços (*hosts*) dentro de uma rede (LANGE, 1998). O grande interesse em usar agente móvel é que, após a sua criação em um ambiente de execução, ele pode transportar o seu código e estado para um outro ambiente de execução e retomar a sua execução do ponto imediatamente antes da migração. O termo código se refere ao código da classe (no contexto de orientação a objetos) necessário para que o agente possa ser executado. O estado corresponde aos valores dos atributos do objeto agente, que caracteriza o seu estado de execução.

Apresentam-se a seguir sete razões para utilização de agentes móveis (LANGE, 1998):

- **Redução do tráfego na rede:** grandes volumes de dados armazenados em diferentes locais podem ser processados em sua própria localidade;
- **Redução da latência da rede:** agentes podem ser disparados para atuar localmente e executar diretamente as interações;
- **Encapsulamento de protocolos:** agentes podem mover-se para locais remotos a fim de estabelecer canais baseados em protocolos proprietários;
- **Execução assíncrona e autônoma:** agentes podem mover-se em redes, e mesmo que estas tenham sua comunicação interrompida, continuam sua execução e quando possível retornam autonomamente ao servidor;
- **Adaptação dinâmica:** agentes móveis têm a capacidade de reconhecer seu ambiente de trabalho e reagir autonomamente a alterações;
- **Heterogêneos:** possuem protocolos independentes e são baseadas em Java, suas execuções podem ser realizadas em ambientes diversos;
- **Robustos e tolerantes a falhas:** podem ficar em *stand-by* caso uma máquina seja desligada ou apresente problemas, e quando o sistema retorna a normalidade, ele continua sua execução.

Outras vantagens ainda são citadas por Mohamad Eid et al (EID, 2005) no que diz respeito ao uso de agentes móveis. São elas:

- **Economia de espaço:** o código e o estado dos agentes móveis não precisam permanecer armazenados em *hosts* que eles rodam;
- **Interação com sistemas de *real-time*:** podem ser instalados em sistemas *real-time* para prevenir atrasos causados pelo congestionamento na rede;
- **Customização de clientes:** podem ser customizados por usuários finais para operações específicas.

Essas características oferecem um novo modelo para programação em redes de computadores e programação distribuída através de suas características como a mobilidade e a autonomia.

Em contraste com os modelos de programação distribuída, nos quais apenas os dados são movidos de *host* para outro (como RPC), os agentes móveis possibilitam a mobilidade do código executável. Isto oferece diversas vantagens conforme listadas acima na descrição das sete razões enumeradas por Danny Lange e Mitsuru Oshima sobre o uso de agentes móveis (LANGE, 1998).

De modo geral, agentes móveis oferecem um modelo de execução uniforme para a programação distribuída, incorporando formas de troca de mensagens síncronas ou assíncronas, transferência de objetos e interação por intermédio de chamadas de objetos estacionários e móveis.

A tecnologia de agentes móveis é uma abstração na comunicação cliente/servidor. O mecanismo de comunicação mais comum usado por agentes móveis é a troca de mensagens realizada independentemente de o mecanismo ser: RPC, RMI ou CORBA.

Como observado anteriormente, os agentes móveis podem ser heterogêneos quando se referem a coleções de dois ou mais agentes com diferentes arquiteturas de agente. Podem ser também colaborativos quando se comunicam, cooperar e interoperar com outros agentes. O requisito chave para essa interoperabilidade é uma linguagem de comunicação que permite que agentes de diferentes tipos possam se comunicar uns com os outros.

2.3.2.1 Plataformas para construção de agentes

Os programadores que desejam implementar agentes, encontram atualmente na Internet algumas plataformas de desenvolvimento de agentes de uso livre. São mencionadas abaixo algumas plataformas e algumas de suas características:

- ***Aglets, Mole*, *Voyager, Odyssey, Concórdia, Grasshopper, Sumatra*, *MuCode, JavaSeal*, *Jamp, JAX, AgentX, Gossip, AgentSpace, Jumping Beans e BeeAgent***. Estas plataformas de desenvolvimento de agentes móveis são baseados na tecnologia Java e trazem suporte à mobilidade de código;

- ***Agent Tcl***, **Ara** e **TACOMA**. Estas plataformas são exemplos de sistemas de agentes móveis baseados em outras linguagens como *Tcl, Scheme e Python*;

- ***MadKit, AgentBuilder*** *e* ***Zeus.*** São plataformas de desenvolvimento de sistemas multi-agente que oferecem aos seus utilizadores classes de agentes equipadas com protocolos de coordenação;

- **JADE** (*Java Agent Development Framework*) é um *middleware* (ambiente de execução) de agentes que implementam uma plataforma de agentes e um *framework* de desenvolvimento. Tem objetivo de simplificar o seu desenvolvimento;

- ***Excalibur***. É uma arquitetura para desenvolvimento de agentes autônomos que posam ser integrados em jogos complexos de computador;

- ***DirectIA, iGEN, Intelligent Agent Factory, JAM*** *e* ***UMPRS*** são bibliotecas usadas para criação de agentes inteligentes com mecanismos de aprendizagem;

- ***MS-Agent*** *(Microsoft Agent)* é uma plataforma voltada para a criação de agentes de interface para ambiente Windows baseada na linguagem *ActiveX*.

Apresenta-se abaixo a tabela 1 contendo os endereços (URLs) de algumas das principais plataformas de agentes citadas acima. Em (BERGAMASCHI, 2002 e LEE, 2000) pode-se encontrar informações e *links* sobre plataformas e *kits* de desenvolvimento de agentes.

Tabela 1 – URLs de sites com informações sobre as principais plataformas de agentes

Plataforma	URL
AgentBuilder	http://www.agentbuilder.com
AgentX	http://www.iks.com
Aglets	**http://www.trl.ibm.com/aglets, http://aglets.sourceforge.net/mantainer.html**
Ara	http://wwwagss.informatik.uni-kl.de/Projekte/Ara/index_e.html
BeeAgent	http://www2.toshiba.com.jp/rdc/beeagent/index.html
Concordia	http://www.meal.com/projects/concordia
D´Agent (Tcl)	http://cs.dartmouth.edu/~agent
Excalibur	http://www.ai-center.com/projects/excalibur
Gossip	http://www.tryllian.com
Grasshoper	http://www.cordis.lu/infowin/acts/analysys/products/thematic/agents/ch4/ch4html
JADE	http://jade.tilab.com
JavaSeal	http://www.jseal2.com
JumpingBeans	http://www.JumpingBeans.com
Mole	http://www.iiia.csic.es/AMEC/Brussels/abstract6.html
MSAgent	http://msdn.microsoft.com
MuCode	http://mucode.sourceforge.net
Tacoma	http://tacoma.cs.uit.no
Voyager	http://recursionsw.com/voyager.htm
Zeus	http://labs.bt.com/projects/agents/zeus

2.3.2.2 Tendências em aplicações com agentes móveis

Apresenta-se neste tópico, uma análise das tendências de utilização de agentes móveis, recém compiladas em um artigo publicado em novembro de 2005 na biblioteca digital ACM com o título de *Trends in Mobile Agent Applications* (EID, 2005). Neste documento foram analisados cerca de 220 artigos publicados entre 1989 e 2004. Dos artigos analisados, 13% fazem parte do domínio de interesse deste trabalho: Busca e Filtragem de Informações. A tabela 2 apresenta uma estatística sobre as áreas pesquisadas.

Tabela 2 – Utilização de agentes móveis em áreas de aplicação segundo artigos publicados

Área	%
1- Monitoramento e Gerenciamento de Redes (Aplicações para SNMP 21%, ATM 23%, TupleSpaces 6%, Gerenciamento avançados de rede 12%, Gerenciamento de Recursos 12%, Detecção de falhas 6% e Roteamento 20%)	**36%**
2- Busca e Filtragem de Informações (Gerenciamento de Sistemas de Banco de Dados Distribuídos 29%, Gerenciamento de Recursos 14% Busca e Filtragem 21%, Recuperação e Reunião de Informações 36%)	**13%**
3- Multimídia (Sistemas Multimídia Distribuídos e Videoconferência 100%)	**10%**
4- Internet (Integração com Servidores Web 43%, Monitoramento de Serviços e Recuperação de Informações 57%)	**8%**
5- Detecção de Intrusões (Métodos de Detecção 35%, Segurança 6%, Ambientes e Arquiteturas 29%, Relatórios e Respostas 12% e Coleção de Dados 18%)	**15%**
6- Telecomunicações (Personalização de Serviços 50% e Sistemas de comunicação de redes sem fio 50%)	**07%**
7- Aplicações Militares (Operações Táticas 100%)	**04%**
8- Outros (Interface de usuário, Manufatura, Simulação e Monitoramento e Desenvolvimento de Aplicações)	**07%**

Entre os artigos analisados e citados no trabalho, apresentou-se a utilização da plataforma “Aglets” (que será utilizada neste trabalho) em diversos domínios. Eles são: (i) Monitoramento e Gerenciamento de Redes (Gerenciamento de Recursos e Detecção de Falhas); (ii) Busca e Filtragem de Informações (Gerenciamento de Sistemas de Banco de Dados Distribuídos, Recuperação e Reunião de Informações); (iii) Internet (Integração com servidores Web); (iv) Detecção de Intrusões (Métodos de detecção).

Verificou-se, através da análise desta pesquisa que 13% dos artigos analisados fazem uso de agentes em busca e filtragem de informações e observa-se que, dentro desta área, 57% dos artigos correspondem ao domínio específico deste trabalho, pois estão relacionados a busca e recuperação de informações em uma rede. Na categoria de busca, foi encontrado dois trabalhos que fizeram uso da plataforma “Aglets”.

2.3.3 Agentes móveis em sistemas distribuídos

Atualmente VRML e X3D são as tecnologias utilizadas para construção de ambientes virtuais distribuídos (NVEs – *Network Virtual Environments*). Junto à infra-estrutura oferecida pelas redes de alta velocidade como RDSI (Rede Digital de Serviços Integrados) e pelos modens ADSL (*Assinchronous Digital Serial Line*) abrem-se novos tipos de aplicações como o CVE (*Collaborative Virtual Environments*), também considerados NVEs (FABRE, 2003).

A VRML estabeleceu o padrão para descrição de cenas 3D em WWW, mas não suporta sozinho mundos virtuais multi-usuários. A utilização de outras linguagens de programação como Java e de tecnologias como a de agentes passa a ser uma extensão das ferramentas para construção de NVEs. Os agentes móveis oferecem uma interação com o usuário e definem uma linguagem para ambientes distribuídos. Eles são uma ferramenta para criar códigos autônomos e dinâmicos em NVEs (FABRE, 2003).

Pesquisas estão sendo realizadas para desenvolver linguagens e tecnologias que suportem agentes móveis, baseados em processos leves (*lightweight process*). Java e JavaScript suportam, por exemplo, redução de requerimento de largura de banda (*Bandwidth*), alcançado com técnicas como *Dead reckoning;* otimização da latência na rede, onde o agente migra e realiza sua execução em outra máquina, e balanceamento de carga, que é explicado pelo fato de os agentes não serem executados em um único *host.*

As possibilidades de uso de agentes em NVEs parecem grandes, porém como os agentes são uma área recente de desenvolvimento, documentos a respeito desta tecnologia são raros. Este trabalho aborda o uso de uma plataforma de agentes móveis para recuperar objetos virtuais e inseri-los em uma cena com possibilidades de interação em NVEs.

Este capítulo apresentou uma visão geral da tecnologia de agentes, onde se observa que seu conceito esbarra a todo instante na definição de *softwares* tradicionais. No caso da utilização de agentes móveis em ambientes

distribuídos, as vantagens se tornam evidentes, principalmente em ambientes distribuídos heterogêneos, com processamento em tempo-real onde as exigências de tempo, necessidade de comunicação persistente e problemas de largura de banda são vitais. A programação remota utilizada pelos agentes permite diminuir o tráfego na rede, pois o agente enviado ao hospedeiro leva consigo o código e o estado de execução com seus dados. Outra justificativa para o uso de agentes é a natureza dos atuais sistemas distribuídos (Web).

Através de propriedades fundamentais, conclui-se que um agente pode ser definido como um *software* que utiliza a comunicação para negociar e coordenar a transferência de informações. É atualmente objeto de estudo nas áreas mais recentes de pesquisa como, por exemplo, as telecomunicações, sistemas distribuídos, realidade virtual e inteligência artificial.

2.3.4 Agentes Baseados em *Applets*

A tecnologia “Aglets” representa o próximo passo na evolução de programas executáveis na Internet, introduzem programas que podem ser transportados em uma rede de computadores com seu estado de informação. (LANGE, 1998; VELLOSO, 2002; NUNES, 2002).

O *Aglets Software Development Kit* (ASDK) introduz o conceito de Aglets (agente + *applet*) que é um programa em execução que pode mover-se de uma máquina (*host*) para outra em uma rede de computadores. A tecnologia “Aglets” estende o modelo de código móvel de *applets* o seu estado de execução. É um ambiente para programação de agentes móveis baseado na linguagem Java desenvolvido pelo *IBM Tokyo Research Laboratory* e que teve seu desenvolvimento continuado por um grupo do *SourceForge*. O professor Luca Ferrari (FERRARI, 2004) iniciou a formação de um novo grupo de desenvolvimento desta plataforma em virtude de um grande interesse e crença na plataforma “Aglets”. Mais informações podem ser encontradas nos endereço aglets.sourceforge.net, ou endereços de e-mail do grupo de desenvolvedores: aglets-developer@lists.sourceforge.net; aglets-commit@lists.sourceforge.net; aglets-users@list.sourceforge.net.

A origem da palavra Aglet vem do termo *lightweight agent* similar a palavra *applet* que deriva de *lightweight application.* Aglet é um agente móvel em Java que possui autonomia e pode definir seu itinerário. Quando um Aglet é executado, ele pode suspender a sua execução em um dado momento, transferir-se para outro *host* e retornar a sua execução do ponto que estava antes da migração. Um Aglet possui: estado, comportamento e identidade, Diferente de um objeto, um Aglet também possui uma localização (contexto).

Um aglet utiliza uma interface para: obter informações do ambiente e para enviar mensagens a este ambiente e a outros aglets ativos. Esta interface oferece meios para o gerenciamento dos aglets no ambiente com segurança contra aglets, ou códigos maliciosos. O contexto “Aglets” é criado por um servidor que tem o objetivo de monitorá-los na rede. Este contexto cuida dos aglets que estão sendo enviados e recebidos no *host* e oferecem um recurso gráfico para visualização. No ASDK este servidor é chamado de Tahiti. Para gerenciar, visualizar, definir segurança e inicializar um aglet, é necessário que este contexto esteja em execução (esteja rodando) em um *host*.

Os aglets comunicam-se através de troca de mensagens. Elas podem ser síncronas ou assíncronas e utilizam comunicação *multicast*. As mensagens são controladas por um gerenciador de mensagens.

A biblioteca “Aglets” de classes Java é chamada de JAAPI (*Java Aglets Application Program Interface*) e tem como objetivos (MILAGRES, 2001):

- Fornecer um modelo de programação para a utilização de agentes móveis, sem implicar em modificações na JVM ou em código nativo;
- Disponibilizar mecanismos de comunicações poderosos e dinâmicos que permitam agentes comunicar-se com outros agentes;
- Projetar uma arquitetura de agentes com extensão e reuso de código;
- Obter uma arquitetura coerente com o modelo tecnológico Web/Java.

O modelo utilizado para que os Aglets cumpram com esses objetivos pode ser descrito a partir do modelo conceitual do sistema de agentes em Java ilustrado na figura 13.

Figura 13 - Interação de agentes com o ambiente através de sensores

O modelo da figura 13 ilustra uma plataforma “Aglets” que fornece uma API com a funcionalidade básica para os processos de gerenciamento, migração e comunicação de agentes. O servidor de Aglets é *multithread* e funciona no topo da JVM. Os elementos dos três níveis superiores da figura 12: **Aglets**, **Aglet API** e **Tahiti** (*Aglet Server* – Ambiente de Execução) determinam as características dos Aglets, com seu ambiente de execução.

Aglets são objetos Java projetados para se mover através de *hosts* de uma rede de computadores. Caracterizam-se por apresentar a própria *thread* de controle, serem dirigidos a eventos e comunicar-se por troca de mensagens (LANGE, 1998).

No capítulo sobre agentes foi apresentado cinco características que definem um agente. Abaixo são relacionadas estas características aos Aglets:

- **Autonomia:** definir um aglet como um objeto com sua própria *thread* é uma forma prática de implementar em Java a característica de autonomia (LANGE, 1998). Para determinar que um programa Java utilize *threads*, deve-se indicar que as classes do programa implementem a interface “Runnable” do pacote “java.lang*”.* Para possuir sua própria *thread* de execução, um aglet é instanciado a partir de uma classe que implementa esta interface. Com o uso de *threads*, um aglet pode executar concorrentemente a outros Aglets no *host*.

- **Habilidade Social:** o modelo de comunicação da plataforma ASDK permite que agentes colaborem entre si realizando tarefas conjuntas. Isto é possível através do mecanismo de troca de mensagens entre os objetos que representam os aglets. As mensagens não são enviadas diretamente aos agentes. Para efetuar interações, é necessário obter previamente um objeto “proxy” que corresponda ao Aglet com quem se deseja comunicar. O “proxy” constitui a interface para a troca de mensagens entre os aglets. O tratamento que um aglet dispensa a uma mensagem específica é determinado pelo método “handleMessage( )”.

- **Reatividade**: para ser reativo, um agente deve oferecer métodos que correspondam aos sensores e atuadores da definição genérica de agentes. Na API Aglets existem métodos que executam ações em resposta a estímulos recebidos. Os sensores são métodos que lêem informações do ambiente. O método “getAgletInfo( )”, por exemplo, é responsável por retornar informações sobre um aglet específico no contexto de execução. O método “handleMessage( )” é um sensor que trata mensagens entre os aglets. Os atuadores, são métodos que lêem e causam modificações no ambiente. O método “setProperty( )” é um exemplo de atuador que permite a um aglet alterar propriedades de seu contexto. O método “sendMessage( )” afeta o comportamento dos agentes através do envio de mensagens.

- **Mobilidade**: os aglets podem se mover entre os *hosts* de uma rede através de chamadas aos métodos “dispatch( )” e “retract( )”. Um aglet adiciona mobilidade aos *applets* transportando de um *host* para outro, além de seu código, os dados de sua execução, o que mantém a persistência de seu estado interno. Um aglet pode ser transportado seqüencialmente entre muitos *hosts* de uma rede, podendo retornar a seu *host* original, sem que seus atributos internos sejam novamente iniciados. Quando chega a um novo *host*, o aglet inicia sua execução em uma nova *thread.* Esta migração é denominada tipo fraca.

- **Persistência (Continuidade Temporal):** os aglets mantém seus

atributos internos ao longo de sua execução. A persistência do estado interno de um aglet pode ser verificada através de seu ciclo de vida, o qual inclui as etapas: criação, clonagem, ativação, desativação e destruição. A serialização de objetos é um mecanismo fundamental no ciclo de vida dos aglets, pois mantém o estado de execução do objeto durante a transferência. Através dela é implementada a persistência nas etapas de clonagem, ativação, desativação e movimentação. Os objetos representantes dos aglets implementam a interface “Serializable” disponível no pacote “java.io”.

Um servidor de agentes é um elemento obrigatório em uma plataforma de agentes móveis (LANGE, 1998). Os aglets são executados em um servidor de agentes denominado “Aglet Server” (Tahiti – figura 14), que possibilita a criação, transferência, destruição de agentes, e a configuração de privilégios de acesso e segurança no servidor. O Tahiti é uma aplicação construída no nível mais alto da “API Aglet”, que possibilita manipular um aglet em todas as etapas de seu ciclo de vida (LANGE, 1998). É um visualizador de contexto de execução dos aglets e uma interface gráfica que gerencia os serviços do “Aglet Server”.

Uma das principais funções do “Aglet Server” é controlar o acesso aos recursos dos *hosts* onde os aglets são executados. Em cada *host*, múltiplos usuários podem ter acesso à plataforma e, cada usuário, pode corresponder a várias instâncias do servidor para a execução dos aglets. A utilização dos servidores é feita pela configuração de portas. Em cada servidor podem ser criados múltiplos contextos para a execução dos aglets. Um aglet está associado apenas a um único contexto em um dado instante.

Na plataforma “Aglets”, as configurações de acesso aos recursos dos *hosts* são semelhantes ao modelo de implementação dos *applets* Java. Da mesma forma que o *browser* limita as atividades que os *applets* podem realizar em *hosts* de uma rede, o servidor “Aglet Server” é responsável pela execução segura dos aglets, mantendo restrições quanto ao que eles são capazes de efetuar nos sistemas em que estão hospedados.

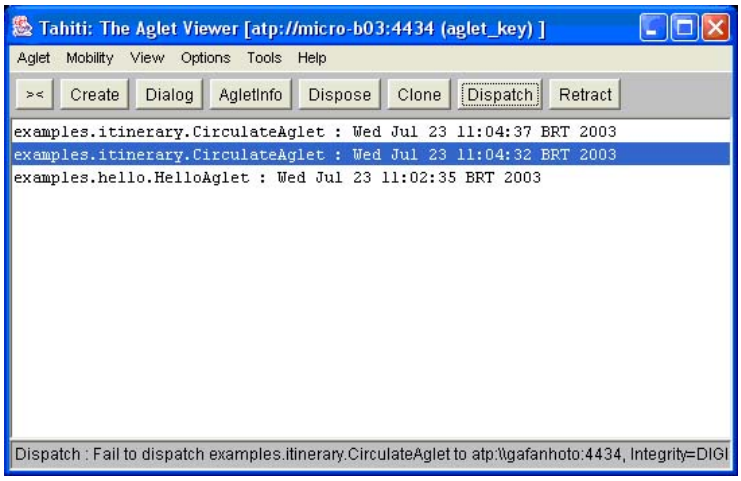


Figura 14 - Ambiente de Execução de um aglet (Tahiti)

O *Aglets Software Development Kit* (ASDK) é uma plataforma para desenvolvimento de agentes móveis (Aglets). Foi desenvolvida na linguagem Java, caracterizada por ser: portável, distribuída, segura e multitarefa, atributos estes essenciais para o desenvolvimento de agentes móveis (LANGE, 1998). Há duas versões para a plataforma ASDK. A primeira, *Aglets Workbench*, foi desenvolvida pelos pesquisadores Danny Lange e Mitsuru Oshima, da IBM Japão em 1996 e funciona com o *Java Development Kit* (JDK) versão 1.1. A plataforma desenvolvida pela IBM não é compatível com as versões mais recentes da linguagem Java. Para o JDK 1.2 e superiores, uma versão de código-livre do ASDK 2.0, ou *Aglets 2.0*, do *SourceForge*, deve ser utilizada. Trata-se de um *framework* para pesquisa e desenvolvimento de agentes móveis em código aberto, licenciado sob os termos da *IBM Public License* (IBM PUBLIC LICENSE), a qual estabelece as condições para que um projeto de propriedade da IBM seja liberado para utilização pública. Apresenta-se abaixo a tabela 3 com informações sobre compatibilidade de plataformas.

Tabela 3 – Compatibilidade entre Java e *Aglets*

Máquina Virtual Java	Versão do ASDK compatível
JDK 1.1	*Aglets Workbench*
JDK 1.2 ou superior (Java 2)	*Aglets 2.0*

Informações sobre *downloads*, instalações e configurações são descritas nos anexos deste trabalho.

A escolha do ASDK para a implementação do protótipo se deve a:

- Estar implementado em Java e oferecer vantagens para a programação de agentes móveis: programação *multithread* (implementa comportamento autônomo); independência de plataforma (permite criar agentes móveis sem conhecer o local de execução); serialização de objetos (permite transferir agentes); execução segura (protege o *host* contra agentes maliciosos, impedindo seu acesso).

- Oferece formas de interação entre agentes móveis através da troca de mensagens síncronas e assíncronas.

- Possibilita o tratamento de eventos como criação, destruição e migração de agentes móveis, favorecendo a implementação de ações dos elementos deste mecanismo na ocorrência destes eventos.

- Possibilita tratamento e distinção entre os tipos de mensagens de acordo com o tipo definido pelo programador.

- Permite a suspensão da *thread* de execução do agente por certo intervalo de tempo e a retomada da execução.

- Permite a invocação de métodos que trata suas mensagens, permitindo tratamento posterior de mensagens e possibilitando sua ordenação.

Os principais conceitos da plataforma ASDK são:

- **Aglet:** objeto Java, autônomo, pois possui a sua própria *thread* de execução e é capaz de migrar de uma máquina para outra na rede.

- **Contexto**: lugar de execução de um aglet. Trata-se de um objeto estacionário que gerencia os aglets em execução, sendo que um nó na rede pode conter mais de que um contexto.

- **Proxy**: é utilizado para a interação com outros agentes. Representa um aglet e protege seus métodos públicos do acesso direto por outros aglets. Isto oferece transparência de localização escondendo a localização verdadeira do aglet.

- **Mensagem**: objeto de comunicação síncrono ou assíncrono dos aglets.
- **Itinerário**: plano de migração de um aglet.
- **Identificador**: identificação do aglet, única e imutável durante sua vida.

O ciclo de vida de um aglet é definido por um conjunto de eventos que causam respectivas ações e modificam o seu estado (LANGE, 1998). Os eventos são: criação (“creation”), clonagem (“clone”), transferência para outro contexto (“dispatch”), volta para o contexto de origem (“retract”), desativação (“deactivation”), ativação (“activation”) e destruição (“dispose”).

A criação de um aglet pode ser realizada de duas maneiras. Criando um objeto ou pela criação de um clone de um aglet existente. Ao chegar ao destino, o aglet pode iniciar sua execução, executar tarefas e voltar para sua origem. Um aglet pode ser armazenado no disco rígido, ficar desativado temporariamente e ser reativado. Quando um aglet termina sua tarefa ele pode ser destruído. Toda esta sistemática está demonstrada na figura 15.

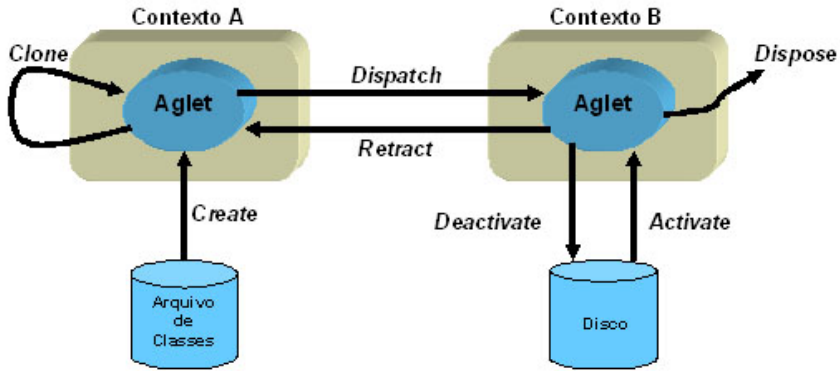


Figura 15 - O ciclo de vida de um aglet

A “API Aglet”, AAPI ou JAAPI foi projetada para beneficiar-se das características de Java que suportam os requisitos do desenvolvimento de agentes móveis. Oferece um modelo para programação de aglets sem modificações na JVM. Por este modelo, não há necessidade de integração a código nativo, sendo a integrada à tecnologia Java/Web existente.

A principal vantagem de utilização da AAPI no desenvolvimento de aglets diz respeito à portabilidade do código dos agentes. Uma vez que um

aglet tenha sido implementado, ele poderá ser executado em qualquer máquina que suporte sua interface de programação, sem que seja necessário considerar aspectos de *hardware*, sistemas operacionais, ou ainda características da AAPI nos *hosts* onde os agentes são executados.

A AAPI é um pacote Java que define as classes e interfaces que implementam as funcionalidades para a construção e o funcionamento dos agentes móveis (LANGE, 1998). As principais classes e interfaces são: “Aglet”, “AgletProxy”, “AgletContext”, “Message”, “FutureReplay” e “AgletID”. No Anexo 1 deste trabalho é descrita toda a AAPI e no anexo 2 é apresentado uma descrição detalhada de suas principais classes e dos modelos de comunicação, eventos, mobilidade, protocolos e ciclo de vida de um aglet.

Este capítulo apresentou conceitos importantes para o entendimento do domínio do projeto “VWBuilder”. A fundamentação teórica passou, na seção 2.1 pelas definições e características da RV e conceituou a RA como uma de suas sub-áreas utilizadas para se fazer realidade misturada entre o mundo real e objetos virtuais. Para finalizar esta seção, foi explicado a ferramenta ARToolkit e mais especificamente o NetARToolkit como uma modificação do *software* original para uso em rede de computadores. Na seção 2.3.2 foi abordado a tecnologia de agentes móveis, fazendo uma revisão sobre a evolução das telecomunicações e localizando-os como uma sub-área de sistemas distribuídos. Foi abordado o conceito de agentes, agentes móveis e especificamente descrito características da “API Aglets”, utilizada para o desenvolvimento dos agentes em Java.

Na sessão 2.4 desta dissertação, está descrito alguns trabalhos analisados que buscam esclarecer o uso de agentes móveis em sistemas distribuídos, busca de informações e aplicações que fazem uso da realidade virtual baseadas em agentes. Estes trabalhos foram analisados e contribuíram para o entendimento e pesquisa do uso de agentes móveis em tecnologias de realidade virtual, busca e apresentação de informações.

2.4 Trabalhos Correlatos

Este capítulo apresenta uma breve descrição de alguns trabalhos relacionados à computação distribuída que fizeram uso da tecnologia de agentes (alguns da plataforma “Aglets”). Os trabalhos apresentam a utilização de agentes em diferentes áreas, não necessariamente na área de RV. Estes trabalhos foram analisados e utilizados como referência no decorrer da dissertação para o entendimento da tecnologia de agentes, e agentes com realidade virtual.

A tecnologia de agentes móveis apresenta um grande potencial para ser utilizada em estruturas de busca e recuperação de informações em sistemas distribuídos implementados, de maneira simples e eficiente (BREWINGTON, 1999). Para fazer a migração de códigos em locais remotos, um agente pode acessar um repositório em um *host* em uma rede e transferir os dados para um outro local. A tarefa de recuperação de Informações (ou IR – *Information Retrieval*) é uma área da ciência da computação que objetiva representar, armazenar e organizar o acesso a informações. O seu objetivo é satisfazer a necessidade do uso da informação da melhor maneira possível. É considerada uma das áreas de utilização de agentes móveis (NUNES, 2002).

Alguns trabalhos são encontrados na literatura e justificam a utilização da tecnologia de agentes móveis em sistemas distribuídos e RV. Outros utilizam-se de agentes móveis, dentre estes alguns usam a plataforma “Aglets” para a busca, filtragem e recuperação de informações distribuídas.

Ao analisar os trabalhos observou-se que a maioria das aplicações busca arquivos para apresentação localmente em diretórios localizados nos próprios *hosts*. Tornou-se padrão carregar, nos mundos virtuais, arquivos locais, através da leitura de grafos de cena nos *hosts* que executam as aplicações. Este fato faz com que as possibilidades de compartilhamento, troca ou busca por novos arquivos, seja anulada, permanecendo as aplicações, resumidas ao uso de uma pequena diversidade de objetos virtuais. A busca de arquivos, para compor um mundo virtual é realizada manualmente utilizando-se de sistemas de busca na Internet, e fazendo uma cópia individual destes arquivos. Pode-se também realizar a modelagem destes objetos virtuais.

O trabalho COUTINHO (2003) aborda o uso de AVC (Ambientes Virtuais Colaborativos) para o desenvolvimento de técnicas de armazenamento e recuperação de objetos virtuais tridimensionais em um sistema de informações geográficas. Também na área de AVC, (MEIGUINS, 2003; ROSA, 2003) fazem uma análise dos problemas encontrados no suporte a comunicação destes sistemas e, apresentam uma proposta sobre um protocolo para utilização de AVC na WWW. Ainda abordando AVC, Onivaldo Rosa Junior (ROSA JR, 2001), apresenta um projeto que elabora a implementação de um sistema interativo de comunicação utilizando RV com mídias simultâneas. Neste trabalho, foi utilizada a tecnologia de agentes inteligentes para atuarem como tutores em um sistema de educação a distância.

O trabalho de Velloso et al (VELLOSO, 2002) apresenta uma proposta de utilização de agentes móveis para gerenciamento de redes, comparando o modelo cliente-servidor com o modelo baseado em agentes através da criação de uma ferramenta baseada em agentes. Como trabalhos futuros, este projeto sugere a possibilidade de se apresentar informações colhidas na rede, podendo ser realizado em um ambiente virtual.

O trabalho de Yoann Fabre (FABRE, 2003) apresenta um estudo das novas tecnologias (VRML, Java3D, X3D) e agentes aplicados para a construção de NVEs. Ainda apresentando tecnologias para construção do Ciberespaço, Jorge Henrique Cabral Fernandes (FERNANDESa, 2000) apresenta tecnologias para construção de mundos virtuais e aborda tecnologia de agentes móveis com uma especial atenção a plataforma “Aglets”.

O trabalho de Bernhard Jung e Jan-Torsten Milde (JUNG, 1999) descreve um protótipo de ambiente virtual construído utilizando VRML e Java. Utiliza uma abordagem cliente-servidor onde os agentes são articulados, heterogêneos e autônomos, utilizados para representar usuários humanos.

O trabalho de Cyril Morcrette (MORCRETTE, 1999) apresenta ferramentas e técnicas de visualização de informações de um banco de dados em ambientes virtuais tridimensionais. Semelhante a este trabalho, apresenta-se uma proposta de trabalho de Marcos Paulo Alves de Souza (SOUZA, 2003) que

propõe a utilização de agentes móveis em ambientes virtuais tridimensionais utilizando a plataforma “Aglets” e a tecnologia Java3D.

O trabalho de Adelinde Uhrmacher e Bernd Kullick (UHRMACHER, 2000) apresenta um ambiente de modelagem e testes baseado em agentes chamado *James.* Este ambiente foi construído sobre a plataforma *Mole* com objetivo ser uma plataforma de simulações de sistemas multi-agentes em NVEs.

O trabalho de Artur Caetano e João Pereira (CAETANO, 2000) apresenta modelos para representação de cenas VRML com populações de agentes interativos, comportamentos autônomos e reativos. Foram utilizadas as linguagens VRML para a construção de cenas e Java para a criação de *Scripts* de interação com o ambiente virtual. A interação entre o usuário e os objetos no mundo virtual se deu através da técnica de EAI.

No trabalho Beat Herrmann (HERRMANN, 2000) apresenta a plataforma DOVRE para construção de ambientes para RV baseada em sistemas distribuídos. DOVRE é uma plataforma orientada a objetos, *multithread,* que permite compartilhamento de mundos virtuais construídos por diversos usuários.

O trabalho de Montesco (MONTESCO, 2001) apresenta um levantamento das linguagens e tecnologias utilizadas para comunicação com agentes e, entre elas, a “API Aglets”. Faz uma comparação entre as linguagens e propõe uma nova linguagem de comunicação para agentes chamada *Universal Communication Language* (UCL). Melo e Borin (MELO, 2001), apresentam um trabalho onde é construído um sistema de agentes móveis em Java. Neste trabalho não foi utilizada uma plataforma, e sim programado um conjunto de agentes que agem como estrutura de comunicação utilizando RMI. Ainda neste mesmo contexto, Vera Nagamuta (NAGAMUTA, 1999) apresenta uma dissertação sobre o problema da coordenação de agentes móveis através de um canal utilizando comunicação *broadcast* e a plataforma “Aglets”.

O trabalho de Ana Paula Piovesan Melchiori Peruzza (PERUZZA, 2003) apresenta um projeto que utiliza a RV e agentes autônomos para a

construção de uma ferramenta (ConstruirRV) para educação a distância. Neste sistema é utilizado o RMI e Java3D para a comunicação entre os mundos virtuais distribuídos. Os agentes têm tarefas como monitorar o ambiente para povoar uma sala de aula quando um usuário entra na sala.

O trabalho LOBATO (2003), apresenta um protótipo utilizando agentes móveis (Aglets) para efetuar uma busca e apresentação de informações banco de dados distribuídos em uma rede de computadores.

O trabalho BAX (2001) apresenta uma discussão sobre a utilização de tecnologias e recursos para o tratamento dos problemas de recuperação e apresentação de informações em sistemas computacionais. Não se trata de um trabalho de Ciência da Computação, mas de uma discussão a respeito da utilização de agentes para busca e recuperação de informações eletrônicas.

Eid (2005) apresenta um estudo realizado através da avaliação de 220 artigos publicados entre os anos de 1989 e 2004 que fizeram uso de aplicações com agentes móveis. Traz uma análise descritiva das áreas pesquisadas e informações estatísticas a respeito da utilização dos agentes em cada uma destas áreas. Pôde-se observar, por exemplo, que foram encontrados 13% dos artigos relacionados à área de pesquisa, filtragem e recuperação de informações. É um trabalho que norteou e abriu caminhos para conhecer outros trabalhos da área pesquisada nesta dissertação.

O trabalho de Helena Nunes e Sofiane Labidi (NUNES, 2002) apresenta uma arquitetura de busca inteligente baseado em um padrão de aprendizagem utilizando a tecnologia de agentes móveis aplicada a um sistema de ensino matemático chamado *MathNe*t. Neste trabalho um agente de busca baseado na plataforma “Aglets” utiliza padrões de aprendizagem para obter informações distribuídas. Foi utilizado a plataforma “ZEUS” no desenvolvimento de agentes para o desenvolvimento do kernel multi-agente.

O artigo de Olguín et al (OLGUIN, 2000) faz a descrição de uma estrutura distribuída baseada em agentes móveis colaborativos utilizando a plataforma “Voyager” para a criação dinâmica e gerenciamento de grupos de

estudo formado por estudantes distribuídos que compartilham material *on-line*.

Theilmann e Rothermel (THEILMANN, 1999) apresentam em seu trabalho uma abordagem baseada em uma *engine* de busca de informações na Web baseada em agentes móveis que filtram informações. Demonstra-se que filtrar informações baseando-se em agentes móveis causa uma significativa redução na carga da rede. O objetivo do trabalho é questionar e propor uma maneira de buscar documentos (artigos científicos) na Web. O experimento apresentado demonstra uma redução de até 60% da carga na rede utilizando-se os agentes móveis.

Papastavrou et al (PAPASTAVROU, 2000) propõem em seu trabalho uma estrutura para acesso à bases de dados distribuídas na Web baseado em agentes móveis em Java. Propõe que processos leves, autônomos e portáveis podem ser utilizados para busca de informações em banco de dados de maneira eficiente e adequado ao ambiente da Internet. Utiliza primeiramente a plataforma “Voyager” e a plataforma “Aglets” em uma reestruturação do sistema onde se percebeu um ganho de performance de 30 a 40%.

O trabalho de Brewington et al (BREWINGTON, 1999) apresenta uma discussão a respeito do uso da tecnologia de agentes móveis para recuperação de informações distribuídas. Defende os pontos onde os agentes móveis apresentam vantagens em relação ao uso de outras tecnologias de comunicação em ambientes distribuídos como RPC, RMI, *Applets* e *Servlets*. Traz uma pesquisa sobre plataformas de agentes móveis e descreve as características de algumas das plataformas apresentadas.

Estes trabalhos foram encontrados em sua grande maioria em sites na Internet e *Proceedings* de congressos e apresentam testes e perspectivas de utilização de agentes móveis (alguns com a plataforma “Aglets”). Não são trabalhos que utilizam as mesmas plataformas de agentes e também não fazem uso de agentes somente em realidade virtual. Com a perspectiva de pesquisar e experimentar a utilização de uma plataforma de agentes móveis em ambientes virtuais é que foi baseada a motivação do trabalho que segue.

3 PROTÓTIPO DO “VIRTUAL WORLD BUILDER”

Virtual World Builder (VWBuilder) (ou Construtor de Mundos Virtuais) é a denominação do protótipo desenvolvido para do uso de agentes móveis aplicados na composição de mundos virtuais com realidade aumentada.

Apresenta-se neste capítulo uma descrição geral do protótipo VWBuilder e sua estrutura em termos de classes e principais funcionalidades. Também é abordado a descrição do NetARToolkit, por se tratar do mecanismo utilizado para a produção da realidade aumentada, conforme ilustra a figura 16.

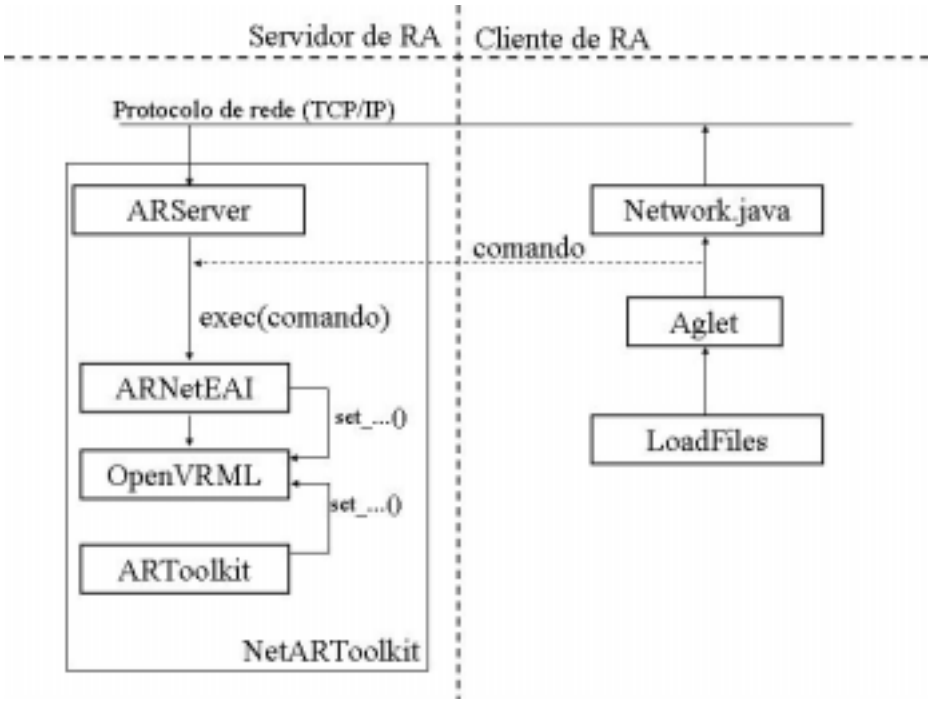


Figura 16 - Estrutura modular do VWBuilder e interação com a biblioteca ”ARToolkit”

A aplicação NetARToolkit implementa um serviço de RA, em que “ARServer” é um processo baseado em *thread* que obtém comandos a partir da rede e os transmite ao objeto “ARNetEAI”. Os comandos implementados nesta classe têm estrutura análoga ao comando apresentado na figura 17. Esse código implementa os mecanismos de acesso ao grafo de cena. Assim, os agentes móveis baseados Aglets, usam a classe “Network” para atuar na cena, enviando mensagens ao servidor. Além disso, é usada a biblioteca “ARToolkit” para implementar o rastreamento das marcas inseridas no mundo real, permitindo que os usuários interajam com as cenas. Desta forma, o ambiente

NetARToolkit permite que os agentes móveis interajam com os usuários, utilizando o espaço virtual. Por exemplo, um agente “LoadFiles” pode inserir objetos na cena, passando a interagir com o usuário.

```
void ARNetEAI::set_transparency(const char* node, float t){
        VrmlScene *sc = sceneGraph->vrmlScene;
        if (sc == NULL)return;
        VrmlNamespace* ns = sc->scope();
        if (ns==NULL)return;
        VrmlNode* p_no = ns->findNode(node);
        if (p_no==NULL) return;
        VrmlNodeMaterial* nd = p_no->toMaterial();
        if (nd==NULL)return;
        VrmlSFFloat transp(t);
        nd->setField("transparency", transp);
        nd->render(sceneGraph);
}
```

Figura 17 - Acesso ao grafo de cena, usando OpenVRML

As principais funcionalidades do VWBuilder são: a busca de identificadores de arquivos; a busca de arquivos; a carga de cenas em uma aplicação de RA. Essa estrutura é constituída de agentes móveis independentes entre si e ocorrem em três etapas distintas, implementadas através de três aglets: (i) o agente móvel “**SearchFileNames**” consulta um arquivo local contendo os URLs de *hosts* a serem visitados, criando o seu itinerário. Percorre o itinerário montando uma lista com todos os identificadores de arquivos com as respectivas URLs, armazenando localmente o resultado da busca; (ii) “**DownloadFile**” é o agente móvel que usa o resultado da pesquisa para buscar os arquivos e armazená-los localmente; (iii) “**LoadFiles**” realiza a inserção dos objetos virtuais no ambiente de RA.

3.1 O Projeto do VWBuilder

3.1.1 Descrição do Projeto

De maneira geral, o VWBuilder é constituído pelas classes: “ContextConfig”, “SearchFileNames”, “DownloadFile” e “LoadFiles”. As classes “SearchFileNames”, “DownloadFile” e “LoadFiles” herdam a classe “Aglet” do pacote “com.ibm.aglets”. O VWBuilder utiliza também a classe “Network” como meio de acesso ao servidor de mundos virtuais ARToolkit. A figura 18 ilustra o diagrama de classes do VWBuilder. Os códigos-fonte e a documentação Java

encontram-se nos anexos 6 e 7 deste trabalho.

A classe “ContextConfig” carrega as URLs que o agente deve utilizar para criar seu itinerário. As URLs estão definidas no arquivo “c:/aglets/listURLs.txt” da máquina local. Esta é uma classe abstrata utilizada pela classe “SearchFileNames” para montar o itinerário que o agente usará para a referência de URLs e arquivos “.wrl” (exemplo: “atp://micro-r02:4435”).

O método “getURL( )” é utilizado para ler as linhas (com os endereços de URLs) do arquivo “c:/aglets/listURLs.txt” da máquina local. Este método irá ler cada linha do arquivo e armazena-las na variável “**url**” formando uma lista de endereços (urls) armazenada em “**list**”. Um objeto “**l**”, instanciado como “ContextConfig” e utilizado para imprimir os valores lidos no console.

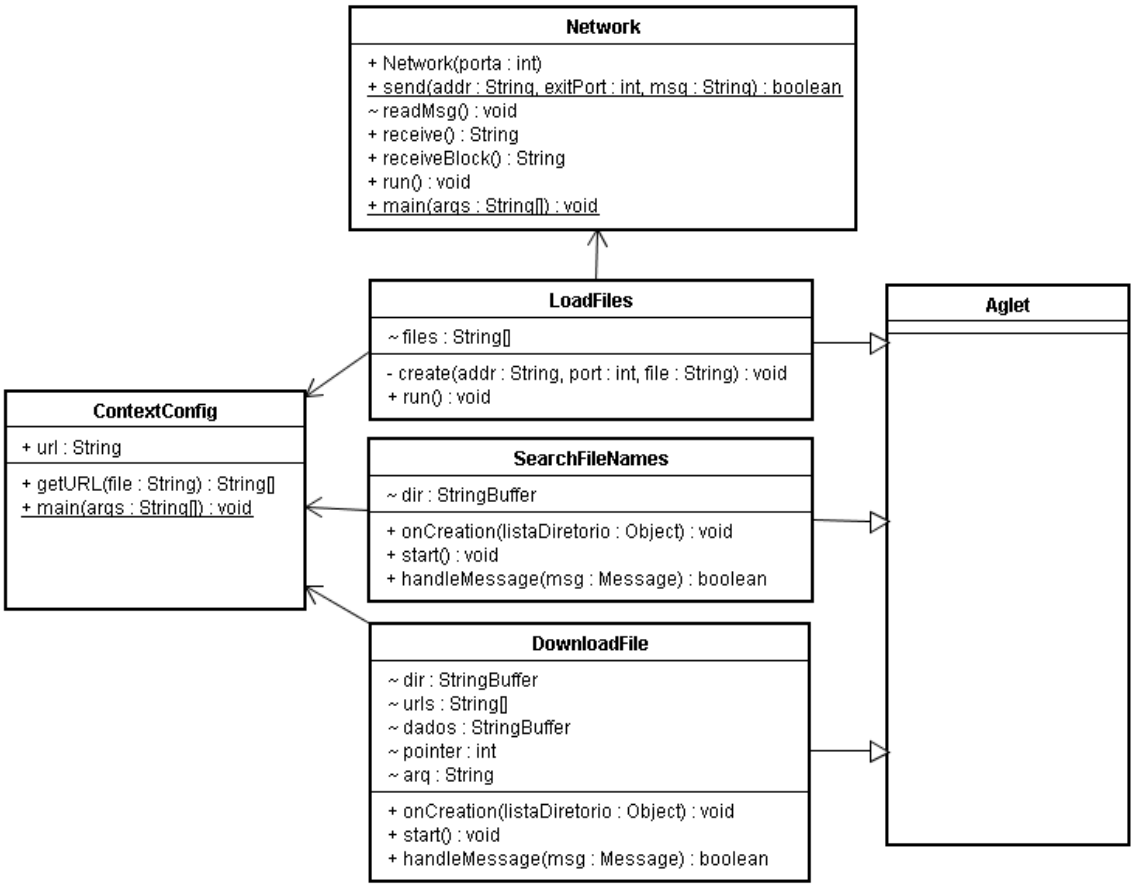


Figura 18 - Diagrama de classes UML do protótipo do VWBuilder

A classe “**SearchFileNames**” estende a classe “Aglet”. Ela realiza duas funções. Primeiro ele verifica o itinerário do agente (lista de URLs que o agente deverá visitar) definida no arquivo “c:/aglets/listURLs.txt” da máquina local e estabelecido pela classe “ContextConfig”. Após verificar o itinerário, o agente busca nos URLs o nome dos arquivos “.wrl” encontrados nos diretórios pesquisados (“c:/aglets/vrml” das máquinas remotas) e monta uma nova lista com o endereço (url) com o arquivo “.wrl” (o formato é por exemplo, “atp://micro-r02:4435/ box.wrl”). Estas informações ficam gravadas no arquivo “c:/aglets/listDir.txt” na maquila local.

Nesta classe, é necessário realizar a importação de alguns pacotes de “Aglets”. A linha “public void onCreation(Object listaDiretorio)” é responsável por criar o agente de busca. O conteúdo do arquivo “c:/aglets/listURLs.txt”, informados pelo objeto “ContextConfig” será utilizado para criar o itinerário que o agente de busca irá percorrer.

O método “itinerary.startTrip( )” faz com que o agente dê início ao processo de busca pelas informações sobre locais e nomes dos objetos virtuais distribuídos na rede, visitando os URLs informados em “ContextConfig”. A linha de comando “public boolean handleMessage(Message msg)” tem como função fazer com que o agente percorra as URLs informadas enquanto o agente de busca receber a mensagem “getDir”. Desta forma, o agente estará copiando as informações de URL + nome do arquivo “.wrl” do diretório “c:/aglets/vrml” da máquina remota pesquisada. O agente continua a busca por informações até que o URL pesquisado seja comparado com o seu próprio endereço (URL). Isto é realizado pelo método “getAgletContext( ).getHostingURL( )” também herdado da classe “Aglet”. Quando é passado a mensagem “saveList” ao agente, ele encerra o trabalho de busca e finaliza a inserção das informações no arquivo “c:/aglets/listDir.txt”.

A classe “**DownloadFile**” estende a classe “Aglet”. Ela realiza o *download* dos arquivos “.wrl” dos diretórios das máquinas pesquisadas na rede, mapeadas através da classe “SearchFileNames” com as informações de url e arquivo “.wrl” gravadas no arquivo local “c:/aglets/listDir.txt” gravando os

arquivos baixados no diretório local “c:/aglets/tempWRL”. Sua funcionalidade é análoga à classe “SearchFileNames”. A diferença fundamental é que ela faz a cópia dos arquivos “.wrl” mapeados no arquivo “c:/aglets/listDir.txt”, para o diretório local “c:/aglets/tempWRL”. A linha “public void onCreation(Object listaDiretorio)” cria um novo agente. Este agente verifica o diretório “c:/aglets/listDir.txt”, monta o itinerário do agente e parte para os *hosts* em busca dos arquivos. Enquanto a mensagem passada a ele for “getFile”, o agente busca no próximo URL novos arquivos. Esta tarefa é repetida até que uma mensagem com conteúdo “saveFile” seja passada para o agente de *download*. Quando isto ocorrer, ele encerra a tarefa de *download*. Esta mensagem é transmitida pelo próprio agente, fazendo-se a comparação do próximo URL a ser pesquisado através do método “getAgletContext( ). getHostingURL( )”, que retorna o URL da máquina local.

Durante sua execução, o agente lê os arquivos dos diretórios remotos “c:/aglets/vrml” pelo método “BufferedReader rd = new BufferedReader (new FileReader("c:/aglets/vrml/"+arq))”; e os copia para o diretório local “c:/aglets/tempWRL” através do método “PrintStream ps = new PrintStream(new FileOutputStream ("c:/aglets/tempWRL/"+arq))”. No final do trabalho, o diretório “c:/aglets/tempWRL” local deve conter uma cópia dos arquivos distribuídos na rede.

A classe “**LoadFiles**” também estende a classe “Aglet”. Ela realiza o carregamento dos arquivos do diretório “c:/aglets/tempWRL” da máquina local, que foram descritos no arquivo “c:/aglets/listDir.txt” e baixados para o diretório local c:/tempWRL para a cena do mundo virtual do aplicativo NetARToolkit. Por se tratar de um aglet, é necessário realizar a importação de alguns pacotes “Aglets”. A linha “private void create(String addr, int port, String file)” define a criação de um novo aglet. Em seguida é realizada a verificação das informações da url e da porta em que o arquivo VRML será carregado. Isto é feito através da linha “System.out.println("Criando --> " + addr + " " + port + " " + file)”. O carregamento do arquivo se dá efetivamente pelo conjunto de linhas apresentados na figura 19:

```
cena="createFromString main ";
StringTokenizer coment;
while ( (buff = in.readLine()) != null){
  if (buff.startsWith("#")) continue;
  if (buff.length() == 0)  continue;
  buff = buff.replace('\n', ' ');
  buff = buff.replace('\t', ' ');
  int pch = buff.indexOf("#");
  if (pch != -1)
     buff = buff.substring(1, pch);
  cena += buff;
}
in.close();
```

Figura 19 - Código para inserção de arquivo VRML na cena

O método “Network.send(addr, port, cena)” da classe “Network” envia o objeto lido na cena para o programa ARToolkit. Como objetivo informativo, é impresso no console informações a respeito das tarefas executadas pelo método “System.out.println ("Saindo --> " + addr + " " + port + " " + file)”. O método “send” é apresentado na figura 20.

O construtor “run( )” desta classe, utiliza-se mais uma vez das informações do arquivo “c:/aglets/lisDir.txt” referentes aos arquivos que deverão ser carregados na cena. As linhas “arq = tk.nextToken()”; e “this.create ("127.0.0.1", 1201, "c:\\Aglets\\tempWRL\\" + arq)”; são as responsáveis por apontar para o arquivo que será lido e informar o endereço IP (utilizado o endereço de *loopback*, a porta utilizada pelo ARToolkit e o arquivo que deverá ser carregado na cena do mundo virtual.

A classe “**Network**” realiza a comunicação com o mundo virtual no renderizador do ARToolkit. Esta classe implementa a interface de comunicação via *socket* (porta 1201) com um servidor que lê os arquivos “.wrl” e os insere no mundo virtual. O método “send” é parte integrante da classe “Network”. Ele permite que se faça o envio do arquivo “.wrl” ao servidor “ARServer” da biblioteca ARToolkit.

Este objeto abre uma conexão via *socket* (porta 1201) utilizando “Socket server = new Socket(InetAddress.getByName(addr), exitPort)”; e envia a *stream* do conteúdo do arquivo lido através do objeto “OutputStream os = server.getOutputStream( )”. Ao finalizar a leitura de todos os arquivos da *string*

“saída”, fecha-se o *socket* com “os.close( )” e finaliza-se o carregamento do objeto virtual na cena.

```
public synchronized static boolean send(String addr, int exitPort, String msg)  {
  try{
    msg = msg.trim();
    Socket server = new Socket(InetAddress.getByName(addr), exitPort);
    OutputStream os = server.getOutputStream();
    String saida =  "" + msg.length();
    while (saida.length() < msgSz)
       saida += " ";
    saida += msg;
    os.write(saida.getBytes(), 0, saida.length());
    os.flush();
    os.close();
      return true;
    }catch (IOException ioe){
       return false;
    }
  }
```

Figura 20 - Método "send" da classe "Network"

3.1.2 Diagrama de Caso de Uso

A figura 21 apresenta o diagrama de caso de uso do projeto VWBuilder apresentando as tarefas e as ralações entre o usuário (ator) e as tarefas realizadas pelas classes. A tarefa “Buscar nomes de arquivos” é executada pela classe “SearchFileNames”; a tarefa “Baixar arquivos relacionados” é executada pela classe “DownloadFile” e a tarefa “Carregar arquivos na cena” é realizada pela classe “LoadFiles”. Esta última depende do renderizador do ARToolkit estar ativado e com um a cena carregada.

Figura 21 - Diagrama de Casos de Uso UML do projeto VWBuilder

3.1.3 Diagramas de Seqüência

Os diagramas de seqüência dão ênfase a ordenação temporal em que as mensagens são trocadas entre os objetos de um sistema. Apresenta-se abaixo os diagramas referentes às três classes que implementam o VWBuilder.

3.1.3.1 Buscar nomes de arquivos

Apresenta-se na figura 22 o diagrama de seqüência da tarefa “Buscar nomes de arquivos”, implementada pela classe “SearchFileNames”.

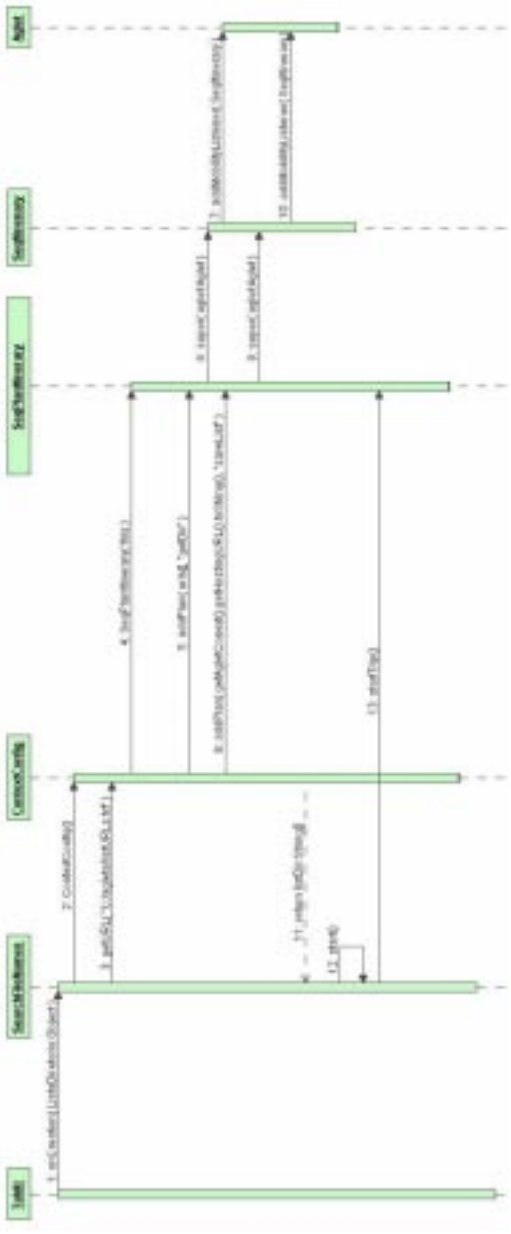


Figura 22 - Diagrama de sequência UML para ”Buscar nomes de arquivos”

3.1.3.2 Baixar arquivos selecionados

A figura 23 apresenta o diagrama de seqüência para a tarefa de “Baixar arquivos selecionados” implementados pela classe “DownloadFile”.

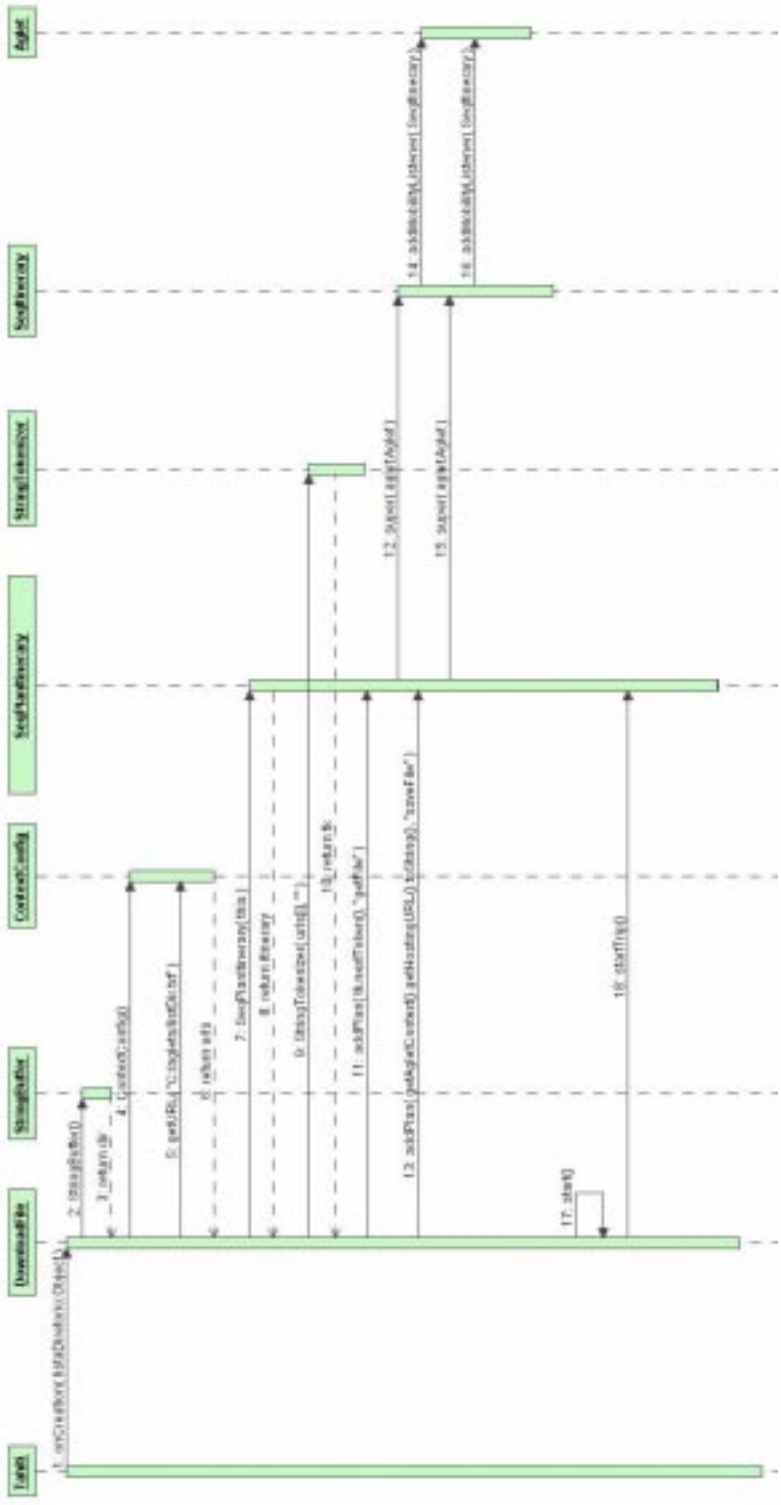


Figura 23 - Diagrama de sequência UML para "Baixar arquivos selecionados"

3.1.3.3 Carregar arquivo na cena

A figura 24 apresenta o diagrama de seqüência da tarefa “Carregar arquivos na cena” que implementa a classe “LoadFiles” e tem associações com a classe “Network”, responsável esta por fazer a comunicação via *socket* com o servidor do ARToolkit.

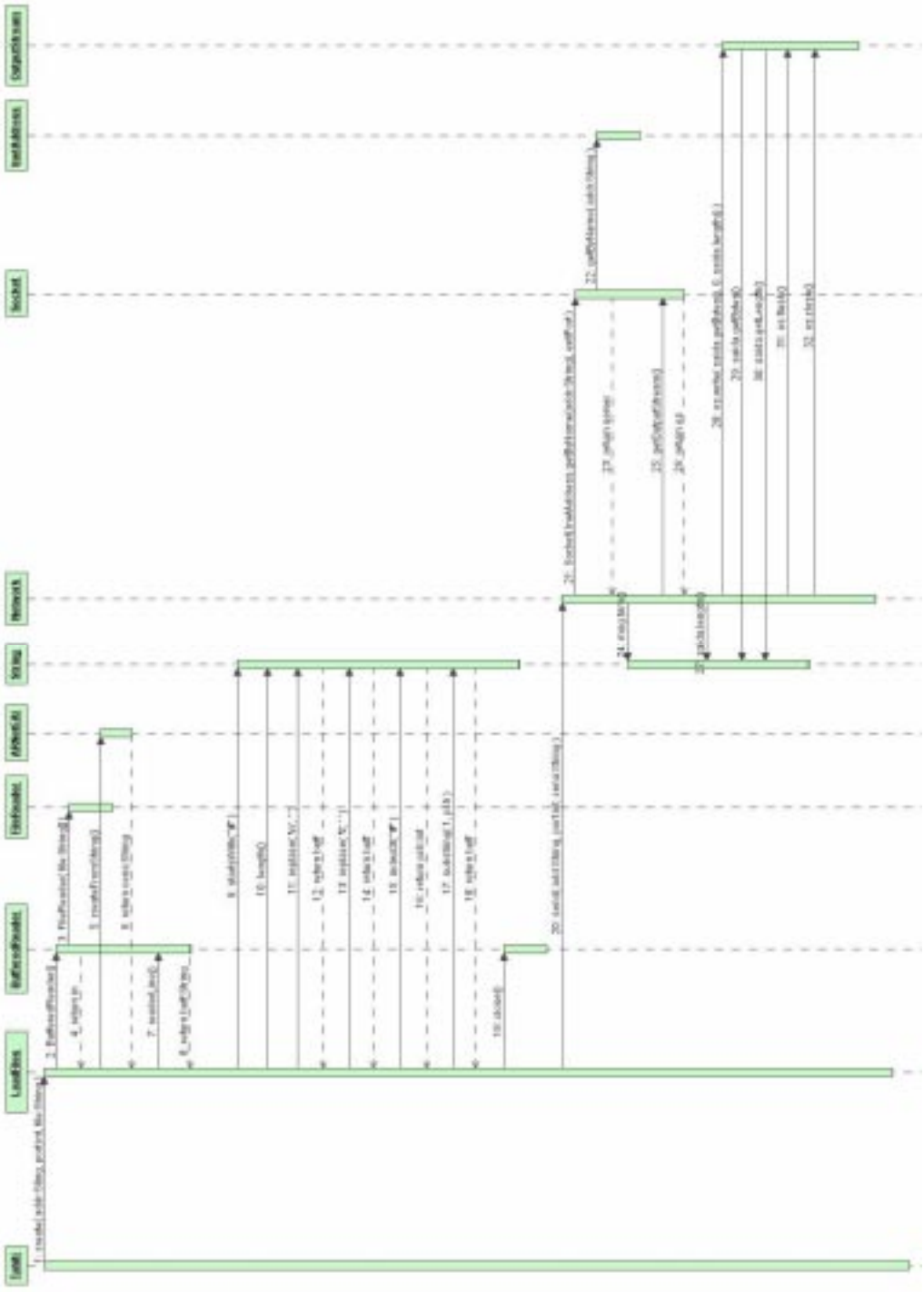


Figura 24 - Diagrama de sequência UML para a tarefa "Carregar arquivos na cena"

3.1.4 Diagramas de Estado

Os diagramas de estados representam os estados possíveis de um objeto em particular. Apresenta-se abaixo quatro diagramas de estados referentes às classes que implementam o projeto “VWBuilder”.

3.1.4.1 Diagrama de Estados: “ContextConfig”

A figura 25 apresenta o diagrama de estados da classe “ContextConfig”, responsável pela leitura dos endereços (url) dos *hosts* que serão visitados pelos agentes “SearchFileNames” e “DownloadFile”. Através do método “getURL()” e da leitura do arquivo c:/aglets/listURLs.txt, esta classe monta o itinerário a ser utilizado pelos outros três agentes para pesquisar, baixar e carregar objetos virtuais na cena do aplicativo de realidade aumentada.

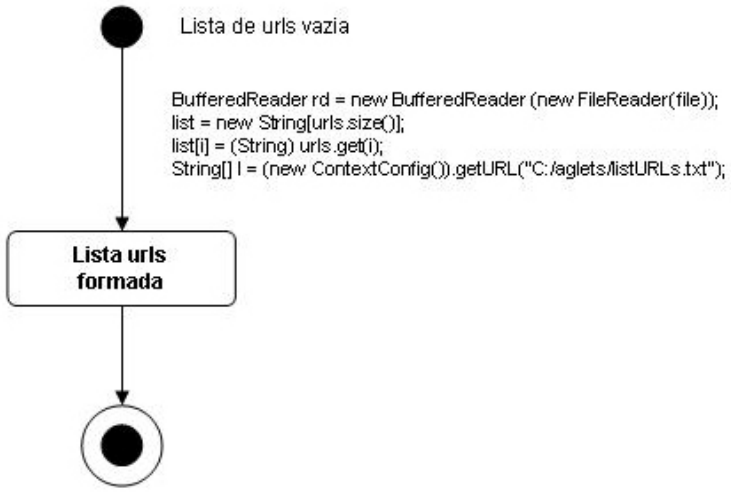


Figura 25 - Diagrama de estados UML da classe "ContextConfig"

3.1.4.2 Diagrama de Estados: "SearchFileNames"

A figura 26 apresenta o diagrama de estados da classe "SearchFileNames". Esta classe irá montar um itinerário para o agente de busca, migrar entre as máquinas que compõem o itinerário e escrever o arquivo "c:/aglets/listDir.txt" com informações a respeito de arquivos localizados nas máquinas distribuídas.

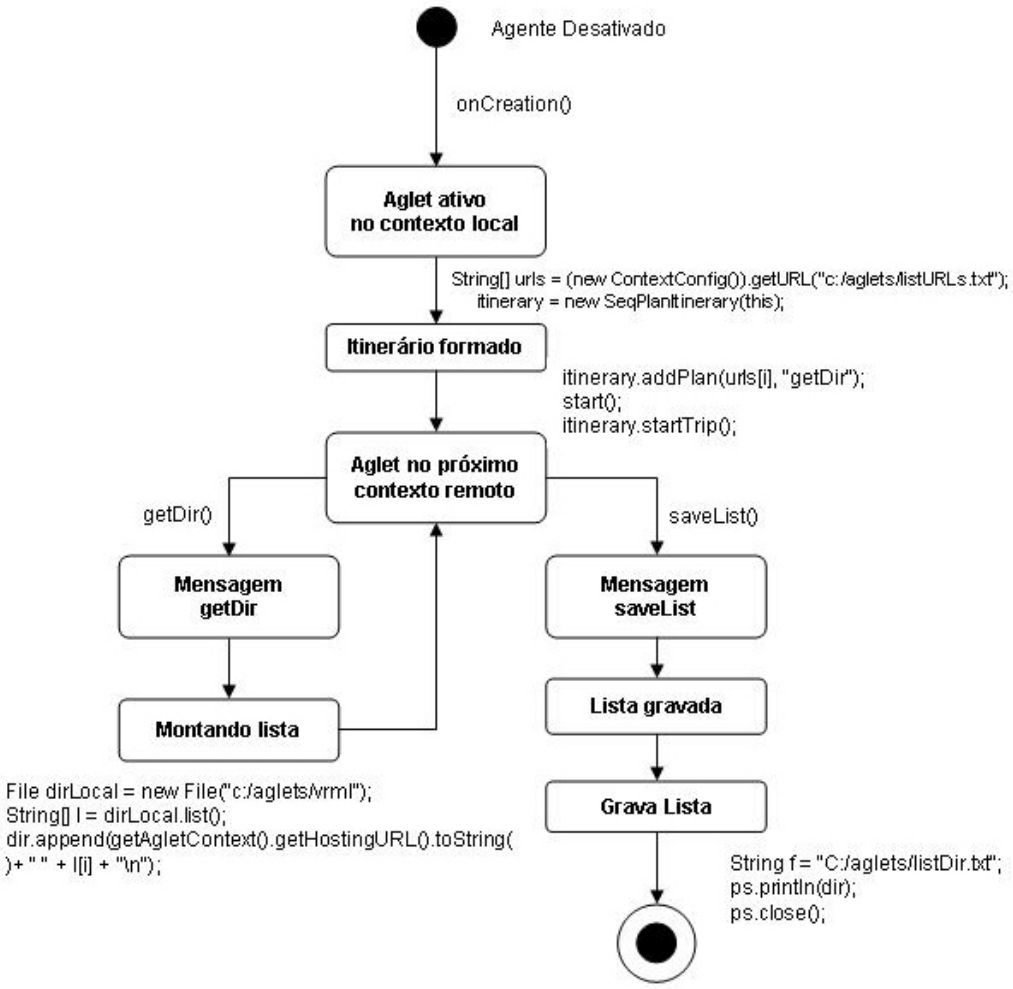


Figura 26 - Diagrama de estados UML da classe "SearchFileNames"

3.1.4.3 Diagrama de Estados: "DownloadFile"

A figura 27 apresenta o diagrama de estados da classe "DownloadFile" que irá fazer o download dos objetos virtuais. As informações sobre os arquivos a serem baixados estão escritas no arquivo "c:/aglets/listDir.txt".

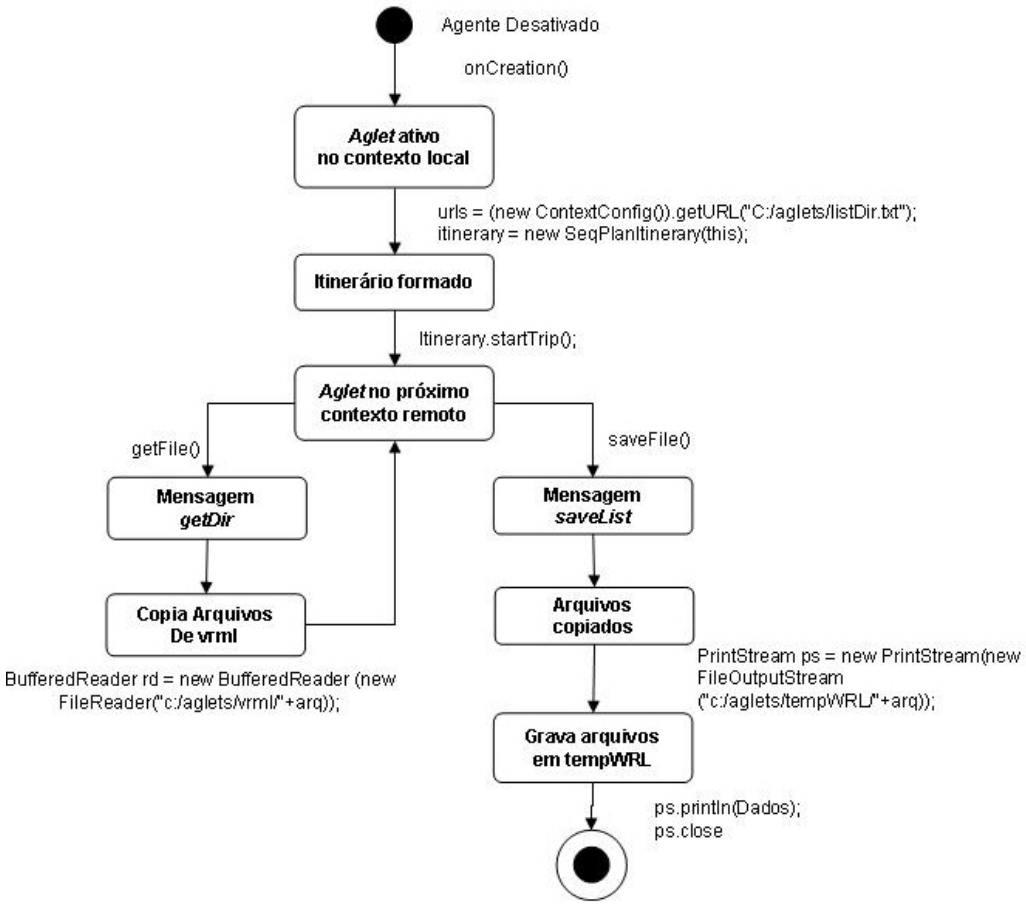


Figura 27 - Diagrama de estados UML da classe "DownloadFile"

3.1.4.4 Diagrama de Estados: "LoadFiles"

A figura 28 apresenta o diagrama de estados da classe "LoadFiles". Esta classe realiza a tarefa de carregamento dos objetos virtuais localizados no diretório c:/aglets/vrml na cena gerada pelo ARToolkit.

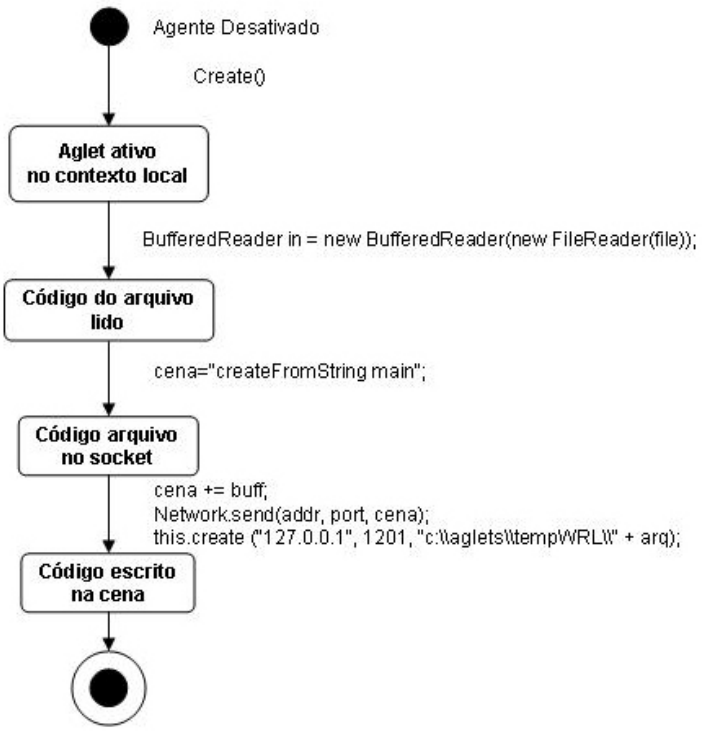


Figura 28 - Diagrama de estados UML da classe "LoadFiles"

4 RESULTADOS E CONSIDERAÇÕES FINAIS

Apresenta-se nesta seção uma discussão dos resultados obtidos com a execução do protótipo na busca, *download* e carga de arquivos na cena. O objetivo é apresentar a real possibilidade de se utilizar a plataforma Aglets, como ferramenta de programação de agentes móveis junto a um ambiente de RA.

Foi realizado um teste com o protótipo "VWBuilder" onde uma máquina chamada de "**micro-r02**" foi o elemento da rede conectado com o ARToolkit e responsável pela criação dos aglets. Outras 5 máquinas fizeram parte do sistema atuando como repositórios de objetos virtuais. Estas máquinas poderiam também executar os agentes. Foram realizados testes em três etapas: (i) Aglet pesquisa e faz *download* de arquivos de apenas uma máquina; (ii) Aglet pesquisa e faz *download* em arquivos distribuídos em cinco máquinas e (iii) Aglets rodam em paralelo, onde as máquinas pesquisam e são pesquisadas ao mesmo tempo.

As máquinas utilizadas para realizar os testes têm a seguinte configuração: "micro-r02": processador de 1.0GHz, 768MB de memória.As outras 5 máquinas ("MICROD01" à "MICROD05") possuem processador de 2.8GHz e 256MB de memória. A rede é uma LAN conectada com um *switch* de 100Mbps, conectada por cabos de par trançado categoria 5.

4.1 Repositórios e Transporte de objetos virtuais

O VWBuilder irá pesquisar e buscar objetos virtuais (arquivos .wrl) em *hosts* da rede. É necessário que se crie uma pasta chamada de "c:/aglets/vrml" em cada máquina que fará parte do sistema. Estes diretórios devem conter arquivos ".wrl". A figura 29 mostra o exemplo de uma pasta "vrml" (repositório).

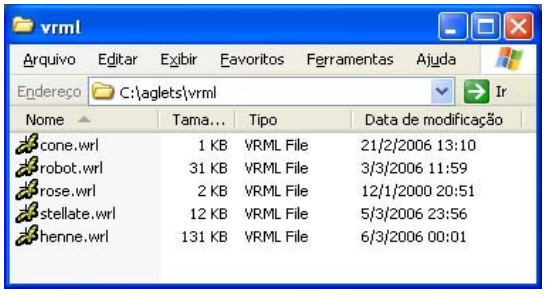


Figura 29 - – Diretório c:/aglets/vrml de uma máquina na rede (MICROD01)

4.2 Ambiente de Execução “Tahiti”

O primeiro passo para se fazer o programa VWBuilder ser executado em uma rede de computadores é realizar a instalação da plataforma “Aglets” nas máquinas que farão parte do sistema. É importante lembrar que, os aglets necessariamente precisam que seu ambiente de execução esteja sendo executado, no caso o Tahiti (Figura 30). Todas as informações a respeito de instalação e configuração de variáveis de ambiente para a execução da plataforma “Aglets” estão disponíveis no Anexo 3 deste trabalho.

Instalada a plataforma e configuradas as variáveis de ambiente, deve-se executar a seguinte linha de comando para abrir um contexto do servidor Tahiti: “agletsd -f c:\aglets\cnf\aglets.props -port 4434”. Onde 4434 é a porta padrão para uso da plataforma “Aglets”, podendo ser alterado. Para dinamizar o trabalho foi criado um arquivo em lote (“4434.bat”) com esta linha de comando.

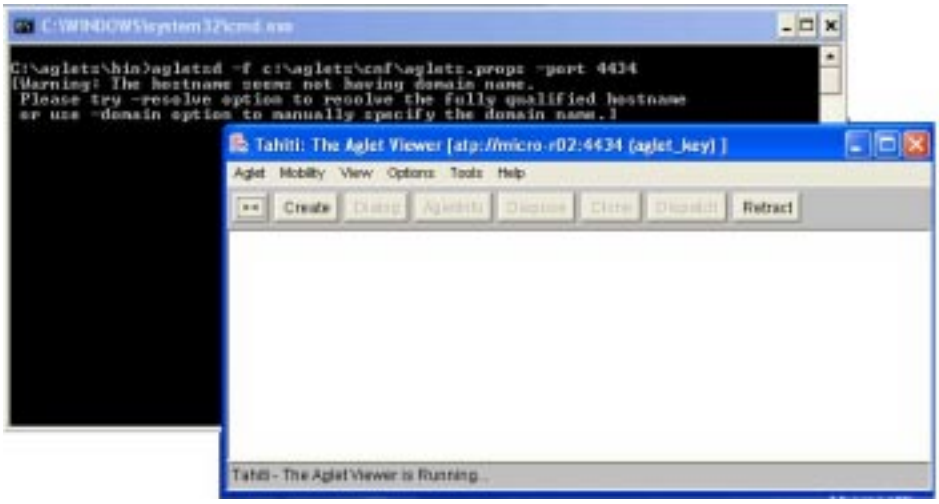
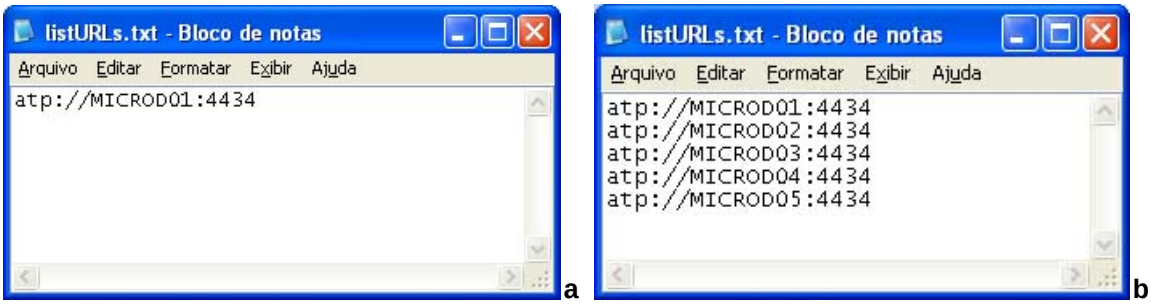


Figura 30 - Ambiente de execução "Tahiti"

4.3 Definição de Itinerário de Busca

Com os repositórios criados e com uma das máquinas contendo os códigos do VWBuilder, a próxima etapa foi configurar manualmente o arquivo “c:/aglets/listURLs.txt” da máquina “**micro-r02**” que terá a função de disparar os agentes na rede. As figuras 31a e 31b mostram exemplos de arquivos “listURLs.txt”.



Figuras 31a e 31b – Arquivos c:/aglets/listURLs.txt do host micro-r02. a) usado no primeiro teste e b) usado no segundo teste

O próximo passo é criar o aglet “SearchFileNames” através do Tahiti. Os arquivos do VWBuilder estão localizados dentro de uma pasta (pacote) em “c:/aglets/public” conforme mostra a figura 32.

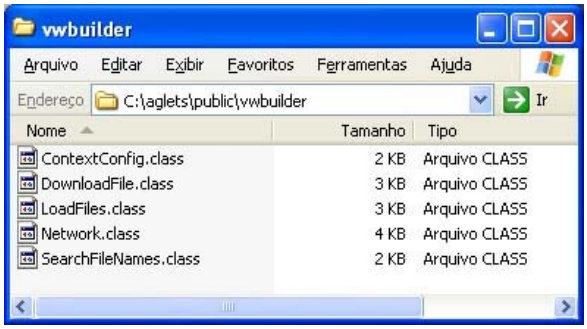


Figura 32 - Pasta c:/agelts/public/vwbuilder com as classes do VWBuilder

Para criar o “SearchFileNames”, deve ser escrito seu caminho completo em “Aglet name” no Tahiti a partir do diretório “c:/aglet/public”. As barras devem ser substituídas por pontos. As figuras 33 e 34 demonstram este passo.

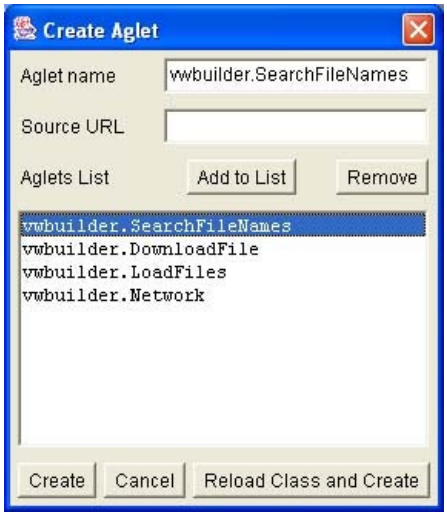


Figura 33 - Criação do aglet “SearchFileNames”

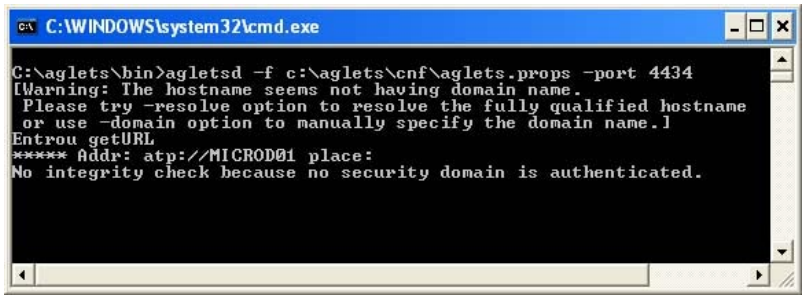
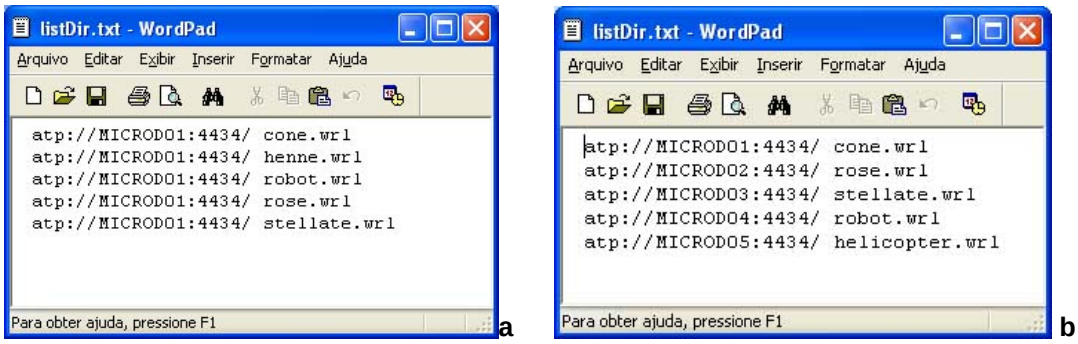


Figura 34 - Console com informações de ação do aglet “SearchFileNames”

Após a execução do aglet “SearchFileNames”, o arquivo “c:/aglets/listDir.txt” deverá ter sido completado com informações a respeito das URLs juntamente com os arquivos “.wrl” encontrados na rede. As figuras 35a e 35b apresentam exemplos de arquivos “listDir.txt” com as informações geradas nos testes.



Figuras 35a e 35b - Visualização do arquivo “listDir.txt” após uso do “SearchFileNames”. a) Teste com arquivos em uma máquina e b) Teste com arquivos em diferentes máquinas

A figura 36 apresenta um gráfico que informa o tempo utilizado pelo agente “SearchFileNames” para percorrer máquinas na rede. Foi observado que o aglet permanece 16 segundos em cada *host* que visita.

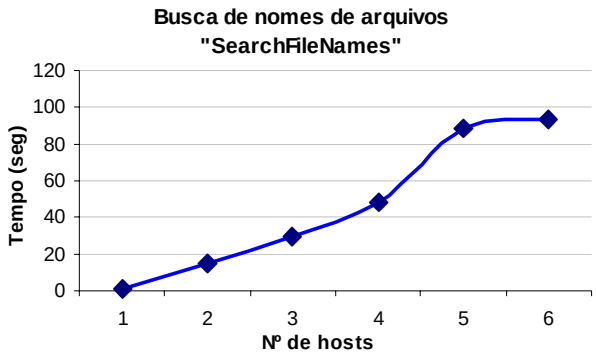


Figura 36 - Gráfico de desempenho (tempo x nº hosts) do "SearchFileNames"

4.4 *Download* de Cenas

Após criar o arquivo “listDir.txt” com as informações a respeito dos arquivos “.wrl” distribuídos nas máquinas na rede, a próxima etapa é criar um agente para fazer o *download* destes arquivos. O aglet “DownloadFile” é o responsável por esta tarefa. Os passos para criação deste agente são os mesmos descritos no item 4.3 (figura 33) no que diz respeito a criação de um aglet. Apresenta-se abaixo a figura 37 com as informações apresentadas no console sobre a cópia dos arquivos “.wrl” distribuídos na rede para o diretório “c:/aglets/tempWRL” da máquina local. A figura 38 apresenta o “Tahiti” com os agentes criados e já retornados ao contexto da maquina local e o diretório com os arquivos copiados.

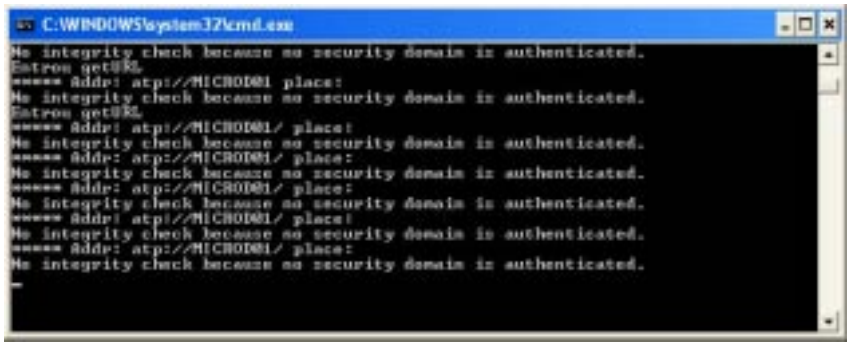


Figura 37 - Console com informações do aglet "DownloadFile"

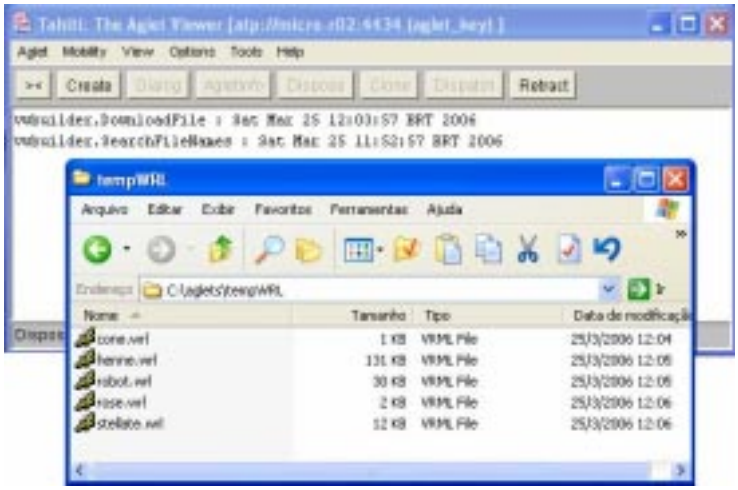


Figura 38 - Agentes no Tahiti e diretório c:/aglets/tempWRL com os arquivos baixados pelo agente “DownloadFile” no segundo teste realizado

Durante os testes observou-se que os aglets, de maneira geral, levam cerca de 16 segundos para migrarem para um novo contexto. Porém, o tempo de transferência de arquivos permaneceu praticamente o mesmo, independente do

tamanho dos arquivos (arquivos variaram de 1K a 1106Kbytes). Apresenta-se na figura 39 um gráfico de performance comparando o tempo que o agente levou para fazer o *download* do arquivo comparado com seu tamanho. Em outro teste, foram baixados 6 arquivos (1, 2, 12, 31, 112, 563 e 1106 Kbytes) com um único aglet. O agente levou 1 minuto e 16 segundos para baixar todos os arquivos.

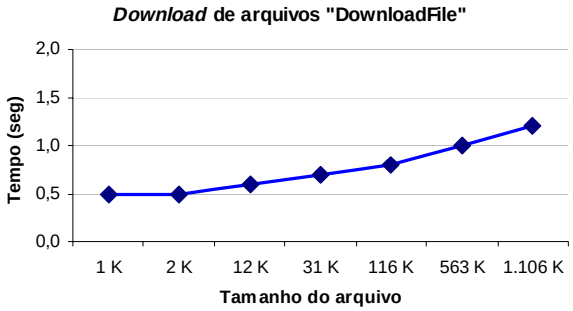


Figura 39 - Tempo de download de objetos virtuais com o agente “DownloadFile”

4.5 Atuação na Cena

Antes de criar o agente “LoadFiles” que carrega os objetos virtuais na cena é necessário executar o aplicativo do ARToolkit. Deve-se primeiro executar o Tahiti e posteriormente executar o ARToolkit. Caso esta seqüência seja invertida verifica-se um erro que impede a execução do Tahiti. Para carregar os objetos virtuais, foi necessário fazer a modificação o arquivo “robot.wrl” para que este possa servir como código base para que a marca relacionada a ele seja utilizada pelo ARToolkit como rastreador para qualquer imagem que estiver no diretório lido pelo agente “LoadFiles”. A figura 40 apresenta o arquivo “robot.wrl” modificado, de forma que o método “main Transform” carregue qualquer arquivo.

```
#VRML V2.0 utf8

DEF   main Transform {
      translation 0 0 0
      rotation 0 0 0 1
      scale 1 1 1
      children [
      ]
}
```

Figura 40 - O arquivo "robot.wrl" modificado para receber qualquer objeto virtual

O processo de criação do aglet “LoadFiles” é exatamente igual aos aglets criados anteriormente. A figura 41 apresenta o agente “LoadFiles” no Tahiti e o console com as informações de carregamento do objeto virtual.

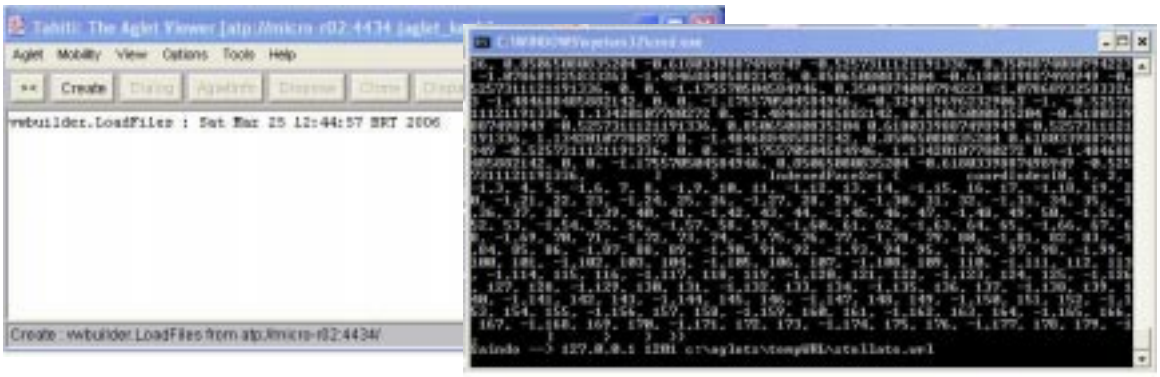
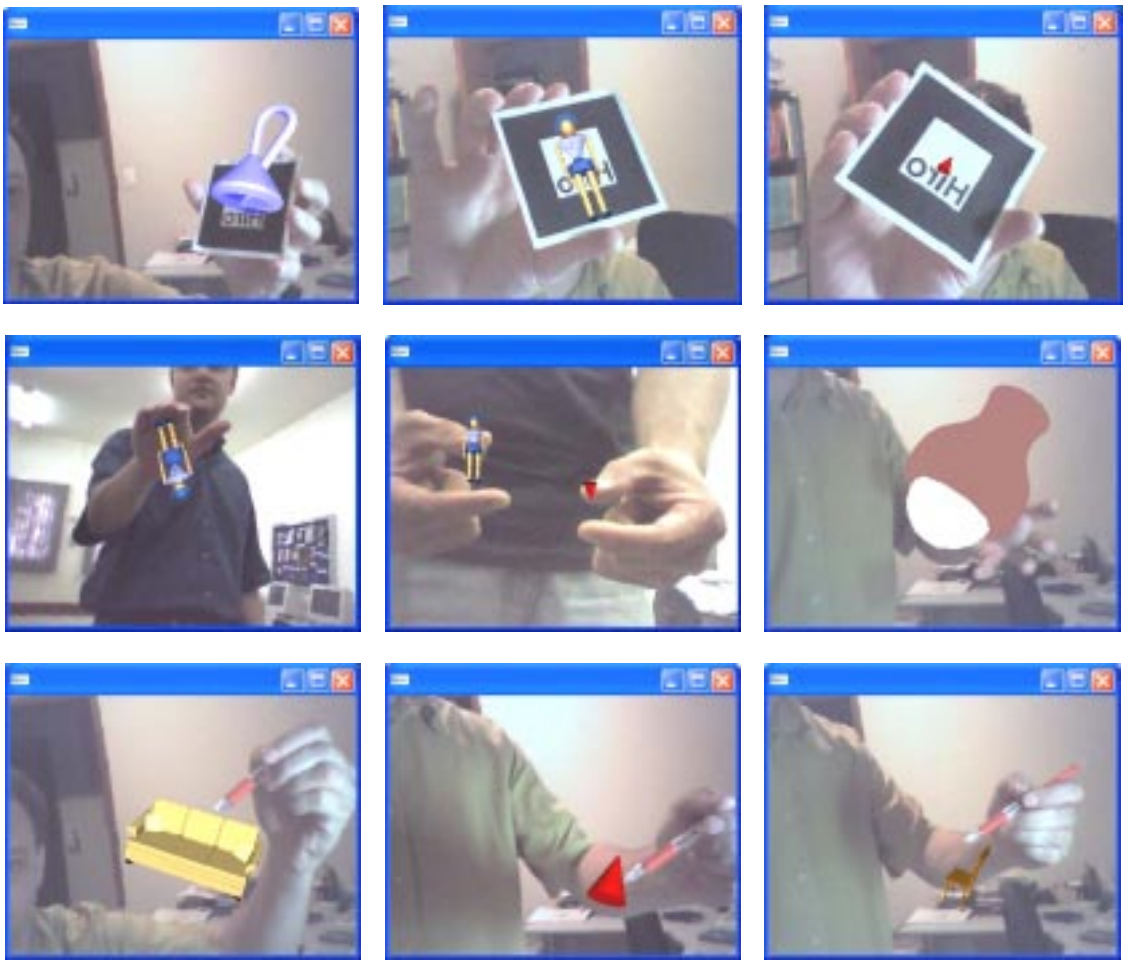


Figura 41 - Agente "LoadFile" no Tahiti e console com informações do arquivo carregado

Apresenta-se abaixo exemplos de cenas (Figuras 42a,b,c,d,e,f,g,h,i) geradas pelo aglet “LoadFiles” durante os testes realizados. Podem ser carregados objetos virtuais separados, um a um em cada cena, e também podem ser carregados todos os objetos virtuais listados no diretório “c:\aglets\tempWRLs”.



Figuras 42 (a,b,c,d,e,f,g,h,i) - Cenas com objetos virtuais carregados pelo "LoadFiles"

A figura 42 apresenta um gráfico que representa o tempo utilizado para carregamento do objeto virtual referente ao tamanho do arquivo.

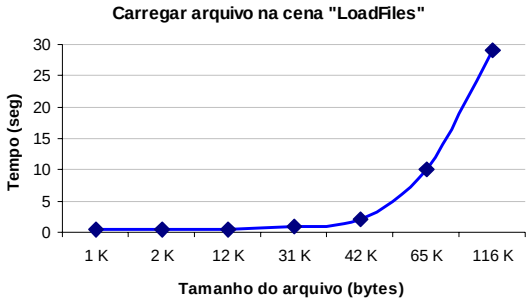


Figura 1 - Gráfico (tempo x tamanho) de arquivos carregados na cena pelo agente "LoadFiles"

Em um outro teste, um único aglet fez a carga de 5 arquivos (1, 1, 2, 31 e 65K), levando 32 segundos para completar a carga dos objetos na cena.

Este capítulo fez uma descrição da estrutura VWBuilder construída para buscar e apresentar objetos virtuais em cenas de realidade aumentada. Foi apresentado a descrição das classes implementadas, os diagramas do projeto e exemplos de funcionalidades do VWBuilder. A demonstração de uso e os gráficos apresentados têm como objetivo apresentar a tecnologia proposta em execução, e confirmar a possibilidade de uso de agentes móveis em um sistema de RA.

4.5.1 Conclusões

Esta dissertação trata do estudo, projeto e implementação de uma estrutura de busca de objetos virtuais (arquivos ".wrl") distribuídos em uma rede de computadores e seu carregamento em uma cena de realidade aumentada. A construção das classes que realizam a busca, *download* e carga na cena é baseado na tecnologia de agentes móveis, e fez uso da biblioteca "Aglets".

Agentes móveis vêm tendo crescimento em interesse da comunidade de pesquisas. Não se espera que as aplicações de agentes móveis substituam as tecnologias utilizadas nas aplicações tradicionais no futuro, mas os agentes têm um grande potencial de dinamizar a realocação de código em execução.

Com relação a plataforma "Aglets" pode-se dizer que a experiência foi positiva pelos motivos: (i) a plataforma está implementada em Java, fazendo que

as classes desenvolvidas possam ser reaproveitadas em trabalhos futuros; (ii) transportar códigos em execução, mantendo o estado dos objetos nos *hosts* visitados abre possibilidades de aplicações em realidade virtual que podem manipular objetos em cenas de uma maneira dinâmica.

Foi apresentada uma estrutura composta por três agentes: o primeiro agente (“SearchFileNames”) é responsável por buscar informações na rede a respeito de objetos virtuais disponíveis. O segundo agente, chamado de “DownloadFile” é responsável por fazer o *download* destes arquivos mapeados pelo agente “SearchFileNames” e gravando-os em um diretório na máquina local. O terceiro agente, chamado de “LoadFiles” é responsável por fazer o carregamento dos objetos virtuais baixados para o diretório local da máquina que criou os agentes em uma cena controlada por um aplicativo de realidade aumentada chamado “ARToolkit”. Neste trabalho, foi utilizada uma versão alterada do “ARToolkit” chamada “NetARToolkit”, que se diferencia do primeiro por ter as funcionalidades de carregamento e interação com objetos virtuais em cenas através de uma rede de computadores. Esta comunicação através de uma classe chamada de “Network” do “NetARToolkit”.

Foram realizados testes com os agentes, onde percebeu-se que, quando agente é criado, ele migra rapidamente para o *host* de destino, permanece neste *host* um determinado tempo, realizando sua tarefa, e retorna com rapidez para o *host* original com a informação já processada. Desta maneira, é possível observar uma das características principais dos agentes móveis que é distribuir o processamento de informações, utilizando-se da rede para enviar e receber informações processadas, através de códigos em execução.

Como resultado principal, foi apresentado evidências da possibilidade de desenvolvimento de aplicações distribuídas de realidade aumentada baseada em agentes móveis. A verificação do comportamento dos agentes pôde ser analisada através da apresentação dos agentes em execução no ambiente Tahiti e a interação com a biblioteca “ARToolkit” onde o objeto virtual foi apresentado.

Uma vantagem, comparada a outros modelos de programação, é que os agentes móveis transportam-se em uma rede, entre *hosts* que possuem o

servidor de agentes (Tahiti) ativado, e possibilita de códigos em execução migrem entre *hosts*, executem suas tarefas, e retornem para o *host* original que o criou. Neste modelo, todo *host* do sistema pode enviar e/ou receber agentes.

Os agentes utilizam um tempo grande para executar suas tarefas nos *hosts*. Isto pode ser visto como uma desvantagem da tecnologia. Também existem problemas de compatibilidade de plataformas Java e de padrões de comunicação, mas isto não impediu que fosse realizada a integração entre a estrutura “VWBuilder” e a biblioteca “NetARToolkit”.

Como contribuição, pretendeu-se desenvolver um estudo que fizesse uso das tecnologias baseadas em agentes móveis, juntamente com a área de RA. O resultado deste estudo deu origem um protótipo formado por agentes que podem ser utilizados como parte de outros projetos que busquem explorar a mobilidade de códigos em execução em sistemas distribuídos de RV e RA.

Como possíveis melhorias e trabalhos futuros, podem-se utilizar o protótipo desenvolvido, através da herança de suas classes, como base para uma aplicação completa que controle o “ciclo de vida” de objetos virtuais em cenas de mundos virtuais distribuídos. Os três agentes que formam o “VWBuilder” podem ser utilizados de maneira independente. Isto permite que possam utilizados em aplicações dinâmicas baseadas em agentes que monitoram ambientes virtuais, ou sistemas reais com representações através de mundos virtuais. Os agentes podem ser disparados para realizar ações nestes ambientes, como também podem permanecer em ambientes distribuídos, monitorando o sistema e enviando para um sistema base informações de atualização. Outra possibilidade ainda, é utilizar os agentes móveis para que realizem o carregamento de imagens em sistemas baseados em mundos virtuais que façam uso do LOD (*Level of Detail*) com o objetivo de otimizar a atualização e objetos virtuais.

5 REFERÊNCIAS BIBLIOGRÁFICAS

AUKSTAKALNIS, S., BLATNER, D. **Silicon Mirage: The Art and Science of Virtual Reality**, Peatchpit Press, Berkeley, CA, 1992.

AZUMA, R. et al. **Recent Advances in Augmented Reality**. IEEE Computer Graphics and Applications, v.21, Nov/Dec. 2001, p.34-37.

BAX, M. P. e SOUZA, R. R. Uma proposta de uso de agentes e mapas conceituais para representação de conhecimentos altamente contextualizados ***in Simpósio Internacional de Gestão do Conhecimento***, 2001**.** Disponível em: http://www.emack.com.br/info/apostilas/nestor/agentes.pdf. Acessado em Fevereiro de 2006.

BERGAMASCHI, S. **The MIRKS Project**, 2002 Disponível em www.dbgroup.unimo.it/Miks/selectedAgentSoftware.html. Acessado em Janeiro de 2006.

BERRE, A. et al. **ARToolkit for dummies**. 2002. Disponível em http://lance.ufrj.br/grva/realidade_aumentada.html. Acessado em Março de 2002.

BREWINGTON, B. et al. Mobile agents in distributed information retrieval. **In Intelligent Information Agents, chapter 12**, Springer-Verlog, 1999. Disponível em http://citeseer.ist.psu.edu/brewington99mobile.html. Acessado em Dezembro de 2005.

BRYSON, S. Virtual Reality in Scientific Visualization**.** ***Communications of the ACM***, vol.39, n.5, p.1-9, 1996. Disponível em http://citeseer.nj.nec.com/bryson95virtual.html. Acessado em Julho de 2003.

BURDEA, G., COIFFET,P. **Virtual Reality Technology**, John Wiley & Sons, New York, New York, 1994.

CÁCERES, E. N. et al. Algoritmos paralelos usando CGM/PVM/MPI. **In XXI Congresso da SBC, Jornada de Atualização de Informática**, 2001, p.219-278. Disponível em: http://www.ime.usp.br/~song/papers/jai01.pdf. Acessado em Março de 2006.

CAETANO, A.; PEREIRA, J. Representação de Agentes Autônomos em VRML 97. ***In Proceedings de 9º Encontro Português de Computação Gráfica.*** Marinha Grande, Portugal, p.145-154, 2000. Disponível em: http://virtual.inesc.pt/9epcg/actas/pdfs/artigo18.pdf. Acessado em Julho de 2003.

CALONEGO JR, N., CONSULARO, L. A. **Manual do ARToolkit**. Faculdade de Ciências Matemática, da Natureza, UNIMEP, 2004.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Distributed Systems - Concepts and Design**, 3th Edition, Addison-Wesley, 2001.

COUTINHO, E., PORTO, F. A., OLIVEIRA, J. ADVICE: Armazenamento e Recuperação de Objetos Virtuais em Ambientes Virtuais Colaborativos para SIG-3D, ***in Proceedings of VI Symposium on Virtual Reality SVR 2003***, Ribeirão Preto, p.453-460, 2003.

DA LUZ, R. P. **Proposta para plataforma para experimentos em Realidade Virtual**. Florianópolis, 2002. Tese apresentada à Faculdade de Engenharia de Produção da Universidade Federal de Santa Catarina. Disp. em http://www.lrv.eps.ufsc.br. Acessado em Janeiro de 2004. 102p.

DEITEL, H. M; DEITEL, P. J. **Java, como programar**, 3ª edição, Porto Alegre, Brasil: ed Bookman, 2001. 1201p.

DEITEL H. M; DEITEL P. J. CHOFFNES, D. R. **Sistemas Operacionais**, 3ª edição, São Paulo, Pearson Pretince Hall, 2005.

EID, M. et al. Trends in Mobile Agent Applications. ***In Journal of Research and Practice in Information Technology***, Vol. 37, n. 4 p323-351, 2005. Disponível em http://www.acs.org.au. Acessado em Janeiro de 2006.

FABRE, Y. A Framework for Mobile-Agents Embodied in X3D Networked Virtual Environment. ***Proceeding of the 8th international conference on 3D Web-technology***, Saint Malo, France, p113-120, 2003.

FERNANDESa, J. H. C, **Ciberespaço: Modelos, Tecnologias, Aplicações e Perspectivas: da vida artificial a busca por uma Humanidade Auto-Sustentável**. Artigo apresentado à EINE2000/SBC na Universidade Federal do Rio Grande do Norte, Recife, 2000.

FERNANDESb, R. F.; PIRES, H. C.; VRML+: uma plataforma de desenvolvimento rápido de mundos VRML. ***In Proceeding of 9º Encontro Português de Computação Gráfica***, Marina Grande, Portugal, p163-170, 2000.

FERRARI, L., **The Aglets 2.0.2 User's Manual. 2004.** Disponível em http://aglets.sourceforge.net. 2004. Acessado em Dezembro de 2005.

FREITAS, F. L. G., BITTENCOURT, G. **Comunicação entre Agentes em ambientes Distribuídos Abertos: o Modelo "*peer-to-peer*"**. UFSC Universidade Federal de Santa Cataria, 2002. Disponível em http://www.sbc.org.br/reic/edicoes/2002e2/informativos/ComunicacaoEntreAgentesEmAmbientesDistribuidosAbertos-OModeloPeerToPeer.pdf. Acessado em Março de 2006.

GREENOP, D. et al. **Cyberspace: The next frontier for the TelCos?** EURESCOM, Schloss-Wolfsbrunnenweg, Heidelberg, Germani, 1999. Disponível em http://www.eurescom.de/~pub-deliverables/P800-deries/P843. Acessado em Setembro de 2003.

HAMIT, F. **Realidade Virtual e a Exploração do Espaço Cibernético**. Berkeley Brasil Editora, Original Sams Publishing, 1993.

HANDCOCK, D. Viewpoint. **Virtual Reality Search of Middleware Ground**, IEEE, 1995.

HERRMANN, B. VENUS – Virtual colaborative environment with next generation multimídia services. ***In EURESCOM, Technology for collaborative applications***. Project P922-PF, Volume 2, 2000. Disponível em http://www.eurescom.de/public/projects/P900-series/p922/default.asp. Acessado em Outubro de 2003.

HOLMES, J. The Availability of VRML Models on the Internet. ***In Departament of eletronics and Computer Science***, Southmpton, UK. 2002. Disponível em http://www.mms.ecs.soton.ac.uk/papers/10.pdf. Acessado em Agosto de 2003.

IBM. **IBM Aglets Home Page**. IBM – International Business Machine. Disponível em http://www.trl.ibm.com/aglets. Acessado em Junho de 2003.

JUNG, B.; MILDE, J. T. An Open Virtual Environment for Autonomous Agents Using VRML and Java. ***In Proceedings for 4th Symposium on Virtual Reality Modeling Language***, Paderbon, Germany, 1999. 5p.

KATO, H.; BILLINGHURST M.; POUPYREV, I. **ARToolkit Manual**, November., 2003.

KIRNER, C. **Sistemas de Realidade Virtual.** Grupo de Pesquisa em Realidade Virtual da UFSCar. São Carlos – SP, 2001. Disponível em http://www.dc.ufscar.br/~grv/tutrv/tutrv.htm. Acessado em Agosto de 2003.

KIRNER, C, TORI, R. Introdução à Realidade Virtual, Realidade Misturada e Hiper-realidade, *in* ***VII Symposium on Virtual Reality***, São Paulo, 2004.

KNAPIK, M; JOHNSON, J. **Developing intelligent agents for distributed systems: exploring architecture, technologies and applications**. New York USA, McGraw-Hill, 1998. 389p.

LANGE, D. B.; OSHIMA, M. **Programming and Deploying Java Mobile Agents with Aglets**. Addison-Wesley, 1998. 225p.

LANGE, D. B. Java Aglet Application Programming Interface (J-AAPI). In **White Paper, IBM Tókio Research Laboratory**. 1997. Disponível em http://www.research.ibm.com/trl/aglets/JAAPI-whitepaper.htm, Acessado em Junho de 2003.

LEA, R.; MATSUDA, K.; MIYASHITA, K. **Java for 3D VRML Worlds**. Ed. New Riders, 1996. 399p.

LEE, H. B. **AGENT Construction Tools**, 2000. Disponível em: http://www.paichai.ac.kr/~habin/research/agent-dev-tool.htm. Acessado em Fevereiro de 2006.

LEITE JR, A. J. M. **ATAXIA: Uma arquitetura para a viabilização de NVE voltados para a educação a distância através da Internet.** Dissertação apresentada ao Mestrado em Ciência da Comutação na Universidade Federal do Ceará, Fortaleza-CE, 2000.

LOBATO, C., A. et al. **Agentes Móveis: aglets na busca de informações**. Trabalho apresentado para conclusão do curso de Ciência da Computação na Universidade Federal do Pará, Belém, 2003.

MEIGUINS B. S., GUEDES, L. A. Arquitetura para Suporte à Comunicação de Aplicações de Realidade Virtual Colaborativas. ***in Proceedings of VI Symposium on Virtual Reality SVR 2003***, Ribeirão Preto, p.467-74, 2003.

MELO, M. A., BORIN, E. **Um Sistema de Agentes Móveis**. Campinas, 2001. Projeto apresentado no curso de Mestrado em Ciência da Computação da Unicamp – Universidade de Campinas.

MILAGRES, F. G.; MOREIRA, E. S. **Especificação e Implementação de Agentes Móveis em um Sistema de Detecção de Intrusão**. Relatório de iniciação científica apresentado ao ICMC da USP, São Paulo, 2001.

MILGRAM, P. et. al. **Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum**. Telemanipulator and Telepresence Technologies, SPIE, V.2351, 1994, p.282-292.

MINAR, N. **Designing an Ecology of Distributed Agents**. Massachusetts, EUA, 1998. 92p. Dissertação apresentada ao MIT – Massachusetts Institute of Technbology para obtenção de grau de Mestre. Disponível em http://www.media.mit.edu/~nelson/. Acessado em Agosto de 2003.

MONTESCO, C. A. E. UCL – **Uma Linguagem de Comunicação para Agentes de Software**. São Carlos, 2001. Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação USP – Universidade de São Paulo. Disponível em http://java.icmc.usp.br /dilvan/papers /2003-TIL/estombelo_moreira_10.pdf. Acessado em Junho de 2004.

MORCRETTE C. **VRML Generation tools for visualization of database content in three dimensions**. 1999. *Thesis of Master of Science*. Disponível em http://ceci.mit.edu/research/database_visualization/papers/thesis.pdf. Acessado em Outubro de 2003.

NAGAMUTA, V. **Coordenação de Agentes Móveis Através do Canal de Broadcast**. São Paulo, 1999. Dissertação apresentada ao IME-USP. Disponível em: http://www.teses.usp.br/teses/disponiveis/45/45134/tde-11102005-123304. Acessado em Junho de 2004.

NUNES, H., LABIDI, S. Mobile agents for information extraction in Mathnet System, ***in 32nd ASEE/IEEE Frontiers in Education Conference, Session F2E*****,** Boston, 2002. Disponível em http://citeseer.ist.psu.edu/nunes02mobile.html. Acessado em Janeiro de 2006.

NWANA, H. S. A brief introduction to software agent technology, ***In ACM Agent Technology: Fundations, Aplications, and Markets****.* New York, Ed Springer, p.29-47, 1998.

OAKS, S. **Segurança de dados em Java.** Rio de Janeiro, editora Ciência Moderna Ltda, 1999.

OLGUIN, C. J. M et al. O uso de agents em ambientes de aprendizagem colaborativos, ***in Proceedings of the XI Simpósio Brasileiro de Informática, 2000.*** Disponível em: http://www.dca.fee.unicamp.br/projects/sapiens/Papers/Sbie/uaaa2000.pdf. Acessado em Dezembro de 2005.

OSHIMA, M., KARJOT, G. **Aglets Specifications (1.0)**, IBM Coorp. 1997 Disponível em http://www.research.ibm.com/trl/aglets/spec10.htm. Acessado em Julho de 2003.

PAPASTAVROU. S., SAMARAS, G., PITOURA, E., Mobile Agents for Word Wide Web Distributed Database Access, ***in IEEE Transaction on Knowledge and Data Engineering, vol. 12, nº 5***, 2000. Disponível em http://citeseer.ist.psu.edu/papastavrou00vobile.html. Acessado em Fevereiro de 2006.

PEREIRA, A. L. V. **O uso da tecnologia de agentes para filtragem e classificação automática de mensagens eletrônicas**. Campinas, São Paulo, 2000. Dissertação (Mestrado) – PUC, Campinas-SP.

PERUZZA, A. P. M. Ferramenta educacional ConstruirRV: Construir conhecimento utilizando realidade virtual. ***In Proceedings of VI Symposium on Virtual Reality – SVR 2003***. 2003. Ribeirão Preto, p.162-171, 2003.

RAPOSO, A. B. et al. **Visão Estereoscópica, Realidade Virtual, Realidade Aumentada e Colaboração. Tecgraf** – Grupo de Tecnologias em Computação Gráfica, Departamento de Informática, PUC - Rio. 2005, Disponível em http://www.tecgraf.puc-rio.br/publications. Acessado em Fevereiro de 2006.

ROQUE, V.M.G.; OLIVEIRA, J.L. CORBA, DCOM e JAVARMI – Uma análise comparativa, ***in Encontro de Engenharia Informática 99 – Ordem dos Engenheiros (EEI'99)***, Braga, Portugal, 1999. p.126-136. Disponível em http://www.ipg.pt/user/~vitor.roque/artigos. Acessado em Dezembro de 2003.

ROSA JR., O. Ambientes Virtuais Cooperativos: LRVCHAT3D, Um Estudo de Caso, ***in Proceedings of 4th SBC Symposium on Virtual Reality***, Florianópolis, Brasil 2001, p.1-11. Disponível em http://www.lrv. eps.ufsc.br/index.php?destino=30. Acessado em Dezembro de 2003.

ROSA, A. L. M., ARAGÃO, A. L. Ambiente Colaborativo Utilizando Realidade Virtual na WWW*, **in Proceedings of VI Symposium on Virtual Reality SVR 2003***, Ribeirão Preto, p.407, 2003.

RUMBAUGH, J., JACOBSON, I., BOOCH, G. **UML: Guia do Usuário**, editora Campus, Rio de Janeiro, 2000.

SCHIAVONI, F, L, **Agentes Autônomos Inteligentes**, UEM, Disponível em http://www.din.uem.br/ia/agentes. Acessado em Junho de 2002.

SHAW, C. et al. Decoupled Simulation in Virtual Reality with MR Toolkit, ***ACM Transection of Information Systems***, 11(3);287-317, July 1993.

SILVA, I. C. F. da. **Desenvolvimento de um ambiente para criação de animações de cenas VRML para Web.** Dissertação de mestrado apresentada a FEEC e Computação da Univ. Federal de Campinas, 2001.

SINGHAL, S.; ZYDA, M. **Network Virtual Environment. Designs and Implementations**. Addison-Wesley, 1999, 331p.

SOUZA, M. P. A. Agentes Móveis em Ambientes Virtuais Tridimensionais – Aglets e Java3D. ***In Proceedings of VI Symposium on Virtual Reality, SVR 2003*****,** Ribeirão Preto, p.363-365. 2003.

TANENBAUM, **Sistemas Operacionais Modernos**, 2ª edição, Rio de Janeiro: ed. Prentice Hall, 2003. 712p.

THEILMANN, W, ROTHERMEL, K. maintaining Specialized Search Engines though Mobile Filter Agents. ***In Proceedings of the Third International Workshop on cooperative Information Agents***. Alemanha, 1999. Disponível em http://citeseer.ist.psu.edu/theilmann99maintaining.html. Acessado em Janeiro de 2006.

UHRMACHER, M.A.; KULLICK, B. G. Plug and Test: Software agents in virtual environments. ***In Proceedings of 32nd conference on winter simulation, Orlando***, Florida, p.1722-1729. 2000.

VELLOSO, P. B.; et al. Uma Ferramenta de gerenciamento de redes baseada em agentes móveis. ***In Proceedings XIV Congresso Brasileiro de Automática***, Natal, Rio Grande do Norte, p-679-684, 2002.

WEB3D, **VRML97 SPECIFICATIONS, 2004**. Disponível em http://www.web3d.org/x3d/specifications/vrml. Acessado em Março de 2006.